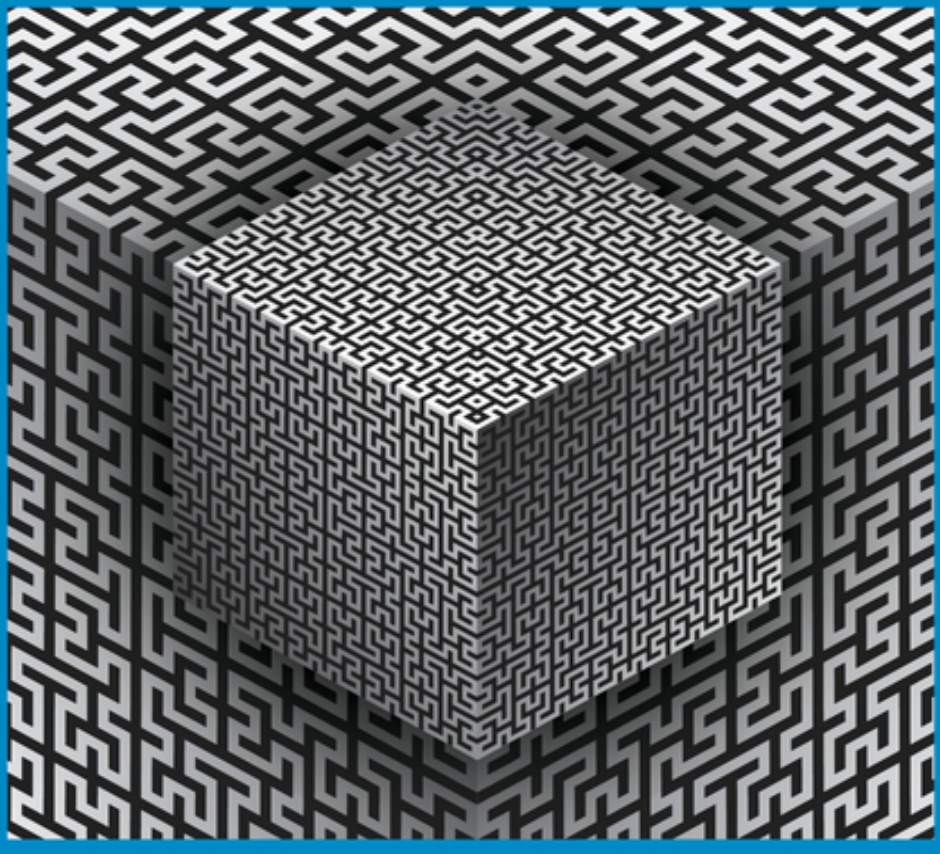
MARIA PETROU
REVISING AUTHOR SEI-ICHIRO KAMATA

IMAGE PROCESSING

DEALING WITH TEXTURE



SECOND EDITION

with website

WILEY

Image Processing

Image Processing

Dealing with Texture

Second Edition

(the late) Maria Petrou
formerly, Imperial College, London, UK

Revising Author
Sei-ichiro Kamata
Waseda University, Tokyo/Kitakyushu, Japan

WILEY

This second edition first published 2021

Edition History
John Wiley & Sons, Ltd (1e 2006)

Registered Offices
John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA
John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

Editorial Office
The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

Library of Congress Cataloging-in-Publication Data

Names: Petrou, Maria, author. | Kamata, Sei-ichiro, author.
Title: Image processing : dealing with texture / Maria Petrou, Imperial College, London, UK, Sei-ichiro Kamata, Waseda University, Tokyo, Japan.
Description: Second edition. | Hoboken : Wiley, 2021. | Includes index.
Identifiers: LCCN 2020028752 (print) | LCCN 2020028753 (ebook) | ISBN 9781119618553 (cloth) | ISBN 9781119618577 (adobe pdf) | ISBN 9781119618591 (epub)
Subjects: LCSH: Image processing–Digital techniques.
Classification: LCC TA1637 .P48 2021 (print) | LCC TA1637 (ebook) | DDC 617.6/342–dc23
LC record available at https://lccn.loc.gov/2020028752
LC ebook record available at https://lccn.loc.gov/2020028753

Cover Design: Wiley
Cover Image: © Trikona/Shutterstock

Set in 9.5/12.5pt STIXTwoText by SPi Global, Chennai, India

10 9 8 7 6 5 4 3 2 1

Contents

Preface to the Second Edition

Image Processing: Dealing with Texture 1st Edition was published in January 2006. This is a comprehensive guidebook to texture analysis techniques with traditional algorithms and their examples. Meanwhile, a lot of methodologies have been developed over the past 14 years. In order to describe the world of texture analysis again, we need to perform a large revision on dealing with texture in image processing. The following topics are mainly revised and/or added in the second edition.

(1) 3.3 Fractals and multifractals: this section is updated based on the newly developed techniques using fractals and multifractals. The multifractals are newly added in part.
(2) 3.4 Image statistics: this section is updated based on the important fundamentals in image statistics by rewriting Sections 3.6 and 3.8 of the first edition.
(3) 3.8 Texture repair: this section is newly added based on the development of texture repair techniques.
(4) 4.4 Local phase features: this section is newly added based on the development of local phase feature techniques.
(5) 4.6 Dual tree complex wavelet transform: this section is newly added based on the dual tree complex extension of wavelet transform technique.
(6) 4.7 Ridgelets and curvelets: this section is newly added based on several extensions of the wavelet transform technique.
(7) 4.12 Deep texture features: This section is newly added based on the recent development of deep neural network techniques.

Image Processing: Dealing with Texture, 2nd Edition, newly includes the MATLAB codes of the various algorithms and examples presented in this book. We hope the codes are helpful for understanding several algorithms behind the technologies dealing with textures in image processing.

For supplementary materials, including most of the executable programs used to create the examples in the book, please visit the accompanying website at www.wiley.com/go/kamataText2.

Preface to the First Edition

It is often said that everybody knows what texture is but nobody can define it. Here is an unconventional definition: texture is what makes life beautiful; texture is what makes life interesting and texture is what makes life possible. Texture is what makes Mozart's music beautiful, the masterpieces of the art of the Renaissance classical and the facades of Barcelona's buildings attractive. Variety in detail is what keeps us going from one day to the next and the roughness of the world is what allows us to walk, communicate and exist. If surfaces were smooth, friction would not exist, the Earth would be bombarded by asteroids and life would not have developed. If surfaces were smooth, pencils would not write, cars would not run, and feet would not keep us upright.

Texture is all around us, and texture is also on the images we create. Just as variation in what we do allows us to distinguish one day in our life from another, texture allows us to identify what we see. And if texture allows us to distinguish the objects around us, it cannot be ignored by any automatic system for vision. Thus, texture becomes a major part of image processing, around which we can build the main core of image processing research achievements.

This book is exactly trying to do this: it uses texture as the motivation to present some of the most important topics of image processing that have preoccupied the image processing research community in recent years. The book covers the topics that have already been well established in image processing research and it has an important ambition: it tries to cover them in depth and be self-contained so that the reader does not need to open other books to understand them.

The book is written on two levels. The top, easy level, is for the reader who is simply interested in learning the basics. This level is appropriate for an undergraduate or Master's level course. The second level goes in depth, demonstrating and proving theorems and concepts.

This level is appropriate for research students. Examples that refer to the advanced level are marked with a B and the theory of this level is presented in boxes with a grey background. In a sense, the book is an interlacing of mainstream material and appendices that cover advanced or even peripheral issues.

The book aspires to be a classical textbook on image processing and not an account of the state of the art. So, the reader who hopes to find here the latest algorithms proposed, will be disappointed.

A large part of this book was written when the first co-author was on sabbatical at the Informatics and Telematics Institute in Thessaloniki, Greece. The support of the Institute as well as the support of our home institutions, namely the University of Surrey for the first co-author when writing this book, and the University Jaume I, throughout this endeavour is gratefully acknowledged.

We would also like to thank the Media Lab of the Massachusetts Institute of Technology for allowing us to use five images from the VisTex database, the Signal and Image Processing Institute of the University of Southern California for allowing us to use three images from their USC-SIPI Image database, and Dr Xavier Llado who supplied the images shown in Figures 1.5 and 1.6.

For the accompanying website please visit www.wiley.com/go/texture.

Maria Petrou and Pedro García Sevilla

Acknowledgements

Sadly the first author, Dr Maria Petrou passed away in 2012. She was a great scientist, and it was a great pleasure to have known her and establish some joint research projects, and to finally be able to make this second edition book. The second author would like to express deep appreciation to her son, Mr Costas Petrou, and Dr Pedro Garcia Sevilla for their contributions to this book.

The second author also appreciates many thanks to colleagues, Ms Haoran Liu and Mr Weili Chen for helping to make the materials.

Sei-ichiro Kamata
Waseda University, Tokyo, Japan

About the Companion Website

This book is accompanied by a companion website: 

www.wiley.com/go/kamataText2

The website includes:

- MATLAB codes.

Scan this QR code to visit the companion web site:

1

Introduction

What is Texture?

Texture is the variation of data at scales smaller than the scales of interest. For example, in Figure 1.1 we show the image of a person wearing a Hawaiian shirt. If we are interested in identifying the person, the pattern on the shirt is considered as texture. If we are interested in identifying a flower or a bird on the shirt, each flower or bird of the pattern is a non-textured object, at the scale of this image, as we can hardly see any detail inside it.

Why are we interested in texture?

We are interested in texture for two reasons.

- Texture may be a nuisance in an automatic vision system. For example, if we were to recognise an object from its shape, texture would create extra lines in the edge map of the object and the shape recognition algorithm would be confused. This is demonstrated in Figure 1.2.
- Texture may be an important cue in object recognition as it tells us something about the material from which the object is made. For example, in the image of Figure 1.3 we may discriminate the city from the woods and the fields using the type of variation the image shows at scales smaller than the objects we are talking about.

How do we cope with texture when texture is a nuisance?

There are algorithms that can isolate the texture regions irrespective of the type of texture. The texture regions then may be identified and treated separately from the rest of the image. Since texture manifests itself as image intensity variation, it gives rise to many edges in the edge map of the image. A trivial algorithm that can isolate texture regions is to consider a scanning window and count inside the window the number of edgels. Any window with number of edgels higher than a certain threshold is considered as belonging to a textured region. Figure 1.4 demonstrates the result of this algorithm applied to the image of Figure 1.3. In Box 1.1 a more sophisticated algorithm for the same purpose is presented.

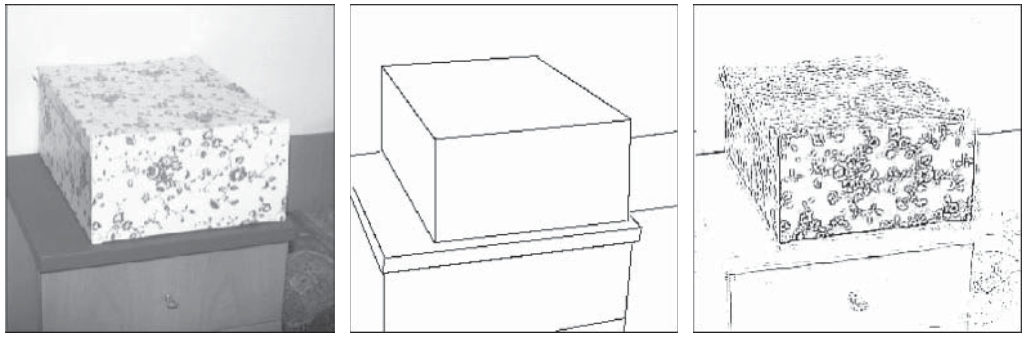
This algorithm is as follows.

Step 1: Perform edge detection in the image to produce edge map A, where all edgels are marked with value 1 and the non-edgels with value 0.

Image Processing: Dealing with Texture, Second Edition. Maria Petrou and Sei-ichiro Kamata.

Companion Website: www.wiley.com/go/kamataText2

Figure 1.1 Costas in bloom. Source: Maria Petrou.



(a) (b) (c)

Figure 1.2 (a) An original image. Source: Maria Petrou. (b) Manually extracted edge map. (c) The automatic edge extraction algorithm is confused by the presence of texture on the box.

Step 2: Create an output array B, the same size as the edge map A, and set all its elements equal to 1 or 255 (all white).

Step 3: Consider an $M \times M$ window, where M could be, for example, 15, 21, etc., depending on the resolution you desire. Select also a threshold T, such that $0 << T < M^2$.

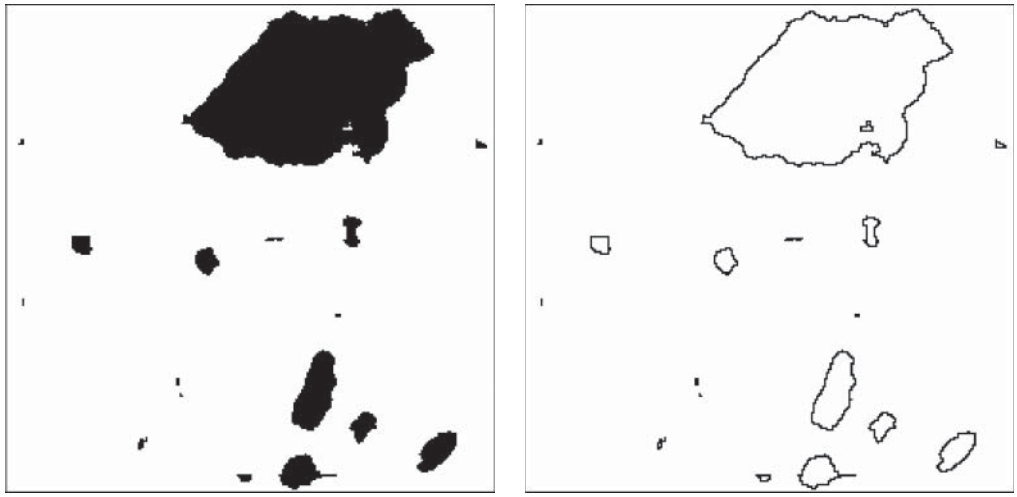
Step 4: Scan A with this window and each time sum up the values of edge map A covered by the window. Assign the output to the central pixel of the window in the output array B. Note that B may be obtained by convolving edge map A with a flat window of size $M \times M$, i.e. a window with all its weights equal to 1.

Step 5: Threshold B, so that if one of its elements has value below T becomes white, while if it has a value above or equal to T it becomes black (value 0). Call this output array C. This way you will obtain an output like that of Figure 1.4a.

Step 6: Consider each black pixel in C, and examine its 3×3 neighbourhood. If even one of those neighbours has value white, keep this as black pixel. If all its neighbours are black, convert it

(a) (b)

Figure 1.3 (a) Blewbury from an aeroplane (size 256 × 256). Source: Maria Petrou. (b) Edge map where the urban area has been annotated. Other texture patches correspond to woods.



(a) (b)

Figure 1.4 (a) When a scanning window of size 17 × 17 was placed in Figure 1.3b with its centre at a pixel marked here as black, the number of edgels inside the window was more than 100. (b) Boundaries of the black regions in (a).

into white. Note that this may be achieved by convolving B with a flat (all elements equal to 1) 3×3 window. If the result of the convolution for a pixel is more than 0, the value of the pixel in the output array should be 0; otherwise it should be 1. (This assumes that the white pixels have value 1.) This way, you will produce a result corresponding to Figure 1.4b.

How does texture give us information about the material of the imaged object?

There are two ways by which texture may arise in **optical images**:

- Texture may be the result of variation of the **albedo** of the imaged surface. For example, the image in Figure 1.2a shows the surface of a box on which a coloured pattern is printed. The change of colours creates variation in the brightness of the image at scales smaller than the size of the box, which on this occasion is the object of interest.
- Texture may be the result of variation of the **shape** of the imaged surface. If a surface is rough, even if it is uniformly coloured, its image will be textured due to the interplay of shadows and illuminated parts. The textures in the image of Figure 1.3 are the result of the roughness of the surfaces of the town and the woods when seen from above.

Are there non-optical images?

Yes, there are. They are the images created by devices that measure something other than the intensity of reflected light. For example, the grey value in **magnetic resonance images (MRI)**, used in medicine, may indicate the proton density of the imaged tissue. Another example concerns images called **seismic sections**, which are created by plotting data collected by seismographs. In this case, the grey value represents the intensity of the reflected seismic wave by the different rocks that make up the crust of the Earth.

What is the meaning of texture in non-optical images?

Texture in non-optical images indicates variation of a certain physical quantity at scales smaller than the scales of interest. For example, in a proton weighted MRI image of the brain, the presence of texture indicates variation of the proton density from one location to the next. Such variation will manifest itself in the image in the same way as variation of the albedo or surface roughness does in an optical image. From the image processing point of view, texture is treated in the same way in optical and non-optical images. So, in this book we shall talk about images irrespective of the sensor with which they have been captured.

What is the albedo of a surface?

The albedo of a surface is a function that characterises the material from which the surface is made. It gives the fraction of incident light this material reflects at each wavelength. When we say that a surface is multicoloured, effectively we say that it has an albedo that varies from location to location.

Can a surface with variable albedo appear non-textured?

Yes, if it is imaged by a single band sensor and the intensity of reflected light by the different coloured patches within the sensor sensitivity band yields constant accumulated recordings.

Can a rough surface of uniform albedo appear non-textured?

Yes, if the roughness of the surface and the incident light are such that no part of the surface is in shadow, all parts of the surface receive the same amount of light and the surface is mat (i.e. the material it is made from reflects equally in all directions, irrespective of the direction from which it is viewed). An example is a rough mat surface seen under totally diffuse ambient light.

Figure 1.5 A surface imaged from two different distances may create two very different image textures. Source: Maria Petrou.

What are the problems of texture that image processing is trying to solve?

Image processing is trying to solve three major problems concerned with texture.

- Identify what a surface represents by analysing the texture in an image of a surface sample: this is the problem of **texture classification**.
- Divide an image into regions of different textures: this is the problem of **texture segmentation**.
- Decide whether a texture is as it is expected to be or contains faults: this is the problem of **texture defect detection**.
- Create a new texture or repair a damaged texture.

What are the challenges in trying to solve the above problems?

The most important challenge is trying to infer information concerning the surface (i.e. its roughness (surface relief) and its albedo) by studying the texture of the image of the surface. The problem is that the same surface may appear totally differently textured under different imaging conditions. Figure 1.5 shows the same surface imaged from different distances. Its appearance changes dramatically and pure image processing without any extra information would never recognise these two images as depicting the same object. Another example is shown in Figure 1.6 where the same surface is imaged from the same distance, but with the illuminating source being at two different positions. Again, pure image processing would never recognise these two images as depicting the same object. These two examples illustrate two important properties of texture:

- Texture is scale-dependent
- Image texture depends on the imaging geometry.

How may the limitations of image processing be overcome for recognising textures?

Image processing that makes use of additional information, e.g. information concerning the imaging geometry, may be able to deal with both the scale dependence of texture as well as its dependence on the imaging geometry. An alternative way is to use computer vision techniques that allow one to recover the missing information suppressed by the imaging process. For example, a rough surface exists in 3D. However, when it is imaged on a 2D medium, the information concerning the third dimension is lost. This information is independent of the imaging process. If one could recover it from the available image, then one would have had a scale-invariant description of the

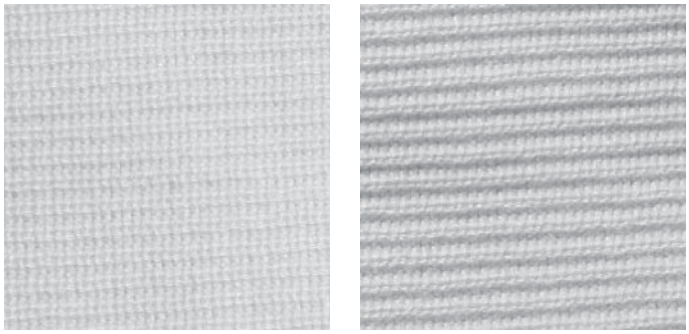


Figure 1.6 A surface imaged under different illumination directions may give rise to two very different image textures. Source: Maria Petrou.

roughness of the surface. The recovery of such information is possible if one uses extra information, or more than one image.

What is this book about?

This book is about the methods used to deal mainly with the first two problems related to texture, namely texture classification and texture segmentation. However, we shall also consider the problem of **in-painting**, i.e. the problem of repairing damaged texture. The various methods will be presented from the point of view of the data rather than the approach used; we shall start from the simplest image textures, i.e. 2D binary images, and proceed to composite grey texture images.

Box 1.1 An algorithm for the isolation of textured regions

Texture in an image implies intensity variation at scales smaller than the image size. It is well known that intensity variation gives rise to edgels in the edge map of the image. A high density of edgels, therefore, characterises textured regions.

Let us imagine that each edgel is a person standing in a flat landscape. The person sees the world around them through the other individuals who might surround them and obscure their view. Suppose that we define the field of view of each individual as the maximum angle through which they can see the world through their neighbours who stand within a certain distance from them. Individuals who are standing in the middle of a textured region will have their views severely restricted and therefore their fields of view will not have large values. On the other hand, individuals standing at the borders of textured regions will be able to see well towards the outside of the textured region and they will enjoy high values of field of view.

The trouble is that even edge pixels that simply separate uniformly coloured regions will enjoy equally high values of the field of view. So, we need to devise an algorithm that will distinguish these two types of edgel: those that delineate textured regions and those that simply form edges between different uniformly shaded regions.

The following are the steps of such an algorithm demonstrated with the help of the image of Figure 1.7a.

Figure 1.7 (a) Original image of size 256×256 showing a town. Source: Maria Petrou. (b) The edge map of the image obtained by using the Sobel edge detector. (c) The edgels that have field of view larger than 90° within a local neighbourhood of size 9×9 (i.e. $n = 9$). (d) The edge map of panel (b) without the edgels of panel (c) and any edgel in a neighbourhood of size 7×7 around them (i.e. $m = 7$). (e) The edgels of panel (d) plus all edgels from panel (b) that are in a neighbourhood of size 7×7 around them (i.e. $\delta m = 0$). (f) The final result keeping only the edgels of panel (e) that have field of view larger than 90° within a neighbourhood of 9×9.

Step 1: Identify the edgels in the image using an edge filter that does very little or no smoothing (e.g. Sobel filters). Call the output edge map A. An example is shown in Figure 1.7b.

Step 2: Consider a neighbourhood of size $n \times n$ around each edgel. Sort the neighbours of the edgel in a clockwise (or anti-clockwise) order around it. Find the angle between any two successive neighbours with the vertex of the angle being the edgel of interest. The largest of these angles is the field of view of the edgel of interest. This process is demonstrated in Figure 1.8.

(Continued)

Box 1.1 (Continued)

Step 3: In A keep only the edgels that have a field of view larger than d degrees. Call this output B. Such a result is shown in Figure 1.7c.

Step 4: Around each edgel you have kept in B, consider a window of size $m \times m$. Delete all edgels in A that are inside this window. Call the result C. An example is shown in Figure 1.7d. Note that in this way you have eliminated all those edgels with high viewing angles that do not delineate texture regions. However, at the same time the texture regions have shrunk.

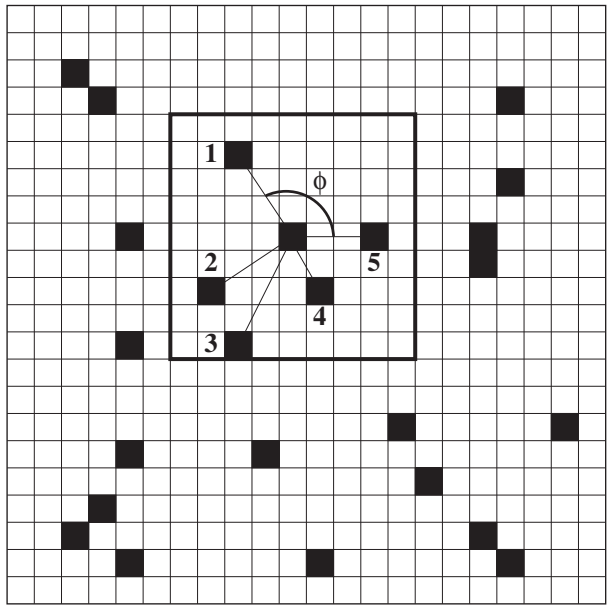


Figure 1.8 Each tile in this figure represents a pixel. The black tiles represent edgels. The windowed region with the thick black line represents the neighbourhood of size $n \times n$ we consider around the edgel in the middle. In this case $n = 9$. The neighbours inside this window are sorted in an anti-clockwise direction. The numbers next to them indicate their rank. We measure the angle between any two successive rays from the central edgel. The largest of these angles, marked here with ϕ, is the field of view of the central edgel.

Step 5: Around each edgel you have kept in C, consider a window of size $(m + \delta m) \times (m + \delta m)$. Restore inside the window all edgels that are present in A. Call the result D. An example is shown in Figure 1.7e. Note that now you have only the edgels that constitute the textured region. Constant δm makes sure that the window used in this step is slightly larger than the window used in the previous step, because we are dealing with discrete points which are at a certain average distance from each other.

Step 6: For each edgel in D calculate the field of view as you did before. Keep only the edgels that have a field of view larger than a threshold d. This is the final result. For the example the final result is shown in Figure 1.7f.

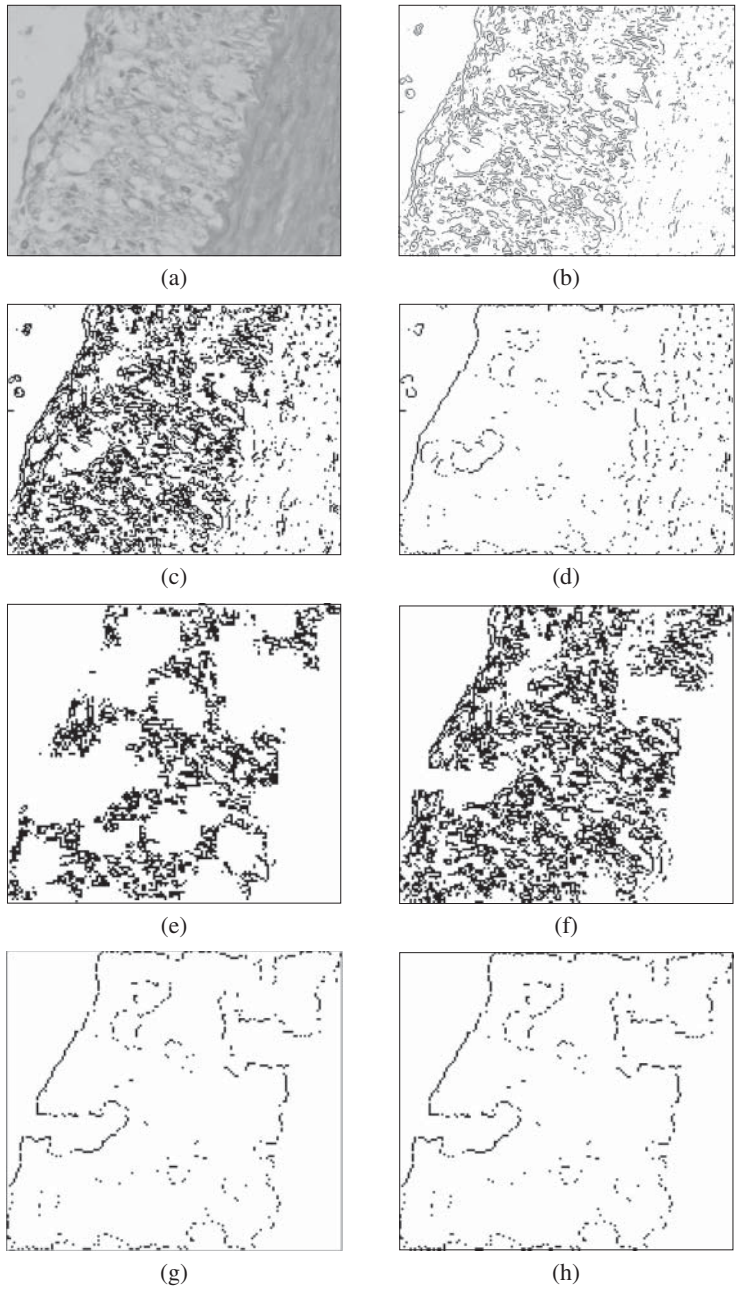


Figure 1.9 (a) An original image of size 382×287, showing the cross section of the aorta of a rabbit that had too much cholesterol. The edgels in the texture region are sparse. Source: Maria Petrou. (b) Edge map produced using a Sobel edge detector. (c) The edge map made denser by replacing every 4 pixels by 1 and marking the new pixel as an edgel, even if only a single pixel in the original 2×2 configuration was an edgel. (d) The edgels that have a field of view larger than 90° within a local neighbourhood of size 5×5 in panel (c). (e) The edgels of panel (c) after the edgels in (d) and their neighbouring edgels inside a window of size 4×4 are removed. (f) The edgels of panel (e) plus all their neighbouring edgels inside a window of size 6×6 in panel (c) restored. (g) The edgels in (f) which have a field of view larger than 90°. (h) The final result in the original size.

(Continued)

Box 1.1 (Continued)

This algorithm works well for texture regions with dense edgels. If the edgels are not very dense, but you still want to identify the textured regions, you may preprocess the edge map using the following step.

Step 1.5: Make the edge map denser by replacing every $l \times l$ tile of pixels by a single pixel. Even if only one of the pixels in the tile is an edgel, make the replacement pixel an edgel too. This way the edge map will shrink but the number of edgels will be preserved and the map will be made denser.

The result may be brought to the full size by the following step.

Step 7: Replace every pixel with a tile of size $l \times l$. All pixels in the tile will be marked either as edgels or background according to how the original pixel was marked. This will yield a thick boundary for the textured region. If this is a problem the boundary may be thinned while preserving its continuity.
Steps 1.5 and 7 are demonstrated in Figure 1.9.

2

Binary Textures

Why are we interested in binary textures?

A grey or colour image contains a lot of information, not all of which is necessary in order to convey its contents. For example, in the image of Figure 2.1 we can easily recognise the trees and the birds although only two brightness levels are used. Texture indicates variation of brightness. The minimum number of brightness levels we can have is two, so, the simplest way of representing an image is to binarise it.

What is this chapter about?

This chapter is about developing some basic tools that will allow us to quantify binary textures.

Are there any generic tools appropriate for all types of texture?

No, there are not. This is because there is a vast variety of textures one may encounter. In fact, every surface we see may appear textured at some resolution. So, the variety of textures is as great as the variety of the world around us.

Can we at least distinguish classes of texture?

Yes, there are some broad texture classes. Look at Figure 2.2 and try to spot the odd one out. You will find the answer in the next paragraph.

What are the texture classes?

There are two broad categories of texture: regular and irregular. Most textures of man-made objects are regular, like those in Figures 2.2a and 2.2b, while most natural textures are irregular. Texture 2.2d is semi-regular; it consists of regularly shaped objects placed at random positions. The odd one out in Figure 2.2 is texture 2.2c, because it shows a random pattern, while all other images exhibit some regularity.

Image Processing: Dealing with Texture, Second Edition. Maria Petrou and Sei-ichiro Kamata.

Companion Website: www.wiley.com/go/kamataText2

Figure 2.1 We can easily recognise the depicted objects, even from a binary image. Source: Maria Petrou.

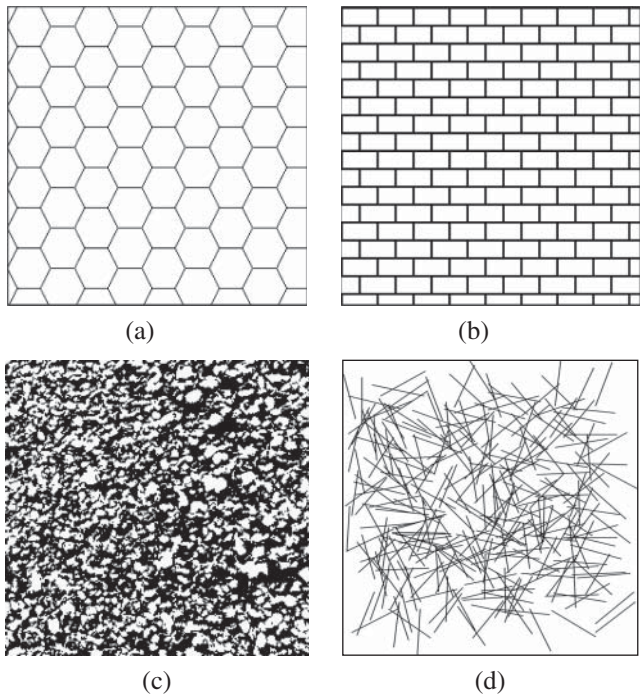


Figure 2.2 Four binary textures: which one is different from the other three?

Which tools are appropriate for each type of texture?

For regular textures we use **shape grammars** to describe them. For irregular textures we use statistical descriptors based on **Boolean models** and **mathematical morphology**.

2.1 Shape Grammars

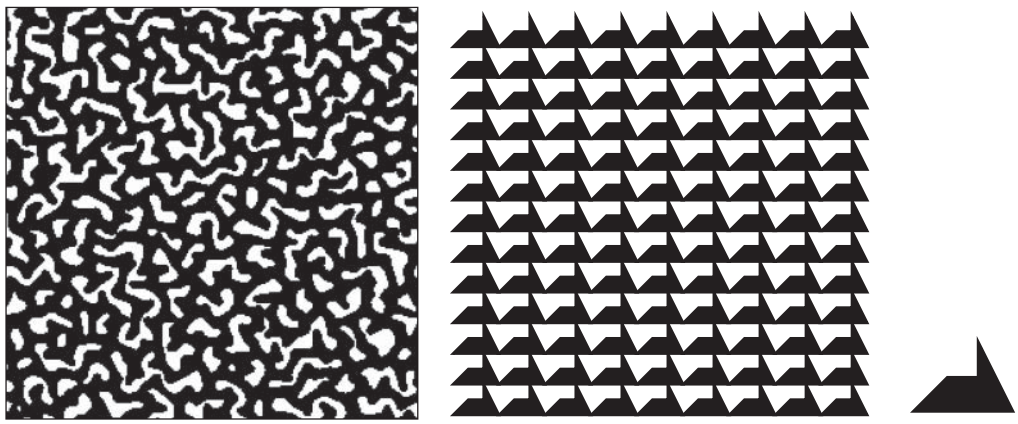
What is a shape grammar?

A shape grammar is a formal way of describing a regular pattern. For example, if we wanted to describe with words the pattern of Figure 2.3a, it would have been quite tedious. However, if we were to describe with words the pattern of Figure 2.3b, we could simply say: it is a rectangular arrangement of the pattern shown in Figure 2.3c, with the patterns tightly packed from all directions. So, in order to describe a pattern we need basically two things: some elementary patterns and a set of placement rules. Formally, we need something more: we need some symbol to mark the position where we "stick" the elementary patterns together, and we need a starting pattern.

Box 2.1 Shape grammars

A shape grammar is defined as a collection of four items.

1. A set V of shapes, called **terminals**. These are in effect the primitive patterns from which we can build up the texture.
2. A set U of shapes called **markers**. These are small markers which indicate the places where the "new bricks" should be placed when building the texture pattern.
3. A mapping between two sets W_1, and W_2, which consist of elements that are elements of sets V and U and combinations of them. The mapping is many to one, allowing us to replace an element of set W_1 by an element of set W_2. These are the placement rules.
4. A starting pattern S. This must be an element of W_1 so that we can start building the texture from it.



(a) (b) (c)

Figure 2.3 (a) A random texture. (b) A texture with a regular primitive pattern and regular placement rules. (c) The primitive pattern of texture (b) scaled up.

Example 2.1

Consider the texture of Figure 2.4. Write a shape grammar to describe it.

Figure 2.4 An example of a regular pattern.

Figure 2.5 There may be more than one primitive pattern that may be used to characterise the same texture.

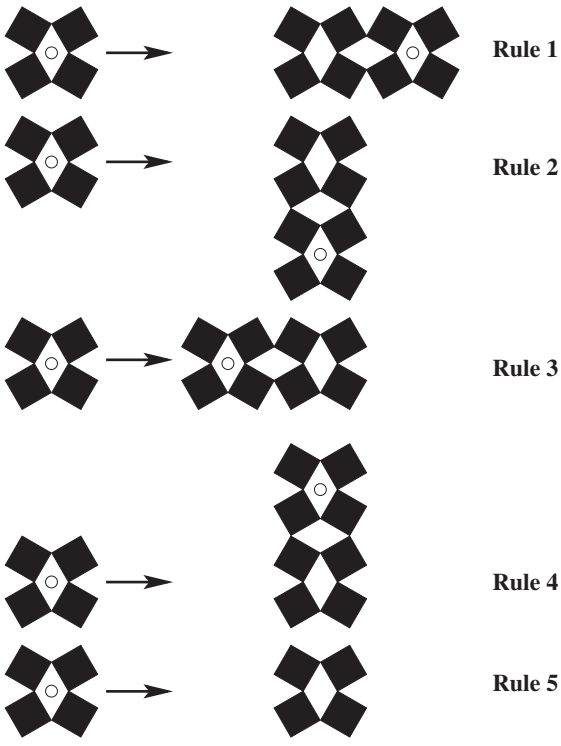


Figure 2.6 The rules of a grammar that may be used to produce the texture of Figure 2.4, using as primitive element the first pattern in Figure 2.5.

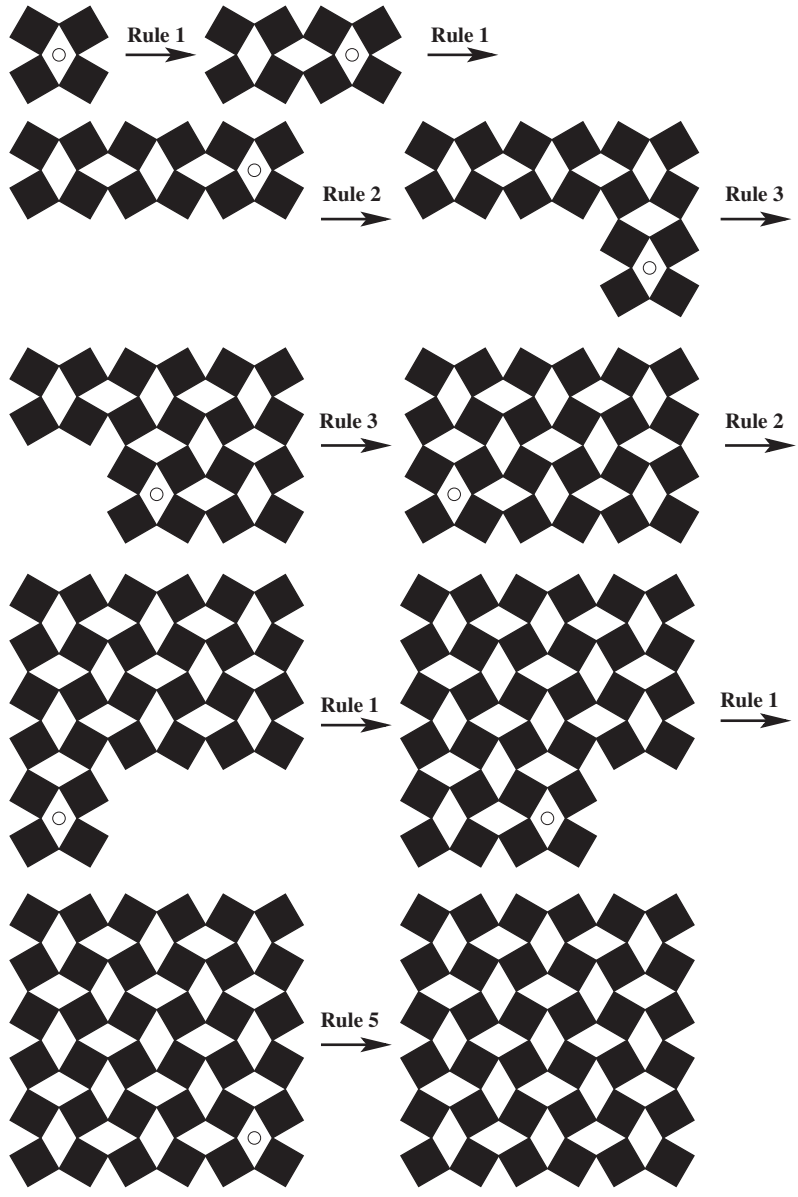


Figure 2.7 Successive application of the rules of Figure 2.6 allows us to reproduce the texture of Figure 2.4. The sequence of rules used characterises the texture.

(Continued)

Example 2.1 (Continued)

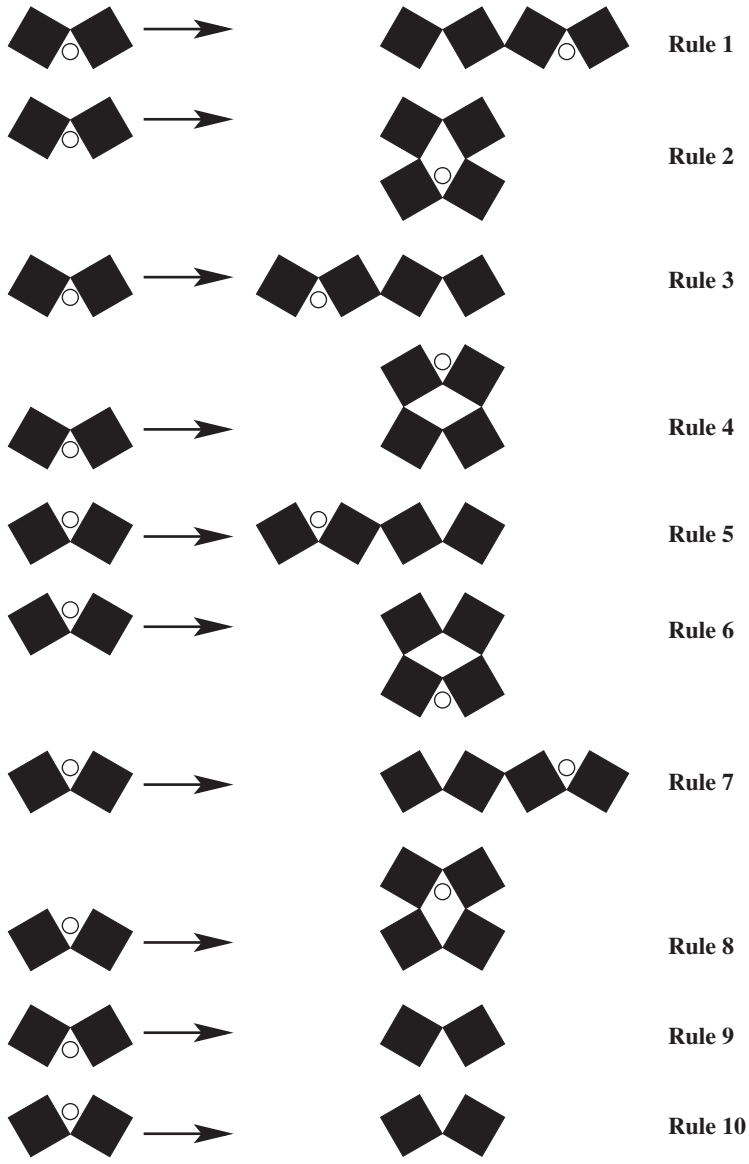


Figure 2.8 The rules of a grammar that may be used to characterise the texture of Figure 2.4, using as primitive element the third pattern in Figure 2.5. The simpler the primitive elements we use, the more rules we need.

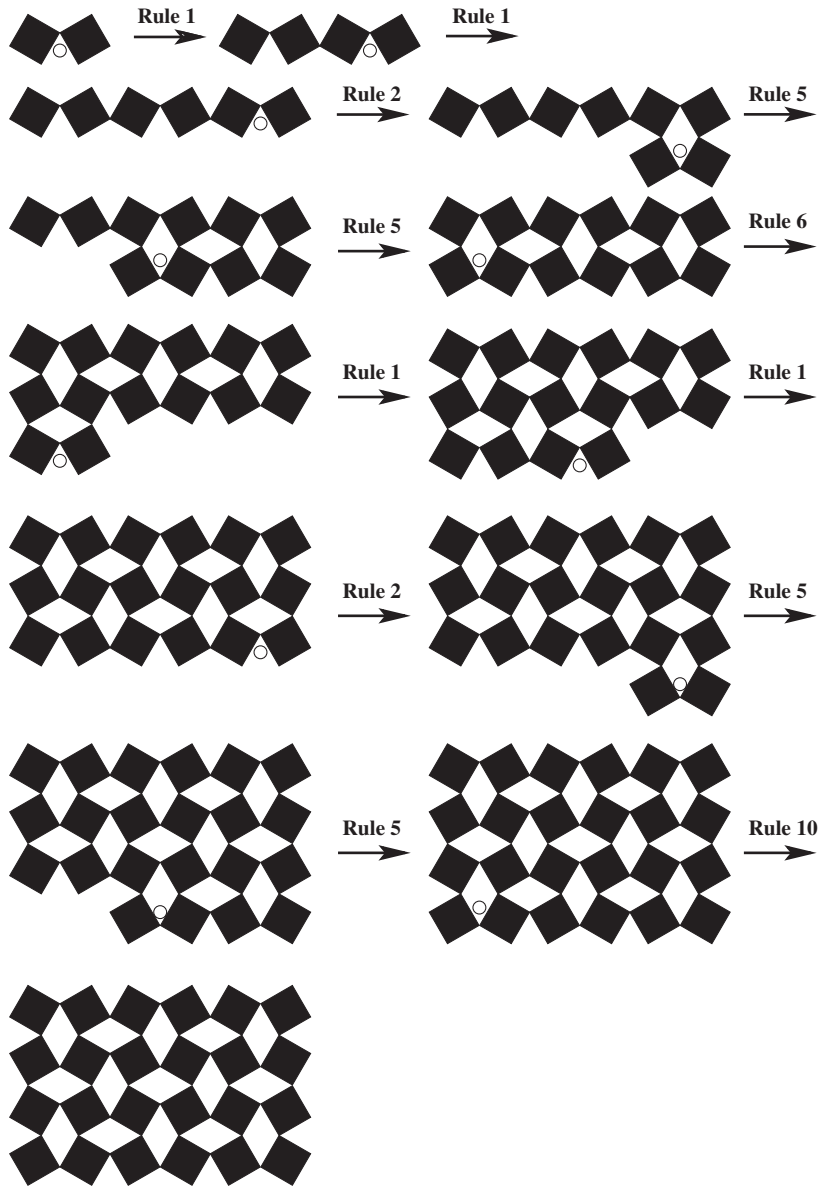


Figure 2.9 Successive application of the rules of Figure 2.8 allows us to reproduce a section of the texture of Figure 2.4.

(Continued)

Example 2.1 (Continued)

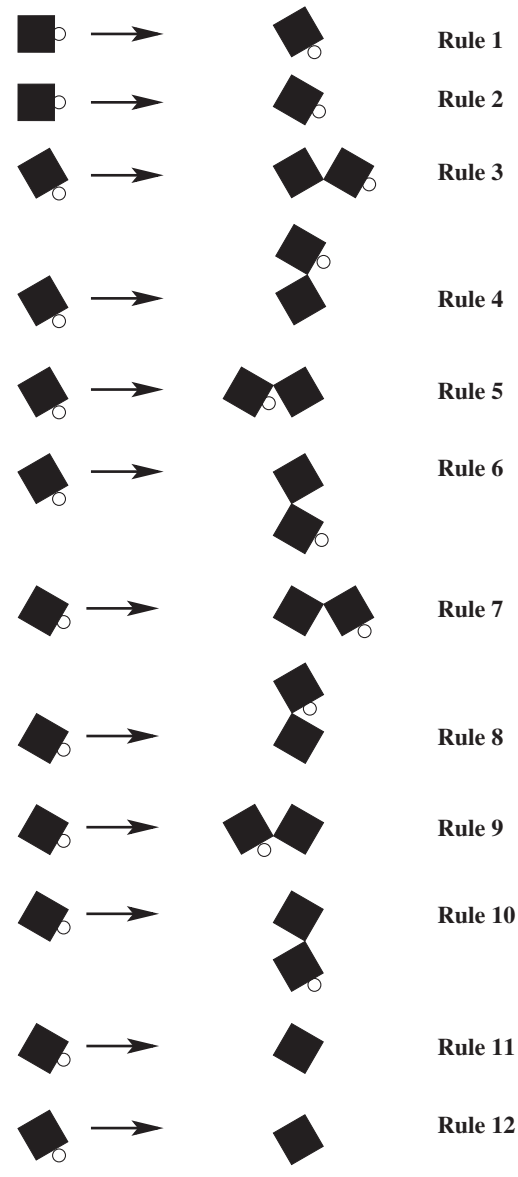


Figure 2.10 The rules of a grammar that may be used to characterise the texture of Figure 2.4, using as primitive element the fifth pattern in Figure 2.5.

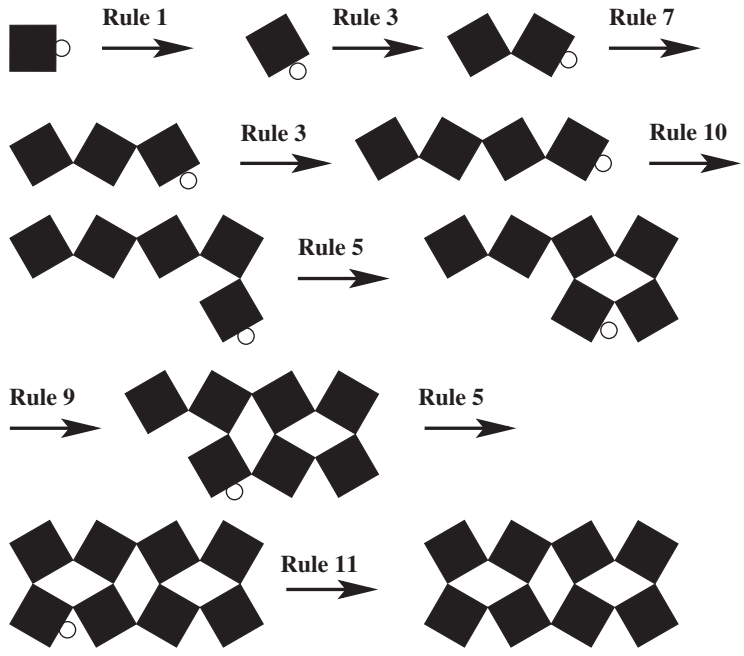


Figure 2.11 Successive application of the rules of Figure 2.10 allows us to reproduce a section of the texture of Figure 2.4.

The first problem we have to solve is the choice of the primitive pattern. We may choose any one of those shown in Figure 2.5.

Let us choose the first one of these patterns. Let us also choose a small circle to be used as a marker. Figure 2.6 is the set of rules we need in order to produce the texture of Figure 2.4. Finally, we need to choose a starting element. Let us choose as the starting element the left-hand side element of rule 1. Figure 2.7 is an example of how texture 2.4 may be produced by the successive application of the proper rules. We can see that the texture we had may be fully described and reproduced on the basis of the five rules of Figure 2.6.

Figure 2.8 shows the set of rules we would need to reproduce the pattern if we had chosen the third primitive pattern of Figure 2.5. Figure 2.9 shows how we may apply these rules to reproduce a section of the texture of Figure 2.4. Notice that now we need the application of more rules to construct a section of the same size as before. This is because our primitive element is now simpler. This becomes more obvious if we choose as our primitive pattern the last pattern of Figure 2.5. The rules needed in this case are shown in Figure 2.10, and their application to produce part of the texture of Figure 2.4 is shown in Figure 2.11.

Example 2.2

Use the rules of Figure 2.10, with starting element the left-hand side of rule 1, to produce a texture pattern different from that of Figure 2.11.

Figure 2.12 An alternative pattern that may be produced by the successive application of the rules of Figure 2.10.

Figure 2.12 shows an example of an alternative pattern that may be constructed by starting with the element on the left-hand side of rule 1 and applying the rules of Figure 2.10 in the following sequence: 2, 7, 3, 7, 3, 7, 3, 7, 3, 7, 3, 7, 6, 10, 6, 10, 6, 10, 6, 10, 6, 10, 6, 9, etc. Note that it is not possible to produce a pattern that consists of disconnected components, as none of the rules we have allows the continuation of the pattern with a gap.

What happens if the placement of the primitive pattern is not regular?

The texture shown in Figure 2.2d is an example of a semi-stochastic pattern; the primitives are regular, but the rules according to which they are placed appear to be random. Such patterns may be described by **stochastic grammars**. In a stochastic grammar the rules are applied in a random order, or they depend on a random variable. They may even be given different weights, i.e. they may be drawn from a random distribution that favours some of the rules more than others. Examples of semi-stochastic textures are those of Figure 2.13.

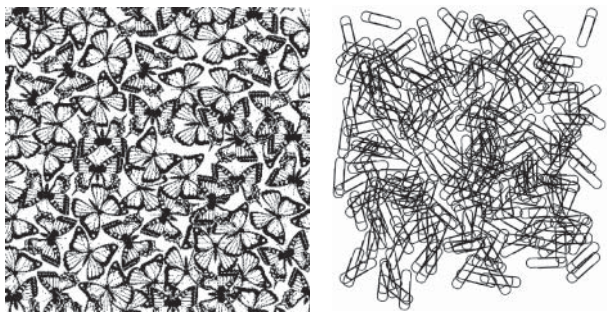


Figure 2.13 Two semi-stochastic textures.

Example 2.3

Use the rules of Figure 2.10, with starting element the left-hand side of rule 1, to produce a semi-stochastic pattern.

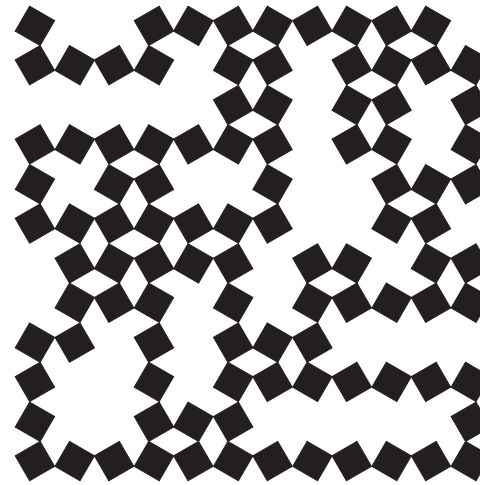


Figure 2.14 A pattern that was produced by the application of the rules of Figure 2.10 in a random order. Starting with the left-hand side of the first rule of Figure 2.10, we proceed by applying rules 2, 10, 3, 7,..., to produce this pattern, growing it from its top left corner.

Figure 2.14 shows a pattern that was produced by the rules of Figure 2.10 applied in a totally random order, but avoiding overlaps. In other words, every time a rule indicated continuation towards a direction where there was already a tile, this rule was abandoned and another one was chosen at random.

What happens if the primitive pattern itself is not always the same?

Then we must use rules that choose primitives from a set of possible primitive patterns every time they are applied.

What happens if the primitive patterns vary in a continuous way?

Then we may use a probability density function to describe the distribution of the possible values of the parameters that describe the shape of the primitive pattern. This leads to the **2D Boolean models** used for texture description.

2.2 Boolean Models

What is a 2D Boolean model?

The Boolean model consists of two independent probabilistic processes: a point process creating the **germs** and a shape process creating the **grains**. The outcome of the point process is a set of locations in the 2D space. The outcome of the shape process is a set of shapes that are placed at the random positions chosen by the point process.

Box 2.2 How can we draw random numbers according to a given probability density function?

Most computers have programs that can produce uniformly distributed random numbers. Let us say that we wish to draw random numbers x according to a given probability density function $p_x(x)$. Let us also say that we know how to draw random numbers y with a uniform probability density function defined in the range $[A, B]$. We may formulate the problem as follows.

Define a transformation $y = g(x)$ that is one-to-one and that is such that if y is drawn from a uniform probability density function in the range $[A, B]$, samples x are distributed according to the given probability density function $p_x(x)$.

Since we assume that relationship $y = g(x)$ is one-to-one, we may schematically depict it as shown in Figure 2.15.

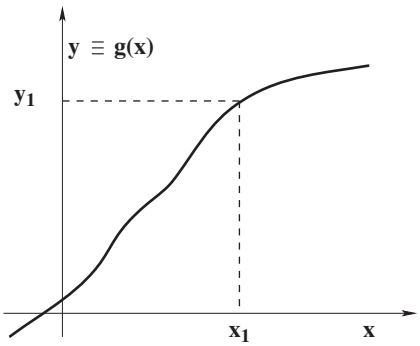


Figure 2.15 A one-to-one relationship between x and y.

It is obvious from Figure 2.15 that distributions $P_y(y_1)$ and $P_x(x_1)$ of the two variables are identical, since whenever y is less than $y_1 = g(x_1)$, x is less than x_1:

$$P_y(y_1) \equiv \mathcal{P}(y \leq y_1) = \mathcal{P}(x \leq x_1) \equiv P_x(x_1). \tag{2.1}$$

The distribution of x is known, since the probability density function of x is known:

$$P_x(x_1) \equiv \int_{-\infty}^{x_1} p_x(x)\mathrm{d}x. \tag{2.2}$$

The probability density function of y is given by:

$$p_y(y) = \begin{cases} \dfrac{1}{B-A} & \text{for } A < y \leq B \\ 0 & \text{otherwise} \end{cases}. \tag{2.3}$$

The distribution of y is then easily obtained:

$$P_y(y_1) \equiv \int_{-\infty}^{y_1} p_y(y)\mathrm{d}y = \begin{cases} 0 & \text{for } y_1 \leq A \\ \dfrac{y_1 - A}{B-A} & \text{for } A < y_1 \leq B \\ 1 & \text{for } B < y_1 \end{cases}. \tag{2.4}$$

Upon substitution in (2.1), we obtain

$$\frac{y_1 - A}{B-A} = P_x(x_1) \tag{2.5}$$

which leads to:

$$y_1 = (B - A)P_x(x_1) + A. \tag{2.6}$$

Random number generators usually produce uniformly distributed numbers in the range $[0, 1]$. For $A = 0$ and $B = 1$ we have:

$$y_1 = P_x(x_1). \tag{2.7}$$

So, to produce random numbers x, distributed according to a given probability density function $p_x(x)$, we use the following algorithm.

Step 1: Compute the distribution of x, $P_x(x_1)$ using Equation (2.2).
Step 2: Tabulate pairs of numbers $(x_1, P_x(x_1))$.
Step 3: Draw uniformly distributed numbers y_1 in the range $[0, 1]$.
Step 4: Use the tabulated numbers as a look-up table, where for each $y_1 = P_x(x_1)$ you look up the corresponding x_1. These x_1 numbers are the random samples distributed according to the desired probability density function.

Example B2.4

Produce 50 random numbers drawn from a Gaussian probability density function with mean $\mu = 2$ and standard deviation $\sigma = 3$. Assume that you have at your disposal a random number generator that produces samples from a uniform probability density function in the range $[0, 1]$.

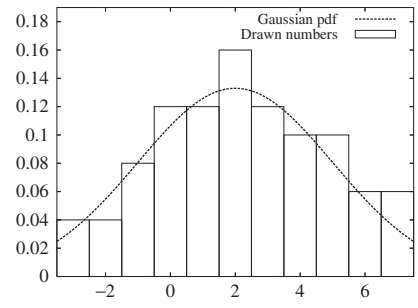


Figure 2.16 A Gaussian probability density function with $\mu = 2$ and $\sigma = 3$ and the histogram of 50 random numbers drawn according to it.

In this case:

$$p_x(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{2.8}$$

Note that $\int_{-\infty}^{+\infty} p_x(x)\mathrm{d}x = 1$ *and that* $\int_{-\infty}^{\mu} p_x(x)\mathrm{d}x = \int_{\mu}^{+\infty} p_x(x)\mathrm{d}x = 1/2$.

The distribution of such a probability density function is computed as follows:

$$P_x(x_1) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{x_1} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \mathrm{d}x. \tag{2.9}$$

(Continued)

Example B2.4 (Continued)

We distinguish two cases: $x_1 \leq \mu$ and $x_1 > \mu$. We shall deal with the second case first:

$$P_x(x_1) = \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\mu} e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx + \frac{1}{\sqrt{2\pi}\sigma}\int_{\mu}^{x_1} e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx$$
$$= \frac{1}{2} + \frac{1}{\sqrt{2\pi}\sigma}\int_{\mu}^{x_1} e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx. \tag{2.10}$$

If we set

$$z \equiv \frac{x-\mu}{\sqrt{2}\sigma} \Rightarrow dx = \sigma\sqrt{2}dz \tag{2.11}$$

we obtain

$$P_x(x_1) = \frac{1}{2} + \frac{1}{\sqrt{\pi}}\int_0^{z_1} e^{-z^2}dz = \frac{1}{2} + \frac{1}{2}erf(z_1) \tag{2.12}$$

where $z_1 \equiv (x_1 - \mu)/(\sqrt{2}\sigma)$ and the error function is defined as:

$$erf(z) \equiv \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt \tag{2.13}$$

When $x_1 \leq \mu$, we write:

$$P_x(x_1) = \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\mu} e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx - \frac{1}{\sqrt{2\pi}\sigma}\int_{x_1}^{\mu} e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx$$
$$= \frac{1}{2} - \frac{1}{\sqrt{2\pi}\sigma}\int_{x_1}^{\mu} e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx. \tag{2.14}$$

In the last expression we change variable of integration from x to $y \equiv -x \Rightarrow dx = -dy$. Then:

$$P_x(x_1) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}\sigma}\int_{-x_1}^{-\mu} e^{-\frac{(-y-\mu)^2}{2\sigma^2}}dy$$
$$= \frac{1}{2} - \frac{1}{\sqrt{2\pi}\sigma}\int_{-\mu}^{-x_1} e^{-\frac{(y+\mu)^2}{2\sigma^2}}dy. \tag{2.15}$$

If we set

$$z \equiv \frac{y+\mu}{\sqrt{2}\sigma} \Rightarrow dy = \sigma\sqrt{2}dz \tag{2.16}$$

we obtain

$$P_x(x_1) = \frac{1}{2} - \frac{1}{\sqrt{2\pi}\sigma}\sigma\sqrt{2}\int_0^{z_1} e^{-z^2}dz = \frac{1}{2} - \frac{1}{2}erf(z_1) \tag{2.17}$$

where here $z_1 \equiv (-x_1 + \mu)/(\sqrt{2}\sigma)$. In summary, we deduced that:

$$P_x(x_1) = \begin{cases} \frac{1}{2} - \frac{1}{2}erf\left(\frac{-x_1+\mu}{\sqrt{2}\sigma}\right) & \text{for } x_1 \leq \mu \\ \frac{1}{2} + \frac{1}{2}erf\left(\frac{x_1-\mu}{\sqrt{2}\sigma}\right) & \text{for } x_1 > \mu \end{cases}. \tag{2.18}$$

The error function may be easily computed by using various rational approximations, one of which is

$$erf(x) = 1 - (a_1 t + a_2 t^2 + a_3 t^3)e^{-x^2} \tag{2.19}$$

where $t \equiv 1/(1+px)$, $p = 0.47047$, $a_1 = 0.3480242$, $a_2 = -0.0958798$ *and* $a_3 = 0.7478556$. *This approximation is valid for* $0 \leq x < \infty$ *and it returns the value of the error function with accuracy better than* 2.5×10^{-5}.

Using this approximation, we create the table of values given in Table 2.1.

We then draw 50 numbers uniformly distributed in the range $[0, 1]$ *and look them up under column* $P_x(x_1)$ *to read the numbers we are really interested in, namely the corresponding* x_1 *numbers. Wherever there is no exact entry in our look-up table, we perform linear interpolation. Some of the drawn numbers and the corresponding* x_1 *numbers worked out from the look-up table are given in Table 2.2. We usually construct the look-up table so that it is very unlikely that we shall draw a number smaller than the smallest one in the table or larger than the largest one in the table. The example we give here uses a rather sparse look-up table. Figure 2.16 shows the theoretical Gaussian probability density function superimposed on the histogram of the 50 numbers drawn.*

Box 2.3 What is a Poisson process?

Imagine that we draw N random numbers uniformly distributed in the range $[0, T]$. Every time we do it, we find yet another combination of N such numbers. So, if we concentrate our attention on a sub-range of this range, say $[t_1, t_2]$, every time we repeat the experiment, we have yet another number of values in this range. Let us say that in the ith realisation we have k_i values in the range $[t_1, t_2]$. The probability of k_i taking a particular value k is given by the Poisson probability density function

$$p(k_i = k) = \mathrm{e}^{-\lambda(t_2 - t_1)} \frac{(\lambda(t_2 - t_1))^k}{k!} \tag{2.20}$$

where $\lambda = N/T$.

If the range is chosen to start from 0, i.e. if $t_1 = 0$, and say $t_2 = t$, then these numbers k_i constitute a **Poisson process** which is a function of t: $k(t)$.

Example B2.5

Create a pattern of 100×100 **pixels using a Poisson process with parameter** $\lambda = 0.005$**. Use a shape process that creates squares with side** l**. The values of** l **are Gaussianly distributed with mean** $l_0 = 5$ **and standard deviation** $\sigma_l = 2$**. Repeat the experiment for** $\lambda = 0.01$**, 0.015, 0.025, 0.05, and 0.1.**

Using the definition of λ *immediately after Equation (2.20), and for a total number of pixels* $T = 100 \times 100 = 10,000$*, we work out that we must draw* $N = 50$ *pixels when* $\lambda = 0.005$. *Therefore, we must identify 50 locations uniformly distributed in the* 100×100 *grid where we shall place black squares of randomly chosen sizes. We draw 100 random numbers uniformly distributed in the range* $[1, 100]$. *Then we combine them two by two to form the coordinates of the 50 pixels we need. We use the method of Example 2.4 to draw random numbers Gaussianly distributed with mean 5 and standard deviation 2 in order to decide the size of each square. Each drawn number is rounded to the nearest integer. The result is shown in Figure 2.17a. The results for the other values of* λ *are shown in Figures 2.17b–2.17f.*

Table 2.1 Pairs $(x_1, P_x(x_1))$ when $P_x(x_1)$ is the distribution function of a Gaussian probability density function with mean $\mu = 2$ and standard deviation $\sigma = 3$.

x_1	$P_x(x_1)$	x_1	$P_x(x_1)$	x_1	$P_x(x_1)$
−2.5	0.06680	0.6	0.32038	3.6	0.70309
−2.4	0.07123	0.7	0.33240	3.7	0.71452
−2.3	0.07587	0.8	0.34459	3.8	0.72574
−2.2	0.08075	0.9	0.35694	3.9	0.73673
−2.1	0.08586	1.0	0.36945	4.0	0.74750
−2.0	0.09120	1.1	0.38209	4.1	0.75803
−1.9	0.09679	1.2	0.39486	4.2	0.76832
−1.8	0.10263	1.3	0.40775	4.3	0.77836
−1.7	0.10871	1.4	0.42073	4.4	0.78814
−1.6	0.11506	1.5	0.43381	4.5	0.79767
−1.5	0.12166	1.6	0.44695	4.6	0.80694
−1.4	0.12853	1.7	0.46016	4.7	0.81594
−1.3	0.13566	1.8	0.47341	4.8	0.82468
−1.2	0.14305	1.9	0.48670	4.9	0.83315
−1.1	0.15072	2.0	0.50000	5.0	0.84135
−1.0	0.15865	2.1	0.51330	5.1	0.84928
−0.9	0.16685	2.2	0.52659	5.2	0.85695
−0.8	0.17532	2.3	0.53984	5.3	0.86434
−0.7	0.18406	2.4	0.55305	5.4	0.87147
−0.6	0.19306	2.5	0.56619	5.5	0.87834
−0.5	0.20233	2.6	0.57927	5.6	0.88494
−0.4	0.21186	2.7	0.59225	5.7	0.89129
−0.3	0.22164	2.8	0.60514	5.8	0.89737
−0.2	0.23168	2.9	0.61791	5.9	0.90321
−0.1	0.24197	3.0	0.63055	6.0	0.90880
0.0	0.25250	3.1	0.64306	6.1	0.91414
0.1	0.26327	3.2	0.65541	6.2	0.91925
0.2	0.27426	3.3	0.66760	6.3	0.92413
0.3	0.28548	3.4	0.67962	6.4	0.92877
0.4	0.29691	3.5	0.69145	6.5	0.93320
0.5	0.30855				

Example B2.6

Create a texture pattern of 100×100 pixels in size, using the following 2D Boolean model. A point process defined as a Poisson process with parameter $\lambda = 0.005$, and a shape process that creates circles with radius r given by a Gaussian probability density function with mean $r_0 = 5$ and standard deviation $\sigma_r = 2$. Approximate each circle by its nearest digital circle. Repeat the experiment for $\lambda = 0.01$, 0.015, 0.025, 0.05, and 0.1.

We work again as for example 2.5. The results for the various values of λ are shown in Figure 2.18.

Table 2.2 The numbers on the left were drawn from a uniform probability density function with range [0, 1]. Each one of them was treated as a value of $P_x(x_1)$ and Table 2.1 was used as a look-up table to find the corresponding value of x_1. The thus deduced x_1 values are given in the corresponding position on the right of this table. They are samples from a Gaussian probability density function with mean $\mu = 2$ and standard deviation $\sigma = 3$.

Uniformly distributed		**Gaussianly distributed**	
numbers in the range [0, 1]		**numbers with $\mu = 2$ and $\sigma = 3$**	
0.13412	0.88634	−1.32152	5.62203
0.82812	0.33051	4.84066	0.68432
0.00055	0.50676	−5.97104	2.05078
0.35441	0.76733	0.87948	4.19036
0.15503	0.22638	−1.04556	−0.25284
0.79651	0.68888	4.48786	3.47826
0.25230	0.21248	−0.00188	−0.39367
0.89950	0.16105	5.83650	−0.97067
0.52185	0.93618	2.16434	6.77440
0.68380	0.01647	3.43533	−5.13699
0.73431	0.53067	3.87797	2.23079
0.23325	0.40219	−0.18480	1.25688
0.24480	0.90535	−0.07308	5.93836
0.09923	0.33897	−1.85813	0.75394
0.35064	0.80364	0.84899	4.56437
0.63024	0.48476	2.99748	1.88541

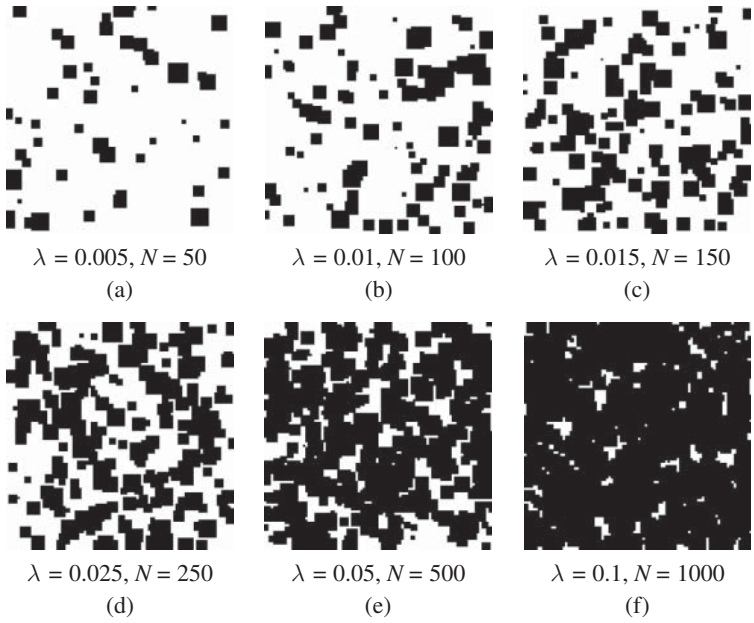


Figure 2.17 2D Boolean patterns created for different values of parameter λ of the Poisson process. The shape process creates squares with side l Gaussianly distributed with mean $l_0 = 5$ and standard deviation $\sigma_l = 2$. Parameter λ determines how many such shapes are created and placed uniformly inside the grid.

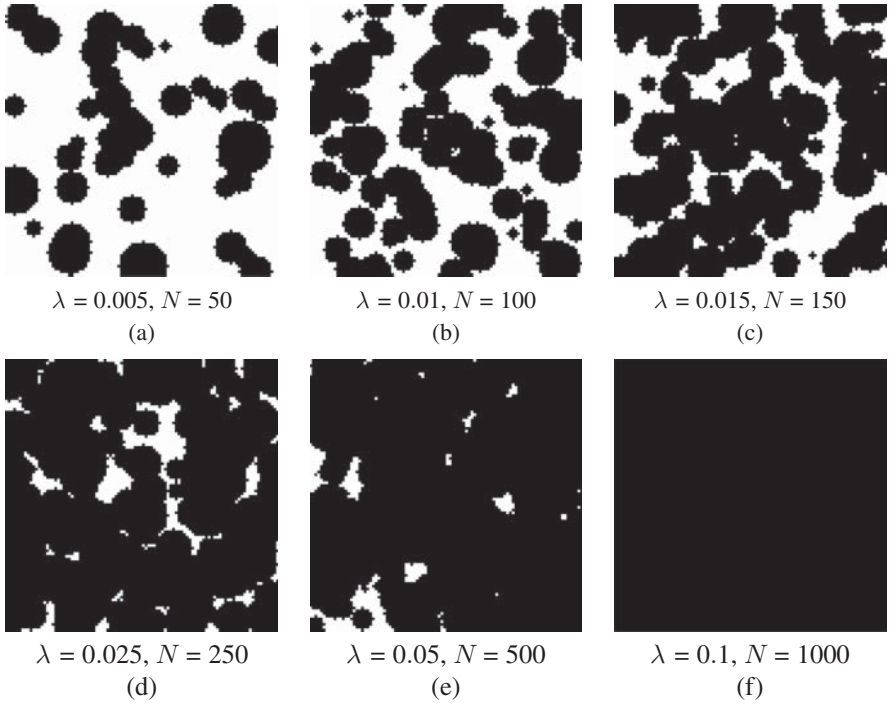


Figure 2.18 2D Boolean patterns created for different values of parameter λ of the underlying Poisson process. The shape process created circles with radius r. The values of r were drawn according to a Gaussian probability density function with mean $r_0 = 5$ and standard deviation $\sigma_l = 2$, using the method described in Example 2.4. For $\lambda = 0.1$ we had far too many circles and obtained complete coverage of the grid. Parameter λ of the Poisson process determines how many such shapes are created and placed uniformly inside the grid.

How can we use the 2D Boolean model to describe a binary texture?

One can distinguish two types of parameter in a binary texture: **aggregate** and **individual** parameters. The aggregate parameters are those that we can measure directly from the image. An example of an aggregate parameter is the average number of black pixels inside a window of fixed size. The individual parameters are those that govern the Boolean model that we assume describes the texture. An example of an individual parameter is the value of λ of the underlying Poisson germ process. The individual parameters are estimated from the values of the aggregate parameters. Both types of parameter may be used to characterise the texture; however, if one needs to reproduce the texture, one must estimate as completely as possible the individual model parameters. In that case, one must also check the validity of the adopted Boolean model, since it is possible that it may not be applicable.

How can we estimate some aggregate parameters of the 2D Boolean model?

Some useful aggregate parameters of a Boolean texture are the **area fraction**, f, the **specific boundary length**, L, and the **specific convexity**, N^+. There are many more aggregate parameters that may be defined, but we chose these particular ones because they are useful in estimating some of the basic individual parameters of the underlying Boolean model in the case when the grains are assumed to be convex shapes and the germ process is assumed to be Poisson.

What is the area fraction?

The area fraction f is defined as the ratio of the black (covered) pixels over all pixels in the image. For example, in the image of Figure 2.19a this parameter is $f = 0.38$ because there are 38 black pixels and the image consists of 100 pixels.

What is the specific boundary length?

The specific boundary length L is defined as the average boundary length per unit area in the image. It may be estimated by counting the number of border pixels in the image. These are all pixels that are black but have at least one white neighbour. When using the word "neighbour" one has to specify whether a neighbour is a pixel that has with the pixel in question a common side or just a common vertex. This determines the type of **image connectivity** we decide to adopt.

What types of image connectivity may we have?

If we accept as neighbours only pixels that share a common side with the pixel being examined, then we are using **4-connectivity**. If in addition we count as neighbours pixels that share a

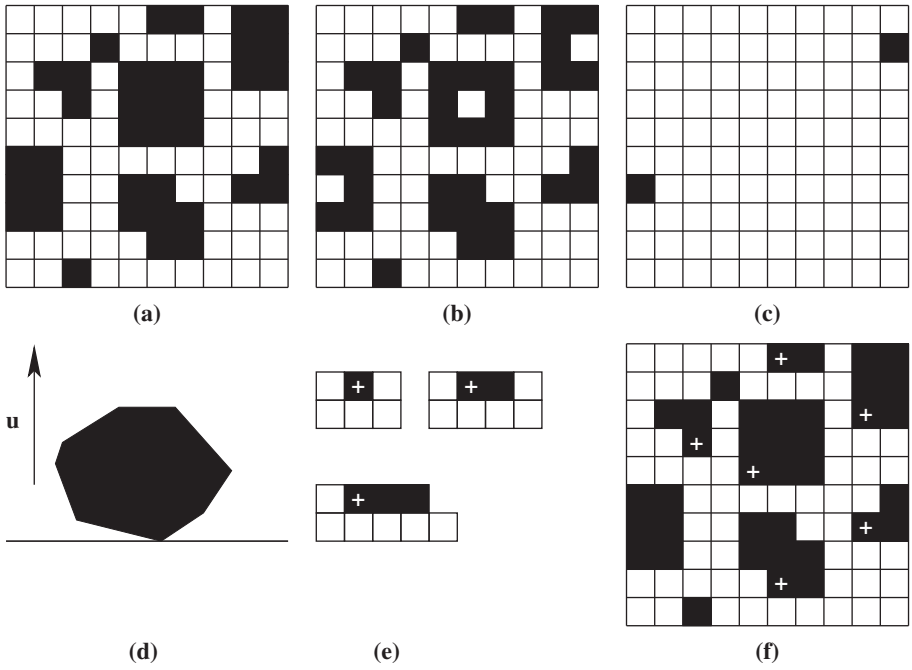


Figure 2.19 (a) A 10×10 binary image. (b)The boundary pixels of the image. These are the interior boundary pixels identified as having at least one white neighbour using 8-connectivity to define which are the neighbours of a pixel. (c) The border pixels that may be counted as boundary pixels as well. (d) A convex grain and the point from which its tangent is orthogonal to vector u. (e) Patterns used to identify boundary pixels from which a tangent can be drawn to the black grain that is perpendicular to the upward vertical direction. The tangent points are marked with a cross. The third pattern is intentionally left incomplete with an apparently missing pixel at its top right corner, in order to indicate that it is indifferent whether that pixel is black or white. (f) The tangent points of tangents perpendicular to the vertical direction identified in the image.

common vertex with the pixel in question, then we are using **8-connectivity**. Very different results may be obtained according to which connectivity we adopt.

How does image connectivity influence the estimation of the specific boundary length?

As the whole theory of 2D Boolean models is based on the assumption of convex grains, it is advisable to use 8-connectivity to decide whether a pixel is a boundary pixel or not. To appreciate the problem of choosing between 4-connectivity and 8-connectivity in deciding the pixels that constitute the boundary pixels of the grains, we present in Table 2.3 the "perimeter" of a digital circle of a certain radius when 4- or 8-connectivity is used and the corresponding value of the perimeter of a circle in the continuous domain. In the same table we also include the areas of the digital and continuous circles. Figure 2.20 shows some digital circles and their boundary pixels defined using 4- and 8-connectivity with the background pixels.

Table 2.3 Perimeter and area of digital and continuous circles. The perimeter of the digital circle was measured using 4-connectivity (second column) and 8-connectivity (third column) in order to decide whether a pixel of the digital circle had a background pixel as a neighbour or not. The digital circle in each case was defined as the set of pixel positions (i, j) that satisfied the inequality $i^2 + j^2 \leq r^2$, with the centre of the circle assumed to be at $(0, 0)$.

	Digital circle			Continuous circle	
Radius r	**Perimeter** Π **4-connectivity**	**Perimeter** Π **8-connectivity**	**Area** A	**Perimeter** $2\pi r$	**Area** πr^2
1	4	5	5	6.28	3.14
2	8	12	13	12.57	12.57
3	16	20	29	18.85	28.27
4	20	28	49	25.13	50.27
5	28	36	81	31.42	78.54
6	32	44	113	37.70	113.10
7	36	52	149	43.98	153.94
8	44	60	197	50.27	201.06
9	48	68	253	56.55	254.47
10	56	76	317	62.83	314.16
11	60	84	377	69.12	380.13
12	64	92	441	75.40	452.39
13	72	100	529	81.68	530.93
14	76	108	613	87.96	615.75
15	84	116	709	94.25	706.86
16	88	124	797	100.53	804.25
17	96	132	901	106.81	907.92
18	100	140	1009	113.10	1017.88

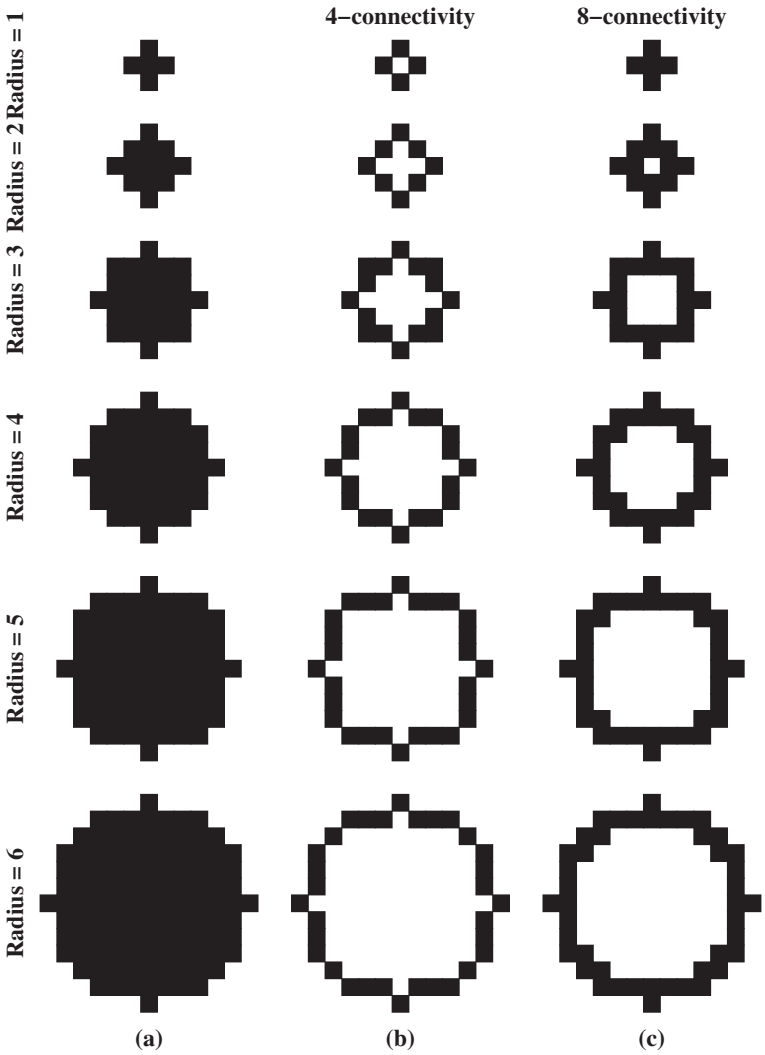


Figure 2.20 (a) Some digital circles. (b) Boundary pixels that share a common side with the background pixels. (c) Boundary pixels that share either a common side or a common vertex with the background pixels.

How do we deal with border effects?

The issue is whether to count as border pixels those that touch the edges of the image or not. Let us assume that we use 8-connectivity in order to decide whether a pixel is a boundary pixel or not. The boundary pixels of image 2.19a, which are interior to the image and are boundary pixels because they touch a white pixel, are shown in Figure 2.19b. We can count 35 such pixels, and therefore, from this count we estimate that the specific boundary length L is 0.35. If in addition we consider as boundary pixels those that touch the border of the image itself, then we must also include the two pixels marked in Figure 2.19c, and in this case $L = 0.37$. A compromise solution is to count only as boundary pixels the pixels that touch either the left or the bottom border of the image. It can be shown that the estimator of the specific boundary length in this case is unbiased. Intuitively we understand this by saying that we make a compromise and it is as if the image continues in a

wrapped round way in the two directions, so counting only the border pixels once makes sense. If we use this unbiased estimator in our example, we must only consider the border black pixel on the left in Figure 2.19c, and this will yield $L = 0.36$.

Is there a way to estimate the specific boundary length L, bypassing the problem of 4- or 8-connectivity?

Yes, with the following algorithm.

Step 1: Choose n digital lines passing through a white point somewhere in the middle of the image.
Step 2: Along each line count how many times it crosses a boundary, i.e. how many transitions there are from white to black and vice versa.
Step 3: Divide this number by the total number of pixels per digital line. The quantity you compute this way is the **specific boundary length per unit length**, $P(\beta_i)$, where β_i is the angle line i forms with the chosen reference direction, say the horizontal axis.
Step 4: The specific boundary length L **per unit area** is given by (see example 2.7):

$$L = \frac{\pi}{2n} \sum_{i=1}^{n} P(\beta_i). \tag{2.21}$$

What is the specific convexity N^+?

The specific convexity N^+ is defined as the average number of boundary points per unit area from which one can draw a tangent to the black grain orthogonal to a fixed direction. For example, if the fixed direction is indicated by vector u in Figure 2.19d, the tangent to the grain is drawn as shown in the same figure. These points are the points of **lower positive tangents**. Such points are identified by the existence of local patterns like those shown in Figure 2.19e and they are marked with a + in Figure 2.19f. It is very easy to identify them in this example. We find that we have six such points, and so $N^+ = 6/100 = 0.06$. However, in practice they have to be identified using the **hit-or-miss** algorithm described later, in Example 2.22 and in Box 2.8, using the patterns shown in Figure 2.19e. In addition, we use more than one orientation for vector u, typically four, with u pointing up, down, left and right, and estimate N^+ as the average of its values over these orientations.

The patterns shown in Figure 2.19e are adequate if we think that the maximum radius of a grain is about 3. If we think that larger grains exist in the image, then extra patterns must also be used with the hit-or-miss transform. These extra patterns should be similar to the first two patterns of Figure 2.19e, but with longer sequences of black pixels, i.e. 3, 4, 5, etc., up to the maximum expected radius of the grains. The last pattern should have its top right corner missing, indicating that it is indifferent whether that pixel is black or white (like the third of the patterns shown in Figure 2.19e).

Example B2.7

Show that if $P(\beta)$ is the specific boundary length per unit length in polar direction β, the specific boundary length per unit area is given by:

$$L = \frac{1}{4}\int_0^{2\pi} P(\beta)\mathrm{d}\beta. \tag{2.22}$$

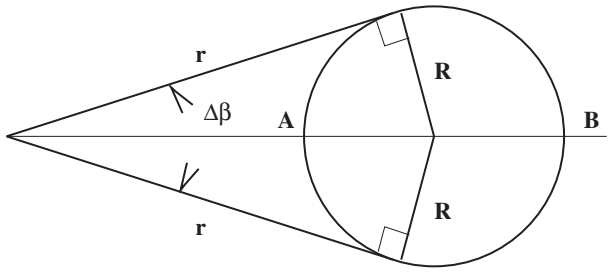


Figure 2.21 A grain at distance r from the origin of the axes will have radius $R = r\tan\frac{\Delta\beta}{2}$.

Since $P(\beta)$ is the probability of boundary crossings per unit length, we expect to find $P(\beta)\mathrm{d}r$ boundary crossings in a length $\mathrm{d}r$. Let us say that $\mathrm{d}r$ is so small that we shall have at most a single boundary crossing inside it. We shall show next that the length of the boundary, if it exists inside $\mathrm{d}r$, is $\pi r\mathrm{d}\beta/2$, where $\mathrm{d}\beta$ is an elementary angle around orientation β. Let us consider a circular grain as shown in Figure 2.21. (Since the grains are assumed convex, to the first approximation they may be represented by circles.) Then the radius R of the grain, in terms of angle $\Delta\beta$ and its distance r from the origin of the polar coordinates we use, is:

$$R = r\tan\frac{\Delta\beta}{2} \simeq r\frac{\Delta\beta}{2}. \tag{2.23}$$

The boundary length of the grain then is $2\pi R = 2\pi r\Delta\beta/2 = \pi r\Delta\beta$. However, this boundary will be counted twice, since it will be encountered once when the ray from the origin crosses the grain at point A and once when it crosses the grain at point B (see Figure 2.21). Therefore, the effective boundary length we may assign to each boundary crossing is $\pi r\Delta\beta/2$. The boundary length then we expect to find inside the area of a circle of radius ρ is:

$$\int_0^{\rho}\int_0^{2\pi} P(\beta)\mathrm{d}r\pi r\frac{\mathrm{d}\beta}{2} = \frac{\pi}{2}\int_0^{\rho} r\mathrm{d}r\int_0^{2\pi} P(\beta)\mathrm{d}\beta = \frac{\pi}{2}\frac{1}{2}\rho^2 P(\beta)\mathrm{d}\beta. \tag{2.24}$$

The specific boundary length per unit area is the above result divided by the area of the circle we considered, i.e. by $\pi\rho^2$, which yields Equation (2.22).

Example B2.8

Consider a circular image of radius $\rho = 1$. At two points centred $\rho/2$ away from the centre of the image, there are two grains of radius $\rho/4$ each. For this image compute the specific boundary length per unit area directly and by applying formula (2.22). Compare the two answers you get.

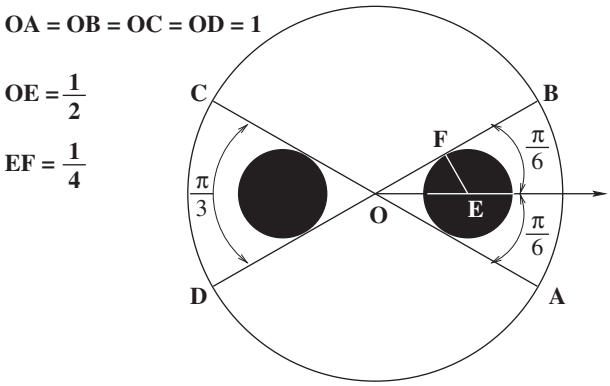


Figure 2.22 A circular image of radius 1 with two grains of radii 1/4.

Since $\rho = 1$, the radius of each grain is $1/4$ and the perimeter of each grain is $2\pi/4 = \pi/2$. So, the total perimeter of both grains is twice that, i.e. π. We must divide this by the area of the circular image, i.e. by $\pi\rho^2 = \pi$ to compute the specific boundary length per unit area, which turns out to be 1.

Let us now apply formula (2.22). For simplicity, let us assume that the two grains are placed in diametrically opposite directions, say at $\beta = 0$ and $\beta = \pi$. This assumption does not affect the result of our calculation. We can see from Figure 2.22 that for triangle OEF we have:

$$\sin(\widehat{EOF}) = \frac{EF}{OE} = \frac{1/4}{1/2} = \frac{1}{2} \Rightarrow \widehat{EOF} = \frac{\pi}{6}. \tag{2.25}$$

As β is measured anti-clockwise from reference direction OE, function $P(\beta)$, which counts the number of boundary crossings per unit length as a function of β, is:

$$P(\beta) = \begin{cases} 2 \text{ for } 0 \le \beta < \frac{\pi}{6} \\ 0 \text{ for } \frac{\pi}{6} \le \beta < \frac{5\pi}{6} \\ 2 \text{ for } \frac{5\pi}{6} \le \beta < \frac{7\pi}{6} \\ 0 \text{ for } \frac{7\pi}{6} \le \beta < \frac{11\pi}{6} \\ 2 \text{ for } \frac{11\pi}{6} \le \beta < 2\pi \end{cases} . \tag{2.26}$$

Then:

$$L = \frac{1}{4}\int_0^{2\pi} P(\beta)\mathrm{d}\beta$$

$$
\begin{aligned}
&= \frac{1}{4}\left(\int_0^{\frac{\pi}{6}} 2\mathrm{d}\beta + \int_{\frac{5\pi}{6}}^{\frac{7\pi}{6}} 2\mathrm{d}\beta + \int_{\frac{11\pi}{6}}^{2\pi} 2\mathrm{d}\beta\right) \\
&= \frac{1}{4}\left(2\frac{\pi}{6} + 2\frac{2\pi}{6} + 2\frac{\pi}{6}\right) \\
&= \frac{\pi}{3} \sim 1. \qquad (2.27)
\end{aligned}
$$

We observe that the formula gives very approximately the right answer.

How can we estimate some individual parameters of the 2D Boolean model?

Individual parameters of the 2D Boolean model are parameter λ of the Poisson process and the parameters that describe the shape and size of the grains. Here we shall restrict ourselves only to the average area A and the average perimeter Π of the grains. In order to estimate them we make use of three equations that relate them to the aggregate parameters we defined earlier:

$$
\begin{aligned}
f &= 1 - \mathrm{e}^{-\lambda A} \\
L &= \lambda(1-f)\Pi \\
N^+ &= \lambda(1-f). \qquad (2.28)
\end{aligned}
$$

Justification of these equations is given in Box 2.4. From the values $f = 0.38$ and $N^+ = 0.06$ we derived for the image of Figure 2.19a, and the third of the above equations, we deduce that $\lambda = 0.06/0.62 = 0.097$. Using this value in the second of these equations alongside the value $L = 0.36$, we deduce that the mean perimeter of the grains is $\Pi = 5.99$ pixels and using it in the first equation we deduce that the mean grain area is $A = 4.93$.

There are also more sophisticated aggregate parameters that may be computed from the image and that may be used to infer more information about the distribution of the grain shape and size so that the image may be characterised better. Of course, if one had known a priori what the primitive pattern of the texture was, then the process would have been more straightforward. Often one makes the assumption that the grains may be described by some very simple parametric shape, e.g. a circle. One can then compute characteristics of the distributions of the parameters of the assumed shape.

Example B2.9

Compute the mean value of a variable that has a probability density function given by the Poisson process:

$$p(k) = \mathrm{e}^{-\lambda}\frac{\lambda^k}{k!}. \qquad (2.29)$$

The mean value of k will be given by:

$$\overline{k} = \sum_{k=0}^{\infty} k\mathrm{e}^{-\lambda}\frac{\lambda^k}{k!} = \mathrm{e}^{-\lambda}\sum_{k=0}^{\infty} k\frac{\lambda^k}{k!}. \qquad (2.30)$$

(Continued)

Example B2.9 (Continued)

Consider the series expansion of e^{λ}*:*

$$e^{\lambda} = \sum_{k=0}^{\infty} \frac{\lambda^k}{k!}. \tag{2.31}$$

Differentiate it with respect to λ *to obtain:*

$$e^{\lambda} = \sum_{k=0}^{\infty} k \frac{\lambda^{k-1}}{k!} = \frac{1}{\lambda} \sum_{k=0}^{\infty} k \frac{\lambda^k}{k!}. \tag{2.32}$$

If we solve then (2.32) for λ*, we have*

$$\lambda = e^{-\lambda} \sum_{k=0}^{\infty} k \frac{\lambda^k}{k!} \tag{2.33}$$

which upon comparison with (2.30) yields $\lambda = \overline{k}$*.*

So, the mean value $\overline{k}$ *of the expected number of events per unit area (or length) of the Poisson process is equal to parameter* λ *of the process.*

Box 2.4 How can we relate the individual parameters to the aggregate parameters of the 2D Boolean model?

We assume that the probability of finding k events in an area of size A is given by a Poisson probability density function of the form:

$$p(k) = e^{-\lambda A} \frac{(\lambda A)^k}{k!}. \tag{2.34}$$

In our case, "event" means having the centre of a grain placed inside the area. The size of the area is measured in pixels. Since we measure all the distances in pixels, it is as if we consider a pixel to be a tile of side 1.

We wish first of all to compute the number of pixels that are expected to be covered in the image. A pixel is covered if it is itself the centre of a grain, or if a grain has been placed near by, no further away than the average radius of the grains. In other words, a pixel is covered if a grain has been placed with its centre anywhere inside a circular neighbourhood, with size equal to the average size of a grain, around it. Let us say that the average size of a grain is A pixels. So, for a pixel to be uncovered, there should be no grain placed in a circular neighbourhood around it of size A pixels. In other words, the number of events happening inside that neighbourhood

should be zero. The probability of getting 0 events in a neighbourhood of size A is:

$$p(0) = \mathrm{e}^{-\lambda A}\frac{(\lambda A)^0}{0!} = \mathrm{e}^{-\lambda A}. \tag{2.35}$$

(Remember that $0! = 1$.)

The probability of a pixel being covered is 1 minus the probability of being uncovered, i.e. $1 - p(0)$. So, for an image of size $N \times N$ we expect to find $N^2(1 - p(0))$ covered pixels, which upon division with the total number N^2 of pixels in the image yields the fraction f of covered pixels:

$$f = 1 - p(0) = 1 - \mathrm{e}^{-\lambda A}. \tag{2.36}$$

This is the first equation that relates the measurable from the image aggregate parameter f to the individual parameters A and λ.

Let us say next that the average grain perimeter is Π. The mean number of grains per unit area is λ (see Example 2.9). Therefore, we expect to have $\lambda\Pi$ boundary pixels per unit area. However, not all of them will be visible, as in order to be visible they have to be uncovered. The probability of a pixel being uncovered was worked out above to be $\mathrm{e}^{-\lambda A}$. So, the probability of $\lambda\Pi$ pixels being uncovered (and therefore visible as boundary pixels) is $\lambda\Pi\mathrm{e}^{-\lambda A}$. If we substitute $\mathrm{e}^{-\lambda A}$ from (2.36) we obtain that the specific boundary length L, i.e. the average number of boundary pixels per unit area is:

$$L = \lambda(1 - f)\Pi. \tag{2.37}$$

This is the second equation that relates the measurable from the image aggregate parameters L and f with the individual parameters Π and λ.

Finally, since we assume that the grains are convex, we must have as many points from which a tangent to the grain perpendicular to a certain direction can be drawn as we have grains (see Figure 2.19d). So, there must be λ such points per unit area. For these points to be visible, they must be uncovered. The probability of a point being uncovered is $\mathrm{e}^{-\lambda A}$ as above, so the specific convexity N^+, which is the average number of such points per unit area, is given by:

$$N^+ = \lambda\mathrm{e}^{-\lambda A} = \lambda(1 - f). \tag{2.38}$$

This is the final equation we present here, which relates the aggregate parameters N^+ and f, that can be directly estimated from the image to the individual parameter λ.

Example B2.10

Create a texture pattern of 100×100 pixels in size using a Boolean model, the point process of which is a Poisson process with parameter $\lambda = 0.005$ and the shape process of that creates digital circles with radius $r = 4$. Estimate the aggregate and individual parameters of the texture pattern.

(Continued)

Example B2.10 (Continued)

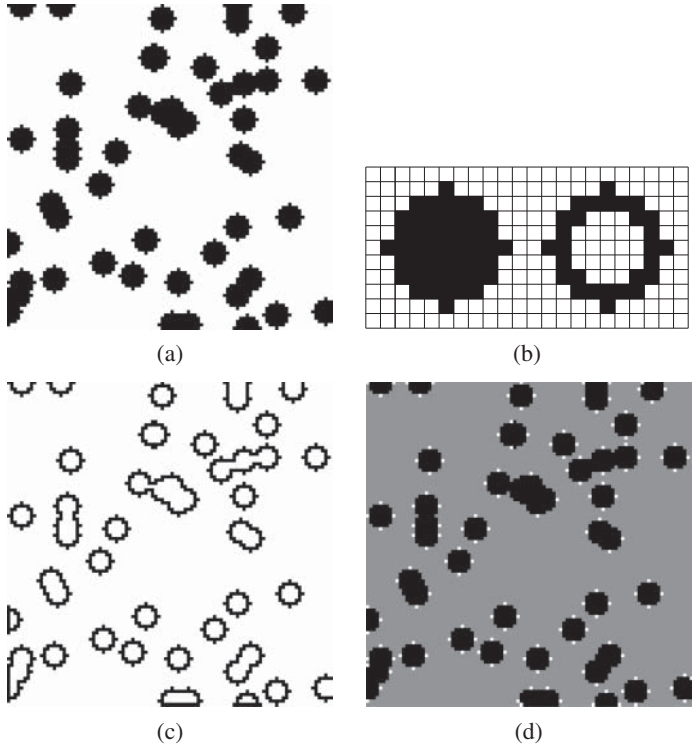


Figure 2.23 Estimating the aggregate parameters of a 2D Boolean model. (a) A realisation of the 2D Boolean model. (b) A digital circle of radius 4 and its boundary pixels using 8-connectivity. (c) The boundary pixels identified in the image. (d) Tangent points in the four chosen directions.

Figure 2.23a shows a random texture pattern created using the given parameters. Figure 2.23b shows a digital circle of radius 4 and its boundary pixels identified using 8-connectivity. Figure 2.23c shows the boundary pixels of the whole image. Note that we included as boundary pixels the pixels where the grains touch the image border at the left and bottom, but not at the right and top. We use the boundary pixels to estimate the value of aggregate parameter L. Figure 2.23d shows the tangent points obtained using four possible directions that allow the estimation of aggregate parameter N^+. One has to count these points and divide by 4 to obtain an estimate of N^+.

From the values of these aggregate parameters and with the help of Equations (2.28) we can estimate the individual parameters of this pattern, namely the density of grains λ, their average area A and their average perimeter Π. Table 2.4 shows the numerical values estimated for the aggregate and individual parameters for this pattern.

We observe that the value of λ is pretty close to the one we used. In addition, a digital circle of radius 4 consists of 49 pixels, and its boundary pixels are 28. Both these values are very close to the estimated ones. However, this should not be surprising. This example is the ideal case for applying this methodology. We know that the 2D Boolean model is applicable since the pattern was created this way, and all grains are of the same shape and size.

Table 2.4 Numerical values measured from the image for the aggregate parameters and values of the individual parameters estimated from the values of the aggregate parameters for the texture of Figure 2.23a.

Aggregate parameter	Measured value from the image	
f	0.2056	
L	0.1088	
N^+	0.0038	
Individual parameter	Estimated value from Equation (2.28)	Real value
λ	0.004783	0.005
Π	28.631579	28.000
A	48.117261	49.000

Example B2.11

Estimate the aggregate and individual parameters of the artificial textures created in Examples 2.5 and 2.6.

Table 2.5 shows the aggregate and individual parameters estimated for each image. The values of Π *and* A *are the average perimeter and area, respectively, of the grains.*

The perimeter of a 5×5 *digital square is* $\Pi = 16$ *and its area is* $A = 25$*. So, for the image with the squares that have sides Gaussianly distributed with mean 5, these are the expected estimates for* Π *and* A*.*

Digital circles are more different from their continuous counterparts than digital squares are from theirs. Table 2.3 gives the values of the perimeter and the area of digital and continuous circles of integer-valued radii in the range $[1, 18]$*. We can see that as the circles become larger, the differences between the digital and continuous case gradually disappear. The perimeter values for the images of Example 2.6 were estimated using 8-connectivity. From Table 2.3 we can see that a digital circle with radius 5 has perimeter* $\Pi = 36$ *for 8-connectivity, and area* $A = 81$*. So, these are the expected values for the image of Figure 2.18. In view of this, we infer that the estimations shown in Table 2.5 are reasonably close to the true values only for small values of* λ*. Note that, as the images are of specific size, the higher the value of* λ *the more the grains overlap, leading to unreliable statistics. This is easily appreciated from the image with circles for* $\lambda = 0.1$ *for which no statistics can be computed since it is totally filled with overlapping grains. In general, for fixed size images, the smaller the* λ *and the smaller the grains, the more reliable the computed statistics are. For higher values of* λ *and/or big grains, one needs a very big image in order to have accurate estimations.*

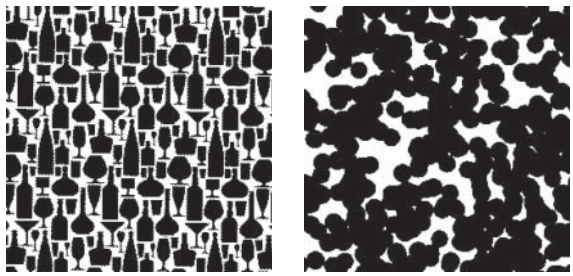
However, one should worry about the accuracy of the estimated parameters only if one wishes to use them for some feedback in the manufacturing process that created the imaged grains, or the characterisation of the size and shape of the grains for some further scientific assessment. If one only wishes to extract some features that characterise the texture pattern for the purpose of recognition and texture classification, even if the computed numbers are not reliably related to the physical properties of the depicted material, they may still be useful as texture discriminators.

Table 2.5 Estimation of the aggregate and individual parameters of the images generated in Examples 2.5 and 2.6. The true value of Π for square grains is 16 and for circular grains and 8-connectivity is 36. The true value of A for square grains is 25 and for circular grains and 8-connectivity is 81.

		Estimated values					
		Aggregate parameters			Individual parameters		
Grain	Real λ	f	L	N^+	λ	Π	A
Square	0.005	0.1274	0.0694	0.0037	0.0042	18.7568	32.1395
	0.010	0.2494	0.1247	0.0054	0.0071	23.3084	40.2493
	0.015	0.3455	0.1672	0.0062	0.0095	26.8594	44.5674
	0.025	0.5064	0.2107	0.0052	0.0105	40.5192	67.0185
	0.050	0.7465	0.2130	0.0028	0.0109	76.7568	125.3698
	0.100	0.9390	0.1307	0.0009	0.0152	141.2973	184.4430
Circle	0.005	0.3555	0.1141	0.0024	0.0037	47.5417	117.9651
	0.010	0.5732	0.1664	0.0039	0.0091	42.6667	93.1781
	0.015	0.7296	0.1543	0.0030	0.0110	51.8655	118.8717
	0.025	0.9102	0.1177	0.0015	0.0161	81.1724	149.2643
	0.050	0.9524	0.0610	0.0008	0.0163	78.7097	187.0172
	0.100	1.0000	0.0199	0.0001	Not applicable		

Example B2.12

Is the simple Boolean model described earlier valid for the image of Figure 2.24a?



(a) (b)

Figure 2.24 Two semi-stochastic textures.

The model is not valid for texture 2.24a for several reasons:

- *The pattern is deterministic and not random.*
- *The grains are not convex.*
- *The grains are too complicated to be adequately described by the shape parameters we use here, namely their mean area and mean perimeter. However, this is no reason by itself to invalidate the Boolean model as there are algorithms and theorems that allow the calculation of higher-order moments of the probability density functions of the grain shape parameters. We simply have not presented them here.*

Example B2.13

Estimate the aggregate and individual parameters of the texture in image 2.24b.

Table 2.6 Estimated parameters for the image in Figure 2.24b, assuming a 2D Boolean model.

Parameter	Value	Parameter	Value
f	0.766663	λ	0.003286
L	0.097748	Π	127.482587
N^+	0.000767	A	442.865387

Table 2.6 shows the results obtained for each aggregate and individual parameter.

Example 2.14

Use the parameters you estimated in Example 2.13 for the texture of Figure 2.24b and create a texture. Comment on its relevance to the original texture from which the parameters were estimated.

Figure 2.25 The original image and two textures created using the 2D Boolean model with parameter values estimated from it.

To create a new texture we use in the Poisson process the estimated value of $\lambda = 0.003286$. *We assume that the grains are digital circles with perimeter equal to the estimated value* $\Pi = 127.48$, *and area* $A = 442.87$. *From Table 2.3 we can see that these values indicate a radius of 16 and 12 respectively. Figure 2.25 shows the textures created using these values. We can easily note that the main difference from the original image is the larger size of grains. It seems that our estimation method overestimated both* $\Pi = 127.48$ *and* $A = 442.87$.

What is the simplest possible primitive pattern we may have in a Boolean model?

It is a straight line segment. This leads to the **1D Boolean models** used to describe binary textures.

What is a 1D Boolean model?

It is a model appropriate for creating and describing binary strings. In this case, the shapes of the Boolean model are simply line segments. For each segment the left end-point is considered to be the origin of the shape. From this origin the shape is said to emanate to the right. Thus, in the 1D Boolean model, a shape is completely identified by the length of the line segment and the position of its origin. In a typical realisation of a 1D Boolean model, shapes tend to overlap each other.

The locations of the origins of the shapes in a 1D Boolean model are the outcomes of the point process with parameter p, which is the probability of a particular point to be the origin of a line segment. This probability is called the **marking probability** and is one of the parameters of the discrete 1D Boolean model. The lengths of the line segments (**primary grains**) are distributed according to some distribution function $C(k)$, where $k = 1, 2, \ldots$. Thus, $C(k)$ is the probability that a line segment has length less than or equal to k, where k is discrete and $C(0) \equiv 0$. The marking probability and the segment length distribution together form the discrete 1D Boolean model.

An observation of a discrete 1D Boolean model consists of a series of covered and uncovered run-lengths (also called **grain segments** and **pore segments**, respectively). An uncovered run-length, that is a sequence of white pixels, is obtained from a sequence of outcomes of the point process that do not consider these points as beginnings of shapes, and that are not covered by other segments. A covered run-length, that is a sequence of black pixels, is due to one or more overlapping outcomes of the shape process.

How may the 1D Boolean model be used to describe textures?

We may use the 1D Boolean model to describe textures by converting first the 2D binary texture into a 1D string, or into a set of 1D strings. Then we may make assumptions about the parametric probability distributions that govern the origin and length of each line segment that appears in these strings. Finally, we may estimate the parameters of these distributions. The values of these parameters are the features that characterise the texture.

How can we create 1D strings from a 2D image?

There are several ways:

1. Read each row of the image separately. If any of the rows starts or ends with a black pixel, we ignore the segment that includes it if we do statistics on the black sequences of pixels. On the other hand, if we compute the statistics of the white sequences, we ignore the sequences that may reach the beginning or the end of a row. This is made clear in Figure 2.26.
2. Read each column of the image separately. Again we ignore incomplete sequences of pixels when we perform statistics. This is made clear in Figure 2.28.
3. Raster scan the image row after row as shown in Figure 2.29a. Note that this method brings next to each other pixels that are at the opposite ends of the image.
4. Raster scan the image column after column as shown in Figure 2.29b. Again, pixels far apart may find themselves next to each other.
5. Raster scan the image row after row, as shown in Figure 2.29c.
6. Raster scan the image column after column as shown in Figure 2.29d.
7. Scan the image using a **Hilbert curve.** An example of such a scan line is shown in Figure 2.29e. Such a scanning path remains in the neighbourhood of a pixel as long as possible before it moves away from it. Such a string, therefore, preserves as much as possible the 2D neighbourhood structure of the data while creating a 1D string out of them.

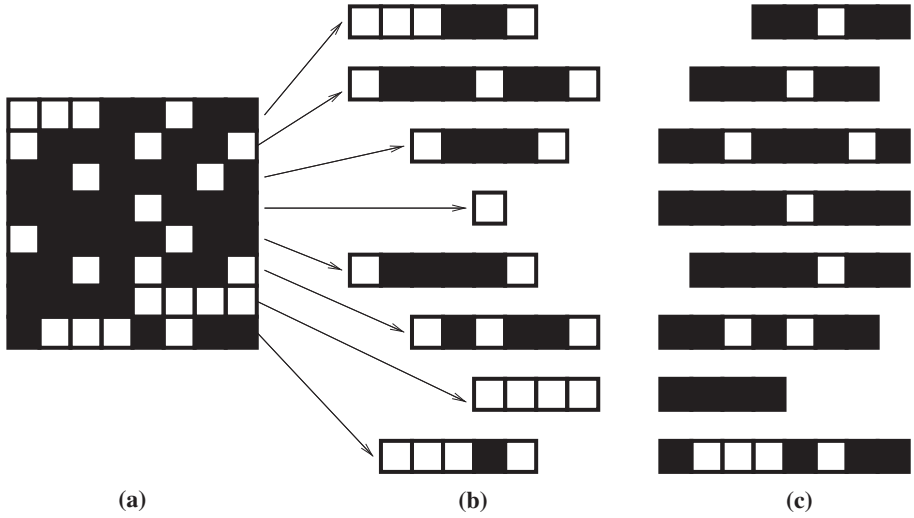


Figure 2.26 (a) An original binary image. The individual strings we can create from it by reading each row separately when we are interested in performing statistics with sequences of (b) black pixels and (c) white pixels. In both cases, sequences that touch the edges of the image are omitted because we are not in a position to know whether they are complete or not.

The strings created by scanning the image of Figure 2.26a in a raster way and by following the Hilbert path are shown in Figure 2.30.

Box 2.5 How to construct Hilbert curves

Step 0: Construct a grid consisting of four quarters.

Step 1: Construct a curve that connects the centres of the four grid elements using three segments, as shown in Figure 2.27a.

Step 2: Reduce the curve you constructed in the previous step to half its size and then copy it four times over to a new grid. The curve is copied without any change in the bottom two quarters, but it is rotated to the left for the top-left quarter and rotated to the right for the top-right quarter.

Step 3: Join the curves in the four quarters with three new segments. Figure 2.27b shows steps 2 and 3 where the three new segments are drawn with dashed lines.

The above steps allow one to construct a Hilbert curve appropriate for scanning a 4×4 image. In the general case, the result of step 2 is reduced to half its size, then copied to a new grid with the required rotations, and finally, three segments are added to join the four curves into one. Figure 2.27c shows the next two steps of the construction of a Hilbert curve, where the added segments are drawn using dotted lines.

Note that, depending on the three segments chosen in the first step, four different curves may be drawn, but they are simply rotated versions of the curves shown in Figure 2.27.

(Continued)

Box 2.5 (Continued)

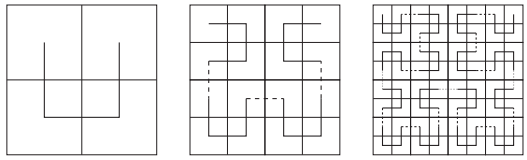


Figure 2.27 The first few steps of drawing a Hilbert curve.

Example B2.15

Construct the Hilbert curve with which you can scan a 4×4 image.

We shall use the algorithm of Box 2.5. Let us consider a square of size 1×1 *and coordinate axes so that the* $(0, 0)$ *point is at the top left corner of the square, with the positive y axis being from top to bottom. Then the coordinates of the four centres of the quarters of the square are* $(0.25, 0.25)$*,* $(0.25, 0.75)$*,* $(0.75, 0.25)$ *and* $(0.75, 0.75)$*. We may visualise this by looking at Figure 2.27a.*

"Construct a curve" in step 2 of the algorithm simply means "read the locations of these centres in the following order: $(0.25, 0.25)$*,* $(0.25, 0.75)$*,* $(0.75, 0.75)$*,* $(0.75, 0.25)$*." Step 3 of the algorithm means that now the coordinates of the sequence of points that make up the curve are:* $(0.125, 0.125)$*,* $(0.125, 0.375)$*,* $(0.375, 0.375)$*,* $(0.375, 0.125)$*.*

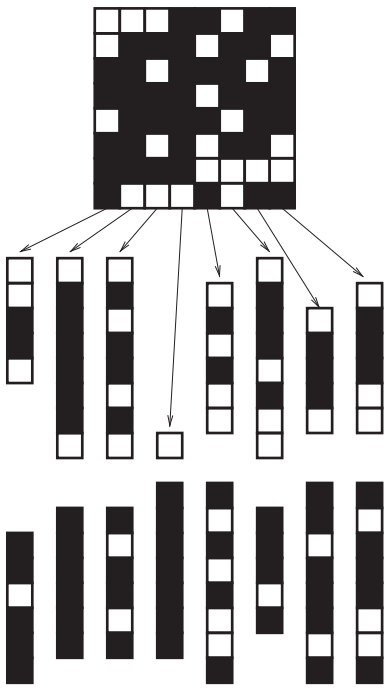


Figure 2.28 At the top, an original binary image. In the middle, the individual strings we can create from it by reading each column separately when we are interested in performing statistics with sequences of black pixels. Note that sequences of black pixels at the ends of the strings have been omitted because they are considered incomplete. At the bottom, the individual strings we can create from the original image by reading each column separately when we are interested in performing statistics with sequences of white pixels. Note that sequences of white pixels at the ends of the strings have been omitted because they are considered incomplete.

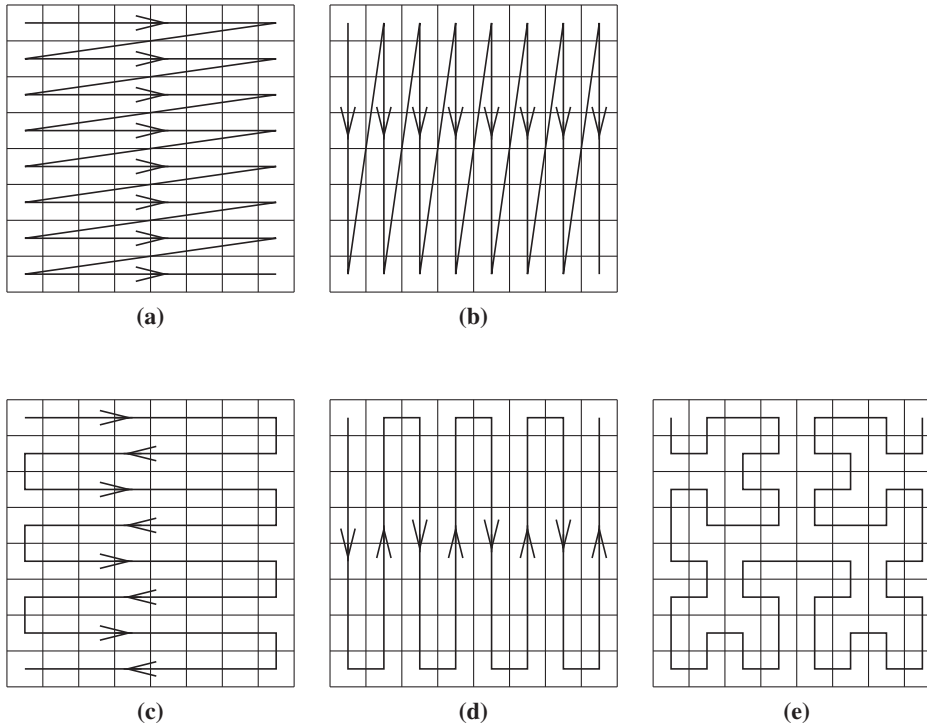


Figure 2.29 (a) A raster scanning line that reads the rows of the image sequentially. (b) A raster scanning line that reads the columns of the image sequentially. (c) A raster scanning line that reads the rows of the image in a boustrophedon way. (d) A raster scanning line that reads the columns of the image in a boustrophedon way. (e) A Hilbert scanning line that remains in the neighbourhood of a pixel as long as possible before it moves away.

To repeat this curve without change, in order to scan the bottom left quarter of the 1×1 *square, simply means shifting these coordinates by* $(0, 0.5)$*, to become* $(0.125, 0.625)$*,* $(0.125, 0.875)$*,* $(0.375, 0.875)$*,* $(0.375, 0.625)$*.*

To scan the bottom right quadrant we simply add to the coordinates $(0.5, 0.5)$*:* $(0.625, 0.625)$*,* $(0.625, 0.875)$*,* $(0.875, 0.875)$*,* $(0.875, 0.625)$*.*

To scan the top left quadrant, we have first to rotate the curve by $90°$ *anticlockwise. This simply means that we permutate the coordinates by one position to the right, so the new order is:* $(0.125, 0.375)$*,* $(0.375, 0.375)$*,* $(0.375, 0.125)$*,* $(0.125, 0.125)$*. However, in order to start the scanning from the top left point, we have also to reverse the order of the points:* $(0.125, 0.125)$*,* $(0.375, 0.125)$*,* $(0.375, 0.375)$*,* $(0.125, 0.375)$*.*

To scan the top right quadrant, we have first to shift the coordinates by $(0.5, 0)$*:* $(0.625, 0.125)$*,* $(0.625, 0.375)$*,* $(0.875, 0.375)$*,* $(0.875, 0.125)$*. Then we have to rotate the curve by* $90°$ *clockwise. This means that the coordinates will be permutated by one position to the left:* $(0.875, 0.125)$*,* $(0.625, 0.125)$*,* $(0.625, 0.375)$*,* $(0.875, 0.375)$*. Finally, in order to finish with the top right point, we have to reverse the order:* $(0.875, 0.375)$*,* $(0.625, 0.375)$*,* $(0.625, 0.125)$*,* $(0.875, 0.125)$*.*

All we have to do now is to concatenate the four sequences we created, starting from the one that scans the top left quadrant, carrying on with the one that scans the bottom left quadrant, then the bottom right quadrant and finally the top left quadrant: $(0.125, 0.125)$*,* $(0.375, 0.125)$*,*

(Continued)

Example B2.15 (Continued)

$(0.375, 0.375)$, $(0.125, 0.375)$, $(0.125, 0.625)$, $(0.125, 0.875)$, $(0.375, 0.875)$, $(0.375, 0.625)$, $(0.625, 0.625)$, $(0.625, 0.875)$, $(0.875, 0.875)$, $(0.875, 0.625)$, $(0.875, 0.375)$, $(0.625, 0.375)$, $(0.625, 0.125)$, $(0.875, 0.125)$.

To work out the pixel coordinates for a 4×4 *image, we multiply all these values with 4 and subtract 0.5 to account for the fact that the centre of the first pixel is at point with continuous coordinates* $(0.5, 0.5)$ *and we call it pixel* $(0, 0)$*. So, the corresponding pixel coordinates are:* $(0, 0)$, $(1, 0)$, $(1, 1)$, $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 3)$, $(1, 2)$, $(2, 2)$, $(2, 3)$, $(3, 3)$, $(3, 2)$, $(3, 1)$, $(2, 1)$, $(2, 0)$, $(3, 0)$.

Example B2.16

You are given the following 4×4 **image:**

$$g = \begin{pmatrix} 1 & 0 & 3 & 5 \\ 6 & 1 & 0 & 5 \\ 2 & 3 & 0 & 6 \\ 1 & 3 & 5 & 2 \end{pmatrix}. \tag{2.39}$$

Scan it by using the Hilbert curve you constructed in Example 2.15, to create a 1D string.

All we have to do is to read the grey values following the sequence of pixels we constructed in Example 2.15: $1, 0, 1, 6, 2, 1, 3, 3, 0, 5, 2, 6, 5, 0, 3, 5$

How can we estimate the parameters of the 1D Boolean model?

As in the 2D case the individual parameters, namely the marking probability p and the distribution function $C(k)$ of the lengths of the line segments, may be estimated from some aggregate parameters. The aggregate parameters used in this case are:

N_p: the number of white points in the sequence;

N_{ps}: the number of white segments in the sequence;

$S(k)$: the number of pairs of white pixels that are k pixels apart, for various values of k.

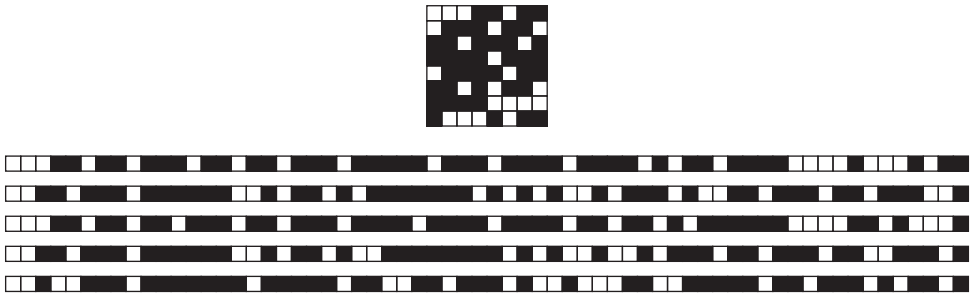


Figure 2.30 An original image and the strings one can create from it by following the scanning lines shown in Figure 2.29(a)–(e) from top to bottom, respectively.

From these aggregate parameters the individual parameters are estimated by using the formulae:

$$p = \frac{N_{\text{ps}}}{N_p} \qquad C(k) = 1 - \frac{1}{p}\left(1 - \frac{S(k+1)}{S(k)}\right). \tag{2.40}$$

The origin of these last two equations is explained in Box 2.6. Note that for characterising the texture one may use the aggregate parameters directly, or one may use the derived individual parameters. The individual parameters have the extra advantage that they allow one to create textures of similar statistical characteristics as the original texture and in addition they give an insight into the process that created the observed pattern. This may be significant for some applications.

Example 2.17

Estimate the aggregate and the individual parameters of the texture in Figure 2.24b, using all scans of Figure 2.29 to convert the image into a 1D string.

The values of the aggregate parameters N_p and N_{ps} for each type of scanning method used to produce the 1D string are shown in Table 2.7. From these we compute the individual parameter p, also shown in the same table. From the values of $S(k)$ (which are aggregate parameters) and the value of p, we compute the values of $C(k)$ that are individual parameters. These are shown plotted versus k in Figure 2.31. Since $C(k)$ is a distribution function, values outside the range $[0, 1]$ are clearly erroneous and they should not be used as texture features. We use only values of k for which $C(k)$ is roughly inside the range $[0, 1]$. We note that for Hilbert scanning, $C(k)$ is not behaving at all like a distribution function. As it is not even monotonically increasing, it indicates negative values of the corresponding probability density function. This behaviour is a strong indication that the Boolean model is not applicable to the string produced from the original image by Hilbert scanning. Another problem with this string is that it consists of very long sequences of black pixels, since the scanning curve remains in the vicinity of a pixel as long as possible. As the total length of the string is fixed, long sequences of black pixels indicate that the string may not be long enough for reliable statistics to be computed.

The other four scanning methods yield approximately similar curves $C(k)$. This should not be surprising as the texture is isotropic, and rotating it by $90°$ should not cause any significant change in the values of the descriptors we are using.

Table 2.7 Aggregate parameters N_p and N_{ps} and individual parameter p estimated from image 2.24b using the scanning methods shown in Figure 2.30.

Scan	N_p	N_{ps}	p
(a)	15292	1512	0.098875
(b)	15292	1539	0.100641
(c)	15292	1464	0.095736
(d)	15292	1503	0.098287
(e)	15292	1462	0.095606

(Continued)

Example 2.17 (Continued)

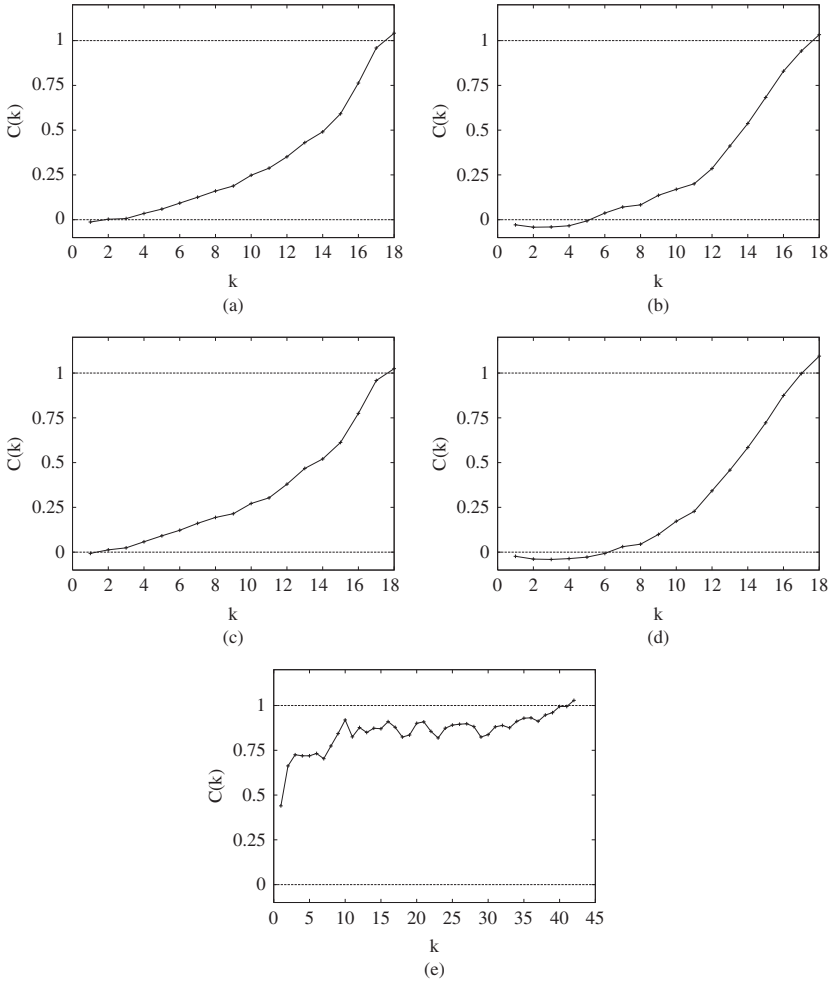


Figure 2.31 Function $C(k)$ estimated from image 2.24b using the scanning methods shown in Figure 2.29, from (a) to (e), respectively.

Example 2.18

Estimate the aggregate and the individual parameters of the texture in Figure 2.24b, using the lines and columns of the image individually, without concatenating them to form continuous strings. Ignore the sequences which touch the edges of the image as we are not in a position to know whether they are complete or not.

The results are shown in Table 2.8, and the values of $C(k)$ are plotted in Figure 2.32.

No noticeable difference appears to be between these results and those in Example 2.17. This is an indication that the image is sufficiently large for the statistics computed not to be significantly affected by the false strings created by the raster and boustrophedon scannings.

Table 2.8 Aggregate parameters N_p and N_{ps} and individual parameter p estimated from image 2.24b using each row and column of the image independently, as shown in Figures 2.26 and 2.28.

Scan	N_p	N_{ps}	p
Rows	15292	1527	0.099856
Columns	15292	1542	0.100837

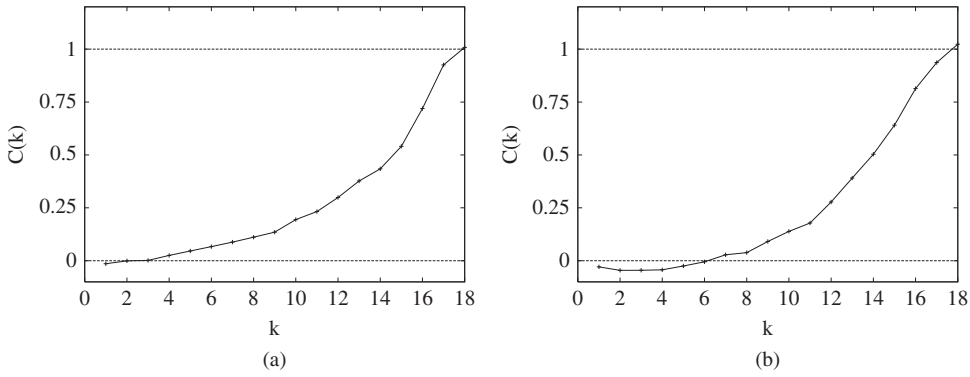


Figure 2.32 Function $C(k)$ estimated from image 2.24b using the image rows (a), and columns (b), separately. Curve (a) of this figure is very similar to curves (a) and (c) of Figure 2.31. This indicates that the erroneous strings created by the wrapping of the scanning line in example 2.17 for cases (a)–(d) have an insignificant effect on the computed values of N_p and N_{ps}.

Box 2.6 Parameter estimation for the discrete 1D Boolean model

Let X be a Boolean sequence of length L characterised by parameters p and $C(k)$, with p being the probability of any particular pixel being the origin of a black line, and $C(k)$ being the probability of finding a line made up with fewer than k black pixels.

Let us consider a pair of neighbouring pixels in the sequence. For this pair to represent a transition from a white pixel to a black pixel, the first pixel must be white, and the second must be the beginning of a black line. The estimated probability of a pixel being white is equal to the fraction of the white pixels in the sequence, i.e. N_p/L, where N_p is the number of white points in the sequence. Since p is the probability of a pixel being the origin of a black line, then the probability of finding a transition pair of pixels is pN_p/L. If we multiply this with the total number L of pixels in the sequence, we have the total number of such transitions that we expect to find: pN_p. And since each white segment of pixels has only one end, the number

(Continued)

Box 2.6 (Continued)

of ends, i.e. transitions we find, must be equal to the number N_{ps} of white segments in the sequence: $N_{ps} = pN_p$. From this we obtain that the marking probability p can be estimated as the ratio of these aggregate parameters: $p = N_{ps}/N_p$, i.e. the ratio of the number of white sequences observed over the total number of white pixels observed.

Let us consider next the probability $S(k)$ of finding two white pixels k positions apart. Let us name the first of these pixels x_0 and the following ones x_1, x_2 and so on. For such a pair of pixels to exist, the first one must be white, the second one should not be the origin of a line longer than $k-1$ pixels, the third one must not be the origin of a line longer than $k-2$ pixels, and so on, until the x_{k-1} pixel, which should not be the origin of a line longer than 1 pixel, and pixel x_k, which should not be the origin of a line. To find the probability $S(k)$, we must multiply the probabilities of these pixels being in the desired state.

Let us remind ourselves that $C(j)$ is the probability of having a line shorter than or equal to j. Then the probability of having a line longer than j must be $1 - C(j)$. The probability of having a line longer than j starting from a particular pixel must then be $p(1 - C(j))$. So, the probability of **not** having a line longer than j starting from a particular pixel is $1 - p(1 - C(j))$. If we apply this for $j = k-1, j = k-2, \ldots, j = 1$, and multiply all these probabilities and also take into consideration the probability $p(w)$ of finding a white pixel in the first place, and the probability p of pixel x_k not being the origin of a line, we find:

$$S(k) = p(w)\prod_{j=1}^{k-1}\left[1 - p(1 - C(j))\right]p. \tag{2.41}$$

Let us apply this for $k+1$:

$$S(k+1) = p(w)\prod_{j=1}^{k}\left[1 - p(1 - C(j))\right]p. \tag{2.42}$$

If we divide these two equations by parts, we obtain:

$$\frac{S(k+1)}{S(k)} = 1 - p(1 - C(k)) \Rightarrow p(1 - C(k)) = 1 - \frac{S(k+1)}{S(k)}$$
$$\Rightarrow \quad 1 - C(k) = \frac{1}{p}\left(1 - \frac{S(k+1)}{S(k)}\right) \Rightarrow C(k) = 1 - \frac{1}{p}\left(1 - \frac{S(k+1)}{S(k)}\right). \tag{2.43}$$

So, $C(k)$ may be computed directly from observable parameters of the sequence.

Example 2.19

Figure 2.33 shows an 8×8 binary image and the binary string created from it by reading its pixels in a boustrophedon way. Estimate the values of p and $C(1), C(2), C(3)$ and $C(4)$ of the texture it depicts, using the white pixels and ignoring boundary effects.

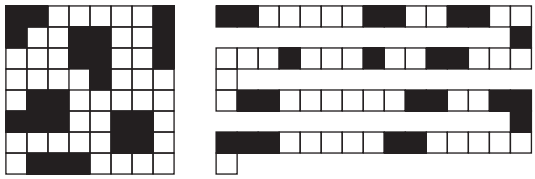


Figure 2.33 An 8×8 binary image and the string created from its pixels by reading them in a boustrophedon way. The string is presented folded to fit in the page.

The total number of white pixels in the image is $N_p = 41$. The total number of strings of white pixels is $N_{ps} = 12$. So, $p = 12/41 = 0.29$. Then we count the number of pairs of white pixels that are next to each other ($k = 1$), the number of pairs of white pixels that have one other pixel in between them ($k = 2$), etc., up to the number of pairs of white pixels that have four other pixels between them ($k = 5$). These are the values of $S(k)$. We then use equation (2.43) to estimate $C(k)$:

$k = 1$ $S(1) = 29$ $C(1) = 1 - \frac{41}{12}\left(1 - \frac{20}{29}\right) = -0.06$

$k = 2$ $S(2) = 20$ $C(2) = 1 - \frac{41}{12}\left(1 - \frac{23}{20}\right) = 1.51$

$k = 3$ $S(3) = 23$ $C(3) = 1 - \frac{41}{12}\left(1 - \frac{27}{23}\right) = 1.59$

$k = 4$ $S(4) = 27$ $C(4) = 1 - \frac{41}{12}\left(1 - \frac{22}{27}\right) = 0.37$

$k = 5$ $S(5) = 22.$

What happens if the primitive patterns are very irregular?

An example of such a texture is shown in Figure 2.2c. Such a pattern requires a purely statistical description, as parametrisation of the observed shapes is very difficult. The best tool for this is **mathematical morphology**.

2.3 Mathematical Morphology

What is mathematical morphology?

Mathematical morphology is a collection of non-linear processes that can be applied to an image to remove details smaller than a certain reference shape, which is called **structuring element**. Some examples of such reference shapes are shown in Figure 2.34. The lines drawn cross at the point that is the centre of each shape. Mathematical morphology relies on two simple operations: **dilation** and **erosion**. Other operations, like **opening** and **closing**, may be defined in terms of them.

What is dilation?

Dilation is the process by which we "dilate" the objects in the image by placing the centre of the structuring element on every object pixel and marking every pixel covered by the structuring element as an object pixel. Figure 2.36a delineates the background pixels which will be turned into object pixels if a structuring element of size 3×3 pixels (shown in Figure 2.34a) were used to dilate the original image of Figure 2.35. Figure 2.36b shows the result of this operation. Notice that the object has lost any gaps narrower than 3 pixels it used to have, and at the same time it has been augmented in size.

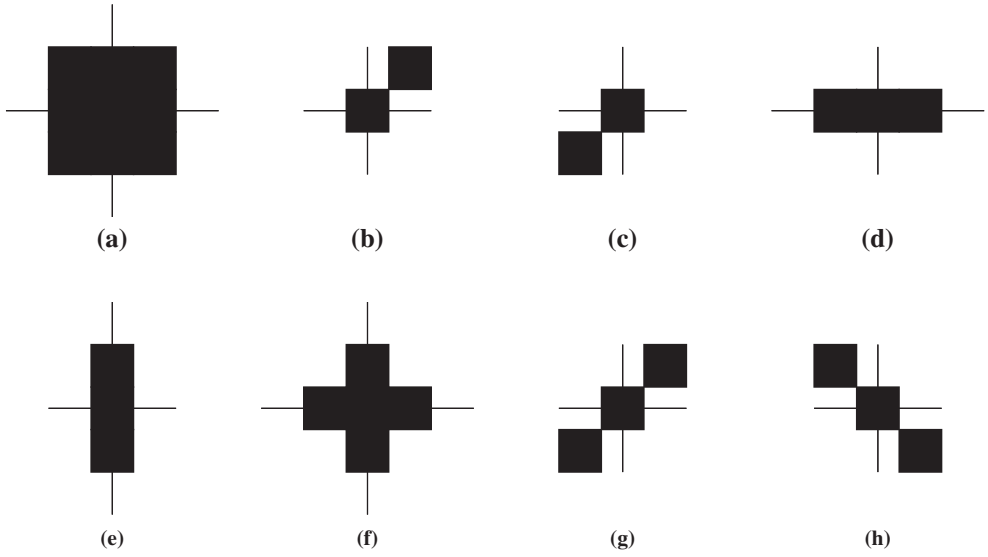


Figure 2.34 Some example structuring elements. The crossed lines mark the centres of the elements. The size of structuring element (a) is 3×3 pixels and elements (d) and (e) are three pixels long.

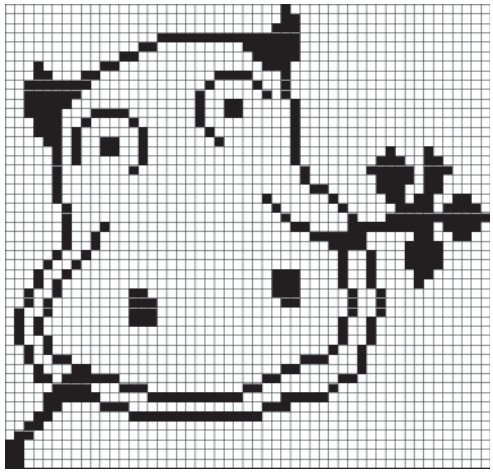
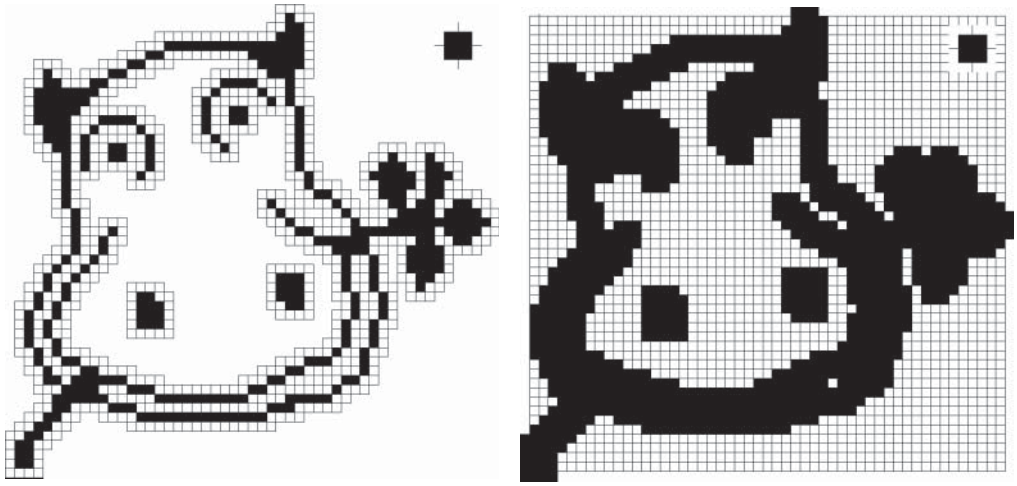


Figure 2.35 The amorous hippopotamus.

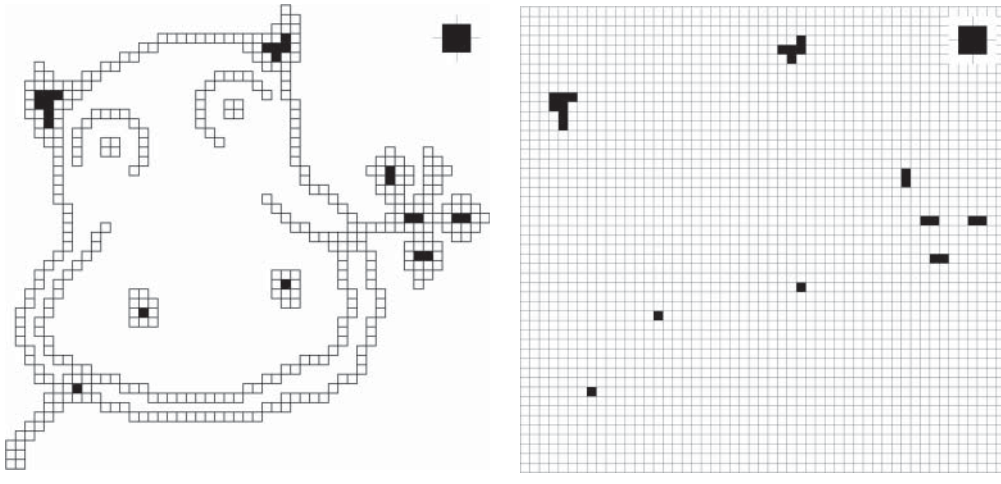
What is erosion?

Erosion is the process by which we "erode" an object by placing the centre of the structuring element on every object pixel and keeping only those pixels that allow the structuring element to fit fully inside the object. Figure 2.37a delineates the object pixels that will be turned into background pixels if a structuring element of size 3×3 pixels (shown in Figure 2.34a) were to be used to erode the original image of Figure 2.35. Figure 2.37b shows the result of this operation. Notice that the object has been reduced only to a few pixels that were at the centre of thick patches. Any details narrower than 3 pixels are lost, and details larger than this have been reduced in size because they lost a layer of pixels all around them.



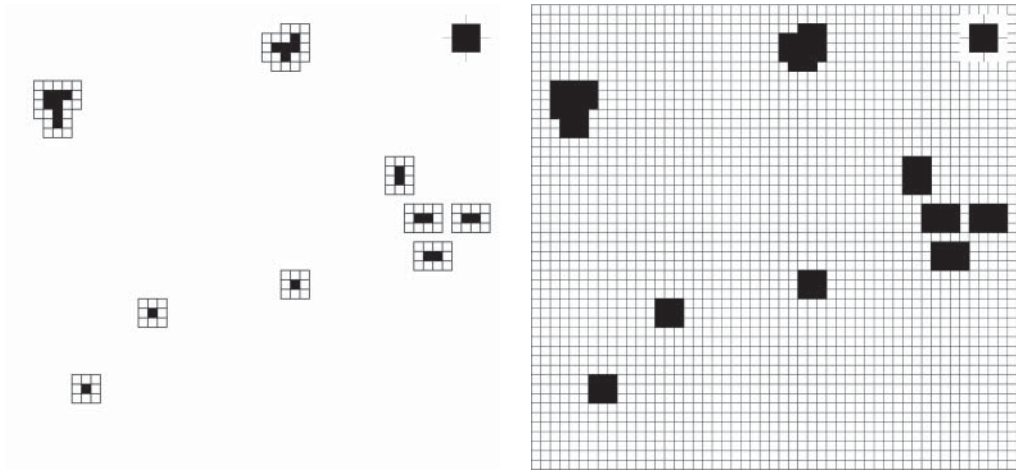
(a) (b)

Figure 2.36 (a) The amorous hippopotamus about to be dilated with the structuring element of Figure 2.34a (shown at the top right here). The white pixels were covered by the structuring element when the latter was placed on each of the black pixels in turn, and they will be converted into black pixels too. (b) The amorous hippopotamus dilated with the structuring element of Figure 2.34a (shown at the top right here). All white pixels in (a) were turned into black.



(a) (b)

Figure 2.37 (a) The amorous hippopotamus about to be eroded with the structuring element of Figure 2.34a (shown at the top right here). The white pixels constitute details narrower than the structuring element, so they will be converted into background pixels. (b) The amorous hippopotamus eroded with the structuring element of Figure 2.34a (shown at the top right here). All white pixels shown in (a) became background pixels.



(a) (b)

Figure 2.38 (a) The amorous hippopotamus about to be opened. All white pixels will be restored to their original status, i.e. they will become object pixels again. These pixels are identified by dilating the eroded image 2.37b. (b) The amorous hippopotamus opened.

Is there any way to lose details smaller than a certain size but leave the size of larger details unaffected?

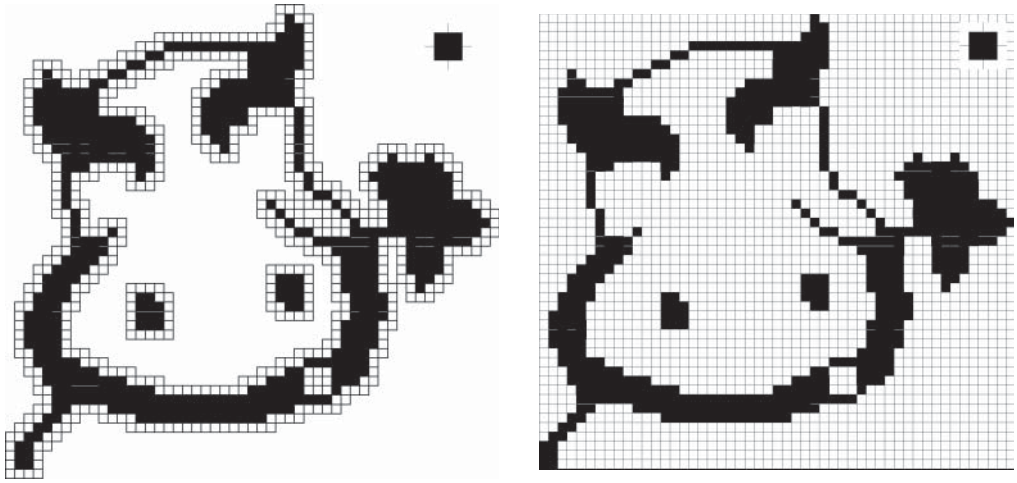
Yes, this can be done by the operations of **opening** and **closing**.

What is opening?

Opening is the operation by which we dilate an image after we have eroded it. Figure 2.38a delineates the background pixels that will be turned into object pixels if a structuring element of size 3×3 pixels (shown in Figure 2.34a) were used to dilate the eroded version of the amorous hippopotamus shown in Figure 2.37b. The final result of this operation is shown in Figure 2.38b. Notice that the thin details of the original object have been lost, while the solid parts have been rounded and they have been restored to their original size. We say that opening removes the "hairs" of an object.

What is closing?

Closing is the operation by which we erode an image after it has been dilated. Figure 2.39a delineates the object pixels that will be turned into background pixels if a structuring element of size 3×3 pixels (shown in Figure 2.34a) were to be used to erode the dilated version of the amorous hippopotamus shown in Figure 2.36b. The final result of this operation is shown in Figure 2.39b. Notice that the narrow gaps in the original object have been filled, while the overall size of the object has not increased. This is because the process of erosion removed the extra layer of pixels around the object which had been added by the process of dilation. We say that closing fills up the narrow "gulfs" of an object.

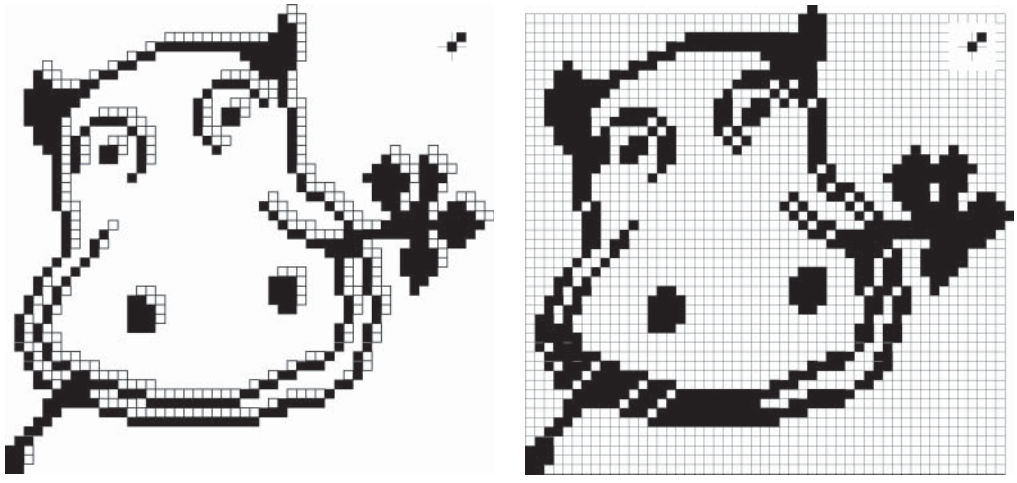


(a) (b)

Figure 2.39 (a) The amorous hippopotamus about to be closed. All white pixels will be returned to their original status, i.e. being background pixels. They were identified by eroding the dilated image 2.36b. (b) The amorous hippopotamus closed.

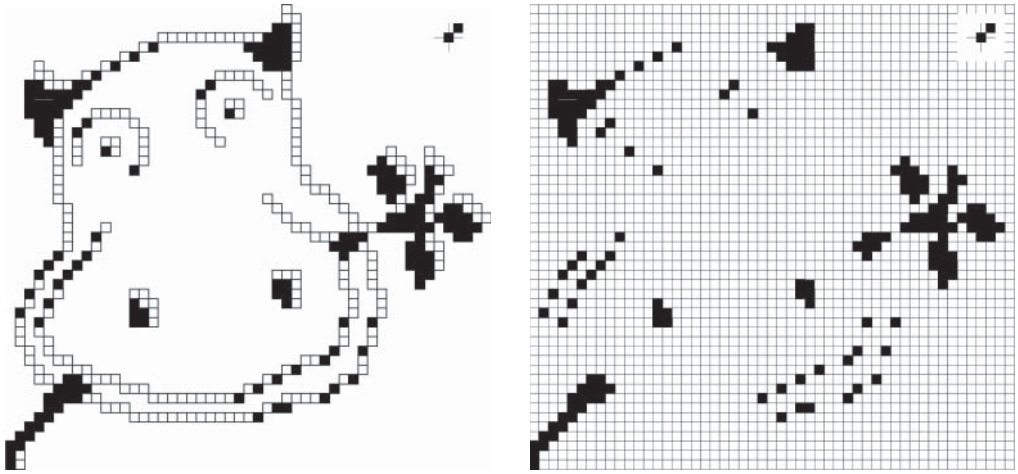
How do we do morphological operations if the structuring element is not symmetric about its centre?

Figures 2.40–2.43 show the results of applying all operations we have discussed so far to the amorous hippopotamus using as structuring element the object of Figure 2.34b. We notice that



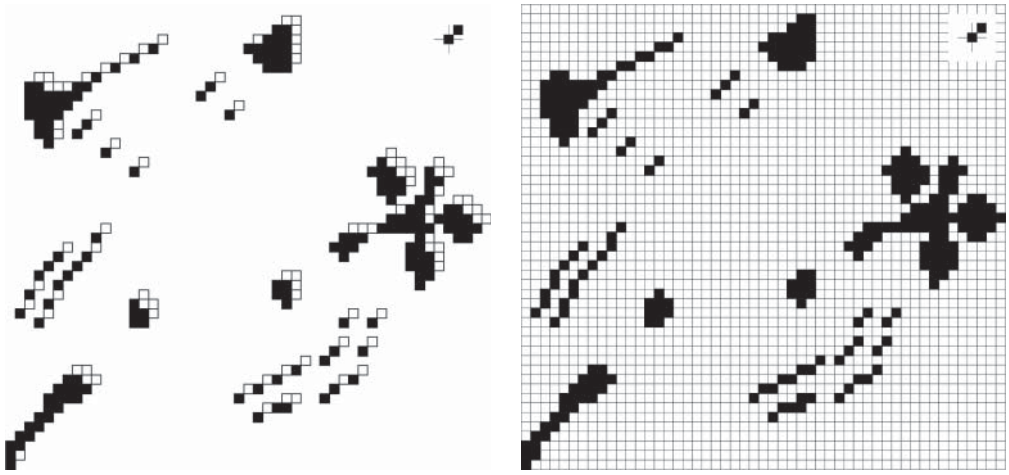
(a) (b)

Figure 2.40 (a) The amorous hippopotamus about to be dilated with the structuring element shown at the top right. All white pixels will be made object pixels. (b) The amorous hippopotamus dilated. Note the shift to the top right in relation to the original grid.



(a) (b)

Figure 2.41 (a) The amorous hippopotamus about to be eroded with the structuring element shown at the top right. The delineated pixels are object pixels that will be turned into background pixels because when the centre of the structuring element was placed on them, the structuring element could not fit fully inside the object. (b) The amorous hippopotamus eroded.



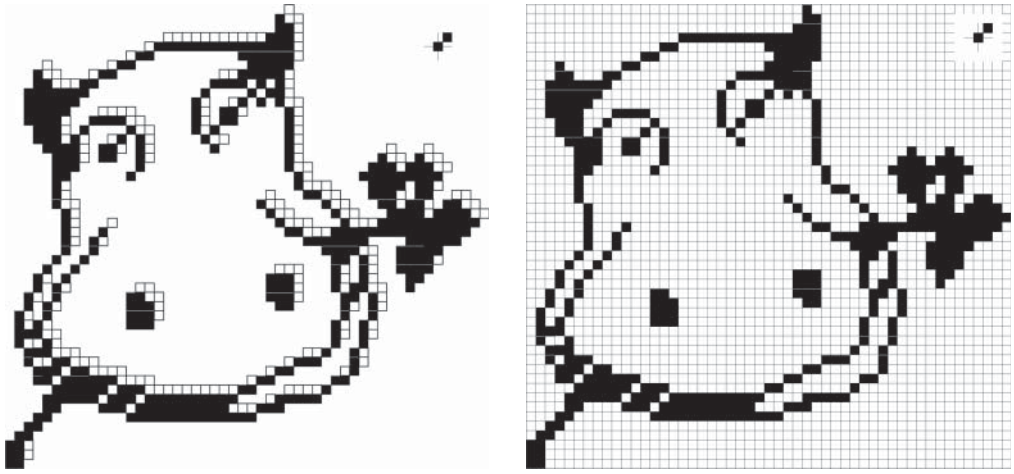
(a) (b)

Figure 2.42 (a) The amorous hippopotamus about to be opened with the structuring element shown at the top right. The delineated pixels are going to be turned into black. (b) The amorous hippopotamus opened. To achieve opening we placed the centre of the structuring element at every black pixel of image 2.41b and converted all pixels it covered into black.

this element treats preferentially details along the diagonal from bottom left to top right. To allow comparison with other results, the superimposed grid remains in its original position in all figures.

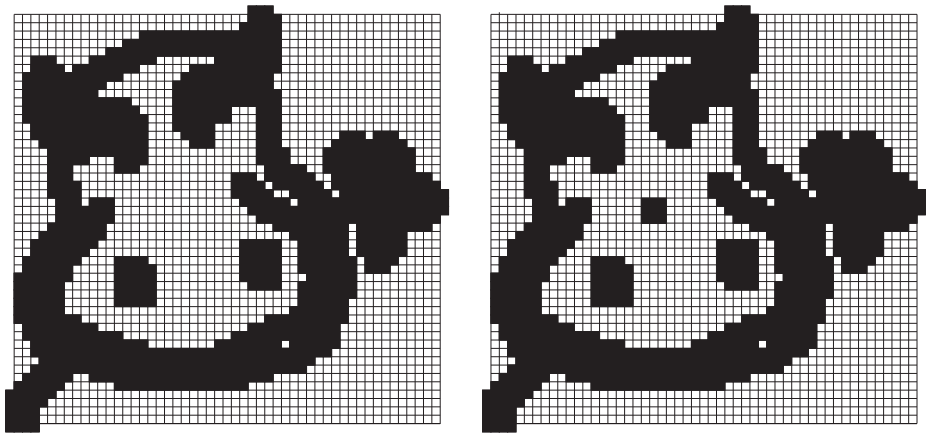
Since the structuring element looks like a small image, can we exchange the roles of object and structuring element?

Yes, we can. As an example, let us dilate the object of Figure 2.34a by using as structuring element the amorous hippopotamus of Figure 2.35. However, we have first to decide which the centre of



(a) (b)

Figure 2.43 (a) The amorous hippopotamus about to be closed with the structuring element shown at the top right. The delineated pixels are going to be removed. (b) The amorous hippopotamus closed. To achieve closing we placed the centre of the structuring element at every black pixel of 2.40b and if the structuring element could not fit fully inside the black region, the pixel was deleted.



(a) (b)

Figure 2.44 The object of Figure 2.34a dilated with the structuring element of Figure 2.36. It is done by placing the centre of object 2.36 in all nine positions of object 2.34a. Depending on whether the centre of object 2.36 is a black or a white pixel, we shall obtain result (a) or result (b), with the 3×3 original object in the second case visible with its centre coinciding with the selected centre of object 2.36.

the amorous hippopotamus image is. Let us assume that the centre is one of its black pixels. The result then is shown in Figure 2.44a which was effectively produced by placing the centre of the amorous hippopotamus to the nine pixels that make up object 2.34a and superimposing the results. By comparing this figure with that of Figure 2.36b we see that there is no difference. Let us next assume that the centre of the amorous hippopotamus is one of the interior pixels of the image, which happens to be a white pixel. By placing this centre on each of the nine pixels that make up object 2.34a and superimposing the results, we obtain the result shown in Figure 2.36b. We see that the difference with Figure 2.36b now is the presence of the original 3×3 black square in the centre

of the new result. So dilation is a **commutative** operation, when the centre of the structuring element is a pixel that belongs to the set of object pixels. In such a case, it does not matter which object we dilate by which. However, this is not true for erosion. If we tried to erode object 2.34a by the amorous hippopotamus, we would get nothing because the amorous hippopotamus cannot fit inside object 2.34a.

Is closing a commutative operation?

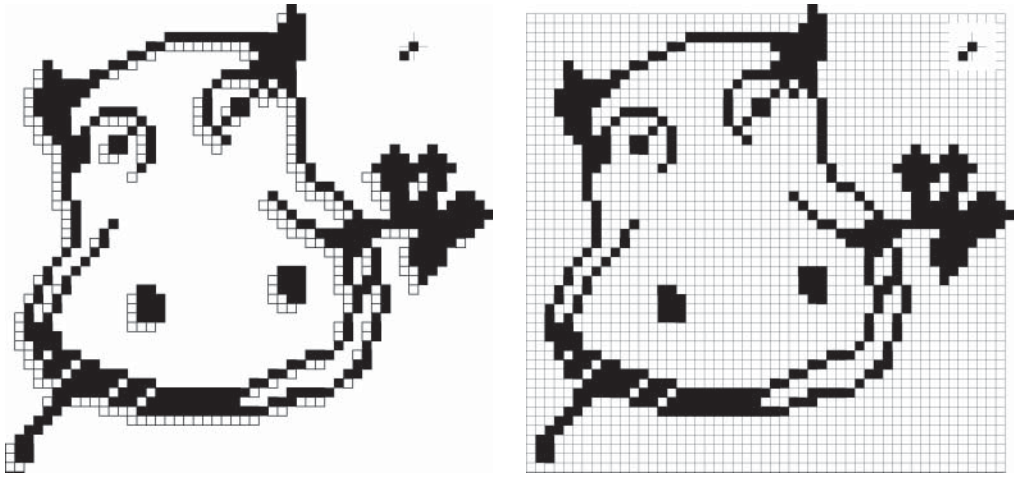
No, it is not. If we apply erosion to either of the dilated images 2.44 using as structuring element 2.35, we shall obtain object 2.34a. We shall not obtain the result we had by closing the amorous hippopotamus with the square shown in Figure 2.34a. This is because erosion is not a commutative operation.

Can we use different structuring elements for the erosion and the dilation parts of the opening and closing operators?

Yes, we can use any elements we like, depending on the application we are interested in. For example, in Chapter 1 we presented an algorithm for finding the boundaries of textured regions from non-textured ones. The operation we did to eliminate edge pixels along boundaries of two uniform regions was a type of erosion and dilation with locally adaptive structuring elements. In addition, mathematicians propose that the structuring elements used for dilation and erosion in the opening and closing operations must be the negative of each other. For example, if erosion is done by using the structuring element of Figure 2.34b, to obtain opening we must use the element of Figure 2.34c for the dilation part. Equivalently, if the dilation is done by using the structuring element of Figure 2.34b, to obtain closing, we must use the element of Figure 2.34c for the erosion part. Figure 2.45a delineates the pixels that have to be removed from the dilation of Figure 2.40b, in order to obtain the closing of the amorous hippopotamus obtained by dilating first with the structuring element of Figure 2.34b, followed by erosion with the structuring element of Figure 2.34c. The result is shown in Figure 2.45b. Notice that the difference from Figure 2.43b is a shift of the image to the **top right** by one pixel. Figure 2.46a delineates the pixels that have to be added to the erosion of Figure 2.41b, in order to obtain the opening of the amorous hippopotamus obtained by eroding first with the structuring element of Figure 2.34c, followed by dilation with the structuring element of Figure 2.34b. Notice that the difference between this result shown in Figure 2.46b and that shown in figure 2.42b is a shift of the new result towards the **bottom left** by one pixel. The shifts observed in results 2.45b and 2.46b are made more obvious in the simple examples shown in Figure 2.47.

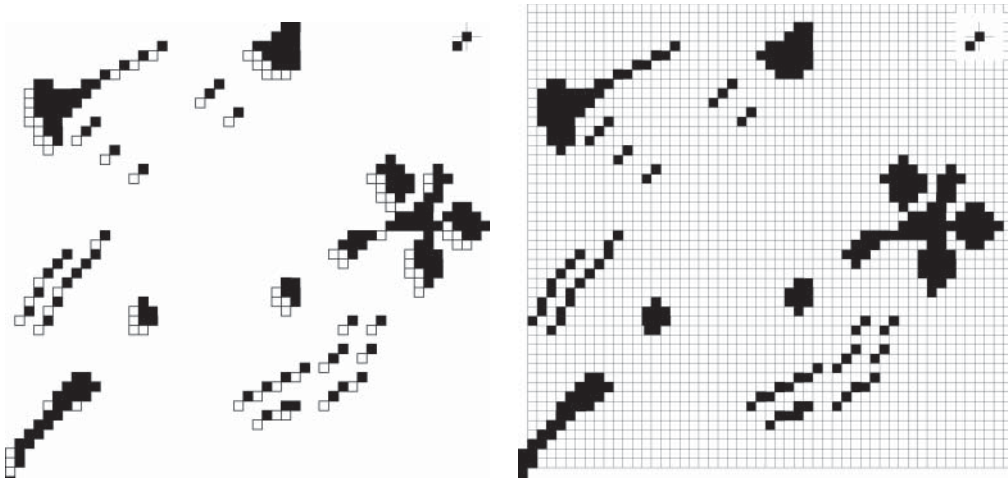
Can we apply morphological operators to the white pixels of an image instead of applying them to the black pixels?

Yes, we can define as the "object" in an image whichever class we choose: either the black pixels or the white pixels. If we apply erosion to the white pixels of image (Figure 2.35) we shall obtain the same result that we would obtain were we to apply dilation with the same structuring element to the black pixels and vice versa. For example, image 2.37b could have been obtained by dilating the white pixels of image 2.36 with a white square structuring element of size 3×3. Note, however, that erosion 2.41b could have been obtained by dilating the white pixels of image 2.36 with a white structuring element in the shape of 2.34c, which is the reflection of that used for eroding the black



(a) (b)

Figure 2.45 (a) The amorous hippopotamus after his dilation with structuring element 2.34b (result in 2.40b), about to be eroded with structuring element 2.34c. The delineated pixels are to be removed. (b) The result. The object was shifted by one pixel to the top right in comparison with image 2.43b.



(a) (b)

Figure 2.46 (a) The amorous hippopotamus after his erosion with structuring element 2.34b (result in 2.41b), about to be dilated with structuring element 2.34c. The delineated pixels are to be converted into object pixels. (b) The result. The object was shifted by one pixel to the bottom left in comparison with image 2.42b.

pixels. This distinction is not obvious for the case of element 2.34a which is symmetric about its origin. We say that erosion and dilation are **dual** operations. Opening and closing are also dual of each other.

Can we apply more than one morphological operator to the same image?

Yes, we can apply as many operators as we like and the image can take. In fact some complex morphological operators are best applied in a cascaded way, one after the other. For example, if we

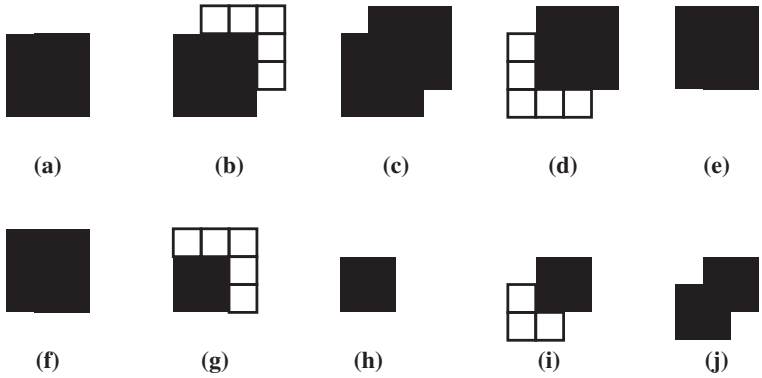


Figure 2.47 The image in (a) is dilated with structuring element 2.34c, to produce (c), which then is eroded with structuring element 2.34b to produce the closing shown in (e). Note that the net effect is a shift of the image towards the top right. In the bottom row, the same original image is first eroded with structuring element 2.34c, to produce (h) and then dilated with structuring element 2.34b to produce the opening shown in (j). Note that the net effect is a shifted copy of the eroded image towards the bottom left added to the originally eroded image.

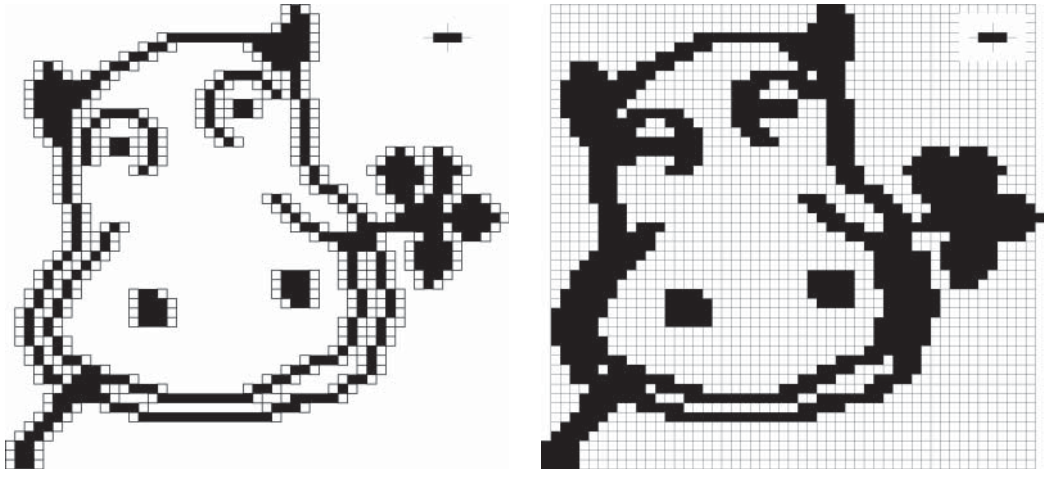


Figure 2.48 (a) The amorous hippopotamus about to be dilated with structuring element 2.34d (shown here at the top right). (b) The amorous hippopotamus dilated.

dilate an image with the structuring element shown in 2.34d and the result with the structuring element shown in 2.34e, we shall obtain the same result as if we were to dilate the original image with the structuring element shown in 2.34a. This is demonstrated in Figure 2.48 which shows the dilation of the amorous hippopotamus with structuring element 2.34d. Figure 2.49 shows what will happen to the result if it is dilated with structuring element 2.34e. If all delineated pixels are made black, this figure will be identical to 2.36b which was the dilation of the original image with structuring element 2.34a. We say that dilation is an **associative** process:

$$\begin{aligned}\text{Result} &= \text{Dilation}(\text{Dilation}(\text{Image\_A by Image\_B}) \text{ by Image\_C})\\ &= \text{Dilation}(\text{Image\_A by Dilation}(\text{Image\_B by Image\_C})).\end{aligned}$$

Figure 2.49 The dilated amorous hippopotamus of Figure 2.48b about to be dilated with the structuring element of Figure 2.34e (shown here at the top right). The result will be the same as that shown in Figure 2.36b.

Is erosion an associative operation as well?

No, the successive erosion of an image by two structuring elements is equivalent to the erosion of the image with a structuring element which is the result of dilating one of the original structuring elements by the other:

$$\begin{aligned}\text{Result} &= \text{Erosion(Erosion(Image\_A by Image\_B) by Image\_C)}\\ &= \text{Erosion(Image\_A by Dilation(Image\_B by Image\_C))}.\end{aligned}$$

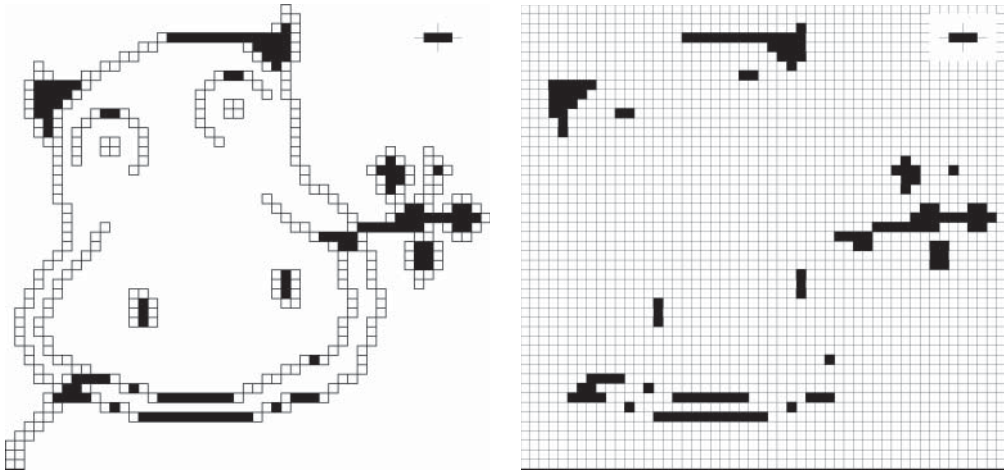
Figure 2.50 demonstrates the erosion of the amorous hippopotamus with structuring element 2.34d. Subsequent erosion of the result by structuring element 2.34e will remove the delineated pixels of Figure 2.51. The result will be the same as that shown in Figure 2.37b which was obtained by directly eroding the original image with 2.34a. This is because structuring element 2.34a may be obtained by dilating structuring element 2.34d with structuring element 2.34e.

This property finds significant applications in the use of mathematical morphology for texture characterisation. For example, if we use a 3×3 square as the structuring element and we erode an image twice, this is equivalent to eroding the image only once with a square structuring element of size 5×5.

How can we use morphological operations to characterise a texture?

We saw how the various morphological operators may remove details of the image smaller than a certain size, either by cutting them off (if they are protrusions), or filling them up (if they are intrusions). For example, if in Figure 2.37a we count the delineated white pixels, ignoring those that are also white in 2.38a, we shall have the number of pixels that form parts of the object smaller than 3×3 pixels in size.

Imagine now that we open an object (effectively the black pixels of a binary texture) with structuring elements of various sizes, say $d \times d$, where $d = 3, 5, 7, \ldots$. After every opening we count the number of pixels that have been removed when the result is compared with the original image. These are the pixels that constitute details smaller than the size of the structuring element in the original image. We shall have a series of numbers then, that constitute a function C that depends



(a) (b)

Figure 2.50 (a) The amorous hippopotamus about to be eroded with structuring element 2.34d (shown here at the top right). The white pixels are to be removed. (b) The amorous hippopotamus eroded.

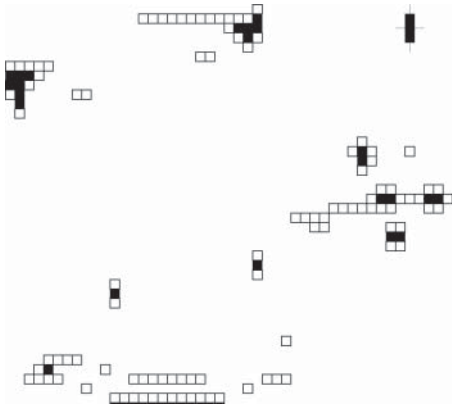


Figure 2.51 The eroded amorous hippopotamus of Figure 2.50b about to be eroded again with the structuring element of Figure 2.34e (shown here at the top right). If we remove the delineated pixels, the result will be the same as that of Figure 2.37b which was obtained by using a structuring element that is the dilation of structuring element 2.34d with structuring element 2.34e.

on the size of the structuring element d. This function $C(d)$ is called the **pattern spectrum** of the texture and it may be used to characterise and classify the texture.

The process of computing $C(d)$ is known as **granulometry** because it counts how many "granules" of a certain size are present in the texture.

Figure 2.52 shows some openings of image 2.1 obtained by using structuring elements of increasing size. Figure 2.53 shows the pattern spectrum $C(d)$ of this image which was computed from these openings.

If we wish to capture the directionality of textures, we may use anisotropic (i.e. not circularly symmetric) structuring elements. However, it is better if they are symmetric about their centres. For example, for exploring the directionality of a texture along the two main diagonal directions, elements 2.34g and 2.34h should be used, instead of elements 2.34b and 2.34c. For directionalities along other orientations, appropriate structuring elements may be constructed.

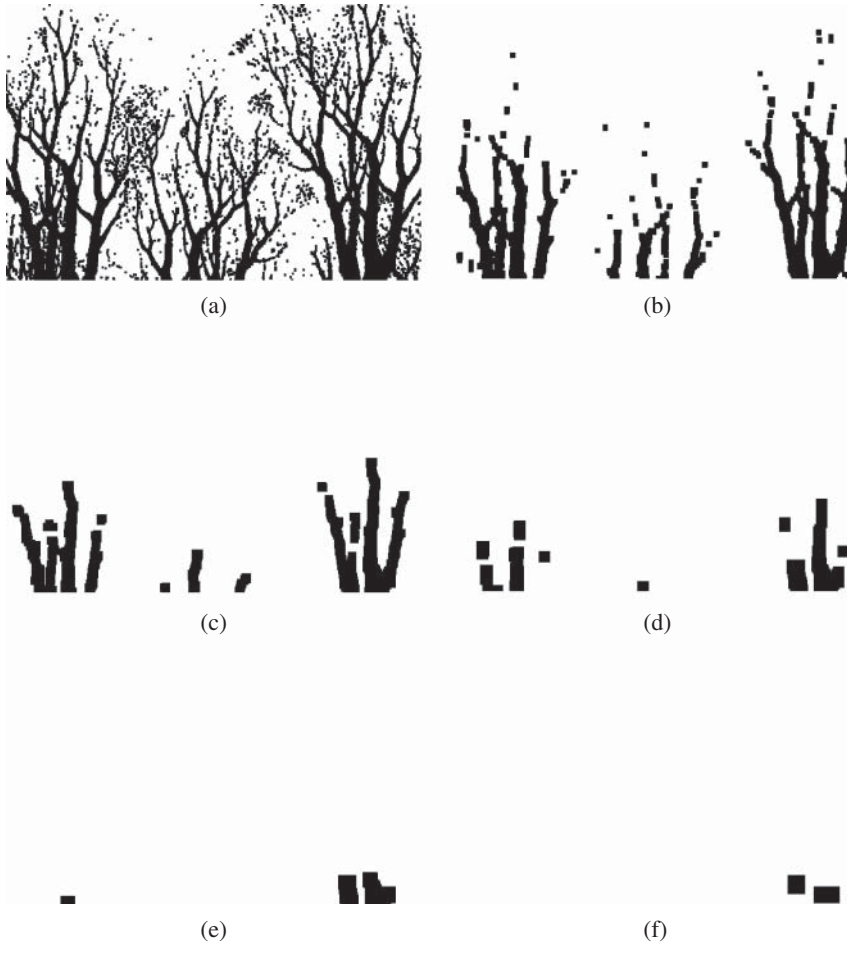
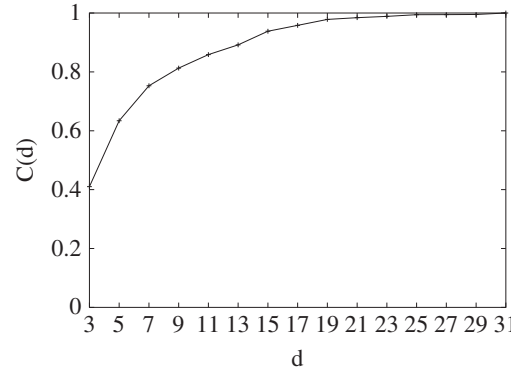


Figure 2.52 Some openings of image 2.1 with structuring elements of size (a) 3×3 ($d = 3$), (b) 7×7 ($d = 7$), (c) 11×11 ($d = 11$), (d) 15×15 ($d = 15$), (e) 19×19 ($d = 19$) and (f) 23×23 ($d = 23$).

Figure 2.53 The pattern spectrum for image 2.1.



Box 2.7 Formal definitions in mathematical morphology

In binary mathematical morphology, an image consists of pixels with values 0 and 1. Each pixel is represented by its position vector x. The centre of the image is taken to be the centre of the co-ordinate axes. "Object" is the set A of pixels x with value 1:

$$A \equiv \{x|\text{value\_of\_}x = 1\}. \tag{2.44}$$

The **complement** A^c of object A is effectively the background:

$$A^c \equiv \{x|x \notin A\}. \tag{2.45}$$

The **reflection** $-A$ of object A is the set of its pixels reflected about the origin of the axes:

$$-A \equiv \{-x|x \in A\}. \tag{2.46}$$

The **translation** of an object by a vector y is defined as:

$$A + y \equiv \{x + y|x \in A\}. \tag{2.47}$$

Then the **dilation** $\text{dil}(A, B)$ of object A by object B is denoted by $A \oplus B$ (or by $A + +B$ by some other authors), and is defined as:

$$\text{dil}(A, B) \equiv A \oplus B \equiv A + +B \equiv \bigcup_{y \in B} A + y \equiv \{x + y|x \in A, y \in B\}. \tag{2.48}$$

This operation is otherwise known as **Minkowski addition** of sets A and B.

The **erosion** $\text{er}(A, B)$ of object A by object B is denoted by $A \ominus B$ (or by $A - -B$ by some other authors), and is defined as:

$$\text{er}(A, B) \equiv A \ominus B \equiv A - -B \equiv \bigcap_{y \in B} A + y \equiv \{x|x + y \in A, y \in B\}. \tag{2.49}$$

This operation is otherwise known as **Minkowski subtraction** of set B from set A.

The properties of **commutativity** and **associativity** of dilation may be expressed as:

$$A \oplus B = B \oplus A \tag{2.50}$$

$$(A \oplus B) \oplus C = A \oplus (B \oplus C). \tag{2.51}$$

The property of successive erosions being equivalent to a single erosion by the two structuring elements dilated by each other may be written as:

$$(A \ominus B) \ominus C = A \ominus (B \oplus C). \tag{2.52}$$

The **duality** of the operations is formally expressed as:

$$A \oplus B = \left[A^c \ominus (-B)\right]^c \tag{2.53}$$

$$A \ominus B = \left[A^c \oplus (-B)\right]^c. \tag{2.54}$$

Example 2.20

Use mathematical morphology to define the outer and the inner boundary of the object in image 2.54a.

The solution is shown in the successive panels of Figure 2.54.

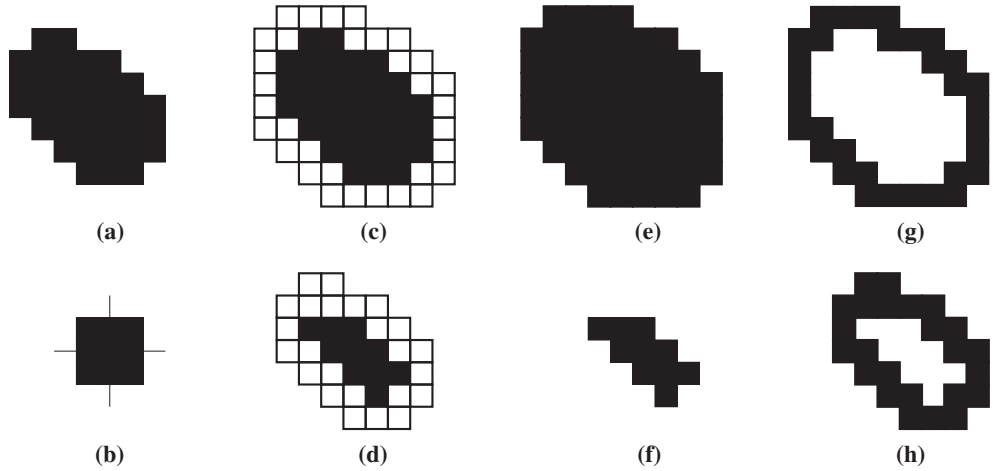


Figure 2.54 (a) The original image of an object. (b) The 3×3 structuring element used for these operations. (c) The delineated pixels will be added to the object if it is dilated by structuring element (b). (d) The delineated pixels will be removed from the object if it is eroded by structuring element (b). (e) The object dilated. (f) The object eroded. (g) The difference of the original object from its dilated version is the **outer object boundary**. (h) The difference of the eroded version of the object from the original object is the **inner object boundary**.

Example 2.21

Identify all 3×3 neighbourhood configurations for which the central pixel plays vital role in local connectivity.

There are 2^9 possible 3×3 binary configurations. In this problem we must consider only those which have the central pixel black, and which contain at least two other black pixels and at least two white pixels, for the question to be meaningful.

Figure 2.55 shows all configurations in which the central pixel is important for local connectivity. Note that, in all cases, if the central pixel is removed, the local connectivity is destroyed. (We assume that we work with 8-connectivity, i.e. a pixel is connected with another pixel even if they share only a vertex.)

(Continued)

Example 2.21 (Continued)

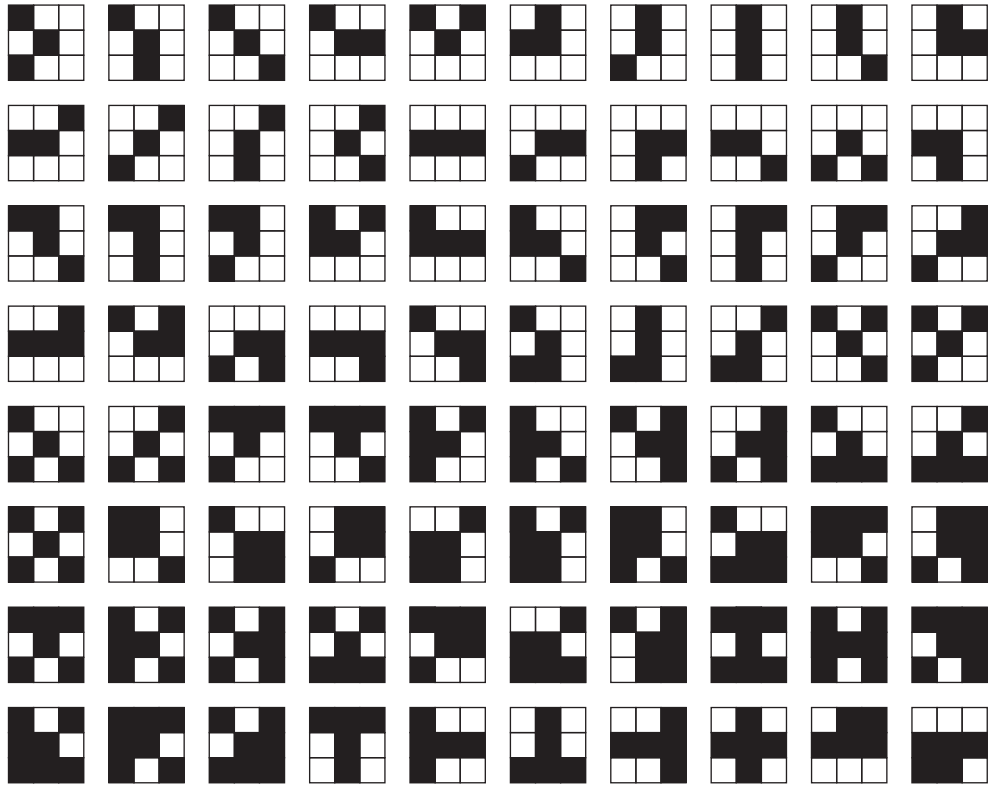


Figure 2.55 If the 3×3 neighbourhood of a pixel looks like any one of these configurations, the central pixel is important for the local connectivity of the shape.

Example B2.22

Imagine fitting circles inside an object. Every pixel of the object may belong to many such circles. For each pixel we keep the largest circle to which it belongs. The centres of these circles form the skeleton of the object, also known as medial axis. The skeleton of an object intuitively is understood as the thinnest possible version of the object that preserves its topological properties (i.e. it does not disconnect it into smaller parts or create holes that were not present in the original object) and consists of object pixels that are equidistant from at least two object boundary pixels. Use mathematical morphology to define the skeleton of the object in Figure 2.56a.

As opening removes "hairs" sticking out of an object, one may first think that by successive openings and subtractions with structuring element 2.34f, the skeleton of the object will be identified. This leads to the "naive" (and wrong) solution of the problem that has been presented in some books.

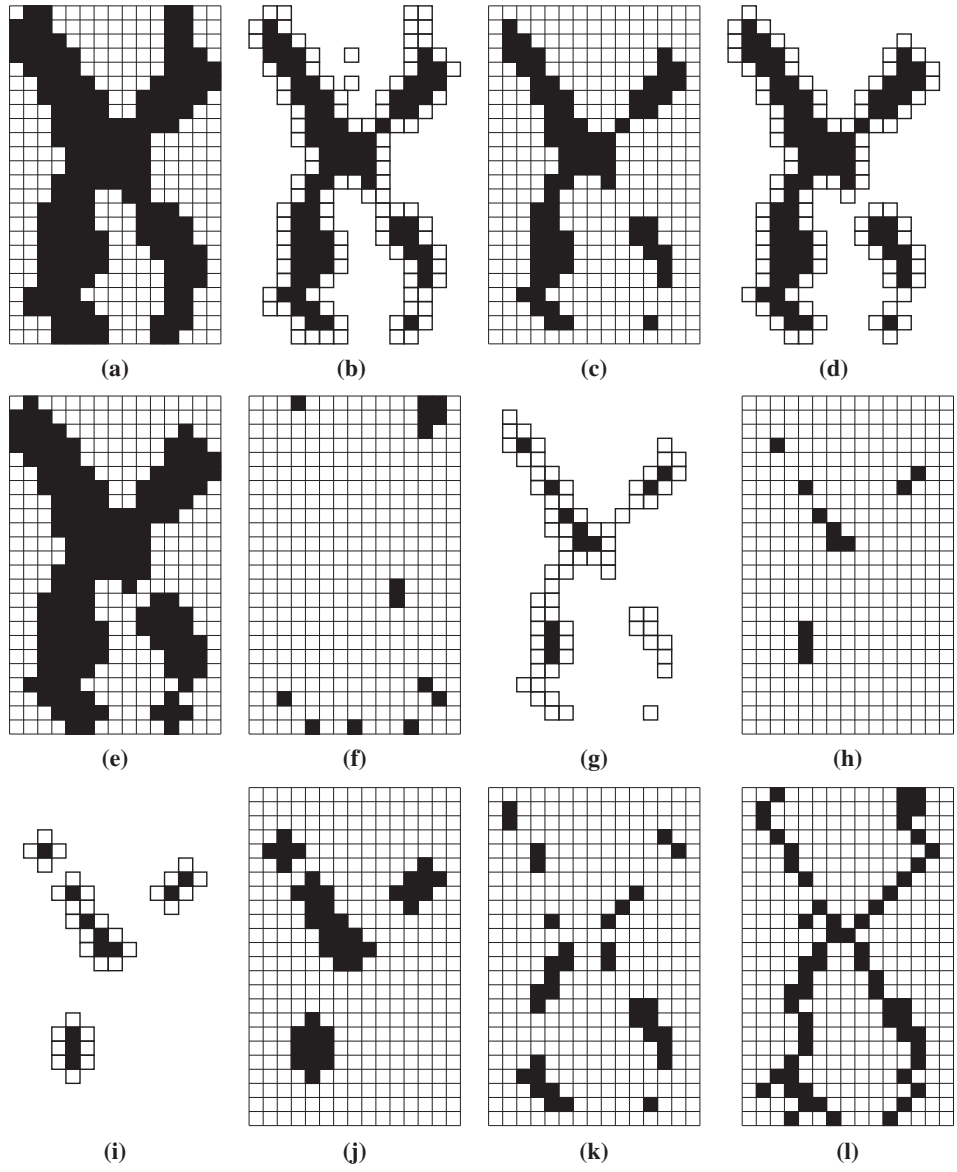


Figure 2.56 (a) The original image. (b) The delineated pixels will be removed. (c) The object eroded. (d) The delineated pixels will be added to the eroded image to produce the opening. (e) The original image opened. (f) The difference between the original image and its opened version. (g) The delineated pixels will be removed from (c). (h) The eroded image 2.56c. (i) The delineated pixels will be added. (j) The opened version of (c). (k) The difference between (c) and its opened version. (l) The final skeleton produced by adding images (f), (h) and (k). This is the wrong result.

Let us perform the following operations.

1. *Erode image 2.56a with structuring element 2.34f to obtain image 2.56c (Figure 2.56b marks the pixels that are removed from 2.56a).*

(Continued)

Example B2.22 (Continued)

2. *Dilate image 2.56c to obtain image 2.56e (the opening of 2.56a). Image 2.56d delineates the pixels added to 2.56c to perform the dilation.*
3. *Find the difference between the original image 2.56a and the opened one, to produce image 2.56f.*
4. *Erode the already once-eroded image 2.56c, by removing the pixels marked in image 2.56g, to produce image 2.56h.*
5. *Dilate image 2.56h by adding the pixels marked in 2.56i to produce the opened image of 2.56j.*
6. *Subtract 2.56j from the starting image 2.56c, to produce image 2.56k.*
7. *Note that if we try to repeat the loop, by opening image 2.56h which has already been eroded twice, we shall get nothing, so we stop here.*
8. *The skeleton is produced by superimposing the two residual images 2.56f and 2.56k, and the image that cannot further be eroded, 2.56h. This is shown in 2.56l.*

The above solution does not lead to a good skeleton. The major problem is the breaking down of connectivity. This is because we tried to erode the image from all four directions simultaneously by using structuring element 2.34f. A much more sensible approach is to proceed by eroding only from one direction at a time, with structuring elements that guarantee the preservation of local connectivity. This can be done by the use of the so-called ***hit-or-miss*** *transform that considers pairs of structuring elements and proceeds by finding pixels that fit both the image and its background. Figure 2.57 shows the pairs of elements we need.*

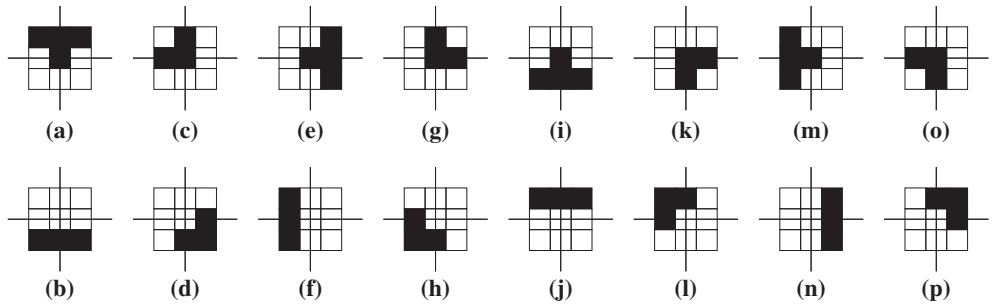


Figure 2.57 Pairs of elements that should be used to produce the skeleton of a binary image.

Figure 2.58a1 shows again the original image and Figure 2.58b1 its complement. Figure 2.58c1 shows the eroded image with structuring element 2.58f1, and 2.58d1, the eroded complement with structuring element 2.58g1. Note that, because element 2.58g1 has its centre outside the black region, when we try to see whether it fits inside the object 2.58b1, we must test all *points of the image, not just the black points (i.e. we must place the centre of the element at every point inside the frame either black or white and see whether the black part of the element is contained inside the black object in the image). Figure 2.58e1 marks the only pixel that is black in both 2.58c1 and 2.58d1, i.e. it is the intersection of the sets of marked (black) pixels in the two eroded images. In effect this is the pixel in the original image at which pattern 2.58h1 matches. This is the result of the hit-or-miss transform.*

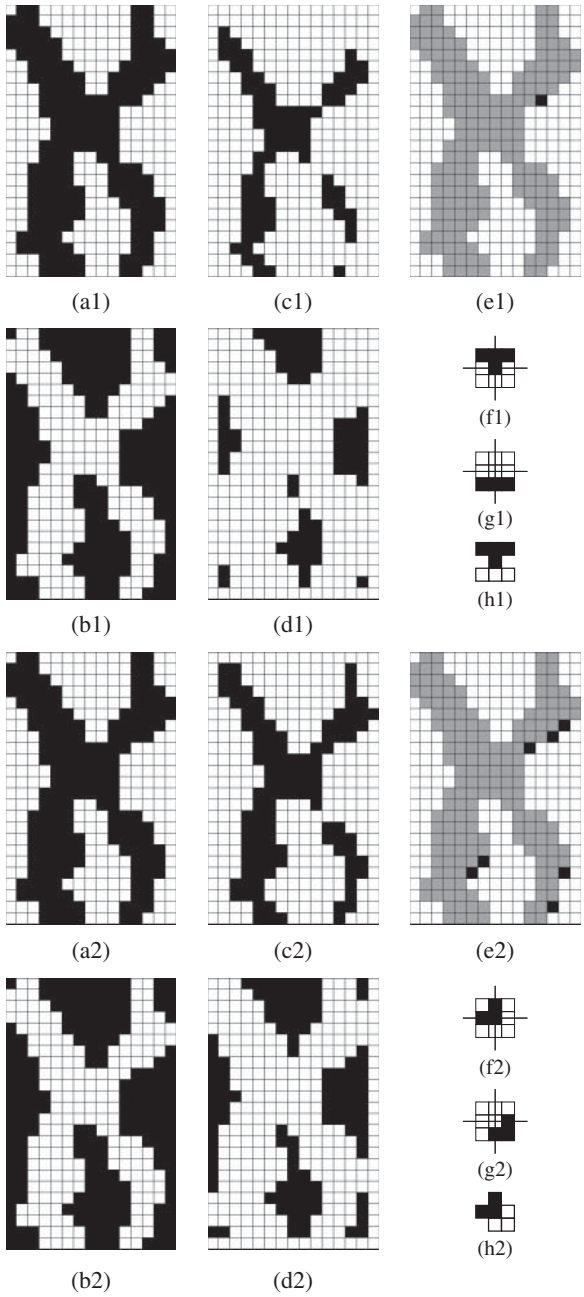


Figure 2.58 Panels (e1) and (e2) are the hit-or-miss transforms of images (a1) and (a2) produced with the pairs of structuring elements (f1)–(g1) and (f2)–(g2), respectively.

The black pixel in Figure 2.58e1 looks as if it were superfluous to the shape of the object, and it may be removed from the original object. After it is removed, we obtain object 2.58a2. Its complement is 2.58b2. Again we erode these two images with the pair of elements 2.58f2 and 2.58g2 respectively, to obtain their eroded versions 2.58c2 and 2.58d2. The common pixels in these two results are marked in 2.58e2, and they indicate the positions in the starting image 2.58a2 where

(Continued)

Example B2.22 (Continued)

pattern 2.58h2 fits. Figures 2.59–2.61 show the results of the hit-or-miss transform when the remaining pairs of elements of Figure 2.57 are applied in a cascaded way.

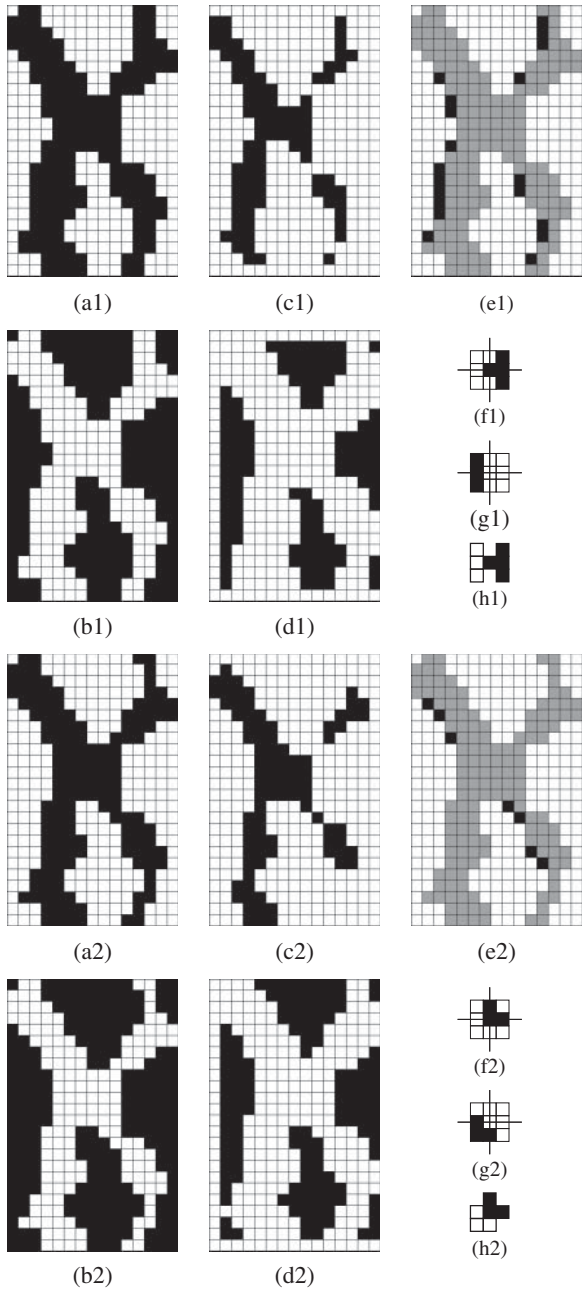


Figure 2.59 Images (a1) and (a2) eroded with structuring elements (f1) and (f2), produce panels (c1) and (c2), respectively.

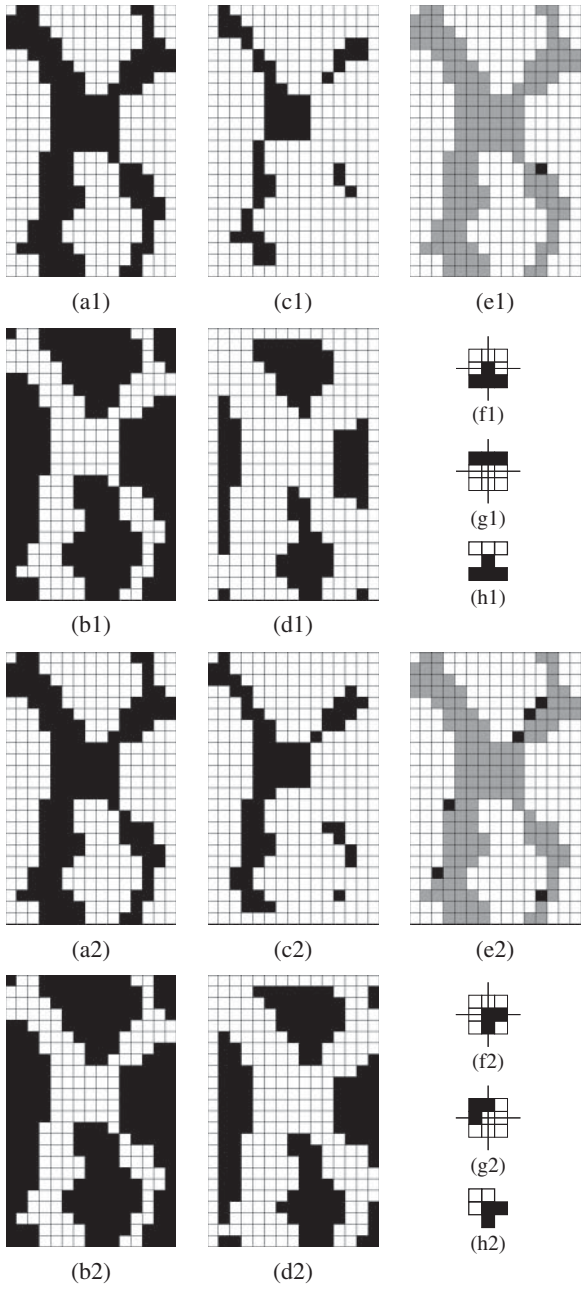


Figure 2.60 Panels (b1) and (b2) are the complements of (a1) and (a2) and they are eroded with elements (g1) and (g2) to produce panels (d1) and (d2), respectively.

(Continued)

Example B2.22 (Continued)

Figure 2.61 Panels (e1) and (e2) are the hit-or-miss transforms that mark the pixels in panels (a1) and (a2) where patterns (h1) and (h2) match, respectively.

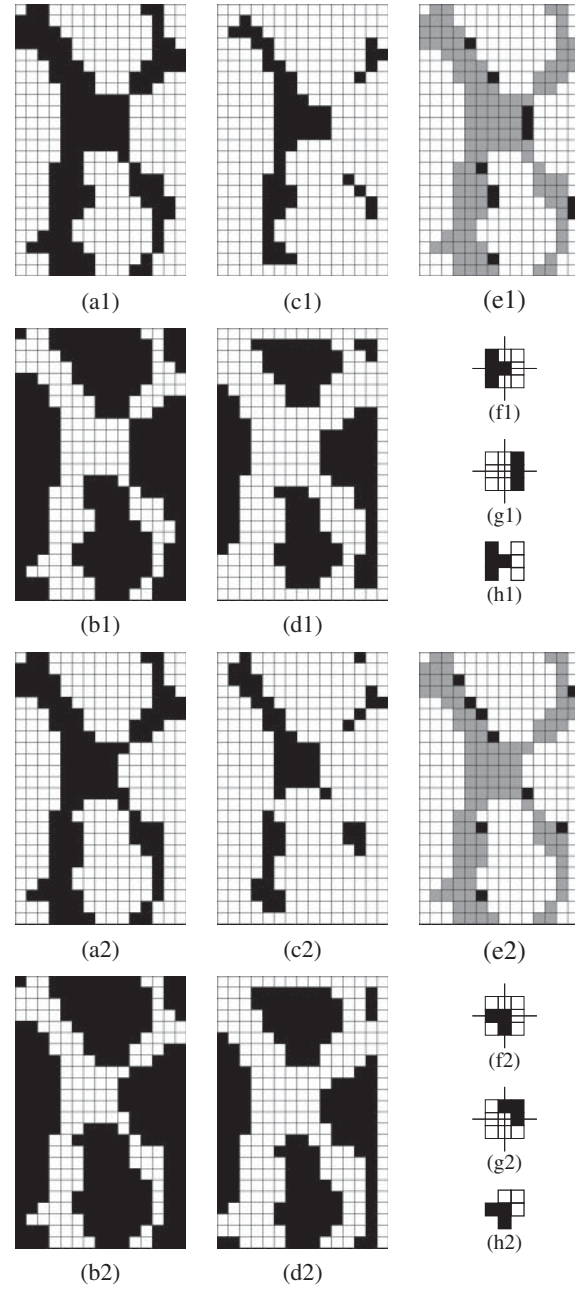


Figure 2.62 shows the successive skeletons of the object produced when elements 2.57 are applied for the second time. Finally, Figure 2.63 shows the remaining intermediate results until the final skeleton is produced. The process stops when all pairs of elements in Figure 2.57 are applied once one after the other and they have no effect on the input image. Note that again this skeleton is not perfect because of the superfluous pixels preserved where the object touches the image border. This is because we started producing the skeleton inside the minimum enclosing rectangle of the object in Figure 2.58a. If we had allowed a white border one pixel wide all around the object, all those artifacts would have not been present and the skeleton would have been perfect, as shown at the bottom right of Figure 2.63.

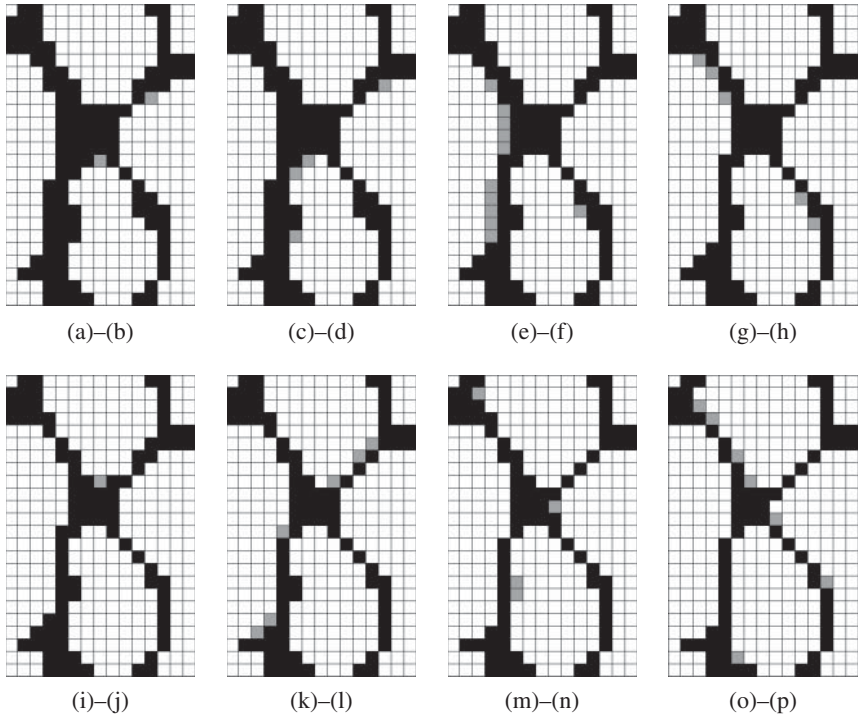


Figure 2.62 The successive thinning stages of the second cycle of applying the hit-or-miss transform to original image 2.58a, using the pairs of structuring elements of Figure 2.57. The starting image is the result produced by removing the black pixels identified in 2.61e2 from 2.61a2. Each panel has been produced from the previous (moving from left to right and from top to bottom), by applying the hit-or-miss transform with the pair of structuring elements from Figure 2.57 identified underneath the panel. The grey pixels are those that are removed at each stage.

(Continued)

Example B2.22 (Continued)

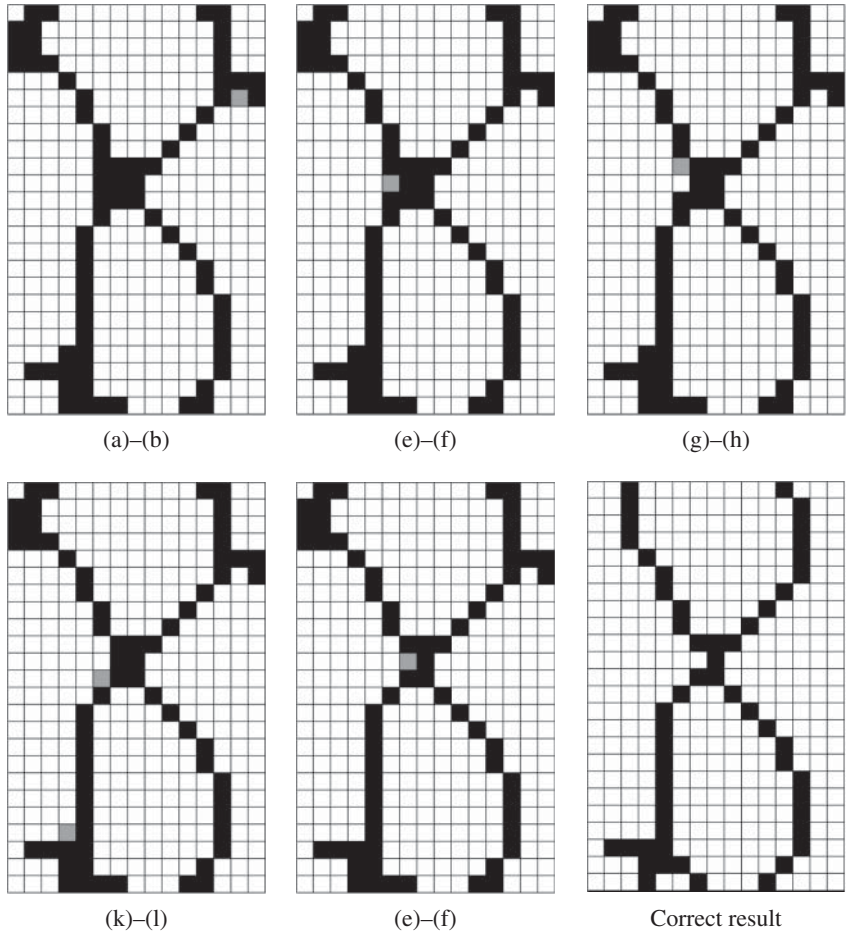


Figure 2.63 Applying successively the elements of Figure 2.57 in turn to intermediate result 2.62 (bottom right panel), eventually produces the skeleton of the object, after two more rounds. Note that some elements do not produce any change to the result of the previous stage, and those intermediate results have been omitted. The letters underneath each panel identify the pair of elements of Figure 2.57 used to produce it from the previous panel. The grey pixels are those that are removed at each stage. The bad behaviour of the algorithm where the skeleton touches the edge of the image is because we did not allow a single pixel border around the object before we started skeletonising it. If we add that border, the results we shall obtain is shown in the bottom right panel.

In general, the hit-or-miss transform may be used to identify pixels in a binary image where a particular local pattern is present. This is very useful in the 2D Boolean model we saw the section What is the specific convexity N^+? where we had to identify all pixels from which horizontal tangents to the grains could be drawn, by identifying the presence of a local configuration of black and white pixels (see Figure 2.19).

Box 2.8 Hit-or-miss algorithm

The hit-or-miss algorithm for skeletonisation is as follows.

Step 0: Make sure that the binary object you wish to skeletonise has a border of white pixels all around it. Call this O_0.
Step 1: Erode O_0 with structuring element 2.57a. Call the result O_a.
Step 2: Take the complement of the object, by turning all white pixels into black and all black pixels into white, and erode it with structuring element 2.57b. Call the result O_b.
Step 3: Find the common pixels between O_a and O_b, i.e. the pixels that are black in both images, and remove them from O_0, to form result O_1.
Step 4: Erode O_1 with structuring element 2.57c. Call the result O_c.
Step 5: Erode the complement of O_1 with 2.57d. Call the result O_d.
Step 6: Find the common pixels between O_c and O_d and remove them from O_1, to form result O_2.
Step 7: Erode O_2 with 2.57e. Call the result O_e.
Step 8: Erode the complement of O_2 with 2.57f. Call the result O_f.
Step 9: Find the common pixels between O_e and O_f and remove them from O_2, to form result O_3.
Step 10: Erode O_3 with 2.57g. Call the result O_g.
Step 11: Erode the complement of O_3 with 2.57h. Call the result O_h.
Step 12: Find the common pixels between O_g and O_h and remove them from O_3, to form result O_4.
Step 13: Erode O_4 with 2.57i. Call the result O_i.
Step 14: Erode the complement of O_4 with 2.57j. Call the result O_j.
Step 15: Find the common pixels between O_i and O_j and remove them from O_4, to form result O_5.
Step 16: Erode O_5 with 2.57k. Call the result O_k.
Step 17: Erode the complement of O_5 with 2.57l. Call the result O_l.
Step 18: Find the common pixels between O_k and O_l and remove them from O_5, to form result O_6.
Step 19: Erode O_6 with 2.57m. Call the result O_m.
Step 20: Erode the complement of O_6 with 2.57n. Call the result O_n.
Step 21: Find the common pixels between O_m and O_n and remove them from O_6, to form result O_7.
Step 22: Erode O_7 with 2.57o. Call the result O_o.
Step 23: Erode the complement of O_7 with 2.57p. Call the result O_p.
Step 24: Find the common pixels between O_o and O_p and remove them from O_7, to form result O_8.
Step 25: If O_8 is the same as O_0, exit. If not, set $O_0 = O_8$ and go to step 1.

What is the "take home" message of this chapter?

This chapter has dealt with binary patterns. Each such texture may be thought of as consisting of a collection of black primitive shapes placed in a 2D space. One has to quantify two processes: the way the shapes are placed in the 2D space and the shapes themselves. On the basis of that, we identified the following cases:

- Shapes that are fixed and finite in variety, placed according to fixed rules in the plane. Examples are the textures of Figures 2.2a, 2.2b, 2.3b, 2.4 and 2.24a. Most of such textures are human-made. They may be described by grammars which, however, are not unique descriptors. The same grammar may be used to describe totally different textures. All textures that may be described by the same grammar form a class of their own called **language**.
- Shapes that are fixed and finite in variety, placed according to randomly chosen rules. Examples are the textures of Figure 2.2d and both textures in Figure 2.13. These are semi-stochastic textures. A lot of them are also human-made. They may be described by stochastic grammars.
- Shapes that are not fixed and finite in variety, but that may be described parametrically, e.g. bubbles of radius r uniformly distributed in a certain range, or Gaussianly distributed with certain mean and standard deviation, placed according to a random process in the 2D space. Examples are the textures of Figures 2.17, 2.18 and 2.24b. Such textures may be described by the 2D Boolean model that estimates the parameters of the statistical processes involved.
- Shapes that are neither fixed, nor parameterisable, placed according to some unknown random process. If we know nothing about the binary texture, we try to capture some of its characteristics by applying binary mathematical morphology and various granulometries. Most natural textures belong to this category. Examples are the textures of Figures 2.1 and 2.2c. The 2D Boolean model may be applied to such textures with the understanding that it will only approximate the shapes of the blobs by fitting them with some parameterisable shapes. The 1D Boolean model may also be applied, without any claim in modelling even approximately the shapes of the granules of the texture.

It is worth noting that none of the above texture descriptors are one-to-one, i.e. none of them uniquely characterises each texture, except for two cases: when we are dealing with fixed shapes and fixed placement rules **and** the sequence by which the rules are applied is also quoted as part of the texture descriptor; and when the 2D Boolean model is fully applicable. The latter happens if the primitive patterns are either fixed or parametric with probability density functions of the shape parameters fully defined by the estimated individual parameters, and they are placed in the plane according to a random process, also fully defined by the estimated individual parameters. An example would be the case of a texture with its primitive shapes (grains) to be circles with

their radii Gaussianly distributed and placed according to a Poisson process. If the estimated individual parameters were the mean radius and the standard deviation of the radii of the grains, as well as the λ parameter of the Poisson distribution, then the texture would have been fully characterised, but **not** defined, since no two realisations of a random texture are expected to be exactly identical.

3

Stationary Grey Texture Images

What is a stationary texture image?

A stationary texture image is an image which contains a single type of texture, i.e. the same texture fills up the whole image and so its local statistical properties are the same everywhere in it. Some examples of stationary texture images are shown in Figure 3.1.

What is this chapter about?

This chapter is about the methodologies commonly used to characterise stationary grey texture images. The image processing problem one tries to solve here is known as **texture classification**: one seeks to identify a few numbers, the so-called **texture features**, that capture the characteristics of a particular texture, so they may be used to identify the texture, even when the texture appears in an image not seen previously. This problem is often related to the following set-up. One has a catalogue of textures, i.e. a database of reference images. Each texture in the database is characterised by a set of parameters, the texture features. When a new texture image comes along, one computes the same features from it, and by comparing them with the values of the corresponding features in the database, tries to recognise the unknown texture.

This chapter will cover: grey scale mathematical morphology, fractals and multifractals, statistical features, including histograms, the Weibull distribution, the variogram, run-length and co-occurrence matrices, Fourier-based features, Gibbs distributions and Markov random fields.

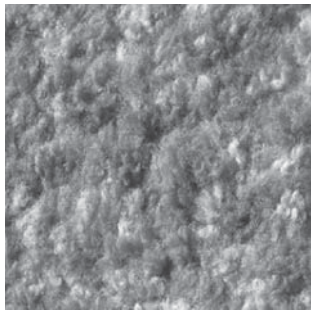
Are any of the methods appropriate for classifying binary textures useful for the analysis of grey textures?

Yes. There are two ways by which we may make use of the techniques developed for binary textures to deal with grey textures.

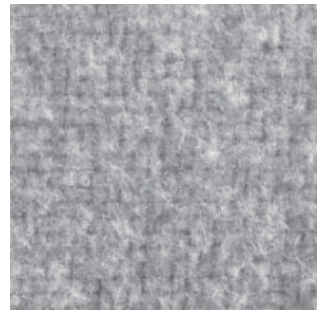
1. Some of the techniques are directly generalisable for grey images. These are, for example, the **mathematical morphology** based techniques.
2. A grey image may be analysed into a set of binary images either by **thresholding** or **bit-slicing**. Then texture features may be derived for each binary image in the set and be used to characterise the grey image.

Image Processing: Dealing with Texture, Second Edition. Maria Petrou and Sei-ichiro Kamata.

Companion Website: www.wiley.com/go/kamataText2

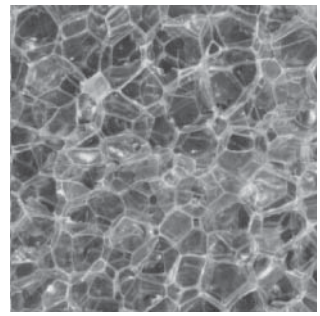


(a) Cloth 1



(b) Cloth 2

(c) Food



(d) Plastic

(e) Fabric 1

(f) Fabric 2

(g) Beans

(h) Sand

Figure 3.1 Some stationary grey texture images. Images (a), (c), (e), (f) and (g) © Massachusetts Institute of Technology. Images (b), (d) and (h) © University of Southern California. Reproduced with permission.

3.1 Image Binarisation

How may a grey image be analysed into a set of binary images by thresholding?

If the range of grey values present in an image is $[G_{\min}, G_{\max}]$, we may choose M values $t_1, ..., t_M$ such that $G_{\min} < t_1 < ... < t_M < G_{\max}$, and analyse the image into a stack of $M+1$ binary slices, by marking black in the first binary slice all pixels which have grey value in the range $[G_{\min}, t_1)$, in the second binary slice all pixels with grey value in the range $[t_1, t_2)$, and so on. One may choose to have as many binary slices as one wishes, up to the maximum value equal to the number of grey levels present in the image. This, however, would be impractical. A few binary slices are usually adequate. Figure 3.2a shows an original 256×256 image and Figures 3.2b–3.2i show its analysis into eight binary slices. The original image had grey values in the range [28,234] as one can see from the histogram of the image shown in Figure 3.2j. The thresholds used were 31, 63, 95, 127, 159, 191 and 223, and they are marked with thin vertical lines in the histogram. It is possible that if the thresholds are chosen at random, some of the slices will end up with very few pixels. This is the case, for example, of slice 1, which corresponds to the range of grey values $[0, 31)$.

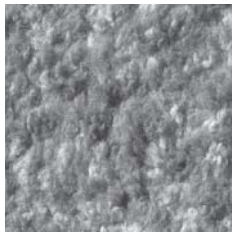
How can we deal with the problem of unequal number of pixels in each range of grey values?

We may plot the histogram of the image first and choose the thresholds so that all ranges of grey values contain the same number of pixels. Let us say that the image contains N^2 pixels. If we want to divide it into eight equally populated slices, each of them must contain $N^2/8$ pixels. We construct the histogram of the image, and then we start from the leftmost bin and start adding the values of the subsequent bins, until we reach roughly the value $N^2/8$. The grey value of the bin before which we stopped is our first threshold t_1. We proceed this way until we define all our thresholds. Figure 3.3 shows the eight slices into which the original image 3.2a may be analysed in this way. The thresholds chosen for this slicing were 77, 92, 103, 113, 123, 135 and 152 and they are marked with the thin vertical lines in the histogram of the image in Figure 3.3j.

Alternatively, we may first apply histogram equalisation (see the book Image Processing, The Fundamentals [75]) to the image and then apply thresholding. Figure 3.4 shows the result of applying histogram equalisation with random additions and the slices we obtain using the same thresholds used for the construction of the binary images in Figure 3.2. Histogram equalisation with random additions introduces some arbitrariness to the process. However, any effect this might have on the image is masked by the binarising process. That is why one cannot see any difference between these results and those of Figure 3.3.

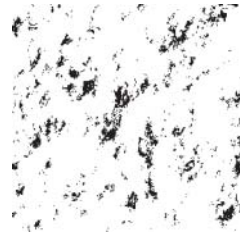
How may a grey image be analysed into a set of binary images by bit-slicing?

The most commonly used grey images take grey values in the range [0,255]. This means that each grey value is represented by 8 bits. In bit slicing, we use one binary slice for each bit, i.e. 8 slices in total. Each pixel is marked in all those binary slices where its corresponding bit has value 1. For example, let us consider a pixel with grey value 176. This number may be written as $176 = 128 + 32 + 16 = 2^7 + 2^5 + 2^4$. Therefore, its binary representation is 10110000. ($10110000 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 176$). This pixel, therefore, will be marked in the binary planes that represent the first, the third and the fourth bits, and in no other plane. Figure 3.5 shows the analysis of an image in its bit planes. It can be seen that the binary images

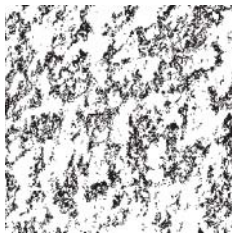


(a) Original image

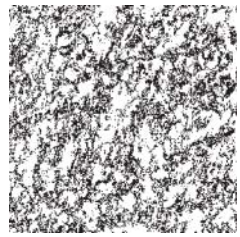
(b) Slice 1



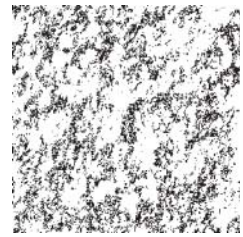
(c) Slice 2



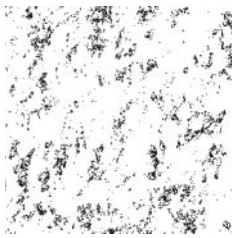
(d) Slice 3



(e) Slice 4



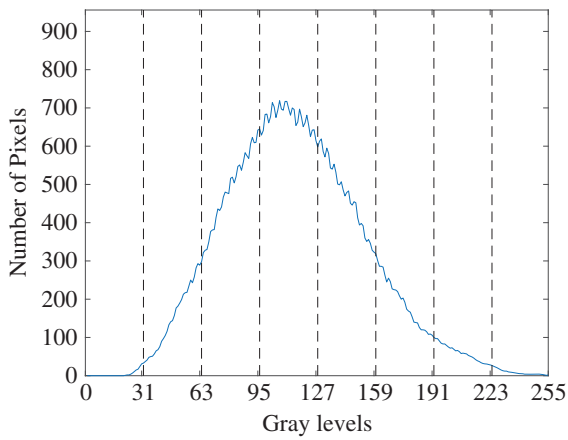
(f) Slice 5



(g) Slice 6

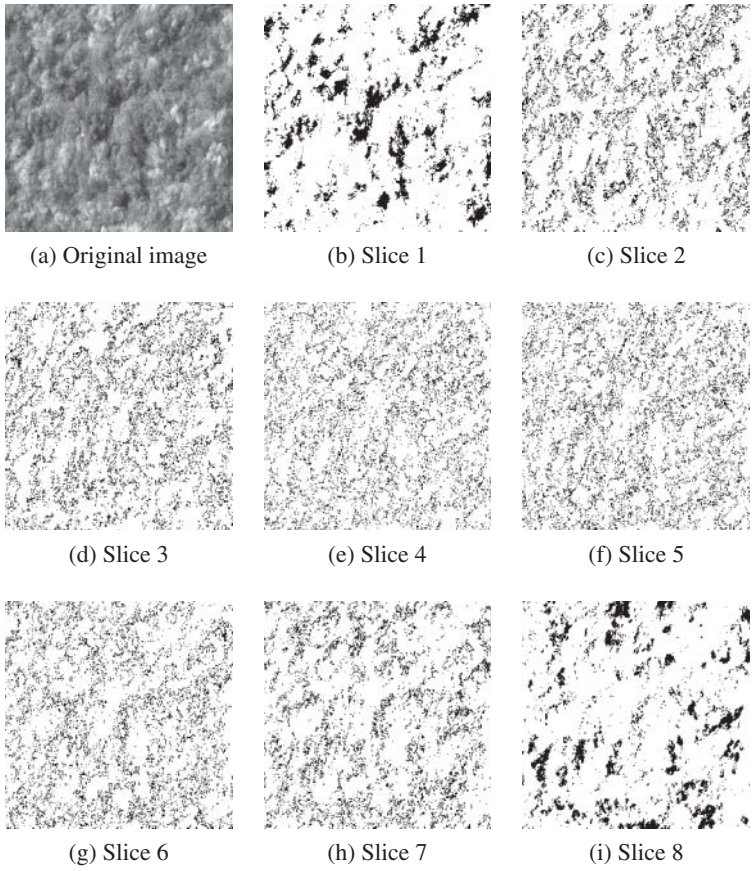
(h) Slice 7

(i) Slice 8



(j) Histogram of the image

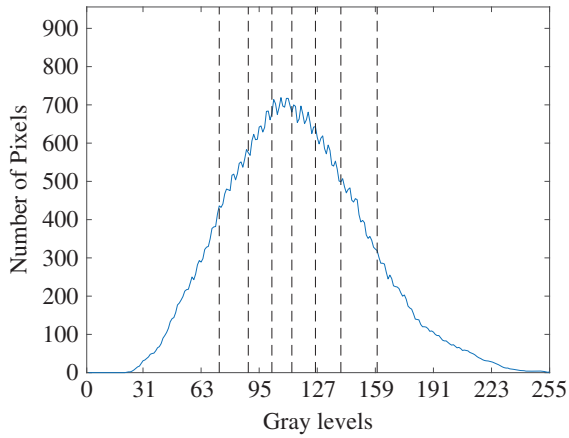
Figure 3.2 Splitting a grey image into a set of binary images by thresholding. (a) Original image. (b)–(i) Binary images obtained by thresholding. Each one of them flags with black the pixels that have grey values in the range between two successive thresholds. Source: Maria Petrou. (j) Histogram of the original image showing the thresholds used.



(a) Original image (b) Slice 1 (c) Slice 2

(d) Slice 3 (e) Slice 4 (f) Slice 5

(g) Slice 6 (h) Slice 7 (i) Slice 8

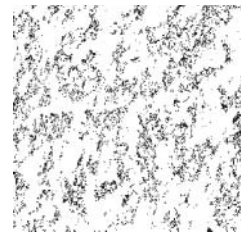


(j) Histogram of the image

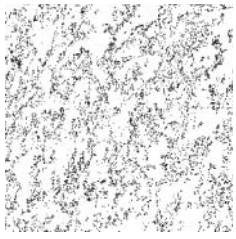
Figure 3.3 Splitting an image into a set of binary images all of which have approximately the same number of black pixels. (a) Original image. (b)–(i) Binary images obtained by thresholding. Source: Maria Petrou. (j) Histogram of the original image. The vertical lines show the threshold values used.

(a) Equalised image

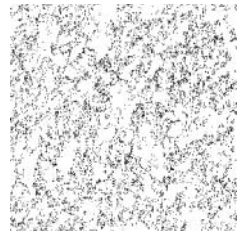
(b) Slice 1



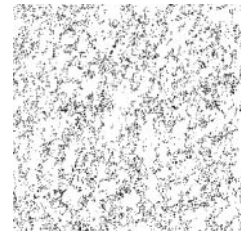
(c) Slice 2



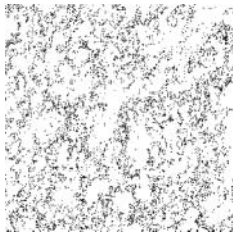
(d) Slice 3



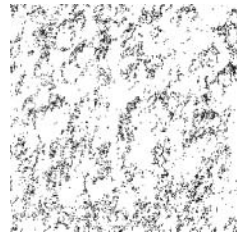
(e) Slice 4



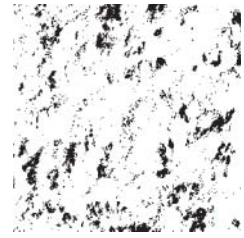
(f) Slice 5



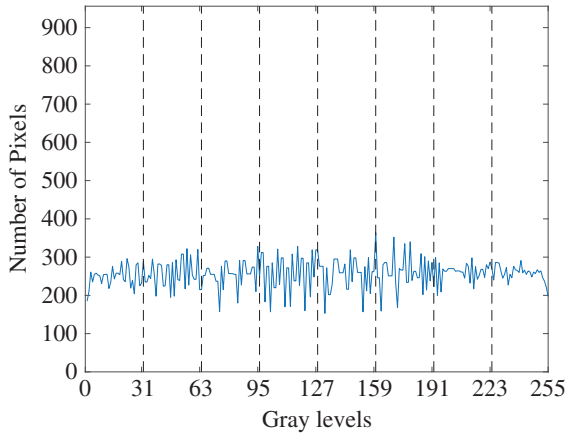
(g) Slice 6



(h) Slice 7

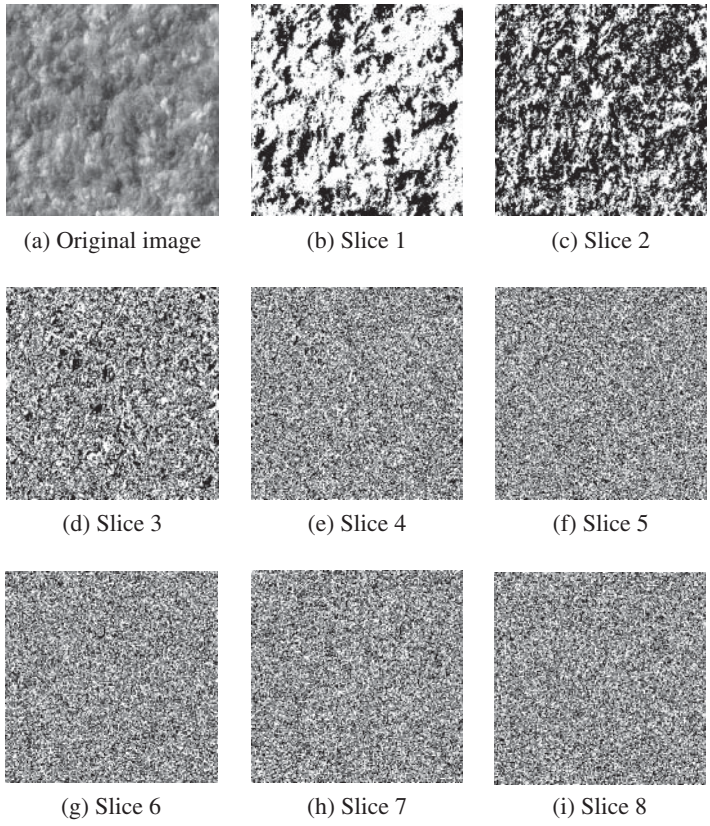


(i) Slice 8



(j) Histogram of the image

Figure 3.4 Splitting an image into a set of binary images after histogram equalisation. (a) Original image after histogram equalisation with random additions. Source: Maria Petrou. (b)–(i) Binary images obtained by thresholding. Source: Maria Petrou. (j) Histogram of the equalised image. The vertical lines show the threshold values used.



(a) Original image (b) Slice 1 (c) Slice 2

(d) Slice 3 (e) Slice 4 (f) Slice 5

(g) Slice 6 (h) Slice 7 (i) Slice 8

Figure 3.5 Splitting an image into a set of binary images by bit-slicing. (a) Original image. (b)–(i) Bit slices, from the most significant bit to the least significant bit, respectively. Source: Maria Petrou.

that correspond to the last few bits do not seem to contain any useful information. As example 3.1 shows, the division in bit planes does not have any physical meaning. It is rather a mathematical tool dictated by the binary technology we use and not by the nature of objects we observe or the human visual system. This, however, does not mean that one may not be able to construct useful features that characterise an image from such binary slicing.

Is there any relationship between the binary planes produced by thresholding and the bit planes?

There is no immediate relationship. However, one may isolate bit planes with flagged pixels that have grey values in a single coherent range. Let us define the subtraction of two sets A and B as follows:

$$A - B \equiv \{x; x \in A \text{ and } x \notin B\}. \tag{3.1}$$

Note that according to this definition, we do not worry whether B contains extra elements that are not in A. We just ignore them. For example, if $A = \{10, 12, 13, 24, 27, 28\}$ and $B = \{12, 24, 27, 28, 30, 32\}$, $A - B = \{10, 13\}$.

Then if we use B_i to indicate the set of flagged pixels in the binary plane which corresponds to the ith bit, starting from the most significant, we can see that

$B_2 - B_1$ flags all pixels with grey value in the range $[64, 127]$
$B_3 - B_2 - B_1$ flags all pixels with grey value in the range $[32, 63]$
$B_3 - B_2$ flags all pixels with grey value in the ranges $[32, 63]$ and $[160, 191]$
$B_3 - B_1$ flags all pixels with grey value in the ranges $[32, 63]$ and $[96, 127]$.

From this we can see that binary planes that result from thresholding may be created as combinations of the bit planes of the image, and therefore some operations that are applied to thresholded versions of the image may be performed directly on the bit planes and thus implemented in hardware.

Example 3.1

Calculate the ranges of grey values contained by each binary slice in the case of an 8-bit grey image.

The first binary slice contains all those pixels that have 1 in the first bit place, irrespective of what value they have in the other bit places. Let us represent the value of a pixel in binary form by $A_1A_2A_3A_4A_5A_6A_7A_8$*, where* A_i *takes either value 0 or 1. After we have identified the pixels with* $A_1 = 1$*, we consider the remaining digits. The smallest value the remaining digits can make is* $0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 0$*, and the largest is* $1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 127$*. If* $A_1 = 1$*, the grey levels that are flagged in this binary slice are those in the range* $[1 \times 2^7 + 0, 1 \times 2^7 + 127]$*, i.e. grey levels in the range* $[128, 255]$.

The second binary slice contains all those pixels that have $A_2 = 1$*. We may again compute the smallest and the largest value that the bits on the right of this bit can make: the smallest value digits* $A_3A_4A_5A_6A_7A_8$ *can make is 0, if all of them are 0, and the largest is* $1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 63$*. Given that* $A_2 = 1$*, we may then say that the smallest and largest values digits* $A_2 \ldots A_8$ *can make are* $1 \times 2^6 + 0$ *and* $1 \times 2^6 + 63$*, respectively, i.e.* 64 *and* 127*. Now, there are two cases for digit* A_1*: it is either 0, in which case the grey value of the pixel is in the range* $[64, 127]$*, or 1, in which case the grey value of the pixel is in the range* $[1 \times 2^7 + 64, 1 \times 2^7 + 127]$*, i.e.* $[192, 255]$*. So, this bit plane flags all those pixels that have grey values in the ranges* $[64, 127]$ *and* $[192, 255]$.

The third binary slice flags all those pixels that have $A_3 = 1$*. The minimum and maximum numbers made up by the remaining digits to the right are 0 and 31, respectively.* $A_3 = 1$ *means we have to add* $2^5 = 32$ *to these numbers. If we consider the digits* A_1 *and* A_2 *to the left, we have four possible combinations in binary form: 00, 01, 10, 11. These correspond to the numbers 0,* $2^6 = 64$*,* $2^7 = 128$ *and* $2^7 + 2^6 = 192$*, respectively. If we add to each one of these numbers the minimum and the maximum numbers of the rest of the digits, we have all ranges of grey values that are flagged in this bit plane:* $[32, 63]$, $[96, 127]$, $[160, 191]$, $[224, 255]$.

In a similar way we may compute the ranges of grey values flagged in each of the remaining binary planes. Note that as we move towards the less significant bits, the ranges flagged together become more fragmented and narrower. For example, the last bit plane, the one that flags pixels with $A_8 = 1$, flags all pixels that have an odd grey value. Table 3.1 gives the ranges of grey values flagged in each bit plane.

Table 3.1 Pixels with grey values in the ranges on the right are flagged in the corresponding bit plane (slice) identified on the left.

Slice	Flagged ranges of grey values
1	[128,255]
2	[64,127] [192,255]
3	[32,63] [96,127] [160,191] [224,255]
4	[16,31] [48,63] [80,95] [112,127]
	[144,159] [176,191] [208,223] [240,255]
5	[8,15] [24,31] [40,47] [56,63] [72,79] [88,95]
	[104,111] [120,127] [136,143] [152,159] [168,175]
	[184,191] [200,207] [216,223] [232,239] [248,255]
6	[4,7] [12,15] [20,23] [28,31] [36,39] [44,47] [52,55] [60,63]
	[68,71] [76,79] [84,87] [92,95] [100,103] [108,111] [116,119]
	[124,127] [132,135] [140,143] [148,151] [156,159] [164,167]
	[172,175] [180,183] [188,191] [196,199] [204,207] [212,215]
	[220,223] [228,231] [236,239] [244,247] [252,255]
7	[2,3] [6,7] [10,11] [14,15] [18,19] [22,23] [26,27] [30,31]
	[34,35] [38,39] [42,43] [46,47] [50,51] [54,55] [58,59] [62,63]
	[66,67] [70,71] [74,75] [78,79] [82,83] [86,87] [90,91] [94,95]
	[98,99] [102,103] [106,107] [110,111] [114,115] [118,119]
	[122,123] [126,127] [130,131] [134,135] [138,139] [142,143]
	[146,147] [150,151] [154,155] [158,159] [162,163] [166,167]
	[170,171] [174,175] [178,179] [182,183] [186,187] [190,191]
	[194,195] [198,199] [202,203] [206,207] [210,211] [214,215]
	[218,219] [222,223] [226,227] [230,231] [234,235] [238,239]
	[242,243] [246,247] [250,251] [254,255]
8	All pixels with odd number grey values

$\text{Dil(A)}_{\text{B}} = \{0\,1\,1\,0\,1\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,1\,0\}$ $\quad A = \{0\,1\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\}$

Wrong: $\{1\,1\,0\,1\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,1\,0\,0\}$ $\quad B = \{\mathbf{1}\ 1\}$

$\text{Er(A)}_{\text{B}} = \{0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,\mathbf{1}\,1\,1\,0\,0\,0\}$ $\quad -B = \{1\ \mathbf{1}\}$

Figure 3.6 A binary sequence A may be dilated with the asymmetric structuring element B by placing the centre of B (indicated in bold) at all samples of A with value 1 and giving value 1 to all samples covered by the structuring element. If instead of doing that we place the centre of B at **every** point of the sequence and take the maximum value of the samples covered by B and assign it to the sample under its centre, we shall get the wrong result, shown underneath the correct dilation result. If, however, we do that using the reflection $-$B of B, we shall get the right answer. Note that erosion of A by B may be produced either by placing the centre of B at every sample with value 1 in A and, if B does not fit inside the sequence of 1s, the sample under its centre is turned to 0, or by placing the centre of B at **every** sample of A and taking the minimum of the sample values covered by B and assigning it to the sample under the centre of B.

3.2 Grey Scale Mathematical Morphology

How is mathematical morphology generalised for grey images?

Mathematical morphology for binary images assumes that an object pixel has value 1 and a background pixel has value 0. When dilation is performed, the structuring element is placed on the top of the image, with its centre at an object pixel. All covered pixels then are turned into object pixels, i.e. they are given value 1, i.e. they are turned into black in the pictorial representation of the process. One may express this by saying that we place the reflected structuring element at **all** pixels (black or white) of the image and we assign to the pixel covered by the centre of the structuring element the maximum value of the pixels covered by the structuring element:

$$\text{New\_value\_at\_}(i,j) = \max(\text{all\_pixels\_covered\_by\_refl\_str\_el\_centred\_at\_}(i,j)). \tag{3.2}$$

Note that if the structuring element is symmetric with respect to its centre, the word "reflected" in the above statement is meaningless. This is made clear in Figure 3.6. In texture analysis we use structuring elements that are symmetric about their centre, so reflection is not usually an issue.

Erosion may be defined as the process of placing the centre of the structuring element at **all** pixels of the image and assigning to the pixel covered by the centre of the structuring element the minimum value of the pixels covered by the structuring element:

$$\text{New\_value\_at\_}(i,j) = \min(\text{all\_pixels\_covered\_by\_str\_el\_centred\_at\_}(i,j)). \tag{3.3}$$

These definitions may be directly generalised to grey images, where the minimum and maximum is computed inside a window that plays the role of the structuring element. Such a structuring element is said to be **flat**.

Example 3.2

Perform erosion, dilation, opening and closing to the image of Figure 3.7a, using a flat structuring element of size 3×3**.**

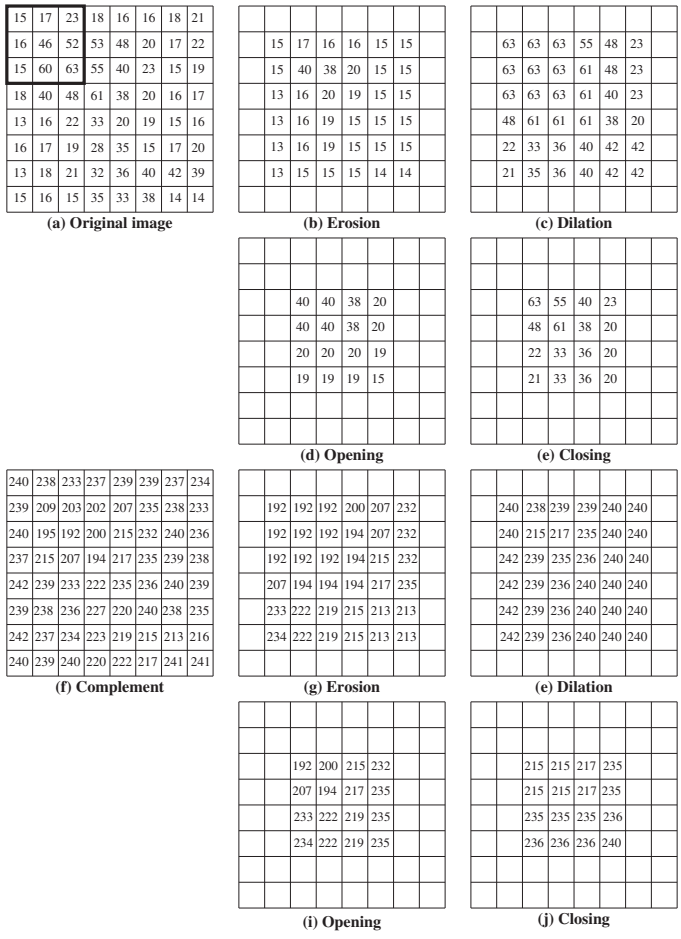


Figure 3.7 (a) A grey image. The highlighted 3×3 frame indicates the structuring element that is used to scan the image. The min and max grey value inside this 3×3 window is computed at each position and assigned to the central pixel in order to create the erosion (shown in (b)) and the dilation (shown in (c)) of the image, respectively. Erosion followed by dilation produces the opening of the image, shown in (d), and dilation followed by erosion produces the closing of the image, shown in (e). (f) The complement of image (a). (g) The erosion of (f). (h) The dilation of (f). (i) The opening of (f). (j) The closing of (f).

This is demonstrated in Figures 3.7a–e.

How is the complement of an image defined for grey images?

The complement of the image is defined by subtracting the grey value of each pixel from its maximum possible value.

Example 3.3

Apply erosion, dilation, opening and closing to the complement of image 3.7a, using a flat structuring element of size 3×3**.**
The complement of image 3.7a is obtained by subtracting its grey values from 255. The result is shown in Figure 3.7f. Its erosion, dilation, opening and closing with a flat structuring element of size 3×3 *are shown in 3.7g–3.7j, respectively.*

What is a non-flat structuring element?

A non-flat structuring element is one that has grey values in its pixels. Erosion using such an element is performed in the same way as with the flat element, except that, as the structuring element slides across the image, at each location, before the minimum is obtained, the values of the structuring element are subtracted from the corresponding grey values of the image underneath. Dilation using such an element is performed in the same way as with the flat element, except that as the structuring element slides across the image, at each location, before the maximum is obtained, the values of the structuring element are added to the corresponding grey values of the image underneath.

Example 3.4

Perform dilation and erosion on the 1D signal of Figure 3.8a with the structuring element of Figure 3.8b, i.e. a structuring element with values $(1, 2, 1)$**.**
The dilation of signal 3.8a with structuring element 3.8b is shown in Figure 3.8c. To produce the dilation, the values of the structuring element are added to the signal values and the maximum between the three consecutive sums is picked up and assigned to the central position.

The erosion of signal 3.8a with structuring element 3.8b is shown in Figure 3.8d. In order to produce the erosion, the negative structuring element of Figure 3.8e is added to the three consecutive samples around each sample position and the minimum is taken.

In 3.8c and 3.8d the grey parts are some selected places where the addition of the structuring element is shown explicitly.

Figure 3.9 shows the results of dilation and erosion of the same signal with a flat structuring element for comparison. The grey areas here represent the difference between the eroded or dilated signal and the original signal.

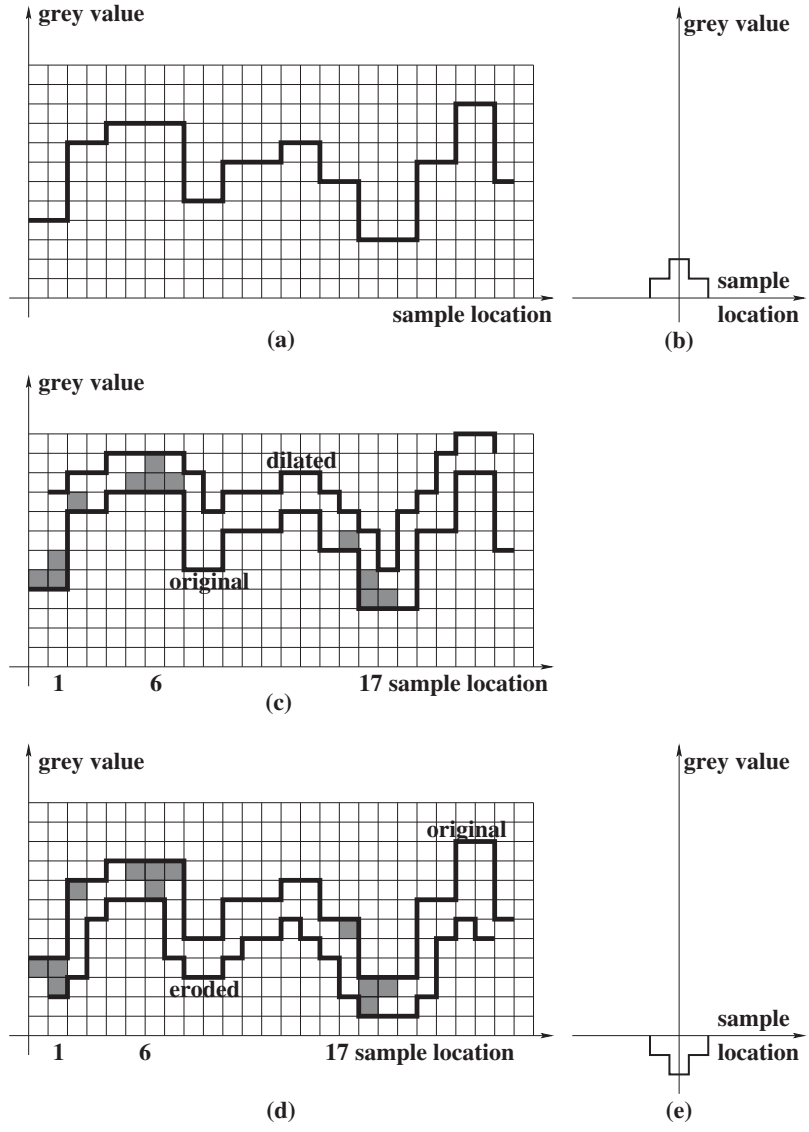


Figure 3.8 (a) The original signal. (b) The structuring element. (c) The top line is the signal dilated. (d) The bottom line is the signal eroded. (e) The negative of the structuring element used to perform the erosion. In (c) and (d) the grey patches show how the structuring element is placed at selected positions 1, 6 and 17. For dilation, the maximum (and for erosion the minimum) over three consecutive positions is identified and assigned to the central position.

(Continued)

Example 3.4 (Continued)

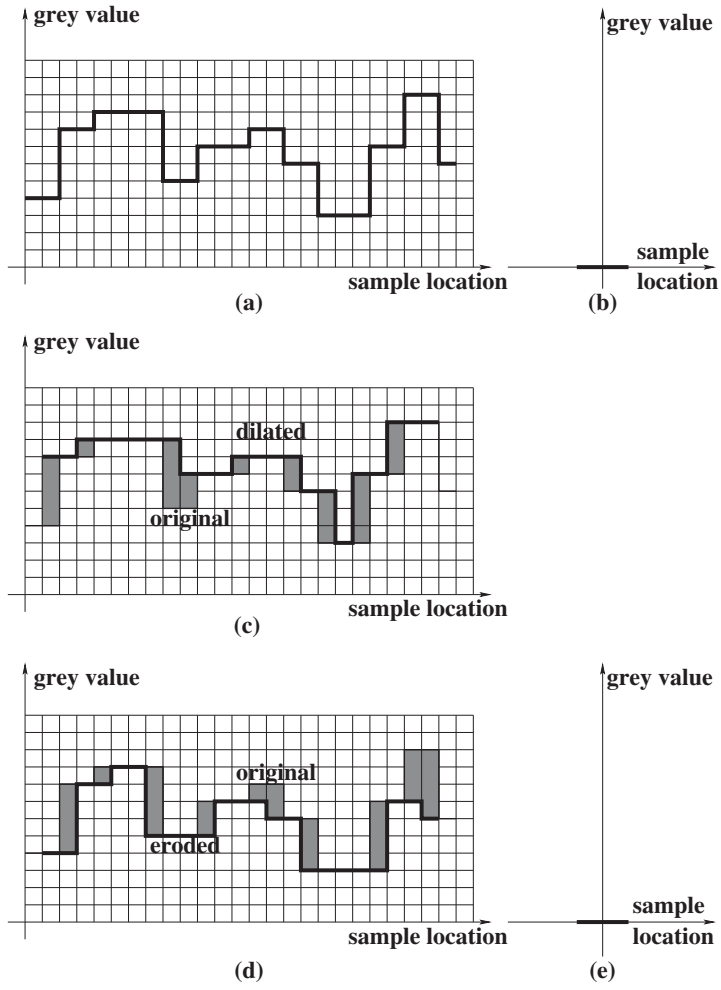


Figure 3.9 (a) The original signal. (b) The flat structuring element of size 3. (c) The top line is the signal dilated. (d) The bottom line is the signal eroded. (e) The "negative" of the structuring element used to perform the erosion (the same as the original one since it is flat). The grey patches show the difference between the original signal and the result.

Example 3.5

Perform erosion, dilation, opening and closing to the image of Figure 3.7a, and its complement, by using a 3×3 structuring element with value 2 in the middle and 1 everywhere else.

The results are shown in Figure 3.10.

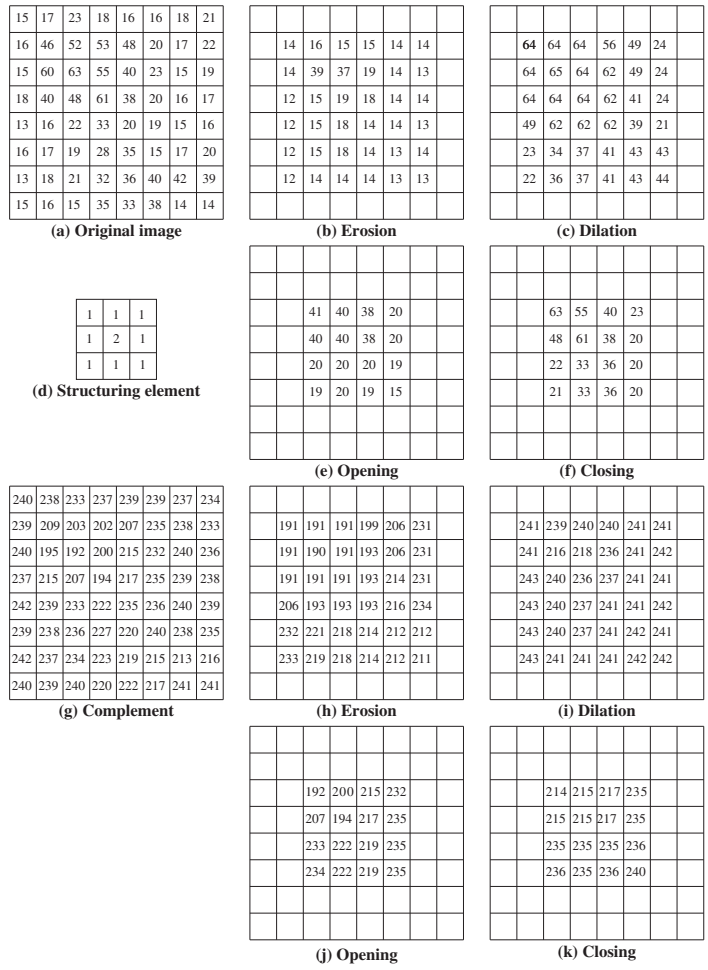


Figure 3.10 (a) The original image. (b) The image eroded by subtracting at each position the value of the structuring element, shown in (d), before taking the minimum. (c) The image is dilated by adding at each position the value of structuring element (d) before taking the maximum. (e) The opening of the image, obtained by dilating (b), i.e. by adding at each position the value of the structuring element and taking the maximum inside each 3×3 window. (f) The closing of the image obtained by eroding (c). (g) The complement of (a) obtained by subtracting its values from 255. (h) Erosion of (g) by (d). (i) Dilation of (g) by (d). (j) Opening of (g) by (d). (k) Closing of (g) by (d).

What is the relationship between the morphological operations applied to an image and those applied to its complement?

For an image A and a symmetric structuring element, the following relations hold:

$$\left(A^{\text{dilated}}\right)^{\text{complement}} = \left(A^{\text{complement}}\right)^{\text{eroded}} \tag{3.4}$$

$$\left(A^{\text{eroded}}\right)^{\text{complement}} = \left(A^{\text{complement}}\right)^{\text{dilated}} \tag{3.5}$$

$$\left(A^{\text{open}}\right)^{\text{complement}} = \left(A^{\text{complement}}\right)^{\text{close}} \tag{3.6}$$

$$\left(A^{\text{close}}\right)^{\text{complement}} = \left(A^{\text{complement}}\right)^{\text{open}}. \tag{3.7}$$

Example 3.6

Assuming that the maximum grey value a pixel can take is G, prove that the erosion of an image is equal to the complement of the dilation of its complement, if the structuring element is symmetric about its centre.

Let us consider a structuring element consisting of M pixels, with values $g_1, g_2, \dots, g_M$. Let us consider a random position of the image where the structuring element is currently placed. Let us say that the grey values of the pixels of the image which are covered by the structuring element are $x_1, x_2, \dots, x_M$. To perform erosion, we compute:

$$\min\{x_1 - g_1, x_2 - g_2, \dots, x_M - g_M\}. \tag{3.8}$$

The complement of the image at the same position will have values $G - x_1, G - x_2, \dots, G - x_M$. To perform the dilation of the complement with the same structuring element, we compute:

$$\max\{G - x_1 + g_1, G - x_2 + g_2, \dots, G - x_M + g_M\}. \tag{3.9}$$

This may be written as:

$$\max\{G - (x_1 - g_1), G - (x_2 - g_2), \dots, G - (x_M - g_M)\}. \tag{3.10}$$

*This now makes it obvious that the minimum value obtained in (3.8) will be the same as the maximum value obtained in (3.10), and therefore that the erosion of an image is the same as the complement of the dilation of its complement, performed using the **same** structuring element.*

Note that this is not valid if the structuring element is not symmetric about its centre, because different pixels will be involved in the erosion and different in the dilation, since for the dilation the structuring element will have to be reflected.

Example 3.7

Demonstrate, using the results of Figure 3.10, that the erosion of an image is equal to the complement of the dilation of its complement, and that the opening of an image is equal to the complement of the closing of its complement and vice versa.

As we have defined the complement of an image to be its difference from a flat image with values 255 everywhere, to show that one image is the complement of another we must show that if we add them we get a flat image with values 255. This is demonstrated in Figure 3.11, where, for example, we add panels (b) and (i) of Figure 3.10 in order to show that erosion of the image with a 3×3 flat structuring element is the complement of the dilation of the complement of the image with the same structuring element.

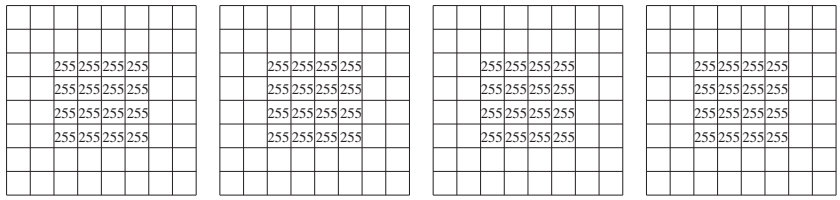
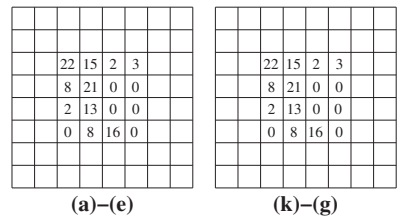


Figure 3.11 Underneath each panel one can see which panels of Figure 3.10 were added to produce the result shown. (b)+(i): The erosion of an image is equal to the complement of the dilation of its complement. (c)+(h): The dilation of an image is equal to the complement of the erosion of its complement. (e)+(k): The opening of an image is equal to the complement of the closing of its complement. (f)+(j): The closing of an image is equal to the complement of the opening of its complement.

Example 3.8

Demonstrate, by using the results of Figure 3.10, that if one subtracts the opening of an image from the original image, one obtains the same result as when one subtracts the complement image from the closing of the complement of the image.
This is demonstrated in Figure 3.12.

Figure 3.12 Underneath each panel one can see which panels of Figure 3.10 were subtracted to produce the result shown.



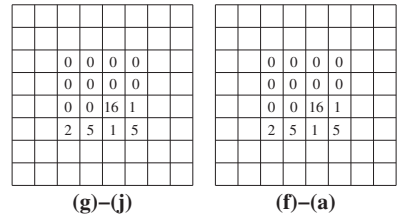
Example 3.9

Demonstrate, by using the results of Figure 3.10, that if one subtracts an image from its closing, one obtains the same result as when one subtracts the opening of the complement image from the complement of the image.
This is demonstrated in Figure 3.13.

(Continued)

Example 3.9 (Continued)

Figure 3.13 Underneath each panel one can see which panels of Figure 3.10 were subtracted to produce the result shown.



What is the purpose of using a non-flat structuring element?

Let us consider the case of erosion with a structuring element with values $g_1, g_2, \ldots, g_c, \ldots, g_M$ in its M positions. Let us also consider the values $x_1, x_2, \ldots, x_c, \ldots, x_M$ of the image covered by the structuring element when the centre of the structuring element is placed at pixel c, with value x_c. Erosion means that we replace x_c with the minimum of values $\{x_1 - g_1, x_2 - g_2, \ldots, x_c - g_c, \ldots, x_M - g_M\}$. Let us assume that the minimum is $x_d - g_d$, where d is one of the pixels covered by the structuring element, other than pixel c. The replacement only takes place if

$$x_d - g_d < x_c - g_c \Rightarrow g_c - g_d < x_c - x_d. \tag{3.11}$$

The significance of this is the following: a local minimum with value x_d wins and changes the value x_c of a neighbouring pixel, by making it $x_d - g_d$, only if it is "deeper" than pixel c by more than $g_c - g_d$. (The word "deeper" is only strictly correct if $g_c > g_d$.)

In a similar way, we may work out that a local maximum wins and replaces the value of a neighbouring pixel, only if it is "sticking above" it by more than $g_c - g_d$. (The term "sticking above" again is only strictly correct if $g_c > g_d$.)

So, the use of grey values for the structuring element serves the purpose of identifying only significant local minima or local maxima that are worth influencing the values of their neighbours. At the same time, we see that the actual values of the structuring element do not matter, but only their relative values. So, for practical purposes, we shall obtain the same results if we remove a constant value from all values of the structuring element. Therefore, it is advisable to use structuring elements with at least one of their values 0. This value serves as a reference level for all other values that have meaning only in relative terms.

Example 3.10

Remove the bias of structuring element Figure 3.10d, and then use it to compute the erosion and dilation of the image of Figure 3.10a. Thus, show that the results are the same as those in Figures 3.10b and 3.10c, apart from the bias.

The bias of the structuring element is removed if we subtract from all its elements 1, in order to make at least one of its elements have value equal to 0. Figure 3.14 shows the results obtained by using this structuring element. Note that the difference from results 3.10b and 3.10c is equal to the removed bias of the structuring element.

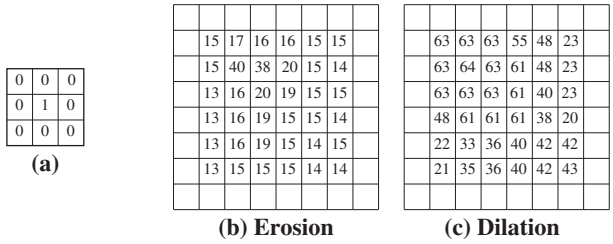


Figure 3.14 (a) Structuring element 3.10d with the bias removed. (b) Erosion and (c) Dilation of image 3.10a, obtained with this new structuring element.

How can we perform granulometry with a grey image?

Once we have generalised the definitions of erosion and dilation for grey images, we can directly generalise the granulometric approach to grey texture analysis. In most situations a flat structuring element is used, and opening is performed with structuring elements of various sizes to produce the pattern spectrum (see Figure 2.53). Here, instead of counting black pixels, like we did in the case of binary images, we subtract the result of opening from the original image pixel by pixel and add the differences.

Example 3.11

Compute a number that expresses how much of the original signal of Figure 3.8a is in peaks narrower than three samples.

We perform erosion with a flat structuring element of size 3, followed by dilation with the same structuring element. This way we obtain the opening of the signal. Then we subtract the result from the original signal. Figure 3.15a shows the erosion of the original signal.

The grey patches show the difference between the original signal and its eroded version. Figure 3.15b shows the dilation of the eroded signal, i.e. the opening of the original signal. The grey patches show the difference between the opened signal and the eroded one. To deal with boundary effects we leave out two samples at either end of the original signal. If we count the grey boxes within these limits in Figure 3.15a, we see that erosion reduced the original signal by 28 boxes. If we count the grey boxes in 3.15b we see that dilation increased the eroded signal by 19 boxes. So, the original signal was reduced by 9 boxes, because it had peaks narrower than three samples occupying 9 boxes.

(Continued)

Example 3.11 (Continued)

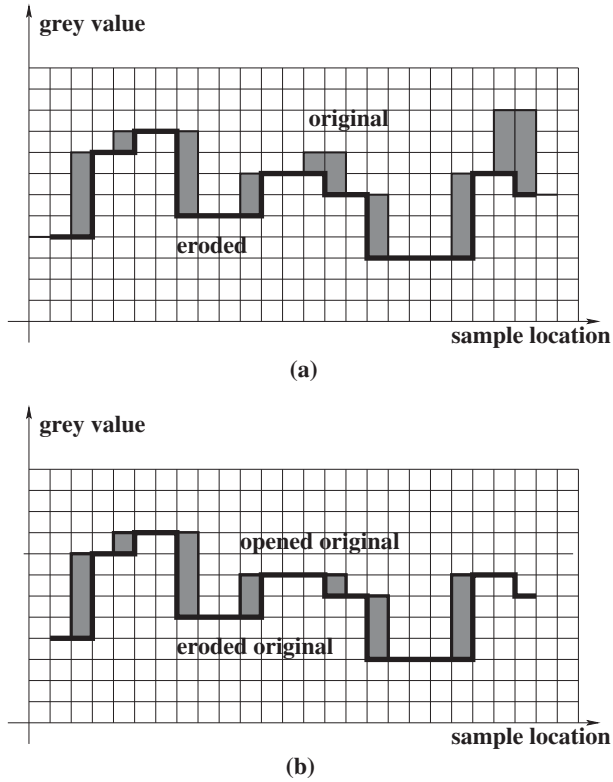


Figure 3.15 (a) The signal of Figure 3.8a eroded with a flat structuring element of size 3. (b) The eroded signal dilated with a flat structuring element of size 3, to produce the opening of the original signal of Figure 3.8a. The grey patches show the difference between the result and the signal to which the operation was applied.

Example 3.12

Compute the number that expresses how much detail the signal of Figure 3.8a has in valleys narrower than three samples.

We perform dilation with a flat structuring element of size 3, followed by erosion with the same structuring element. This way we obtain the closing of the signal. Then we subtract the result from the original signal.

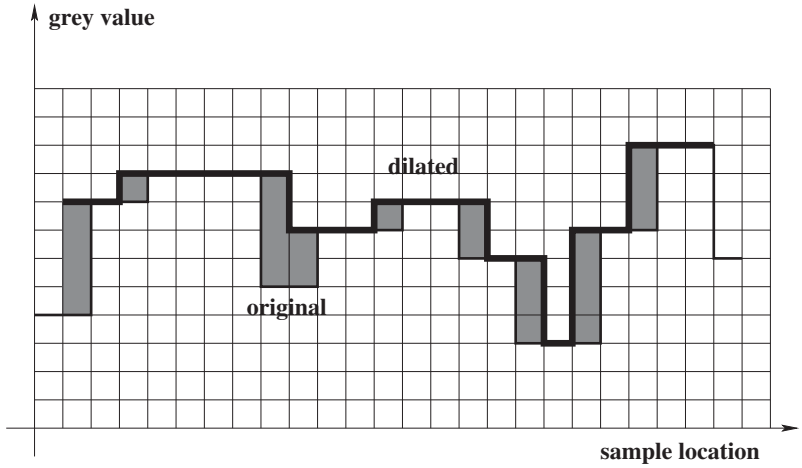
Figure 3.16a shows the dilation of the original signal. The grey patches show the difference between the original signal and its dilated version.

Figure 3.16b shows the erosion of the dilated signal, i.e. the closing of the original signal. The grey patches show the difference between the closed signal and the dilated one.

To deal with boundary effects we leave out two samples at either end of the original signal.

If we count the grey boxes within these limits in Figure 3.16a, we see that dilation increased the original signal by 24 boxes. On the other hand, counting the grey boxes in 3.16b we see that erosion decreased the dilated signal by 16 boxes.

So, the original signal was increased by 8 boxes, because it had valleys narrower than three samples occupying 8 boxes.



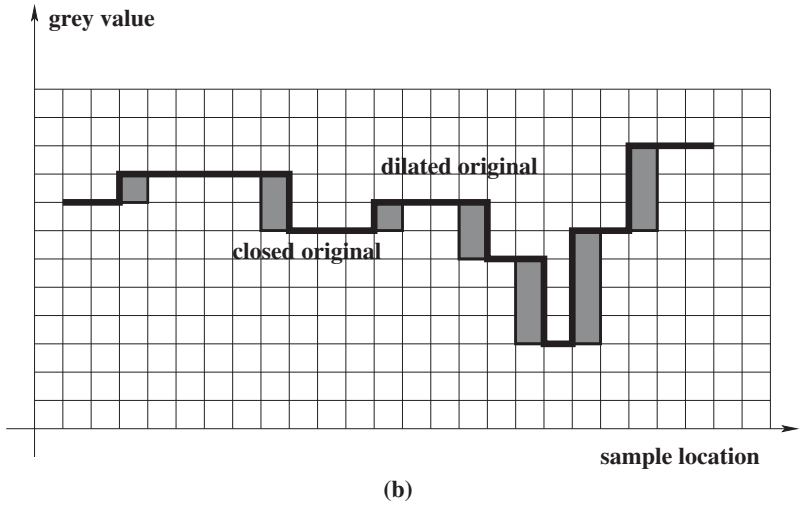


Figure 3.16 (a) The signal of Figure 3.8a dilated by a flat structuring element of size 3. (b) The dilated signal eroded by a flat structuring element of size 3, to produce the closing of the original signal of Figure 3.8a. The grey patches show the difference between the result and the signal to which the operation was applied.

Can we extract in one go the details of a signal, peaks or valleys, smaller than a certain size?

Yes, this can be done by the so-called **blanket method**.

What is the blanket method?

One dilates and erodes the original signal with a flat structuring element and then takes the difference of the two results pixel by pixel and adds the differences to produce a single number. The process is repeated for structuring elements of various sizes in order to produce the pattern spectrum.

Example 3.13

Produce the pattern spectrum of the signal of Figure 3.8a for structuring elements of size 3, 5 and 7 using the blanket method.

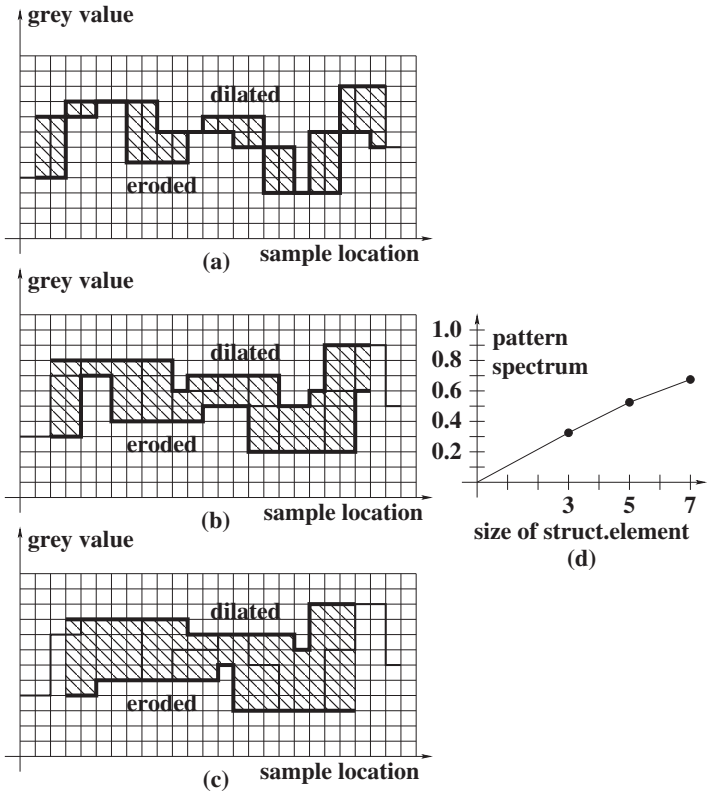


Figure 3.17 (a) The signal of Figure 3.8a dilated and eroded by a flat structuring element of size 3. (b) The same signal dilated and eroded by a flat structuring element of size 5. (c) The same signal dilated and eroded by a flat structuring element of size 7. In all cases the thick line enveloping the graph from the top is the dilation, and the thick line enveloping the graph from the bottom is the erosion. The shaded boxes indicate the difference between erosion and dilation. Their numbers normalised by the number of boxes that make up the signal (excluding those near the borders) are plotted in (d) to form the pattern spectrum.

The result is shown in Figure 3.17. Counting boxes in (a) gives 52 shaded boxes that constitute details smaller than three samples. The total number of boxes occupied by the signal itself (excluding those under the first and last samples) is 158. This gives as value of the pattern spectrum for structuring element of size 3: $52/158 = 0.329$. *For structuring element of size 5 we get* $76/144 = 0.528$ *and for size 7:* $87/126 = 0.690$. *The pattern spectrum of this signal is shown in Figure 3.17d.*

How can we deal with border effects without losing part of the image?

Let us say that the structuring element is of size $(2M + 1) \times (2M + 1)$ and the image is of size $N \times N$. We repeat the first M columns of the image to the left, reflecting them with respect to the left image border. We also repeat the last M columns of the image to the right, reflecting them about the right image border. Then we reflect the top M rows of the image to the top, reflecting them about the top image border, and the bottom M rows of the image to the bottom, reflecting them about the bottom image border. We apply then the morphological operations to the enlarged image, leaving its border pixels unprocessed. Figure 3.18 demonstrates this process for a 7×7 image and a 5×5 ($M = 2$) structuring element. As erosion and dilation involve taking minimum and maximum values over windows, this repetition of the pixels near the border does not affect the result, assuming that we are using a symmetric structuring element.

On the contrary, if we either assume 0 values for the pixels outside the image border, or assume toroidal boundary conditions, where the image is wrapped around in four directions, artifacts will occur.

How do we apply the blanket method in practice?

Perform the following process for various values of M, using a symmetric structuring element.

Step 0: Sum up the grey values of the input image I, to produce number A.

Step 1: Enlarge the image by reflection, as explained in Figure 3.18, so that, if the structuring element is of size $(2M + 1) \times (2M + 1)$ and the image is of size $N \times N$, the enlarged image is of size $(N + 2M) \times (N + 2M)$. Call this image I_0.

Step 2: Dilate image I_0 with the structuring element by placing the centre of the structuring element at all pixels of the image, but leaving a border of M unprocessed pixels all around. Produce output image I_d, of size $N \times N$.

1	0	2	3	1	4	0
5	4	3	1	2	0	2
3	2	0	1	4	5	3
4	1	2	5	3	1	0
3	5	2	4	1	3	5
0	1	2	3	0	3	4
2	1	3	0	1	5	3

4	5	5	4	3	1	2	0	2	2	0
0	1	1	0	2	3	1	4	0	0	4
0	1	1	0	2	3	1	4	0	0	4
4	5	5	4	3	1	2	0	2	2	0
2	3	3	2	0	1	4	5	3	3	5
1	4	4	1	2	5	3	1	0	0	1
5	3	3	5	2	4	1	3	5	5	3
1	0	0	1	2	3	0	3	4	4	3
1	2	2	1	3	0	1	5	3	3	5
1	2	2	1	3	0	1	5	3	3	5
1	0	0	1	2	3	1	3	4	4	3

Figure 3.18 Enlargement of the image in order to deal with boundary effects when applying erosion or dilation.

Step 3: Erode image I_0 with the structuring element, by placing the centre of the structuring element at all pixels of the image, but leaving a border of M unprocessed pixels all around. Produce output image I_e, of size $N \times N$.

Step 4: Subtract I_e from I_d, pixel by pixel.

Step 5: Take the sum of all pixel values of image $I_d - I_e$. Call this number B.

Step 6: Calculate B/A to work out the value of the pattern spectrum of the original image for structuring element of size $(2M + 1) \times (2M + 1)$.

Example 3.14

Perform erosion, dilation, opening and closing to the images of Figure 3.1 using flat structuring elements of sizes 3×3, 7×7 **and** 11×11**. Then plot their pattern spectra.**

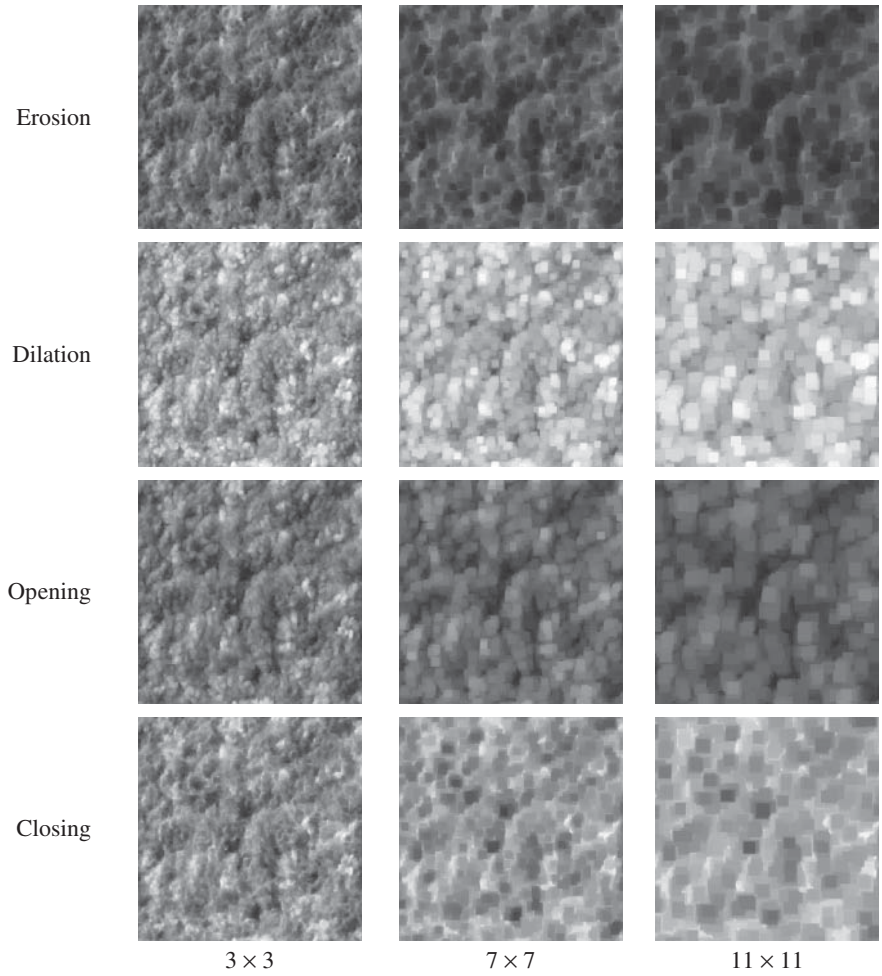


Figure 3.19 Morphological operations applied to the image of Figure 3.1a with structuring elements of sizes indicated at the bottom of each column. Note the blocky effect created by the use of a square structuring element instead of a round one. Source: Maria Petrou.

Figure 3.19 shows the results of the morphological operations for the various structuring elements, for image 3.1a. Figure 3.20 shows the pattern spectra of six of these images, computed using structuring elements of sizes 3×3 *to* 21×21.

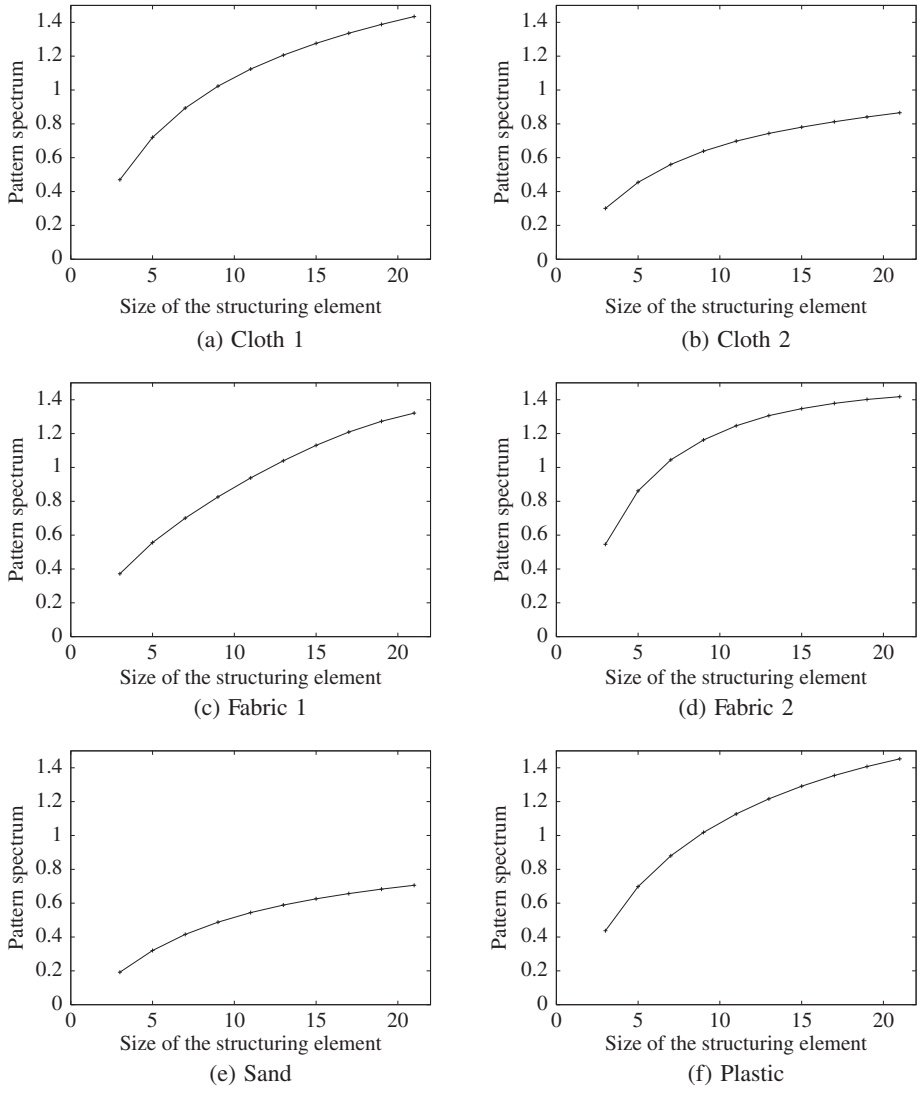


Figure 3.20 Pattern spectra for some images of Figure 3.1. They may be used to discriminate the different textures.

How can we use the pattern spectrum to classify textures?

The shape of the pattern spectrum is characteristic of the texture. We may use it as is, i.e. a series of numbers that are features that characterise the texture. Alternatively, one may model it by a parametric curve. The values of the parameters characterise the texture. For example, if the pattern spectrum in log–log coordinates can be fitted by a straight line, the texture is said to be approximated by a **fractal**. The slope of the line that fits the pattern spectrum in log–log coordinates may be used to estimate the so-called **fractal dimension** of the texture, which may serve as a feature characterising the texture.

3.3 Fractals and Multifractals

What is a fractal?

A **fractal** is a shape that has the same structure at all scales. An example of a fractal curve is shown in Figure 3.21 which depicts a fractal curve called the **von Koch snowflake**, first proposed in 1904.

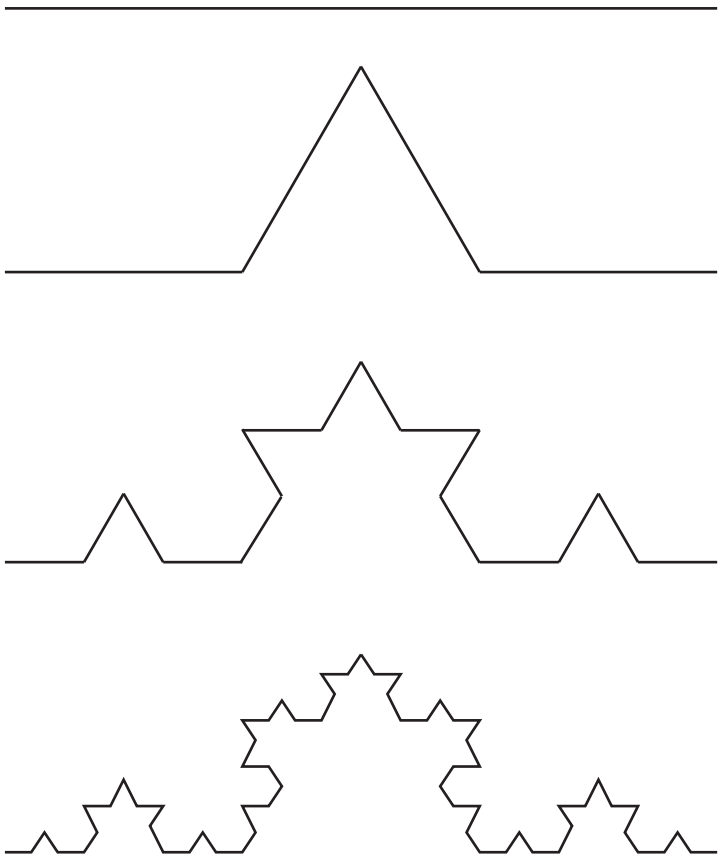


Figure 3.21 To create a von Koch snowflake curve, start with a straight line segment of length l. Divide it into three equal segments and replace the middle one by two segments of equal size, so that the replaced segment with the two new segments form an equilateral triangle. Repeat this ad infinitum for every line segment of the curve.

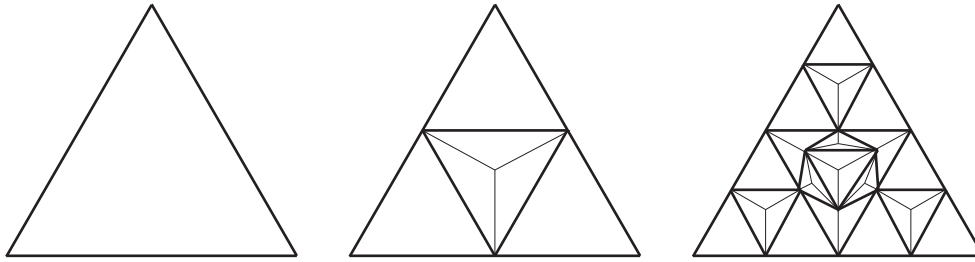


Figure 3.22 To create a fractal surface from a **Sierpinski triangle**, start with an equilateral triangle with side of length l. Divide it into four equal triangles and replace the middle one with a regular tetrahedron of side $l/2$. Repeat this ad infinitum for every equilateral triangle of the surface.

An example of a fractal surface is shown in Figure 3.22. These are examples of deterministic fractals that may be generated by applying a specific rule repeatedly.

One may also have fractals which, instead of having the same **shape** at all scales, have the same **statistical properties** at all scales.

These are **self-similar** fractals. So, a self-similar fractal remains the same either statistically or literally if all its coordinates are scaled by the same factor. If different coordinates have to be scaled by different factors, the fractal is called **self-affine**.

A fractal is characterised by, among other things, its **fractal dimension**.

What is the fractal dimension?

The **fractal dimension** of a curve is defined as

$$D \equiv \frac{\ln N}{\ln(1/r)} \tag{3.12}$$

where N expresses how many replicas of the curve, scaled by a factor $1/r$, we can fit in the curve.

The more accurately we try to measure the length of a fractal curve, the longer we find it. We may say that as the size of the measure we use tends to zero, the length of the curve tends to infinity. This is demonstrated in Example 3.15.

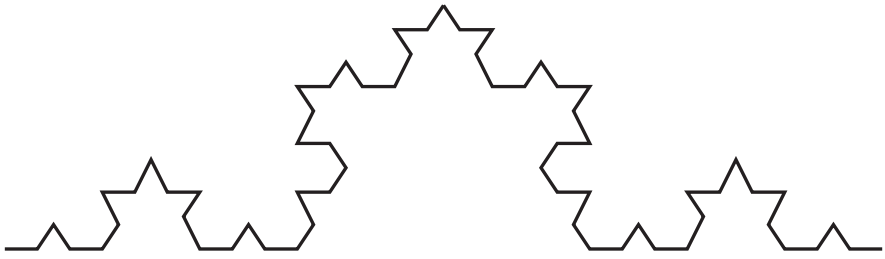
To express the fact that the length of the curve is always longer than what we actually measure, we say that we use a "corrective" exponent in the measuring units we use. For example, in everyday life, if we measure a line, we may say that it is three metres long and write 3 m. If we measure an area, we may say it is three square metres and write 3 m^2. Here, we may say that the length of the curve is 3 m^D, where D is a number between 1 and 2 and attempts to correct for the underestimate of the length of the curve as represented by number 3. This exponent D is the fractal dimension of the curve.

Example 3.15

Measure the length of the von Koch snowflake curve of Figure 3.23a using as measuring units those shown in Figure 3.23b, which are such that each one is smaller than the previous one by a factor of 3.

(Continued)

Example 3.15 (Continued)



(a)

(b)

Figure 3.23 The line segments in (b) are assumed to be rigid measuring rods that have to be placed on the top of the curve in (a) to see how many times they fit along the curve. The smaller the rod, the better it will follow the details of the curve.

The measuring process entails taking the measuring unit and checking how many times it fits on the curve. We must imagine that our measuring unit is like a solid rigid rod, and we physically pick it up and place it along the curve.

When we do this with the topmost rod, we see that we can only fit it on the curve once. So, the length of the von Koch snowflake curve is found to be 1 L, where L is our unit. Assume now that we pick up the second rod in Figure 3.23b. This one has length $L/3$*, and because it is smaller allows us to follow the details of the curve better. We find that it fits four times along the curve, so now we find that the length of the curve is* $4 \times (L/3)$*, i.e.* $4/3$ *L. Let us pick up the third rod. This is* $L/9$ *in length and, being even smaller, allows us to follow even more details of the curve. Now we find the length of the curve to be* $16 \times (L/9)$*, i.e.* $16/9$ *L. With the last rod we find that the length of the curve is* $64/27$ *L.*

A real von Koch snowflake curve will have even smaller details than those shown in Figure 3.23a, as the process of its creation is supposed to carry on ad infinitum. Therefore, it is not difficult to see that as the rods we use become smaller and smaller, more and more details are being picked up, and the length of the curve we compute tends to infinity, since at each scale we find the length we found at the previous scale multiplied with $4/3$*.*

Example 3.16

Compute the fractal dimension of the von Koch snowflake curve.
From example 3.15 we see that $4\ (unit/3)$ *is the new length we find when the unit we use is* $1/3$ *the previous unit. In order to compensate for this increase in length, we say that we shall use an exponent D for the unit, so that the length remains the same as before:* $4\ (unit/3)^D = 1\ (unit)^D$. *This leads to* $4(1/3)^D = 1$. *By taking the logarithm of both sides, we obtain* $\ln 4 - D \ln 3 = 0$, *which means that* $D = \ln 4/\ln 3 = 1.2618$.

We obtain the same answer if we apply equation (3.12). We obtain four equal replicas of the curve every time we scale it down by 3, so in that formula, $N = 4$ *and* $r = 1/3$.

Example 3.17

Compute the fractal dimension of the surface of Figure 3.22.
We notice that when we scale down the triangle by 2, we can fit four of them to the flat surface. Then the central one is replaced by a tetrahedron, i.e. one of those triangles is replaced by three equal ones, so the total surface now consists of six equilateral triangles that have a quarter of the area of the original one. Therefore, we may apply Equation (3.12) with $r = 1/4$ *and* $N = 6$ *to obtain* $D = \ln 6/\ln 4 = 1.29248$.

Example 3.18

Compute the fractal dimension of the von Koch snowflake curve using the box-counting method.
According to this method, we cover the shape with boxes of size $1/2^n$, *where n takes values* $0, 1, 2, \ldots$ *and count each time the number of boxes* $N(n)$ *of this size that are needed to cover the shape. The fractal dimension is then defined as:*

$$D = \lim_{n\to\infty} \frac{\ln N(n)}{\ln 2^n}. \tag{3.13}$$

Figure 3.24 demonstrates the application of this formula for the calculation of the fractal dimension of the von Koch snowflake. By counting the boxes we get the following table:

$n = 0$	N=1	-
$n = 1$	N=2	$\frac{\ln N(1)}{\ln 2^1} = 1.00$
$n = 2$	N=6	$\frac{\ln N(2)}{\ln 2^2} = 1.29$
$n = 3$	N=14	$\frac{\ln N(3)}{\ln 2^3} = 1.27$
$n = 4$	N=32	$\frac{\ln N(4)}{\ln 2^4} = 1.25.$

If we carry on, eventually the computed number will approach 1.26 which is the theoretically calculated value.

(Continued)

Example 3.18 (Continued)

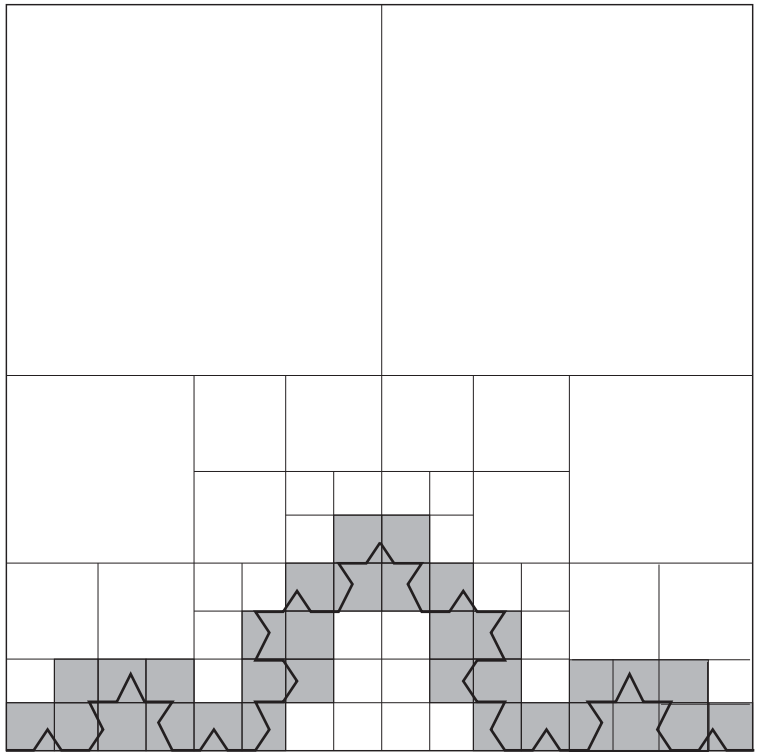


Figure 3.24 The box-counting method for the computation of the fractal dimension of the von Koch snowflake. For $n = 1$ we have four boxes, only two of which contain parts of the curve. So $N(1) = 2$. For $n = 2$, out of the 16 boxes needed to cover the whole space, only six contain parts of the curve. For $n = 4$, only the 32 shaded grey boxes contain parts of the curve.

Example 3.19

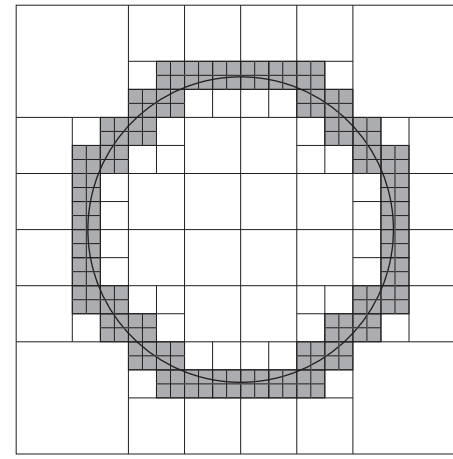
Use the box-counting method to work out the fractal dimension of a circle.

Figure 3.25 shows the successive stages of dividing the unit square that contains a circle into smaller and smaller squares. We can easily count the occupied boxes at each division:

$n = 0$	N=1	-
$n = 1$	N=4	$\frac{\ln N(1)}{\ln 2^1} = 2.00$
$n = 2$	N=12	$\frac{\ln N(2)}{\ln 2^2} = 1.79$
$n = 3$	N=20	$\frac{\ln N(3)}{\ln 2^3} = 1.44$
$n = 4$	N=44	$\frac{\ln N(4)}{\ln 2^4} = 1.36$
$n = 5$	N=80	$\frac{\ln N(5)}{\ln 2^5} = 1.26.$

If we carry on, eventually the numbers will converge to 1, since the circle is a smooth curve and not a fractal.

Figure 3.25 Calculating the fractal dimension of a circle by successive divisions of the occupied cells.



Example B3.20

Consider the surface created in Figure 3.22. The first stage of the creation of the surface is reproduced in Figure 3.26a. Imagine a cross-section of the surface along line *OA* as marked in that figure. The line forms angle $\alpha < 30°$ with the height *OF* of the original triangle. Draw this cross-section and compute all lengths that constitute it as functions of angle α and the side of the original triangle *l*.

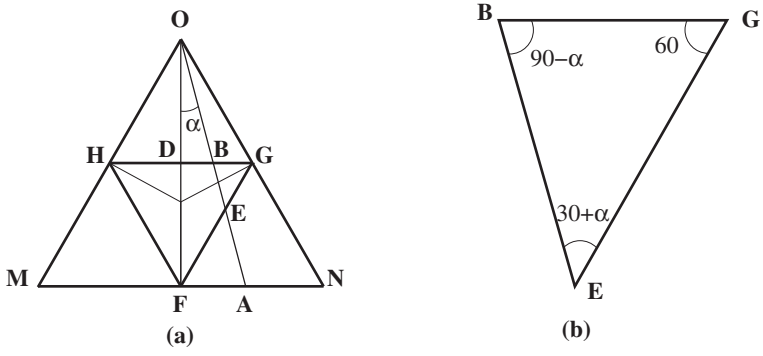


Figure 3.26 (a) The first stage of constructing the fractal surface of figure 3.22. (b) Triangle *BGE* in magnification.

(Continued)

Example B3.20 (Continued)

We must compute lengths OB, BE and EA first. Triangle OMN is equilateral with side l, and it is easy to see from Figure 3.26a that:

$$\begin{aligned} ON &= l \\ OF &= l\cos 30^\circ = l\sqrt{3}/2 \\ OD = DF &= OF/2 = l\sqrt{3}/4 \\ DB &= OD\tan\alpha = (l\sqrt{3}\tan\alpha)/4 \\ DG &= l/4 \\ BG &= DG - DB = l(1-\sqrt{3}\tan\alpha)/4. \end{aligned} \tag{3.14}$$

Let us define:

$$u \equiv OA = \frac{OF}{\cos\alpha} = \frac{l\sqrt{3}}{2\cos\alpha}. \tag{3.15}$$

So, length OB is:

$$OB = BA = OA/2 = u/2. \tag{3.16}$$

Then we must compute length BE:
Consider triangle BEG as shown in Figure 3.26b, and apply the sine rule to it:

$$\frac{BE}{\sin 60^\circ} = \frac{BG}{\sin(\alpha + 30^\circ)}. \tag{3.17}$$

Substituting BG from the last of Equations (3.14) into the above expression, we obtain:

$$BE = \frac{l\sqrt{3}(1-\sqrt{3}\tan\alpha)}{8\sin(\alpha+30^\circ)}. \tag{3.18}$$

This equation may be rearranged as follows:

$$BE = \frac{l\sqrt{3}}{2\cos\alpha} \times \frac{(1-\sqrt{3}\tan\alpha)\cos\alpha}{4\sin(\alpha+30^\circ)}. \tag{3.19}$$

Let us define a function f of α:

$$f(\alpha) \equiv \frac{(1-\sqrt{3}\tan\alpha)\cos\alpha}{4\sin(\alpha+30^\circ)}. \tag{3.20}$$

Then using the definition of u from (3.15) and f from (3.20) into (3.19), we obtain:

$$BE = uf(\alpha). \tag{3.21}$$

To compute EA we observe that:

$$EA = BA - BE = OA/2 - BE = u/2 - uf(\alpha) = u(1/2 - f(\alpha)). \tag{3.22}$$

Let us recapitulate. The process of creating the fractal surface divides segment OA into three parts with lengths:

$$\begin{aligned} OB &= \frac{u}{2} \\ BE &= uf(\alpha) \\ EA &= u\left(\frac{1}{2} - f(\alpha)\right). \end{aligned} \tag{3.23}$$

When we replace triangle HGF with a tetrahedron, segment BE is replaced by two other segments. These are segments BS and SE defined in Figure 3.27a which shows a 3D impression of the regular tetrahedron that sits on triangle HGF of Figure 3.26a. We need to compute the lengths of segments BS and SE in order to complete the answer to this problem.

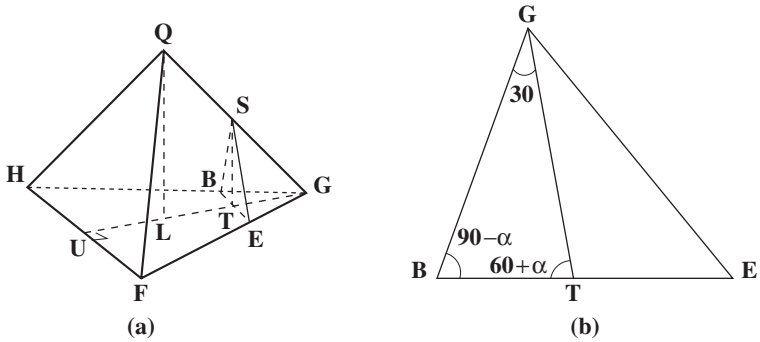


Figure 3.27 (a) The regular tetrahedron with which we replace triangle *HGF* of Figure 3.26a at the first stage of constructing the fractal surface. (b) Triangle *BGE* in magnification.

We start by observing that GU is the height of an equilateral triangle with side $l/2$, and L is the centre of the triangle. The side of the tetrahedron is also $l/2$. So, we know that:

$$\begin{aligned} QG = QH = QF = GF = HF = HG &= l/2 \\ UF = HF/2 &= l/4 \\ GU = \sqrt{GF^2 - UF^2} &= l\sqrt{3}/4 \\ GL = 2GU/3 &= l/(2\sqrt{3}) \\ QL = \sqrt{QG^2 - GL^2} &= l/\sqrt{6}. \end{aligned} \tag{3.24}$$

For obvious reasons QL is parallel to ST, so triangle STG is similar to triangle QLG. So, we have:

$$\frac{ST}{QL} = \frac{GT}{GL}. \tag{3.25}$$

Let us examine triangle BGT as shown in Figure 3.27b. By applying the sine rule to this triangle we obtain:

$$\frac{BG}{\sin(60° + \alpha)} = \frac{GT}{\sin(90° - \alpha)} = \frac{BT}{\sin 30°}. \tag{3.26}$$

From Figure 3.26a, we can see that:

$$BG = DG - DB = HG/2 - OD\tan\alpha = l/4 - l\sqrt{3}\tan\alpha/4 = l(1 - \sqrt{3}\tan\alpha)/4. \tag{3.27}$$

Combining Equations (3.26) and (3.27), we obtain:

$$\begin{aligned} BT &= \frac{l(1 - \sqrt{3}\tan\alpha)}{8\sin(\alpha + 60°)} \\ GT &= \frac{l(1 - \sqrt{3}\tan\alpha)\cos\alpha}{4\sin(\alpha + 60°)}. \end{aligned} \tag{3.28}$$

(Continued)

Example B3.20 (Continued)

We may use the expression for GT and the values of QL and GL from (3.24) into (3.25) to obtain the value of ST:

$$ST = \frac{\sqrt{2}l(1-\sqrt{3}\tan\alpha)\cos\alpha}{4\sin(\alpha+60^{o})}. \tag{3.29}$$

If we define a function g of α

$$g(\alpha) \equiv \frac{\sin(30^{\circ}+\alpha)}{\sqrt{3}\sin(60^{o}+\alpha)} \tag{3.30}$$

then one can see that:

$$BT = uf(\alpha)g(\alpha) \qquad ST = 2\sqrt{2}\cos\alpha\ uf(\alpha)g(\alpha). \tag{3.31}$$

One may easily prove that:

$$1-g(\alpha) = \frac{\cos\alpha}{\sqrt{3}\sin(60^{\circ}+\alpha)}. \tag{3.32}$$

Then from the right-angled triangles BST and STE one can compute the lengths of BS and SE:

$$BS = uf(\alpha)g(\alpha)\sqrt{1+8\cos^{2}\alpha}$$
$$SE = uf(\alpha)\left[1-g(\alpha)\right]\sqrt{1+8\sin^{2}(30^{\circ}+\alpha)}. \tag{3.33}$$

Figure 3.28 shows the profile of this cross-section for $u = 1$ and $\alpha = 15^{\circ}$. For these values we find $f(15^{\circ}) = 0.183$ and $g(15^{\circ}) = 0.4226$. Segment OA is divided into three parts of length 0.5, 0.183 and 0.317. The middle part is further divided into two parts of length $BT = 0.07734$ and $TE = 0.10566$. At point T we draw a line perpendicular to segment OA and measure along it length $TS = 0.2113$ to mark point S. Then we join point S with points B and E to form the profile. The lengths of segments BS and SE are 0.225 and 0.2363, respectively.

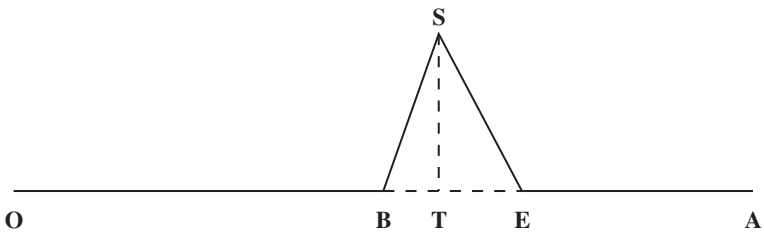


Figure 3.28 The cross-section of the surface created at the first stage of the creation of the fractal surface of Figure 3.26, along a line that forms a 15° angle with the north–south axis of the first triangle, and passes through its top vertex.

Example B3.21

Draw the profile of the fractal surface of Example 3.20 for $\alpha = 0^{\circ}$.

Let us assume that the length of segment OA, which in this case coincides with OF, is 1 (i.e. $u = 1$). Then for $\alpha = 0°$ we find from the corresponding equations:

From (3.20):	f	=	$1/2$
From (3.30):	g	=	$1/3$
From (3.23):	OB	=	$1/2$
	BE	=	$1/2$
	EA	=	0
From (3.31):	BT	=	$1/6$
	ST	=	$\sqrt{2}/3$
From (3.33):	BS	=	$1/2$
	SE	=	$1/\sqrt{3}$.

Figure 3.29a shows this profile. Its total length is:

$$OB + BS + SE = \frac{1}{2} + \frac{1}{2} + \frac{1}{\sqrt{3}} = 1 + \frac{1}{\sqrt{3}}. \tag{3.34}$$

From the lengths of the segments we can compute the tangents of angles ϕ and ψ:

$$\tan\phi = \frac{ST}{BT} = 2\sqrt{2} \qquad \tan\psi = \frac{ST}{TE} = \frac{ST}{BE - BT} = \sqrt{2}. \tag{3.35}$$

At the next stage of the fractal construction, shown in Figure 3.22, segment SE will not be affected. In fact it will not be affected at any stage as it coincides with an edge. Segment OB will be replaced by a replica of profile 3.29a scaled down by 2, while segment BS will be replaced by the same scaled-down profile reflected left to right. This will bring at vertex B two angles ψ and one angle ϕ. The tangent of the total angle is:

$$\tan(\phi + 2\psi) = \frac{\tan\phi + \tan 2\psi}{1 - \tan\phi \tan 2\psi}. \tag{3.36}$$

We need the tangent of 2ψ:

$$\tan 2\psi = \frac{2\tan\psi}{1 - \tan^2\psi} = -2\sqrt{2}. \tag{3.37}$$

Then:

$$\tan(\phi + 2\psi) = \frac{2\sqrt{2} - 2\sqrt{2}}{1 + 8} = 0. \tag{3.38}$$

This shows that the edges of the tetrahedra that make up the scaled-down segments will touch each other. As fractals continue ad infinitum at all scales, we may say that segment SE is not part of the line profile either, since being the edge of a tetrahedron it will also be in contact with another edge of the structure at the next larger scale. The new profile is shown in 3.29b. Notice that this new profile is the line $OB_1S_1B_2SE$. The lengths of segments OB_1, B_1S_1, S_1B_2 and B_2S are $1/4$. So the total length of the profile remains the same.

At the subsequent steps, segments OB_1, B_1S_1, S_1B_2 and B_2S will all be replaced by segments of similar shape but half their size, and the total length of the line will also remain the same. The next two stages are shown in Figures 3.29c and 3.29d.

(Continued)

Example B3.21 (Continued)

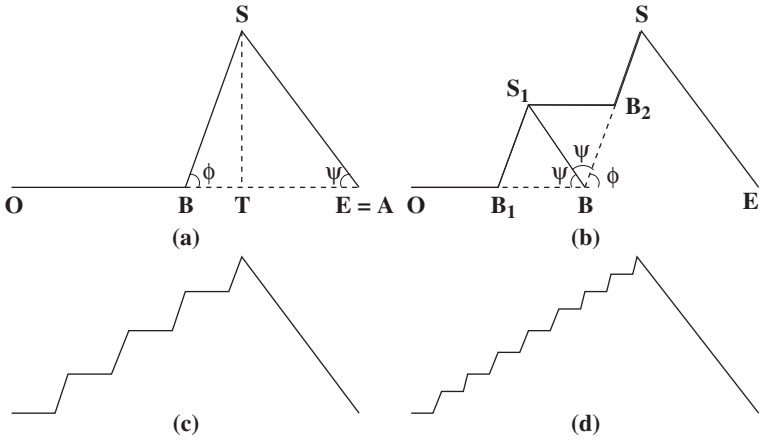


Figure 3.29 The cross-section of the surface created at the first four stages of the creation of the fractal surface of Figure 3.26, along a line that forms 0° angle with the north–south axis of the first triangle, and passes through its top vertex.

Example B3.22

Compute the fractal dimension of the curve created in Example 3.21.

The process of creating the curve of Example 3.21 may be summarised as follows.

Replace shape 3.30a with shape 3.30b, which consists of two scaled-down versions, by a factor of 2, of the original shape. It can be easily shown, from the original geometry, that angle ω is equal to $\phi/2$ defined in Figure 3.29, and such that $\sin\omega = 1/\sqrt{3}$. Figure 3.30d shows the application of the box-counting method for computing the fractal dimension. By counting boxes of size $1/2^n$, for $n = 0, 1, 2, \ldots$, we form the following table:

$n = 0$	N=1	-
$n = 1$	N=2	$\frac{\ln N(1)}{\ln 2^1} = 1.00$
$n = 2$	N=4	$\frac{\ln N(2)}{\ln 2^2} = 1.00$
$n = 3$	N=8	$\frac{\ln N(3)}{\ln 2^3} = 1.00$
$n = 4$	N=16	$\frac{\ln N(4)}{\ln 2^4} = 1.00.$

The curve becomes smoother and smoother, and it tends to a straight line with "fractal" dimension 1. So, this is not a fractal at all.

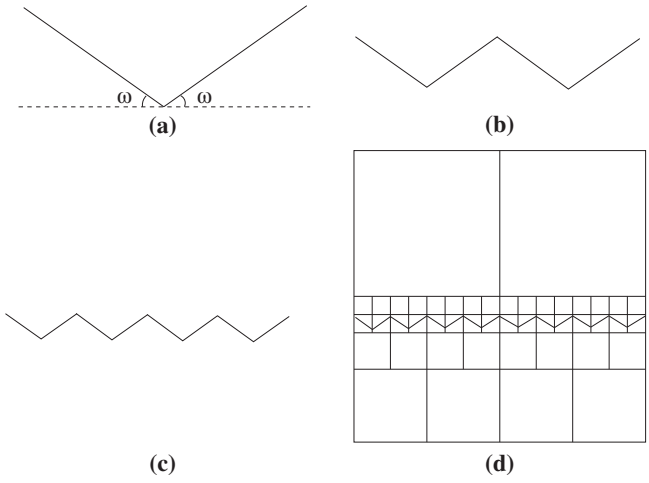


Figure 3.30 The first four stages of creating the curve of Example 3.21 and the application of the box-counting method to find its fractal dimension.

Example B3.23

Think of a simple procedure that will convert a number into its binary form. Apply it to write 175 as a binary number.

Let us call this number X. We shall do the following.

Step 1: *Compute* $X_1 \equiv \lfloor \log_2 X \rfloor$.
Step 2: *If* $X - 2^{X_1} = 0$, *exit. The binary representation of X is a 1 followed by* $X_1 - 1$ *zeros.*
Step 3: *If* $X - 2^{X_1} = X_2 \neq 0$, *compute* $X_3 \equiv \lfloor \log_2 X_2 \rfloor$.
Step 4: *If* $X_2 - 2^{X_3} = 0$ *exit. The binary representation of X is a 1, followed by* $X_1 - X_3 - 1$ *zeros, followed by 1, followed by* $X_2 - 1$ *zeros.*
Step 5: *If* $X_2 - 2^{X_3} = X_4 \neq 0$, *compute* $X_5 \equiv \lfloor \log_2 X_4 \rfloor$.
Carry on in a similar way, until you exit the algorithm.

Set $X = 175$. *We have* $\log_2 175 = 7.4512$. *Then* $X_1 = 7$.
$175 - 2^7 = 175 - 128 = 47$, *so,* $X_2 = 47$.
$\log_2 47 = 5.5546$, *so,* $X_3 = 5$.

(Continued)

Example B3.23 (Continued)

$47 - 2^5 = 47 - 32 = 15$, *so,* $X_4 = 15$.
$\log_2 15 = 3.9069$, *so,* $X_5 = 3$.
$15 - 2^3 = 7$, *so,* $X_6 = 7$.
$\log_2 7 = 2.8074$, *so,* $X_7 = 2$.
$7 - 2^2 = 7 - 4 = 3$, *so,* $X_8 = 3$.
$\log_2 3 = 1.5850$, *so,* $X_9 = 1$.
$3 - 2^1 = 1$, *so,* $X_{10} = 1$.
$\log_2 1 = 0$, *so,* $X_{11} = 0$, *and we exit the algorithm.*
So the binary representation of 175 *is* 1, *followed by* $X_1 - X_3 - 1 = 1$ *zeros,*
followed by 1, *followed by* $X_3 - X_5 - 1 = 1$ *zeros,*
followed by 1, *followed by* $X_5 - X_7 - 1 = 0$ *zeros,*
followed by 1, *followed by* $X_7 - X_9 - 1 = 0$ *zeros,*
followed by 1, *followed by* $X_9 - X_{11} - 1 = 0$ *zeros, followed by* 1*:*
10101111.

Example B3.24

Consider a square image of size 128×128**. Divide the image into four quadrants. By considering the binary representation of the indices of the pixels, work out what the first bit of the pixel indices in each quadrant will be.**

The top left quadrant will consist of all pixels with indices (i, j) *both in the range* $[0, 63]$*. The top right quadrant will have the pixels with indices* (i, j) *in the ranges* $[64, 127]$ *and* $[0, 63]$*, respectively, the bottom left will consist of all pixels with indices* (i, j) *in the ranges* $[0, 63]$ *and* $[64, 127]$*, respectively, and the bottom right will consist of all pixels with both indices* (i, j) *in the range* $[64, 127]$*. Since the image is of size* 128×128*, and* $128 = 2^7$*, all pixel indices are 7-bit binary numbers. If we write these indices in binary form, we shall see that the top left quadrant will have* (i, j) *indices both starting from 0, the top right quadrant index i starting from 1 and index j from 0, the bottom left quadrant index i starting from 0 and index j from 1, while the bottom right quadrant will have both indices starting from 1. This is shown schematically in Figure 3.31*

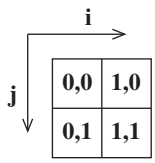


Figure 3.31 The first bit of the indices of the pixels in each quadrant of an image are always those indicated here.

Box 3.1 The box-counting method for binary images

As this method keeps dividing the image into quadrants (see Example 3.16), we assume that the image is square, with dimensions a power of 2. Let us assume then that we have expressed all

pixel coordinates in binary form (see Example 3.23). With some careful thinking (see Example 3.24), we can soon realise that the first bit of each pixel coordinate tells us in which first division quadrant the pixel is. The second bit tells us in which quadrant the pixel is in the second division, the third bit in the third division and so on. At each division level, the arrangement of the bits follows that shown in Figure 3.31. So, for example, a pixel with coordinates that have as first three bits $(101, 011)$, belongs to the top right quadrant in the first division $(1, 0)$, the bottom left quadrant of the top right quadrant in the second division $(0, 1)$, and the bottom right quadrant of the bottom left quadrant of the top right quadrant in the third division $(1, 1)$, as shown in Figure 3.32.

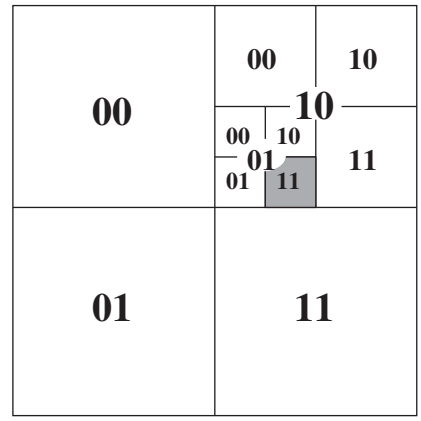


Figure 3.32 The largest font is used to indicate the first bit of the index of a pixel, the medium font indicates the second bit and the smallest font indicates the third bit. The grey square indicates the cell to which a pixel, with indices the first three bits of which are $(101, 011)$, belongs.

So, if we want to know the fractal dimension of a binary curve, like that of Figure 3.24, we assume that the pixels of the curve have value 1 and the background has value 0. Then we apply the following algorithm.

Step 0: Consider the input binary image $I(i,j)$, of size $2^{n_0} \times 2^{n_0}$.
Step 1: Consider the binary indices (i,j) of each pixel. These binary numbers consist of n_0 bits. Let us indicate by $P(n;x)$ the nth bit of number x.
Step 2: Set up n_0 arrays:
$M(1;k)$ where $k = 1, 2, 3, 4$
$M(2;k)$ where $k = 1, 2, \dots, 16$
$\dots$
$M(n;k)$ where $k = 1, 2, \dots, 4^n$
$\dots$
$M(n_0;k)$ where $k = 1, 2, \dots, 4^{n_0}$
Set all elements of all these arrays equal to 0.
Step 3: For $i = 0, 1, \dots, 2^{n_0} - 1$ and $j = 0, 1, \dots, 2^{n_0} - 1$, and if $I(i,j) \neq 0$, do:
for $n = 1, 2, \dots, n_0$, set
$k_1 = 2^{n-1}P(1;i) + 2^{n-2}P(2;i) + \cdots + 2^0 P(n;i)$
$k_2 = 2^{n-1}P(1;j) + 2^{n-2}P(2;j) + \cdots + 2^0 P(n;j)$
$k = 2^n k_2 + k_1 + 1$
$M(n,k) = 1$.
Step 4: For $n = 1, 2, \dots, n_0$, set $N(n) = \sum_k M(n,k)$.

(Continued)

Box 3.1 (Continued)

Step 5: Use $N(n)$ in (3.13) to work out the various approximations of the fractal dimension of the curve.

Note that step 2 sets up the cells for all divisions of the grid. It is obvious from Figure 3.24 that in the first division we have only 4 cells, in the second 16, and so on, until the last one where each cell consists of a single pixel, so we have $2^{n_0} \times 2^{n_0} = 4^{n_0}$ cells. Note that the cells of each division are indexed sequentially by index k. A cell is considered occupied even if a single pixel of the curve belongs to it. In step 3, and for each bit of the coordinate position of a curve pixel (which also identifies which division we consider), we identify the cell to which this pixel belongs and mark it as occupied. The calculation of k_1 and k_2 identifies the cell of the division under consideration, by considering all bits of the coordinates of the pixel on the left of the current bit (division). In step 4 we count how many cells for each division are occupied.

Obviously, if we are only interested in calculating the fractal dimension for the largest value of n, i.e. for the finest division, the algorithm is trivial, since a cell of the finest division consists of a single pixel. Then, for a $2^{n_0} \times 2^{n_0}$ binary image, we simply count the number of black pixels that make up the depicted object. This is number $N(n_0)$. We use it then in (3.13) to compute the fractal dimension of the binary image. Seen this way, we realise that when we compute the fractal dimension of a binary image, we do not take any consideration of the spatial distribution of the pixels that make up the depicted shape. That is why the **generalised fractal dimensions** (see section on What is the generalised fractal dimension of a binary shape?) and the **local connected fractal dimension** (see section on How do we compute the local connected fractal dimension of a binary image?) were introduced.

However, to compute the fractal dimension it is better not to use just the last division into blocks, where each block consists of a single pixel, but a few previous divisions too. Then the fractal dimension will be computed as the slope of the line fitted to the pairs of points $(\ln 2^n, \ln N(n))$. In fact experiments have shown that the last couple of divisions where the bocks consist of 4 or 1 pixel only yield unreliable results and it is better if they are not used at all (see Example 3.53).

Example B3.25

For the 16×16 image of Figure 3.24 apply the algorithm of Box 3.1 to work out the fractal dimension of the curve.

Note that $n_0 = 4$. The 32 pixels that contain parts of the curve have coordinates:

$(0, 15)$, $(1, 14)$, $(1, 15)$, $(2, 14)$, $(2, 15)$, $(3, 14)$, $(3, 15)$, $(4, 15)$
$(5, 13)$, $(5, 14)$, $(5, 15)$, $(6, 12)$, $(6, 13)$, $(6, 14)$, $(7, 11)$, $(7, 12)$
$(8, 11)$, $(8, 12)$, $(9, 12)$, $(9, 13)$, $(9, 14)$,$(10, 13)$, $(10, 14)$, $(10, 15)$,
$(11, 15)$, $(12, 14)$, $(12, 15)$, $(13, 14)$, $(13, 15)$, $(14, 14)$, $(14, 15)$, $(15, 15)$.

As we scan the image in a raster way, the first pixel we shall encounter with non-zero value will be pixel $(7, 11)$. *The binary representations of these indices are:* $(0111, 1011)$.

For $n = 1$, $P(1; 7) = 0$ *and* $P(1; 11) = 1$. *Then:*

$k_1 = 2^0 \times 0 = 0$ *and* $k_2 = 2^0 \times 1 = 1$. *Then* $k = 2^1 \times 1 + 0 + 1 = 3$. *So:*

$M(1, 3) = 1$

For $n = 2$, $P(2; 7) = 1$ *and* $P(2; 11) = 0$. *Then:*

$k_1 = 2^1 \times 0 + 2^0 \times 1 = 1$ *and* $k_2 = 2^1 \times 1 + 2^0 \times 0 = 2$. *Then:* $k = 2^2 \times 2 + 1 + 1 = 10$.
So:
$M(2, 10) = 1$
For $n = 3$, $P(3; 7) = 1$ *and* $P(3; 11) = 1$. *Then:*
$k_1 = 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1 = 3$ *and* $k_2 = 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 = 5$. *Then:*
$k = 2^3 \times 5 + 3 + 1 = 44$. *So:*
$M(3, 44) = 1$
For $n = 4$, $P(4; 7) = 1$ *and* $P(4; 11) = 1$. *Then:*
$k_1 = 2^3 \times 0 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 1 = 7$ *and* $k_2 = 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1 = 11$.
Then: $k = 2^4 \times 11 + 7 + 1 = 184$. *So:*
$M(4, 184) = 1$
In the same way we have to check for all other curve pixels in the image. At the end, we shall have to count the number of occupied cells at each level, by simply adding the elements of the corresponding M matrix up.

Box 3.2 The box-counting algorithm for grey images

Step 0: Consider the input grey image $I(i, j)$, of size $2^{n_0} \times 2^{n_0}$ and with 2^{m_0} grey levels.
Step 1: For each pixel, consider its binary indices and its grey value in binary form: (i, j, m). The first two of these binary numbers consist of n_0 bits, while the last one of m_0 bits. Let us indicate by $P(n; x)$ the nth bit of number x. Set $L = \max\{n_0, m_0\}$ and express all three numbers as L-bit binary numbers.
Step 2: Create L empty sets:
$M_1 = \emptyset$ which will contain elements in the range $[1, 8]$
$M_2 = \emptyset$ which will contain elements in the range $[1, 64]$
…
$M_n = \emptyset$ which will contain elements in the range $[1, 8^n]$
…
$M_L = \emptyset$ which will contain elements in the range $[1, 8^L]$.
Step 3: For $i = 0, 1, \ldots, 2^{n_0} - 1$ and $j = 0, 1, \ldots, 2^{n_0} - 1$, if $I(i, j) \neq 0$ execute step 4.
Step 4: For $n = 1, 2, \ldots, L$, set
$k_1 = 2^{n-1}P(1; i) + 2^{n-2}P(2; i) + \cdots + 2^0 P(n; i)$
$k_2 = 2^{n-1}P(1; j) + 2^{n-2}P(2; j) + \cdots + 2^0 P(n; j)$
$k_3 = 2^{n-1}P(1; I(i, j)) + 2^{n-2}P(2; I(i, j)) + \cdots + 2^0 P(n; I(i, j))$
$k = 4^n k_3 + 2^n k_2 + k_1 + 1$
$M_n = M_n \cup \{k\}$, adds element k to set M_n.
Step 5: For $n = 1, 2, \ldots, L$, set
$N(n) = cardinality(M_n)$, where cardinality is the number of elements in the set.
Step 6: Use $N(n)$ in Equation (3.13) to create a sequence of approximations of the fractal dimension of the image.

If we are only interested in the finest box division, we essentially have to count the number of non-zero pixels, and use that as $N(n_0)$ in (3.13). Clearly this is not going to be particularly informative for the structure of the image, so several intermediate values of n may be used instead, to yield a sequence of numbers characterising the image.

Example B3.26

Apply the algorithm of Box 3.2 to the image of Figure 3.33.

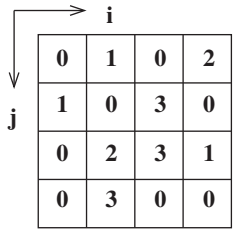


Figure 3.33 A 4×4 2-bit image.

For this image $n_0 = m_0 = L = 2$. The algorithm of Box 3.2 fills the 3D image space with boxes, numbered as shown in Figure 3.34. Each of these boxes will be further subdivided into 8 boxes in the same way. For each level of division, the algorithm counts how many of the boxes are occupied by the image when it is interpreted as a landscape.

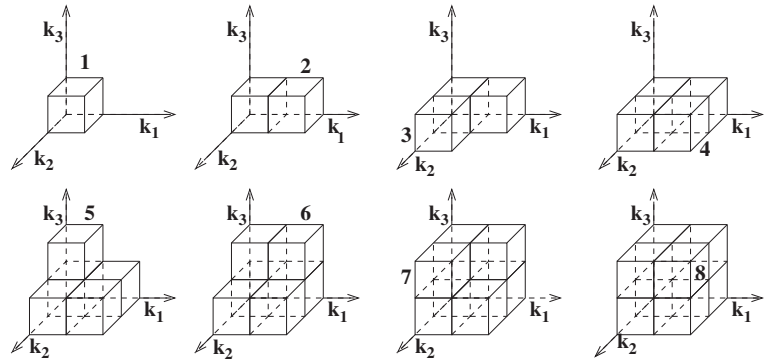


Figure 3.34 Treating the image as a landscape, we imagine that it exists in a 3D space, which we may fill with 3D boxes. Here we show the boxes used for a 4×4 image, with four grey levels, and for the first level of division of the image volume, by dividing the range of values along each axis into two halves.

We must consider the binary representations of each pixel's coordinates and grey value. In what follows, each of these numbers is first given in the decimal system and then in the binary. The first non-zero pixel has $i = 1 = 01$, $j = 0 = 00$ and $I(i,j) = 1 = 01$. Thus, for $n = 1$, it has $k_1 = 0$, $k_2 = 0$ and $k_3 = 0$, yielding an index $k = 1$, indicating that it is in box number 1 in the first division. Therefore, the index of this box is added to set M_1: $M_1 = \{1\}$.

The same pixel, for $n = 2$, will yield $k_1 = 1$, $k_2 = 0$ and $k_3 = 1$, yielding an index $k = 18$. So, for the second level of division, we add index 18 to set M_2: $M_2 = \{18\}$.

The second non-zero pixel has $i = 3 = 11, j = 0 = 00$ and $I(i,j) = 2 = 10$. So, for $n = 1$, we have $k_1 = 1$, $k_2 = 0$ and $k_3 = 1$, resulting in $k = 6$ and thus $M_1 = \{1, 6\}$. For $n = 2$, $k_1 = 3$, $k_2 = 0$ and $k_3 = 1$, we obtain $k = 20$ and $M_2 = \{18, 20\}$.

In a similar way we may deal with the other pixels:

Pixel $(0,1) = (00, 01)$ and $I(0,1) = 1 = 01$.

For $n = 1$ $k_1 = 0$, $k_2 = 0$, $k_3 = 0$, $k = 1$ and $M_1 = \{1, 6\}$. Note that element 1 already belongs to set M_1, so it is not repeated.

For $n = 2$, $k_1 = 0$, $k_2 = 1$, $k_3 = 1$, $k = 21$ and $M_2 = \{18, 20, 21\}$.

Pixel $(2, 1) = (10, 01)$ and $I(2, 1) = 3 = 11$.

For $n = 1$ $k_1 = 1$, $k_2 = 0$, $k_3 = 1$, $k = 6$ and $M_1 = \{1, 6\}$.

For $n = 2$, $k_1 = 2$, $k_2 = 1$, $k_3 = 3$, $k = 55$ and $M_2 = \{18, 20, 21, 55\}$.

Pixel $(1, 2) = (01, 10)$ and $I(1, 2) = 2 = 10$.

For $n = 1$ $k_1 = 0$, $k_2 = 1$, $k_3 = 1$, $k = 7$ and $M_1 = \{1, 6, 7\}$.

For $n = 2$, $k_1 = 1$, $k_2 = 2$, $k_3 = 2$, $k = 42$ and $M_2 = \{18, 20, 21, 55, 42\}$.

Pixel $(2, 2) = (10, 10)$ and $I(1, 2) = 3 = 11$.

For $n = 1$ $k_1 = 1$, $k_2 = 1$, $k_3 = 1$, $k = 8$ and $M_1 = \{1, 6, 7, 8\}$.

For $n = 2$, $k_1 = 2$, $k_2 = 2$, $k_3 = 3$, $k = 59$ and $M_2 = \{18, 20, 21, 55, 42, 59\}$.

Pixel $(3, 2) = (11, 10)$ and $I(3, 2) = 1 = 01$.

For $n = 1$ $k_1 = 1$, $k_2 = 1$, $k_3 = 0$, $k = 4$ and $M_1 = \{1, 6, 7, 8, 4\}$.

For $n = 2$, $k_1 = 3$, $k_2 = 2$, $k_3 = 1$, $k = 28$ and $M_2 = \{18, 20, 21, 55, 42, 59, 28\}$.

Pixel $(1, 3) = (01, 11)$ and $I(1, 3) = 3 = 11$.

For $n = 1$ $k_1 = 0$, $k_2 = 1$, $k_3 = 1$, $k = 7$ and $M_1 = \{1, 6, 7, 8, 4\}$.

For $n = 2$, $k_1 = 1$, $k_2 = 3$, $k_3 = 3$, $k = 62$ and $M_2 = \{18, 20, 21, 55, 42, 59, 28, 62\}$.

So, we may count now how many of the level $n = 1$ boxes and how many of level $n = 2$ boxes are occupied:

$$N(1) = \textit{cardinality}(M_1) = 5 \qquad N(2) = \textit{cardinality}(M_2) = 8. \tag{3.39}$$

From these values we may identify two successive estimates of the fractal dimension, using Equation (3.13):

$$\begin{aligned} &\textit{For } n = 1 \qquad D = \frac{\ln 5}{\ln 2} = 2.32 \\ &\textit{For } n = 2 \qquad D = \frac{\ln 8}{\ln 4} = 1.50 \end{aligned} \tag{3.40}$$

Note a shortcut for the calculation of the last level of division: if this level results in boxes consisting of single pixels, to find how many boxes are occupied all we have to do is to count the non-zero pixels. However, a better estimate is obtained if the level of division stops at an earlier point, or, if several successive high values of n are used, to work out a single value by fitting the pairs of points $(\ln 2^n, \ln N(n))$ with a straight line (see Box 3.3 and Example 3.53).

What is the generalised fractal dimension of a binary shape?

In the box counting method for computing the **fractal dimension** of a binary shape, we cover the image plane with boxes of progressively smaller size and we simply check whether a box contains part of the shape or not. To define the **generalised fractal dimension** of the shape, we have to count the individual pixels of the shape that happen to be inside the box. If all points that make up the shape are N and there are N_i points inside the ith box, then we assign to that box the value $p_i \equiv N_i/N$, which indicates the probability of finding a point of the shape inside a box of that size, say $\varepsilon \times \varepsilon$. In this context the image is considered to be of size 1×1, so ε is a number smaller than

1. Then the generalised fractal dimension of the shape is defined as:

$$D_q \equiv \frac{1}{q-1}\lim_{\varepsilon\to 0}\frac{\ln\sum_i p_i^q}{\ln\varepsilon}. \tag{3.41}$$

We have the following special cases. For $q = 0$,

$$D_0 = -\lim_{\varepsilon\to 0}\frac{\ln\sum_i 1}{\ln\varepsilon} = -\lim_{\varepsilon\to 0}\frac{\ln(\text{number\_of\_non-empty\_boxes})}{\ln\varepsilon}. \tag{3.42}$$

This is the fractal dimension we compute by the ordinary box counting method (see Equation (3.13)). To distinguish from the other dimensions, we call it **capacity dimension**. For $q = 1$, it can be shown (see Example 3.27) that definition (3.41) reduces to:

$$D_1 = \lim_{\varepsilon\to 0}\frac{-\sum_i p_i \ln p_i}{\ln\frac{1}{\varepsilon}}. \tag{3.43}$$

As $-\sum_i p_i \ln p_i$ is the entropy used to measure the information conveyed by random variable p_i, this dimension is known as the **information dimension** of the shape. For $q = 2$, we have:

$$D_2 = \lim_{\varepsilon\to 0}\frac{\sum_i p_i^2}{\ln\varepsilon}. \tag{3.44}$$

Note that $\sum_i p_i^2$ is the probability with which we can find two points within the same box of size ε. This is related to the two-point correlation function that expresses the probability of two points being less than ε distance apart. So, this dimension is known as the **correlation dimension**.

Other values of q are also used. D_q plotted versus q constitutes the **multifractal spectrum** of the image. Non-fractals and mono-fractals (i.e. ordinary fractals) have flatter multifractal spectra than multifractals.

What is the role of q in the definition of the generalised fractal dimension?

It can be seen from (3.41) that when q is positive, boxes with small values of p_i (almost empty regions of the shape) have their values suppressed more than boxes with big values (dense regions of the shape). For example, if $p_i = 0.7$, for $q = 2$ the value of p_i^q is 0.49, not very different from the original p_i value. If, however, $p_i = 0.1$, for $q = 2$ the value of p_i^q is 0.01, i.e. 10 times smaller than p_i. When q is negative, small numbers (almost empty regions of the shape) are exaggerated much more than larger numbers (dense regions of the shape). For example, if $p_i = 0.7$, for $q = -2$ the value of p_i^q is $1/0.49 \simeq 2$, but if $p_i = 0.1$, for $q = -2$ the value of p_i^q is 100, much larger than 0.1. So, q acts like a non-linear filter that exaggerates differently sparse and dense regions of a shape.

Example B3.27

Show that for $q \to 1$, (3.41) yields (3.43).
We must use de l'Hospital's rule to work out what the limit of the right-hand side of (3.41) is for $q \to 1$.

$$D_1 = \lim_{\varepsilon\to 0}\left(\frac{1}{\ln\varepsilon}\lim_{q\to 1}\frac{\ln\sum_i p_i^q}{q-1}\right)$$

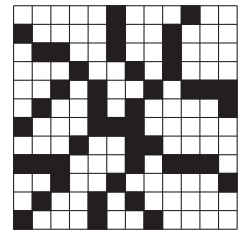
$$= \lim_{\varepsilon \to 0} \left(\frac{1}{\ln \varepsilon} \lim_{q \to 1} \frac{\frac{d(\ln \sum_i p_i^q)}{dq}}{\frac{d(q-1)}{dq}} \right)$$
$$= \lim_{\varepsilon \to 0} \left(\frac{1}{\ln \varepsilon} \lim_{q \to 1} \frac{\frac{\sum_i p_i^q \ln p_i}{\sum_i p_i^q}}{1} \right)$$
$$= \lim_{\varepsilon \to 0} \left(\frac{-1}{\ln \frac{1}{\varepsilon}} \left[\sum_i p_i \ln p_i \right] \right). \tag{3.45}$$

The last result is because $\sum_i p_i = 1$.

Example 3.28

Figure 3.35 shows the drawing of a ganglion cell. Compute its generalised fractal dimensions and its multifractal spectrum for $q = -2, -1, 0, 1, 2$**.**

Figure 3.35 A ganglion cell.



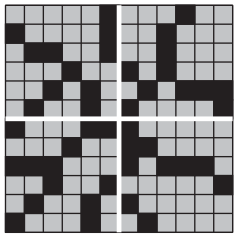
We remember that the size of the image is considered to be 1×1. *Figure 3.36 shows the boxes with which we cover the image plane (delineated with white lines), with sizes* $\varepsilon = 1/2, 1/3, 1/4, 1/6$. *Figure 3.37 shows the number of object pixels that have been identified inside each box. These numbers have to be divided with 45, the total number of object pixels, to produce numbers* p_i *that have to be used in formula (3.41).*

Applying (3.41) for $\varepsilon = 1/2$, *we have:*

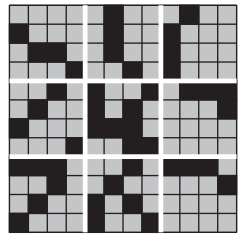
$$D_q = \frac{1}{q-1} \frac{\ln \left[\left(\frac{11}{45} \right)^q + \left(\frac{11}{45} \right)^q + \left(\frac{13}{45} \right)^q + \left(\frac{10}{45} \right)^q \right]}{\ln \left(\frac{1}{2} \right)}. \tag{3.46}$$

(Continued)

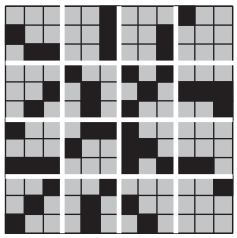
Example 3.28 (Continued)



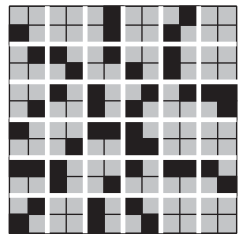
(a) $\varepsilon = 1/2$



(b) $\varepsilon = 1/3$



(c) $\varepsilon = 1/4$



(d) $\varepsilon = 1/6$

Figure 3.36 We cover the image with boxes of decreasing size and count the number of shape pixels inside each box.

11	11
13	10

(a) $\varepsilon = 1/2$

4	4	4
4	9	4
6	6	4

(b) $\varepsilon = 1/3$

3	3	2	1
2	3	4	4
4	3	5	2
3	3	2	1

(c) $\varepsilon = 1/4$

1	0	2	0	2	0
1	2	1	1	2	0
1	1	2	2	1	3
1	1	2	3	0	0
2	2	1	1	2	2
2	0	2	2	0	0

(d) $\varepsilon = 1/6$

Figure 3.37 The number of object pixels inside each box.

For $q = -2$, we have:

$$D_{-2} = -\frac{1}{3}\frac{\ln\left[\frac{2\times 11^{-2}+13^{-2}+10^{-2}}{45^{-2}}\right]}{-\ln 2} = 2.0126. \tag{3.47}$$

For the other values of q, we obtain:

$$D_{-1} = 2.0064 \qquad D_0 = 2 \qquad D_1 = 1.9934 \qquad D_2 = 1.9865. \tag{3.48}$$

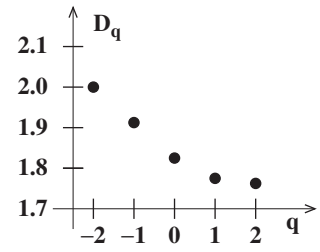
In a similar way, we work out the values for the other values of ε. These are shown in Table 3.2. Each one of the D_q is computed as a limit, for $\varepsilon \to 0$. The numbers in each column of the table

are fitted in the least square error sense with an equation of the form $D = \alpha\varepsilon + \beta$. The value of D_q then is equal to β. This limit is given at the bottom row of this tale. The multifractal spectrum of this image is plotted in Figure 3.38, consisting of the sequence of numbers in the bottom row of the table.

Table 3.2 The last row shows the values of the various generalised fractal dimensions of image 3.35, each computed as the limiting value of the values in the column above it.

ε	D_{-2}	D_{-1}	D_0	D_1	D_2
$1/2$	2.0126	2.0064	2.0000	1.9934	1.9865
$1/3$	2.0629	2.0351	2.0000	1.9571	1.9077
$1/4$	2.1575	2.0722	2.0000	1.9444	1.9019
$1/6$	1.9167	1.8790	1.8394	1.8021	1.7696
$\varepsilon \to 0$	2.0066	1.9207	1.8394	1.7689	1.7100

Figure 3.38 The multifractal spectrum of image 3.35.



Example 3.29

Compute the generalised fractal dimensions and its multifractal spectrum for $q = -3, -2, -1, 0, 1, 2, 3$, for Figure 3.39.

Figure 3.39 A 256×256 area of the image in Figure 2.1. Source: Maria Petrou.

(Continued)

Example 3.29 (Continued)

The results are shown in Table 3.3 and the multifractal image is plotted in Figure 3.40.

Table 3.3 The numbers of each column have been fitted with the least squares error to obtain the limiting values listed in the bottom row.

ϵ	D_{-3}	D_{-2}	D_{-1}	D_0	D_1	D_2	D_3
1/2	2.0235	2.0156	2.0077	2.0000	1.9924	1.9851	1.9779
1/4	2.0471	2.0294	2.0137	2.0000	1.9883	1.9783	1.9698
1/8	2.0560	2.0335	2.0149	2.0000	1.9882	1.9786	1.9705
1/16	2.1616	2.0765	2.0255	2.0000	1.9846	1.9733	1.9641
1/32	2.3372	2.1602	2.0379	1.9986	1.9799	1.9670	1.9568
$\epsilon \to 0$	2.2184	2.1056	2.0298	1.9994	1.9825	1.9704	1.9608

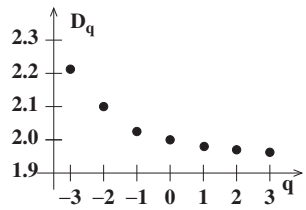


Figure 3.40 The multifractal spectrum of image 3.39.

Box 3.3 Least squares error fitting of data with a straight line

Assume we have N pairs of values (x_i, y_i) and we want to fit them with a straight line of the form: $y(x) = Ax + B$. We want to define the values of A and B so that the square difference between y_i and $y(x_i)$ is minimal. We define the total square error we wish to minimise as:

$$E \equiv \sum_{i=1}^{N} (Ax_i + B - y_i)^2. \tag{3.49}$$

To minimise E with respect to A and B, we differentiate it with respect to these two variables and set the derivatives to 0:

$$\frac{\partial E}{\partial A} = 2\sum_{i=1}^{N} (Ax_i + B - y_i)x_i$$

$$= 2A\sum_{i=1}^{N} x_i^2 + 2B\sum_{i=1}^{N} x_i - 2\sum_{i=1}^{N} y_i x_i$$

$$\frac{\partial E}{\partial B} = 2\sum_{i=1}^{N} (Ax_i + B - y_i)$$

$$= 2A\sum_{i=1}^{N} x_i + 2B\sum_{i=1}^{N} 1 - 2\sum_{i=1}^{N} y_i. \tag{3.50}$$

Setting the right-hand sides to 0 leads to the following system of linear equations with respect to A and B:

$$\left(\sum_{i=1}^{N} x_i^2\right) A + \left(\sum_{i=1}^{N} x_i\right) B = \sum_{i=1}^{N} y_i x_i$$
$$\left(\sum_{i=1}^{N} x_i\right) A + NB = \sum_{i=1}^{N} y_i. \tag{3.51}$$

The solution is

$$A = \frac{1}{D}\left[N\sum_{i=1}^{N}(x_i y_i) - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right)\right]$$
$$B = \frac{1}{D}\left[\left(\sum_{i=1}^{N} y_i\right)\left(\sum_{i=1}^{N} x_i^2\right) - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} x_i y_i\right)\right] \tag{3.52}$$

where:

$$D \equiv N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2. \tag{3.53}$$

Box 3.4 Robust line fitting using the RanSac method

If P denotes the set of points that we want to fit with a straight line, the algorithm is as follows.

Step 1: Select two points from P randomly and determine the line L defined by them.

Step 2: For every point in P compute its distance from line L, and then determine the subset S (**consensus set**) consisting of the points with distances less than a certain threshold E (error tolerance).

Step 3: If the number of points in the consensus set S is greater than a given threshold, compute the line that fits set S using the least square error method. Otherwise, repeat the above process until you find an appropriate consensus set S, or notify failure after a predefined number of attempts.

Step 4: If no consensus set with at least as many as the minimum acceptable number of points was found in the allowed number of attempts, select the largest consensus set found and compute the parameters of the line using it.

How do we compute the local connected fractal dimension of a binary image?

Step 1: We consider every point of the shape in turn and place a window of size M around it. It could be a circular window of radius M, or a square window of size $(2M+1)\times(2M+1)$. Inside the window we count the number of points $N(M)$ that are **connected** with the central pixel.

Step 2: We repeat it for windows of various sizes.

Step 3: We plot $\log N(M)$ versus $\log M$ and fit it with a straight line using least squares error (see Box 3.3). The slope of the line is the connected fractal dimension of the point under consideration.

Step 4: We then make a histogram of the local connected fractal dimensions of the various pixels. This histogram characterises the binary shape.

Example 3.30

Prove that when you compute fractal dimensions using logarithms it does not matter what basis you use for the logarithm.

Consider that the logarithm of a number x in base a is s and in base b is t. We shall work out the way these two logarithms are related. By definition:

$$\left.\begin{array}{l} s = \log_a x \Rightarrow x = a^s \\ t = \log_b x \Rightarrow x = b^t \end{array}\right| \Rightarrow a^s = b^t \Rightarrow s\log_b a = t \Rightarrow \log_a x \log_b a = \log_b x. \tag{3.54}$$

We note that the logarithms of the same number in two different bases differ only by a constant factor. When we use ratios of logarithms, as long as numerator and denominator are in the same base, it makes no different to the result what that base is.

Example 3.31

For each black pixel of image 3.41a calculate its local connected fractal dimension by considering squares of size 3×3, 5×5, etc. Then construct a histogram of the identified local connected fractal dimensions, the entries of which may be used to characterise the texture.

At each black pixel of the image, we place the centre of a window of size $(2M+1) \times (2M+1)$ and count how many black pixels inside the window are connected to the central pixel. We use 8-connectivity for that. We assign this number to the corresponding pixel. Figures 3.41b–f and 3.41h–l show this process. For each pixel and each window size M we consider the corresponding number $N(M)$ of pixels we found, and plot $\ln N(M)$ versus $\ln M$. The slope of the line is the local connected fractal dimension of the neighbourhood of the pixel. Since we need at least two window sizes to compute this number, we apply this process only to the pixels inside the white frame in Figure 3.41c. We identify these pixels with a sequential index shown in Figure 3.41g.

Table 3.4 shows the values of M and $N(M)$, as well as their logarithms that have to be used for the calculation of the fractal dimension.

Figure 3.42 shows the histogram of the local connected fractal dimensions, using bins 0.2 wide. We note that the dominant local connected fractal dimension is in the range $[1.4, 1.6)$, with frequency $7/24 = 0.292$.

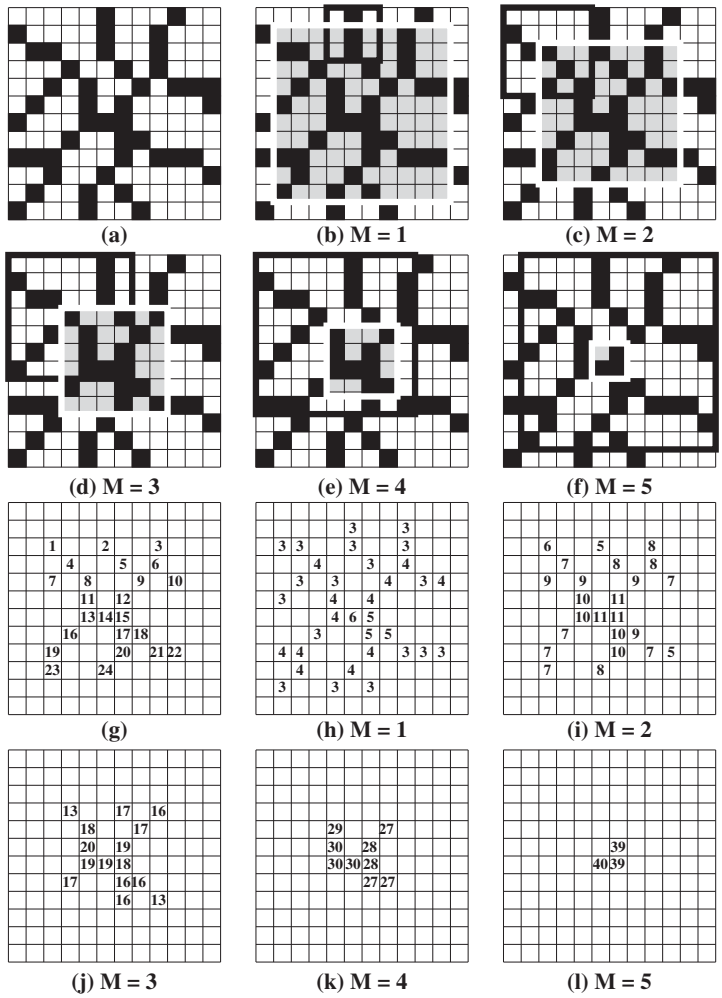
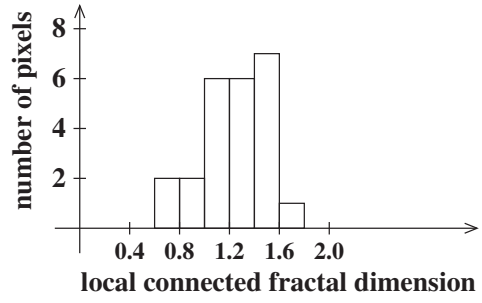


Figure 3.41 (a) An original binary image. (b)–(f) The thick black frames indicate the windows that will be used for counting the black pixels connected to the black pixel on which the centre of the window will be placed. The white margin shows the pixels that will be processed, each time omitting a margin of M pixels around the image to avoid boundary effects. The window will be placed with its centre on the top of each black pixel inside the white frame. (g) The sequential numbering of the pixels inside the white frame in panel (c). (h)–(l) The number of connected neighbours each processed pixel has, for each size neighbourhood window used.

Figure 3.42 The histogram of the local connected fractal dimensions, computed from the entries of Table 3.4. It can be used as a feature vector to characterise the structure of the binary pattern of the ganglion cell of Figure 3.41a.



(Continued)

Example 3.31 (Continued)

Table 3.4 The local connected fractal dimension of the pixels of Figure 3.41.

	$M = 1$	$M = 2$	$M = 3$	$M = 4$	$M = 5$	D_{connect}
$\log M$	0.000	0.301	0.477	0.602	0.699	
Pixel 1	3	6				
$\log N(M)$	0.477	0.778				1.000
Pixel 2	3	5				
$\log N(M)$	0.477	0.699				0.737
Pixel 3	3	8				
$\log N(M)$	0.477	0.903				1.415
Pixel 4	4	7	13			
$\log N(M)$	0.602	0.845	1.114			1.044
Pixel 5	3	8	17			
$\log N(M)$	0.477	0.903	1.230			1.561
Pixel 6	4	8	16			
$\log N(M)$	0.602	0.903	1.204			1.234
Pixel 7	3	9				
$\log N(M)$	0.477	0.954				1.585
Pixel 8	3	9	18	29		
$\log N(M)$	0.477	0.954	1.255	1.462		1.638
Pixel 9	4	9	17	27		
$\log N(M)$	0.602	0.954	1.230	1.431		1.372
Pixel 10	3	7				
$\log N(M)$	0.477	0.845				1.222
Pixel 11	4	10	20	30		
$\log N(M)$	0.602	1.000	1.301	1.477		1.466
Pixel 12	4	11	19	28	39	
$\log N(M)$	0.602	1.041	1.279	1.447	1.591	1.406
Pixel 13	4	10	19	30		
$\log N(M)$	0.602	1.000	1.279	1.477		1.451
Pixel 14	6	11	19	30	40	
$\log N(M)$	0.778	1.041	1.279	1.477	1.602	1.194
Pixel 15	5	11	18	28	39	
$\log N(M)$	0.699	1.041	1.255	1.447	1.591	1.269
Pixel 16	3	7	17			
$\log N(M)$	0.477	0.845	1.230			1.541
Pixel 17	5	10	16	27		
$\log N(M)$	0.699	1.000	1.204	1.431		1.182
Pixel 18	5	9	16	27		

Table 3.4 (Continued)

	$M = 1$	$M = 2$	$M = 3$	$M = 4$	$M = 5$	$D_{connect}$
log M	**0.000**	**0.301**	**0.477**	**0.602**	**0.699**	
$\log N(M)$	0.699	0.954	1.204	1.431		1.192
Pixel 19	4	7				
$\log N(M)$	0.602	0.845				0.807
Pixel 20	4	10	16			
$\log N(M)$	0.602	1.000	1.204			1.268
Pixel 21	3	7	13			
$\log N(M)$	0.477	0.845	1.114			1.323
Pixel 22	3	5				
$\log N(M)$	0.477	0.699				0.737
Pixel 23	4	7				
$\log N(M)$	0.602	0.845				0.807
Pixel 24	4	8				
$\log N(M)$	0.602	0.903				1.000

Example 3.32

Compute the histogram of local connected fractal dimensions for Figure 3.39.
We used 15 different window sizes, from 3×3 *to* 31×31 *(*$M = 1, 2, \ldots, 15$*). Figure 3.43 shows the histogram of the local connected fractal dimensions constructed with bin width 0.1.*

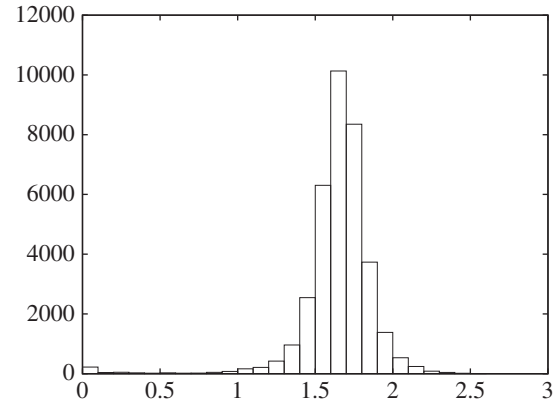


Figure 3.43 The histogram of the local connected fractal dimensions for image 3.39.

Which statistical properties remain the same at all scales in non-deterministic fractals?

The random fractals we use are characterised by self-similar second-order statistics. These are expressed either by the power spectrum, or by the autocorrelation function. Some self-affine fractals used very often are the so-called **fractional Brownian motions**. They are characterised by the self-affine scaling of the variance of their pairwise point differences.

Box 3.5 What is self-affine scaling?

When we plotted the von Koch snowflake and computed its fractal dimension, in the section What is the fractal dimension?, we treated it as a binary curve drawn in an image plane. Alternatively, we may treat it as a 1D function plotted against its independent variable. Then, we may consider that when we scale the independent variable by r, the function itself is scaled by a function of r, say $a(r)$,

$$f(rx) = a(r)f(x) \tag{3.55}$$

where $f(x)$ is then said to be a self-affine fractal function.

We shall show next that $a(r) = r^H$ where H is a constant that characterises the fractal.

Let us assume that we scale the independent variable by $r = r_1 r_2$. Then:

$$f(r_1 r_2 x) = a(r_1 r_2) f(x). \tag{3.56}$$

We may also write:

$$f(r_1 r_2 x) = f(r_1(r_2 x)) = a(r_1) f(r_2 x) = a(r_1)a(r_2)f(x). \tag{3.57}$$

By comparing (3.56) and (3.57) we deduce that:

$$a(r_1 r_2) = a(r_1)a(r_2). \tag{3.58}$$

Let us consider a function $Q(d) \equiv \ln[a(e^d)]$, where $d \equiv \ln r$. By definition, we have:

$$Q(d_1 + d_2) = \ln[a(\mathrm{e}^{d_1+d_2})]. \tag{3.59}$$

Then, by using (3.58), we obtain:

$$Q(d_1 + d_2) = \ln[a(\mathrm{e}^{d_1})a(\mathrm{e}^{d_2})] = \ln[a(\mathrm{e}^{d_1})] + \ln[a(\mathrm{e}^{d_2})] = Q(d_1) + Q(d_2). \tag{3.60}$$

From this result we conclude that if a is a continuous function, Q is a linear function, and, therefore, it must have the form $Q(d) = Hd$. So, $\ln[a(\mathrm{e}^d)] = Hd = H \ln r = \ln r^H$ and, therefore, $a(r) = r^H$. Therefore, for a self-affine fractal

$$f(x) = \frac{1}{r^H} f(rx) \tag{3.61}$$

where H is called the **Hurst dimension** of the fractal.

Box 3.6 What is the relationship between the fractal dimension and exponent *H*?

Let us consider a segment of a self-affine fractal, as shown in Figure 3.44. Let us say that we need $N(n-1)$ boxes to cover it at a certain scale $n-1$. Let us say now that we halve the size of the boxes along both axes. This means that we scale the x axis with $r = 1/2$. Then the curve

drawn in the figure will be scaled by $r^H = 2^{-H}$. How many boxes of the new size do we need to cover the curve? Let us consider a column of boxes intersected by the curve and marked grey in Figure 3.44. Halving the size of the boxes is equivalent to scaling the independent variable by 1/2. The particular vertical part of the curve after the scaling will have a span 2^{-H} times the original span. So, we shall need 2^{-H} more boxes to cover the vertical span of the curve. However, the new boxes are half the vertical size of the original ones, so we shall need twice as many: 2×2^{-H} to cover a column. The boxes are also half the size in the horizontal direction, so we shall need twice as many to cover the extent of the function along the x axis. This makes the total number of new boxes $N(n) = 2 \times 2 \times 2^{-H} N(n-1)$. If we use this equation recursively, we may write

$$N(n) = 2^{2-H} N(n-1) = \left[2^{2-H}\right]^2 N(n-2) = \ldots = \left[2^{2-H}\right]^n N(0) = \left[2^{2-H}\right]^n \tag{3.62}$$

where we have used $N(0) = 1$ as this by definition is the largest scale where a single box of side $1/2^0 = 1$ covers the whole fractal.

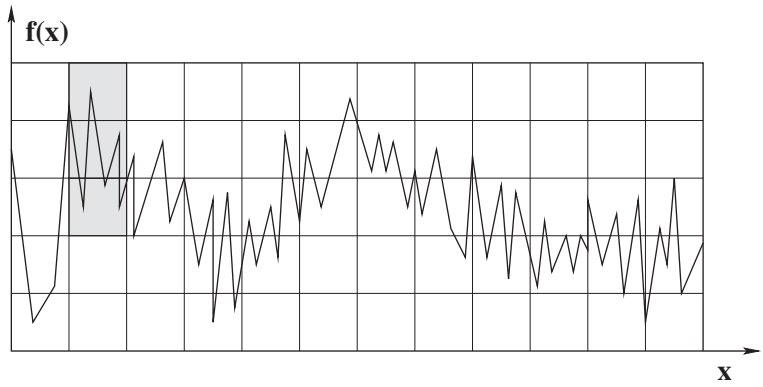


Figure 3.44 Fractal dimension and self-affinity.

The fractal dimension as defined by Equation (3.13) is given by:

$$D = \lim_{n\to\infty} \frac{\ln N(n)}{\ln 2^n} = \lim_{n\to\infty} \frac{\ln\left[2^{2-H}\right]^n}{\ln 2^n} \Rightarrow$$

$$D = \lim_{n\to\infty} \frac{\ln\left[2^n\right]^{2-H}}{\ln 2^n} \Rightarrow D = \lim_{n\to\infty} \frac{(2-H)\ln 2^n}{\ln 2^n} \Rightarrow D = 2 - H. \tag{3.63}$$

Note that if the fractal were a self-affine surface, instead of a self-affine curve, we would have had another independent variable y in addition to x. Our boxes would have been cubes rather than squares, and the columns of the function would have to be filled along the y axis with twice as many boxes as previously, so we would have had another factor of 2 appearing in (3.62). The result would have been:

$$D_{\text{surface}} = 3 - H. \tag{3.64}$$

(Continued)

Box 3.6 (Continued)

In general

$$D = D_T + 1 - H \tag{3.65}$$

where D_T is the **topological** dimension of the structure, i.e. 1 if the fractal is a curve, 2 if it is a surface and so on.

Box 3.7 What is the range of values of *H*?

If $H > 1$, when we scale the boxes down by a factor of 2 in order to compute the fractal dimension in Box 3.6, instead of needing more boxes to cover a column of the function, we would have needed fewer, since 2^{1-H} would have been less than 1. Then the curve would have looked smoother and smoother as we looked at it in more and more detail and it would have not been a fractal at all (see Example 3.22). On the other hand, if $H < 0$, Equation (3.65) would have implied that the fractal dimension of the structure would have been larger than the topological dimension by more than 1, i.e. a fractal surface would have had fractal dimension more than 3, meaning it would have been a rough volume, not a rough surface. So, the range of values of H must be $0 < H < 1$. This is an intuitive explanation of why H should be in the range $(0, 1)$. For a proper proof, based on the definition of **fractional Brownian motions**, see Box 3.8.

Example B3.33

Prove that if x and y are two random variables,

$$[E\{xy\}]^2 \leq E\{x^2\}E\{y^2\} \tag{3.66}$$

where $E\{z\}$ is the expectation value of z.

Let us consider a random variable defined as $ax - y$, for some parameter a, and write the trivially obvious identity:

$$E\{(ax - y)^2\} = a^2E\{x^2\} - 2aE\{xy\} + E\{y^2\}. \tag{3.67}$$

The left-hand side of this expression is the expectation value of non-negative numbers, so it is always non-negative. So the right-hand side should also be always non-negative, independently of the value of a. The right-hand side is a quadratic polynomial in a and in order to be always positive its discriminant must be non-positive

$$4[E\{xy\}]^2 - 4E\{x^2\}E\{y^2\} \leq 0 \tag{3.68}$$

which proves (3.66).

Example B3.34

Prove that if σ_X is the standard deviation of a zero mean random process X, then

$$\sigma_{A+B} \leq \sigma_A + \sigma_B \tag{3.69}$$

where A and B are two zero mean random processes.
The above expression may be equivalently written according to the notation of Example 3.33 as:

$$\sqrt{E\{(A+B)^2\}} \leq \sqrt{E\{A^2\}} + \sqrt{E\{B^2\}}. \tag{3.70}$$

Since all terms in the above expression are positive, we may square both sides and so (3.70) may equivalently be written as

$$E\{A^2 + B^2 + 2AB\} \leq \left(\sqrt{E\{A^2\}} + \sqrt{E\{B^2\}}\right)^2 \tag{3.71}$$

which, upon expansion of the square on the right-hand side and splitting of the terms on the left-hand side, is equivalent to

$$E\{A^2\} + E\{B^2\} + 2E\{AB\} \leq E\{A^2\} + E\{B^2\} + 2\sqrt{E\{A^2\}}\sqrt{E\{B^2\}} \tag{3.72}$$

which is equivalent to:

$$E\{AB\} \leq \sqrt{E\{A^2\}}\sqrt{E\{B^2\}}. \tag{3.73}$$

This is true according to the result of example 3.33.

Example B3.35

Consider a function

$$f(x) \equiv (1+x)^\alpha - b - x^\alpha \qquad \text{for } x > 0 \tag{3.74}$$

where b is some constant. Prove that this function is monotonic, and that it is always increasing if $\alpha > 1$ or $\alpha < 0$, while it is decreasing for $0 < \alpha < 1$.
A function is monotonic if its first derivative is always positive or always negative. Upon differentiation we obtain:

$$f'(x) = \alpha(1+x)^{\alpha-1} - \alpha x^{\alpha-1} = \alpha\left[(1+x)^{\alpha-1} - x^{\alpha-1}\right]. \tag{3.75}$$

Since $x > 0$, $1 + x > x > 0$ and for $\alpha > 1$ the quantity inside the square bracket is always positive, thus $f'(x) > 0$ always, and $f(x)$ is an increasing function of x. If $\alpha < 0$, the quantity inside the square bracket is negative and when multiplied with the negative α in front, makes the derivative again positive and the function again increasing. When $0 < \alpha < 1$, the square bracket is negative, but the α in front is positive, so $f'(x)$ is negative, which implies a decreasing function $f(x)$.

What is a fractional Brownian motion?

A fractional Brownian motion is a random function $v(t)$ that is characterised by

$$\mathcal{P}\left(\frac{v(t+\Delta t) - v(t)}{|\Delta t|^H} < y\right) = F(y) \qquad \text{for all } t \text{ and all } \Delta t \tag{3.76}$$

where $F(y)$ is a cumulative distribution function. In other words, a fractional Brownian motion is a random variable, which is being characterised not by its distribution function, but by the distribution function of the differences between the values it takes at two positions Δt apart, scaled by

$(\Delta t)^H$, where H is some parameter. Usually, $F(y)$ is assumed to be the cumulative distribution of a Gaussian probability density function with zero mean and variance σ^2:

$$F(y) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{y} \mathrm{e}^{-\frac{x^2}{2\sigma^2}} \mathrm{d}x. \tag{3.77}$$

Example B3.36

Show that:

$$\int_{-\infty}^{\infty} \mathrm{e}^{-x^2} \mathrm{d}x = \sqrt{\pi}. \tag{3.78}$$

Consider the integral:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-(x^2+y^2)} \mathrm{d}x\mathrm{d}y = \int_{-\infty}^{\infty} \mathrm{e}^{-x^2} \mathrm{d}x \int_{-\infty}^{\infty} \mathrm{e}^{-y^2} \mathrm{d}y = \left\{ \int_{-\infty}^{\infty} \mathrm{e}^{-x^2} \mathrm{d}x \right\}^2. \tag{3.79}$$

We may compute the double integral by using polar coordinates defined as $r \equiv \sqrt{x^2+y^2}$ *and* $x = r\cos\theta,\ y = r\sin\theta$, *for which* $\mathrm{d}x\mathrm{d}y = r\mathrm{d}r\mathrm{d}\theta$*:*

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-(x^2+y^2)} \mathrm{d}x\mathrm{d}y = \int_{0}^{\infty}\int_{0}^{2\pi} \mathrm{e}^{-r^2} r\mathrm{d}r\mathrm{d}\theta = 2\pi \int_{0}^{\infty} \mathrm{e}^{-r^2} r\mathrm{d}r =$$
$$2\pi \int_{0}^{\infty} \mathrm{e}^{-r^2} \mathrm{d}(-r^2) \left(-\frac{1}{2}\right) = -\pi[\mathrm{e}^{-r^2}|_0^{\infty} = \pi. \tag{3.80}$$

By equating the right-hand sides of (3.79) and (3.80) we obtain the result.

Example B3.37

Prove that

$$\int_0^{\infty} s\mathrm{e}^{-s^2} \mathrm{d}s = \frac{1}{2} \qquad \text{and} \qquad \int_{-\infty}^{\infty} s^2 \mathrm{e}^{-as^2} \mathrm{d}s = \frac{1}{2a}\sqrt{\frac{\pi}{a}}. \tag{3.81}$$

We observe that $s\mathrm{d}s = \frac{1}{2}\mathrm{d}s^2$ *and so we may write*

$$\int_0^{\infty} s\mathrm{e}^{-s^2} \mathrm{d}s = \int_0^{\infty} \mathrm{e}^{-s^2} \frac{1}{2} \mathrm{d}s^2 = \frac{1}{2}\left[-\mathrm{e}^{-s^2}\right|_0^{\infty} = \frac{1}{2} \tag{3.82}$$

which proves the first of formulae (3.81). For the second integral we proceed as follows:

$$\int_{-\infty}^{\infty} s^2 \mathrm{e}^{-as^2} \mathrm{d}s = -\frac{1}{2a} \int_{-\infty}^{\infty} s\mathrm{e}^{-as^2} \mathrm{d}(-as^2) =$$
$$-\frac{1}{2a}\left[s\mathrm{e}^{-as^2}\right|_{-\infty}^{+\infty} + \frac{1}{2a} \int_{-\infty}^{\infty} \mathrm{e}^{-as^2} \mathrm{d}s = \frac{1}{2a} \int_{-\infty}^{\infty} \mathrm{e}^{-as^2} \mathrm{d}s. \tag{3.83}$$

In the last integral we define a new variable of integration $x \equiv \sqrt{a}s$, *which implies that* $\mathrm{d}s = \mathrm{d}x/\sqrt{a}$. *Then:*

$$\int_{-\infty}^{\infty} s^2 \mathrm{e}^{-as^2} \mathrm{d}s = \frac{1}{2a\sqrt{a}} \int_{-\infty}^{\infty} \mathrm{e}^{-x^2} \mathrm{d}x. \tag{3.84}$$

The last integral is given by (3.78), so we finally obtain:

$$\int_{-\infty}^{\infty} s^2 \mathrm{e}^{-as^2} \mathrm{d}s = \frac{1}{2a\sqrt{a}}\sqrt{\pi}. \tag{3.85}$$

Example B3.38

Starting from Equation (3.76), and assuming that $F(y)$ is given by (3.77), prove that a fractional Brownian motion satisfies equation

$$E\{|v(t+\Delta t) - v(t)|\} = \sigma\sqrt{\frac{2}{\pi}}|\Delta t|^H \tag{3.86}$$

where $E\{z\}$ is the expectation value of z.
Let us define:

$$z \equiv \frac{v(t+\Delta t) - v(t)}{|\Delta t|^H} \tag{3.87}$$

Then using Equations (3.76) and (3.77) we may write:

$$\mathcal{P}(z < y) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{y} \mathrm{e}^{-\frac{x^2}{2\sigma^2}} \mathrm{d}x. \tag{3.88}$$

The problem we wish to solve is the estimation of the mean value of $|z|$. To do that we need the probability density function $p_z(z)$. This is given by the derivative of the distribution function $\mathcal{P}(z < y)$ with respect to y, which is a Gaussian, since we chose $\mathcal{P}(z < y) = F(y)$ to be the distribution function of a Gaussian. So:

$$p_z(y) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{y^2}{2\sigma^2}}. \tag{3.89}$$

Then:

$$E\{|z|\} = \int_{-\infty}^{\infty} |z| p_z(z) \mathrm{d}z = 2\int_0^{\infty} z p_z(z) \mathrm{d}z = \frac{2}{\sigma\sqrt{2\pi}} \int_0^{\infty} z \mathrm{e}^{-\frac{z^2}{2\sigma^2}} \mathrm{d}z. \tag{3.90}$$

If we change the variable of integration to $s \equiv z/\left(\sqrt{2}\sigma\right)$ and observe that the limits of s are 0 and ∞, and $\mathrm{d}z = \sqrt{2}\sigma\mathrm{d}s$, we obtain:

$$E\{|z|\} = \frac{2}{\sigma\sqrt{2\pi}} \int_0^{\infty} 2\sigma^2 s \mathrm{e}^{-s^2} \mathrm{d}s = \frac{2}{\sqrt{\pi}}\sqrt{2}\sigma \int_0^{\infty} s \mathrm{e}^{-s^2} \mathrm{d}s. \tag{3.91}$$

Then we use the first of formulae (3.81) to obtain:

$$E\{|z|\} = \sqrt{2}\sigma \frac{2}{\sqrt{\pi}} \frac{1}{2} = \sigma\sqrt{\frac{2}{\pi}}. \tag{3.92}$$

By substituting z from (3.87), we obtain Equation (3.86).

Example B3.39

Starting from equation (3.76), and assuming that $F(y)$ is given by (3.77), prove that a fractional Brownian motion satisfies equation

$$E\left\{|v(t+\Delta t)-v(t)|^2\right\}=\sigma^2|\Delta t|^{2H} \tag{3.93}$$

where $E\{z\}$ is the expectation value of z. This equation is known as the power law of the fractional Brownian motion. The quantity on the left-hand side is known as the variogram of the random process (see the section What is the variogram of an image?).

Working as in the previous example, we deduce that:

$$E\{z^2\}=\int_{-\infty}^{\infty}z^2p_z(z)\mathrm{d}z=\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty}z^2\mathrm{e}^{-\frac{z^2}{2\sigma^2}}\,\mathrm{d}z. \tag{3.94}$$

To compute this integral we apply the second one of formulae (3.81) with $a=1/(2\sigma^2)$:

$$E\{z^2\}=\frac{1}{\sigma\sqrt{2\pi}}\sigma^2\sqrt{\pi 2\sigma^2}=\sigma^2. \tag{3.95}$$

By substituting z from (3.87), we derive Equation (3.93).

Example 3.40

Use Equation (3.93) to compute the fractal dimension of the images in Figure 3.1.

We re-write equation (3.93) here to refer to pixels (i,j) and (k,l) and their grey values $g(i,j)$ and $g(k,l)$ respectively, as

$$E\left\{|g(i,j)-g(k,l)|^2\right\}=\sigma^2d^{2H} \tag{3.96}$$

where $d\equiv\sqrt{(i-k)^2+(j-l)^2}$. To use this equation to compute H, and from it D, we consider many pairs of pixels at a particular distance from each other and compute their average square difference. This corresponds to the value of the left-hand side of Equation (3.96) for a given value of d^2. We repeat this for several different values of d. We then plot the average square differences we computed versus d^2 in log–log *axes and from the slope of the curve we estimate H:*

$$\log\left(E\left\{|g(i,j)-g(k,l)|^2\right\}\right)=2\log\sigma+H\log(d^2). \tag{3.97}$$

Figure 3.45 shows the plots of the average square differences of the grey values of pairs of pixels versus the square of their distances, in log–log *coordinates, for all textures in Figure 3.1. We can immediately see from these graphs that the model of Equation (3.97) is not valid. According to this equation, the graphs should be straight lines. We observe, however, that all graphs may be fitted by a straight line with a slope different from 0, only up to a small value of distance $d=4$ or $d=\sqrt{32}\simeq 5.7$. Then the model may be fitted just to that part of each graph. Table 3.5 shows the values for H obtained by fitting Equation (3.97) to the linear part of each curve. For each value of H we compute the corresponding value of the "fractal dimension" of the texture. The term "fractal dimension" is inside quotation marks here, because it is obvious that these textures are not fractals. The fractal model, however, allows us to compute a feature for each texture (the*

value of D) which somewhat characterises the 2-point relationships between pixels up to relative distances $d \sim 5$.

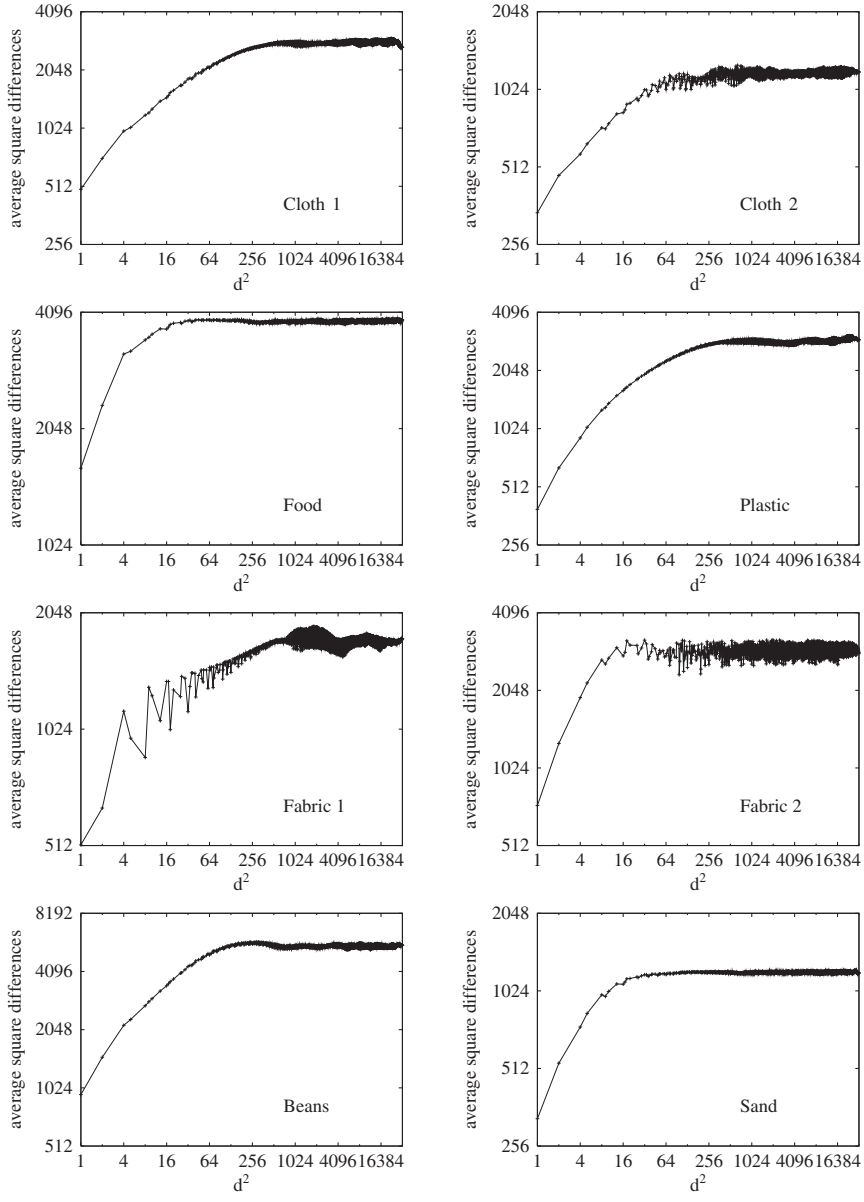


Figure 3.45 Average square differences versus d^2 for the textures of Figure 3.1. Both axes are logarithmic. The congestion of points towards the right is caused by small relative changes in the logarithm of d^2 when d is large. In reality, each point is the average of many more paired pixel values for small values of d than for larger values.

(Continued)

Example 3.40 (Continued)

Table 3.5 Values of H and D estimated from the log–log plots of the average square differences of pairs of pixels at relative distance d from each other, versus d^2. The fractal model is fitted only to the part of the curve that is linear, either up to $d \simeq 5.7$ (first two columns of results) or up to $d = 4$ (last two columns of results).

	$1 \le d^2 < 32$		$1 \le d^2 < 16$	
Image	H	D	H	D
Cloth 1	0.356020	2.643980	0.384457	2.615543
Cloth 2	0.293155	2.706845	0.318384	2.681616
Food	0.213984	2.786016	0.288067	2.711933
Plastic	0.438988	2.561012	0.503764	2.496236
Fabric 1	0.243839	2.756161	0.324784	2.675216
Fabric 2	0.369560	2.630440	0.495040	2.504960
Beans	0.411235	2.588765	0.456610	2.543390
Sand	0.333166	2.666834	0.430733	2.569267

Example 3.41

Compute the fractal dimension of the images in Figure 3.1, using the box-counting method of Box 3.2. Comment on these results in view of the conclusions of Example 3.40.

The box-counting method fits the fractal model to the signal/image as a whole. From example 3.40 it is clear that the fractal model is not applicable to these textures. So, the "fractal dimension" we shall compute by the box-counting method will not really be a fractal dimension. However, it may still be used as a feature, since it is a number that is obtained from the data and somewhat characterises the data.

Table 3.6 shows the "fractal dimension" D computed for each image for the largest values of n we used. Remember that n is the parameter that defines the size of the box we use (see Example 3.18 and Equation (3.13)). We use $n = 2, 3, 4, 5$ and 6. Although the images are of size 256×256 and values of $n = 7$ and 8 were possible, these values lead to far too small boxes, consisting of 4 or 1 pixel, respectively. They are not used as they do not offer credible image statistics. Once we have worked out the number of occupied boxes $N(n)$ for each value of n, we fit with a least squares error line the pairs of points $(\ln 2^n, \ln N(n))$. Let us say that the equation of this line is $\ln N(n) = \alpha \ln 2^n + \beta$. Then we have:

$$\lim_{n\to\infty} \frac{\ln N(n)}{\ln 2^n} = \lim_{n\to\infty} \alpha + \lim_{n\to\infty} \frac{\beta}{\ln 2^n} \Rightarrow D = \alpha. \tag{3.98}$$

So, the slope of the fitted line is the estimated fractal dimension of the image.

For a true fractal and large values of n, the computed value of D should converge to the fractal dimension.

Of course, here, what we compute is not really a fractal dimension and that is why the values of D we get are almost all near 3: the results we get are equivalent to trying to fit straight lines to the full curves of Figure 3.45. The curves are dominated by their flat parts on the right and each fitted line would have been shallower than the linear part of the curve on the left, i.e. it would produce a smaller value of H than the true H, which corresponds to the linear left part, i.e. it would result in a larger value of D than the value of D, which corresponds to the part of the curve that agrees with the fractal model.

Table 3.6 The fractal dimension computed for the textures of Figure 3.1 using the box-counting method, and stopping it for a maximum value of $n = 2, 3, 4, 5$ or 6 (see Equation (3.13)). For a true fractal, the values of D would converge to the true fractal dimension as n increases. N here is the number of boxes of size $1/2^n \times 1/2^n \times 1/2^n$ needed to cover the whole image treated as a landscape with the grey value of each pixel indicating its height. Note that the image volume is assumed to be $1 \times 1 \times 1$.

	$n=2$		$n=3$		$n=4$		$n=5$		$n=6$		
Image	N	D	N	D	N	D	N	D	N	D	**Final D**
Cloth 1	64	3.000	429	2.915	2646	2.842	14123	2.757	43768	2.570	2.6135
Cloth 2	52	2.850	334	2.795	2144	2.767	11840	2.706	40951	2.554	2.5996
Food	64	3.000	436	2.923	3051	2.894	19390	2.849	50927	2.606	2.6882
Plastic	64	3.000	396	2.876	2522	2.825	13094	2.735	39388	2.544	2.5956
Fabric 1	62	2.977	403	2.885	2385	2.805	10896	2.682	34504	2.512	2.5439
Fabric 2	63	2.989	418	2.902	2990	2.886	17041	2.811	47169	2.588	2.6638
Beans	64	3.000	507	2.995	3473	2.940	17204	2.814	44434	2.573	2.6398
Sand	59	2.941	378	2.854	2342	2.798	11387	2.695	36828	2.528	2.5675

Example B3.42

Prove that the increments of a fractional Brownian motion constitute a self-affine fractal in a statistical sense.

Let us define $\Delta v(\Delta t) \equiv v(t + \Delta t) - v(t)$. Let us also consider scaling Δt by a factor $r > 0$. Then from (3.93), on page 138, we have:

$$\begin{aligned} E\left\{\Delta v(\Delta t)^2\right\} &= \sigma^2 |\Delta t|^{2H} \\ \Rightarrow E\left\{\Delta v(r\Delta t)^2\right\} &= \sigma^2 |r\Delta t|^{2H} \\ &= r^{2H}\sigma^2 |\Delta t|^{2H} \\ &= r^{2H} E\left\{\Delta v(\Delta t)^2\right\} \\ \Rightarrow \sqrt{E\left\{\Delta v(r\Delta t)^2\right\}} &= r^H \sqrt{E\left\{\Delta v(\Delta t)^2\right\}}. \end{aligned} \tag{3.99}$$

The last expression shows that the square root of the mean square increment scales like an affine fractal with Hurst dimension H (see Equation (3.61)).

Box 3.8 Prove that the range of values of *H* for a fractional Brownian motion is (0, 1)

Consider the fractional Brownian motion $v(t)$ and its values at arguments $t+\tau_1+\tau_2$, $t+\tau_1$ and t, where τ_1 and τ_2 are some positive increments of the independent variable. The increments of the function are random variables. We observe that $v(t+\tau_1+\tau_2)-v(t) = \left[v(t+\tau_1+\tau_2)-v(t+\tau_1)\right]+\left[v(t+\tau_1)-v(t)\right]$. Then we may apply the result of Example 3.34 to the standard deviations of these three random variables:

$$\sqrt{E\left\{\left[v(t+\tau_1+\tau_2)-v(t)\right]^2\right\}} \leq \sqrt{E\left\{\left[v(t+\tau_1+\tau_2)-v(t+\tau_1)\right]^2\right\}} + \sqrt{E\left\{\left[v(t+\tau_1)-v(t)\right]^2\right\}}. \tag{3.100}$$

We make use of Equation (3.93) to write:

$$\begin{aligned}&\sigma(\tau_1+\tau_2)^H \leq \sigma\tau_2^H+\sigma\tau_1^H \Leftrightarrow\\ &(\tau_1+\tau_2)^H \leq \tau_2^H+\tau_1^H.\end{aligned} \tag{3.101}$$

Let us define $s \equiv \tau_2/\tau_1$. Then the last expression may equivalently be written as

$$\begin{aligned}&\tau_1^H(1+s)^H \leq \tau_1^H(1+s^H) \Leftrightarrow\\ &(1+s)^H \leq 1+s^H \Leftrightarrow\\ &(1+s)^H-1-s^H \leq 0\end{aligned} \tag{3.102}$$

where we made use of the fact that $\sigma, \tau_1, \tau_2 > 0$.

Let us consider function $f(s) \equiv (1+s)^H-1-s^H$. This is the same as the function of Example 3.35 for $b=1$. Note that $f(0)=0$ and this function will be decreasing if $0<H<1$, according to the result of Example 3.35. Function $f(s)$ being decreasing means that if $s>0, f(s)<f(0)=0$. So, for (3.102) to be valid, H must be in the range $(0,1)$.

Example B3.43

Prove that

$$2(a-b)(c-d) \equiv (a-d)^2+(b-c)^2-(a-c)^2-(b-d)^2 \tag{3.103}$$

where a, b, c and d are some real numbers.

The proof is trivial if we expand the squares on the right-hand side and factorise the remaining terms after the cancellation of the squares.

Box 3.9 What is the correlation between two increments of a fractional Brownian motion?

Consider a point t and three increments τ_1, τ_2 and τ_3, such that $\tau_3 > \tau_2 > \tau_1 > 0$. The correlation between two different increments is:

$$E\left\{\left[v(t+\tau_3)-v(t+\tau_2)\right]\left[v(t+\tau_1)-v(t)\right]\right\}. \tag{3.104}$$

According to (3.103), this may be re-written as:

$$\frac{1}{2}E\left\{\left[v(t+\tau_3)-v(t)\right]^2+\left[v(t+\tau_2)-v(t+\tau_1)\right]^2-\left[v(t+\tau_3)-v(t+\tau_1)\right]^2-\left[v(t+\tau_2)-v(t)\right]^2\right\}=$$
$$\frac{1}{2}E\left\{\left[v(t+\tau_3)-v(t)\right]^2\right\}+\frac{1}{2}E\left\{\left[v(t+\tau_2)-v(t+\tau_1)\right]^2\right\}-\frac{1}{2}E\left\{\left[v(t+\tau_3)-v(t+\tau_1)\right]^2\right\}-\frac{1}{2}E\left\{\left[v(t+\tau_2)-v(t)\right]^2\right\}=$$
$$\frac{\sigma^2}{2}\left[\tau_3^{2H}+|\tau_2-\tau_1|^{2H}-|\tau_3-\tau_1|^{2H}-\tau_2^{2H}\right]. \tag{3.105}$$

The last expression was derived because of (3.93).

Consider first $H=1/2$. The right-hand side of (3.105) is zero, and therefore the increments of the fractional Brownian motion are uncorrelated.

In general however,

$$E\left\{\left[v(t+\tau_3)-v(t+\tau_2)\right]\left[v(t+\tau_1)-v(t)\right]\right\}=$$
$$\frac{\sigma^2}{2}\tau_1^{2H}\left[s_3^{2H}+(s_2-1)^{2H}-(s_3-1)^{2H}-s_2^{2H}\right] \tag{3.106}$$

where $s_3\equiv\tau_3/\tau_1>1$ and $s_2\equiv\tau_2/\tau_1>1$.

Let us define some new variables $u_3\equiv s_3-1$ and $u_2\equiv s_2-1$. Then we have:

$$E\left\{\left[v(t+\tau_3)-v(t+\tau_2)\right]\left[v(t+\tau_1)-v(t)\right]\right\}=$$
$$\frac{\sigma^2}{2}\tau_1^{2H}\left[(1+u_3)^{2H}-u_3^{2H}-((1+u_2)^{2H}-u_2^{2H})\right]. \tag{3.107}$$

Since $\tau_2<\tau_3$, then $s_2<s_3$, and $s_2-1<s_3-1$ and $u_2<u_3$. Consider function $f(u)\equiv(1+u)^{2H}-u^{2H}$. This is the same as the function of Example 3.35 for $b=0$ and $\alpha=2H$. For $0<2H<1$, therefore, this function is decreasing and, when $u_2<u_3$, $f(u_2)>f(u_3)$ and the function on the right-hand side of (3.107) is negative. When $2H>1$, $f(u)$ is increasing, and so for $u_2<u_3$, we have $f(u_2)<f(u_3)$, which implies a positive correlation.

We conclude, therefore, that the **increments** of a fractional Brownian motion are negatively correlated (i.e. anti-correlated) for $0<H<1/2$, uncorrelated for $H=1/2$ and positively correlated for $1/2<H<1$. The case $H=1/2$ is known as **Brownian motion** or the **Wiener process**.

Example B3.44

Show that for a discretised fractional Brownian motion, with Hurst exponent H, the increment between the first two samples is correlated with the increment between samples k and $k+1$ according to:

$$\rho(k)=\frac{\sigma^2}{2}\left[(k+1)^{2H}+(k-1)^{2H}-2k^{2H}\right] \tag{3.108}$$

where σ^2 is the variance of the samples.

Let us consider the increment between the first two samples, by setting $t=0$ and $\tau_1=1$ on the left-hand side of (3.105), and its correlation with the increment between samples k and $k+1$ by setting $\tau_2=k$ and $\tau_3=k+1$. Then (3.108) follows trivially from (3.105).

Example B3.45

Show that for $k \to \infty$ the correlation between the increments of a fractional Brownian motion, as worked out in Example 3.44, goes like:

$$\rho(k) \simeq \sigma^2 H(2H-1)k^{2H-2}. \tag{3.109}$$

When $k \to \infty$, $\frac{1}{k} \to 0$. So, we may re-arrange (3.108) to isolate the small parameter and then use Taylor expansion to work out how $\rho(k)$ behaves at that limit:

$$\begin{aligned}\rho(k) &= \frac{\sigma^2}{2}\left[k^{2H}\left(1+\frac{1}{k}\right)^{2H} + k^{2H}\left(1-\frac{1}{k}\right)^{2H} - 2k^{2H}\right] \\ &\simeq \frac{\sigma^2 k^{2H}}{2}\left[1+2H\frac{1}{k}+\frac{2H(2H-1)}{2}\frac{1}{k^2}+1-2H\frac{1}{k}+\frac{2H(2H-1)}{2}\frac{1}{k^2}-2\right] \\ &= \sigma^2 k^{2H-2}H(2H-1).\end{aligned} \tag{3.110}$$

Example B3.46

A random process is said to exhibit long range dependence if its increments are stationary and

$$\sum_k \frac{\rho(k)}{\sigma^2} \to \infty \tag{3.111}$$

where $\rho(k)$ is the autocorrelation function of the increments of the process and σ^2 the variance of the increments. Show that the fractional Brownian motion for $1/2 < H < 1$ exhibits long range dependence.

We know that the increments of the fractional Brownian motion are stationary. The long range behaviour of the autocorrelation function is according to Equation (3.110). We note that for $1/2 < H < 1$, $-1 < 2H-2 < 0$ while $0 < 2H-1 < 1$. So, the correlation coefficient $\rho(k)/\sigma^2$ is positive and decays very slowly with k, more slowly than $1/k$, which corresponds to the harmonic series $\sum_k(1/k)$, which diverges. So, $\sum_k \rho(k)/\sigma^2$ also diverges, and the fractional Brownian motion exhibits long range dependence for $1/2 < H < 1$.

Example B3.47

Calculate the autocorrelation matrix A between two one-step increments at two different locations, for a fractional Brownian motion of Hurst exponent H.

Let us consider an element A_{lk} of the matrix, expressing the correlation between increments l and $l+1$, and k and $k+1$. In (3.105) we set:

$$\begin{aligned} t &= k \\ \tau_1 &= 1 \\ t+\tau_2 &= l \Rightarrow \tau_2 = l-k \\ t+\tau_3 &= l+1 \Rightarrow \tau_3 = l-k+1. \end{aligned} \tag{3.112}$$

Then:

$$A_{lk} = E\{[v(l+1) - v(l)][v(k+1) - v(k)]\}$$
$$= \frac{\sigma^2}{2}\left[|l-k+1|^{2H} + |l-k-1|^{2H} - 2|l-k|^{2H}\right]. \quad (3.113)$$

Example B3.48

A Toeplitz matrix is a diagonal constant matrix, i.e. a matrix for which $A_{ij} = A_{i-1,j-1}$. Show that the autocorrelation matrix of a fractional Brownian motion is a symmetric Toeplitz matrix.

We note from (3.113) that A_{lk} depends only on $l - k \equiv \alpha$, which is constant along the diagonals of the matrix, as shown in Figure 3.46. The elements along each diagonal are given by:

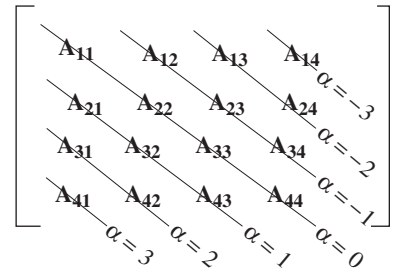
$$A_{lk} = \frac{\sigma^2}{2}\left[|\alpha+1|^{2H} + |\alpha-1|^{2H} - 2|\alpha|^{2H}\right]. \quad (3.114)$$

Note also that if we replace α with $-\alpha$, we have:

$$A_{kl} = \frac{\sigma^2}{2}\left[|-\alpha+1|^{2H} + |-\alpha-1|^{2H} - 2|-\alpha|^{2H}\right]. \quad (3.115)$$

By comparing the right-hand sides of (3.114) and (3.115), we conclude that $A_{kl} = A_{lk}$, i.e. that the matrix is symmetric.

Figure 3.46 The value of α, which appears in (3.114), is constant along each diagonal of the matrix.



Example B3.49

Show that if the autocorrelation matrix of a process is written as $A = LL^T$, and v is a vector made up from samples that form white noise, with unit variance, the sequence created by multiplying vector v from the left with L will have autocorrelation matrix A.

(Continued)

Example B3.49 (Continued)

White noise means that the autocorrelation matrix of the samples is diagonal with values along the diagonal equal to the variance of the noise (see Book I). So, we may write

$$E\{\mathbf{v}\mathbf{v}^T\} = \sigma^2 I = I \tag{3.116}$$

since $\sigma^2 = 1$ for the noise we consider. The new random sequence will consist of the elements of matrix $L\mathbf{v}$. Its autocorrelation matrix will be:

$$E\{L\mathbf{v}(L\mathbf{v})^T\} = E\{L\mathbf{v}\mathbf{v}^T L^T\} = LE\{\mathbf{v}\mathbf{v}^T\}L^T = LIL^T = A. \tag{3.117}$$

Example B3.50

The *Cholesky decomposition* of a symmetric semi-definite matrix A is $A = LL^T$, where L is a lower triangular matrix. Work out the formulae that will allow you to perform the Cholesky decomposition of a 4×4 matrix A.

Let us define matrix L as

$$L \equiv \begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix}. \tag{3.118}$$

Then we have:

$$\begin{aligned} A &= \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = LL^T \\ &= \begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} & l_{41} \\ 0 & l_{22} & l_{32} & l_{42} \\ 0 & 0 & l_{33} & l_{43} \\ 0 & 0 & 0 & l_{44} \end{pmatrix} \\ &= \begin{pmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} & l_{11}l_{41} \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32} & l_{21}l_{41} + l_{22}l_{42} \\ l_{31}l_{11} & l_{31}l_{21} + l_{32}l_{22} & l_{31}^2 + l_{32}^2 + l_{33}^2 & l_{31}l_{41} + l_{32}l_{42} + l_{33}l_{43} \\ l_{41}l_{11} & l_{41}l_{21} + l_{42}l_{22} & l_{41}l_{31} + l_{42}l_{32} + l_{43}l_{33} & l_{41}^2 + l_{42}^2 + l_{43}^2 + l_{44}^2 \end{pmatrix}. \end{aligned} \tag{3.119}$$

We note that:

$$\begin{aligned} &l_{11} = \sqrt{a_{11}} \\ &l_{21} = \frac{a_{12}}{l_{11}} \qquad l_{31} = \frac{a_{13}}{l_{11}} \qquad l_{41} = \frac{a_{14}}{l_{11}} \\ &l_{22} = \sqrt{a_{22} - l_{21}^2} \qquad l_{32} = \frac{1}{l_{22}}(a_{23} - l_{21}l_{31}) \qquad l_{42} = \frac{1}{l_{22}}(a_{24} - l_{21}l_{41}) \\ &l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} \qquad l_{43} = \frac{1}{l_{33}}(a_{34} - l_{31}l_{41} - l_{32}l_{42}) \end{aligned}$$

$$l_{44} = \sqrt{a_{44} - l_{41}^2 - l_{42}^2 - l_{43}^2}. \tag{3.120}$$

These formulae may be generalised to:

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$$

$$l_{ij} = \frac{1}{l_{jj}}\left(a_{ji} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\right) \qquad \text{for } i > j. \tag{3.121}$$

*These formulae constitute the **Cholesky algorithm**. They can be used to obtain the Cholesky decomposition of any symmetric positive semi-definite matrix A.*

Example B3.51

Show that you may achieve the Cholesky decomposition of a positive semi-definite 4×4 matrix A by considering the decomposition:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \begin{pmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{pmatrix} \begin{pmatrix} 1 & l_{21} & l_{31} & l_{41} \\ 0 & 1 & l_{32} & l_{42} \\ 0 & 0 & 1 & l_{43} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.122}$$

Let us calculate the right-hand side of (3.122):

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \begin{pmatrix} d_1 & d_1 l_{21} & d_1 l_{31} & d_1 l_{41} \\ 0 & d_2 & d_2 l_{32} & d_2 l_{42} \\ 0 & 0 & d_3 & d_3 l_{43} \\ 0 & 0 & 0 & d_4 \end{pmatrix} =$$

$$\begin{pmatrix} d_1 & d_1 l_{21} & d_1 l_{31} & d_1 l_{41} \\ l_{21} d_1 & d_1 l_{21}^2 + d_2 & l_{21} d_1 l_{31} + d_2 l_{32} & l_{21} d_1 l_{41} + d_2 l_{42} \\ l_{31} d_1 & l_{31} d_1 l_{21} + l_{32} d_2 & l_{31}^2 d_1 + l_{32}^2 d_2 + d_3 & l_{31} d_1 l_{41} + l_{32} d_2 l_{42} + d_3 l_{43} \\ l_{41} d_1 & l_{41} d_1 l_{21} + l_{42} d_2 & l_{41} d_1 l_{31} + l_{42} d_2 l_{32} + l_{43} d_3 & l_{41}^2 d_1 + l_{42}^2 d_2 + l_{43}^2 d_3 + d_4 \end{pmatrix}. \tag{3.123}$$

Then we have:

$$d_1 = a_{11} \qquad l_{21} = \frac{a_{12}}{d_1} \qquad l_{31} = \frac{a_{13}}{d_1} \qquad l_{41} = \frac{a_{14}}{d_1}$$

$$d_2 = a_{22} - d_1 l_{21}^2 \qquad l_{32} = \frac{1}{d_2}(a_{23} - l_{21} d_1 l_{31}) \qquad l_{42} = \frac{1}{d_2}(a_{24} - l_{21} d_1 l_{41})$$

$$d_3 = a_{33} - l_{31}^2 d_1 - l_{32}^2 d_2 \qquad l_{43} = \frac{1}{d_3}(a_{34} - l_{31} d_1 l_{41} - l_{32} d_2 l_{42})$$

$$d_4 = a_{44} - l_{41}^2 d_1 - l_{42}^2 d_2 - l_{43}^2 d_3. \tag{3.124}$$

(Continued)

Example B3.51 (Continued)

These formulae may be generalised as follows.

$$d_i = a_{ii} - \sum_{k=1}^{i-1} d_k l_{ik}^2$$
$$l_{ij} = \frac{1}{d_j}\left(a_{ji} - \sum_{k=1}^{j-1} d_k l_{ik} l_{jk}\right) \qquad \text{for } i > j. \tag{3.125}$$

Finally, if we define $D^{\frac{1}{2}}$ to mean a diagonal matrix with values along the diagonal the square roots of the values of matrix D, then we may write:

$$A = LDL^T = LD^{\frac{1}{2}}D^{\frac{1}{2}}L^T = \left(LD^{\frac{1}{2}}\right)\left(D^{\frac{1}{2}}L^T\right) = \left(LD^{\frac{1}{2}}\right)\left(LD^{\frac{1}{2}}\right)^T. \tag{3.126}$$

Obviously then, the Cholesky decomposition of A is with the help of the lower triangular matrix $LD^{\frac{1}{2}}$.

These formulae may be used to calculate the Cholesky decomposition of any symmetric semi-positive definite matrix A, without having to take the square roots of differences like when using the algorithm in Example 3.50.

Box 3.10 Synthesising a 1D fractal

Step 0: Select the size of the fractal you wish to construct. Call it N.
Step 1: Calculate the autocorrelation matrix, of size $N \times N$, of the fractal you wish to construct, using (3.113). Call it A.
Step 2: Apply Cholesky decomposition to matrix A, so you can write it as $A = LL^T$ (see Box 3.11).
Step 3: Construct a random sequence of white noise with variance 1, consisting of N samples. Call it $\mathbf{v}$.
Step 4: Multiply $\mathbf{v}$ from the left with L to obtain the fractal sequence.

To construct a random sequence of white noise with variance σ^2, do the following:

Step 0: Select the length of the sequence to be an odd number, let us say $2M + 1$.
Step 1: Select $2M + 1$ random numbers $p(m)$, uniformly distributed in the range $[0^\circ, 360^\circ)$.
Step 2: Construct the sequences:

$$\begin{aligned} F_R(m) &= \sigma\cos(p(n)) \quad \text{for } -M \le m \le M \\ F_I(m) &= \sigma\sin(p(n)) \quad \text{for } 0 \le m \le M \\ F_I(m) &= -F_I(-m) \quad \text{for } -M \le m < 0. \end{aligned} \tag{3.127}$$

Step 3: Treat sequences $F_R(m)$ and $F_I(m)$ as the real and the imaginary parts of the discrete Fourier transform (DFT) of a signal made up of $2M + 1$ samples, and take the inverse DFT. The result is a sequence of white noise.

Figure 3.47 shows two fractal signals consisting of 500 samples each, constructed with this algorithm with $H = 0.1$ and $H = 0.9$.

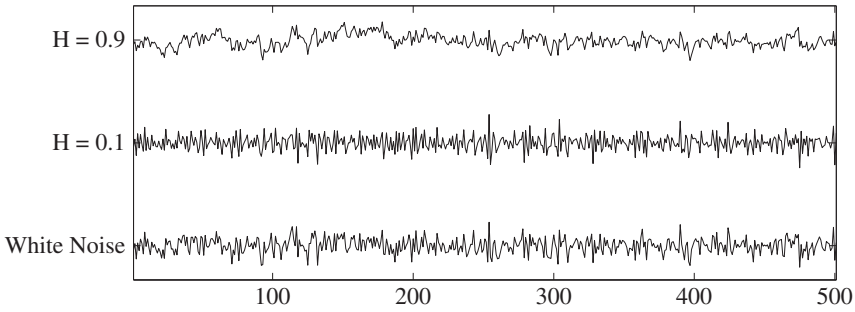


Figure 3.47 Synthesising 1D fractals from a sequence of white noise with $\sigma = 1$.

Box 3.11 The Shur algorithm for Cholesky decomposition of a symmetric Toeplitz matrix

The Shur algorithm for Cholesky decomposition of a symmetric Toeplitz matrix is the result of theoretical considerations that are beyond the scope of this book. So, it is given here only as a recipe without any justification of the various steps it involves.

Step 0: Assume that the matrix you wish to decompose is $N \times N$:

$$T = \begin{pmatrix} 1 & t_1 & t_2 & \cdots & t_{N-1} \\ t_1 & 1 & t_1 & \cdots & t_{N-2} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ t_{N-1} & t_{N-2} & t_{N-3} & \cdots & 1 \end{pmatrix}. \tag{3.128}$$

Step 1: Construct the $2 \times N$ matrix G:

$$G = \begin{pmatrix} 1 & t_1 & t_2 & \cdots & t_{N-1} \\ 0 & t_1 & t_2 & \cdots & t_{N-1} \end{pmatrix}. \tag{3.129}$$

Step 2: Construct $N \times N$ matrix C, with the first row the same as that of matrix G, and all the rest 0:

$$C = \begin{pmatrix} 1 & t_1 & t_2 & \cdots & t_{N-1} \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}. \tag{3.130}$$

Step 3: Shift one position to the right the first row of matrix G, removing the rightmost element and inserting a 0 on the left. Call the result G':

$$G' = \begin{pmatrix} 0 & 1 & t_1 & t_2 & \cdots & t_{N-2} \\ 0 & t_1 & t_2 & t_3 & \cdots & t_{N-1} \end{pmatrix}. \tag{3.131}$$

Step 4: Assume that the columns of matrix G' are indexed with i, taking values from 1 to N. For $i = 2$ to N, inclusive, do the following.

(Continued)

Box 3.11 (Continued)

Step 4.1: Set:

$$\rho = -\frac{G'_{2i}}{G'_{1i}}. \tag{3.132}$$

Step 4.2: Create matrix:

$$R = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}. \tag{3.133}$$

Step 4.3: Multiply from the left with matrix R the submatrix of G' formed by the columns i to N inclusive. Call the result $\hat{G}$.

Step 4.4: Divide each element of $\hat{G}$ by $\sqrt{1-\rho^2}$. Call the result $\tilde{G}$.

Step 4.5: Replace the columns of G' you used to produce $\hat{G}$ with $\tilde{G}$. Call this new G.

Step 4.6: Copy the elements of the new G in the first row from column i to column N, in the corresponding positions of row i of matrix C.

Step 4.7: Shift the first row of matrix G one position to the right, removing the right-most element and inserting a 0 on the left, to create new matrix G'.

Step 5: At the end, matrix C is the L^T matrix of the Cholesky decomposition of the original matrix T.

Example B3.52

Use the algorithm of Box 3.11 to construct the Cholesky decomposition of matrix:

$$T = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.5 \\ 0.3 & 0.5 & 1 \end{pmatrix}. \tag{3.134}$$

We first construct matrices G and C:

$$G = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0 & 0.5 & 0.3 \end{pmatrix} \qquad C = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.135}$$

Then form matrix G':

$$G' = \begin{pmatrix} 0 & 1 & 0.5 \\ 0 & 0.5 & 0.3 \end{pmatrix}. \tag{3.136}$$

Set $i = 2$. Compute $\rho = -0.5/1 = -0.5$ and form matrix R:

$$R = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}. \tag{3.137}$$

Construct matrix $\hat{G}$:

$$\hat{G} = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0.5 \\ 0.5 & 0.3 \end{pmatrix} = \begin{pmatrix} 0.75 & 0.35 \\ 0.00 & 0.05 \end{pmatrix}. \tag{3.138}$$

Divide every element of $\hat{G}$ *by* $\sqrt{1-0.5^2} = \sqrt{0.75} = 0.866$*, to form matrix* $\tilde{G}$*:*

$$\tilde{G} = \begin{pmatrix} 0.87 & 0.40 \\ 0.00 & 0.06 \end{pmatrix}. \tag{3.139}$$

Create a new matrix G:

$$G = \begin{pmatrix} 0 & 0.87 & 0.40 \\ 0 & 0.00 & 0.06 \end{pmatrix}. \tag{3.140}$$

Update matrix C:

$$C = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0 & 0.87 & 0.40 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.141}$$

Create from new matrix G the new matrix G'*:*

$$G' = \begin{pmatrix} 0 & 0 & 0.87 \\ 0 & 0 & 0.06 \end{pmatrix}. \tag{3.142}$$

Now set $i = 3$*. Compute* $\rho = -0.06/0.87 = -0.07$ *and form matrix R:*

$$R = \begin{pmatrix} 1 & -0.07 \\ -0.07 & 1 \end{pmatrix}. \tag{3.143}$$

Construct new matrix $\hat{G}$*:*

$$\hat{G} = \begin{pmatrix} 1 & -0.07 \\ -0.07 & 1 \end{pmatrix} \begin{pmatrix} 0.87 \\ 0.06 \end{pmatrix} = \begin{pmatrix} 0.87 \\ 0.00 \end{pmatrix}. \tag{3.144}$$

Divide every element of $\hat{G}$ *by* $\sqrt{1-0.07^2} = 0.998$*, to form matrix* $\tilde{G}$*:*

$$\tilde{G} = \begin{pmatrix} 0.87 \\ 0.00 \end{pmatrix}. \tag{3.145}$$

Create a new matrix G:

$$G = \begin{pmatrix} 0 & 0 & 0.87 \\ 0 & 0 & 0.00 \end{pmatrix}. \tag{3.146}$$

Update matrix C:

$$C = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0 & 0.87 & 0.40 \\ 0 & 0 & 0.87 \end{pmatrix}. \tag{3.147}$$

To check the validity of the result, we multiply $C^T C$*, i.e.* LL^T*, and check whether we get matrix T:*

$$LL^T = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 0.87 & 0 \\ 0.3 & 0.4 & 0.87 \end{pmatrix} \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0 & 0.87 & 0.40 \\ 0 & 0 & 0.87 \end{pmatrix} = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1.01 & 0.5 \\ 0.3 & 0.5 & 1.01 \end{pmatrix}. \tag{3.148}$$

This is equal to the original matrix T within the accuracy of the calculation.

Box 3.12 What is the power spectrum of a fractal?

Given that the fractal is a non-stationary process with stationary increments, the power spectrum of the fractal function itself cannot be defined. Nevertheless, we may define a **generalised** power spectrum and work out its scaling property.

The scaling property of the Fourier transform tells us that if $F(\omega)$ is the Fourier transform of a function $f(x)$, then the Fourier transform of function $f(rx)$ is $\frac{1}{r}F\left(\frac{\omega}{r}\right)$ (see the book Image Processing, The Fundamentals [75]). We also know that the Fourier transform of $\alpha f(x)$ is $\alpha F(\omega)$. So, the Fourier transform of $r^{-H}f(rx)$ is $\frac{1}{r^{H+1}}F\left(\frac{\omega}{r}\right)$. Therefore, we have the following Fourier transform pairs:

$$f(x) \leftrightarrow F(\omega)$$

$$f(rx) \leftrightarrow \frac{1}{r}F\left(\frac{\omega}{r}\right) \tag{3.149}$$

$$\frac{1}{r^H}f(rx) \leftrightarrow \frac{1}{r^{H+1}}F\left(\frac{\omega}{r}\right). \tag{3.150}$$

If $f(x)$ is a fractal function, then $r^{-H}f(rx) = f(x)$, according to Equation (3.61). So,

$$\begin{aligned} F(\omega) &= \frac{1}{r^{H+1}}F\left(\frac{\omega}{r}\right) \\ F^*(\omega) &= \frac{1}{r^{H+1}}F^*\left(\frac{\omega}{r}\right) \\ F(\omega)F^*(\omega) &= \frac{1}{r^{2H+2}}F\left(\frac{\omega}{r}\right)F^*\left(\frac{\omega}{r}\right) \\ |F(\omega)|^2 &= \frac{1}{r^{2H+2}}\left|F\left(\frac{\omega}{r}\right)\right|^2. \end{aligned} \tag{3.151}$$

To estimate the power spectral density of the fractal function, we make use of the periodogram, i.e. we consider a finite range X of the domain of the function and allow X to tend to infinity:

$$P(\omega) \equiv \lim_{X\to\infty}\frac{1}{X}|F(\omega)|^2. \tag{3.152}$$

Substituting $|F(\omega)|^2$ on the right-hand side of (3.152) from (3.151), we obtain:

$$P(\omega) = \lim_{X\to\infty}\frac{1}{X}\frac{1}{r^{2H+2}}\left|F\left(\frac{\omega}{r}\right)\right|^2. \tag{3.153}$$

However, when $0 \le x \le X$, $0 \le rx \le rX$. So, we may define a new variable $Y \equiv rX$, over which we shall take the power spectrum of the scaled function. As $X \to \infty$, $Y \to \infty$ too:

$$\begin{aligned} P(\omega) &= \lim_{X\to\infty}\frac{1}{Xr}\frac{1}{r^{2H+1}}\left|F\left(\frac{\omega}{r}\right)\right|^2 \\ &= \lim_{Y\to\infty}\frac{1}{Y}\frac{1}{r^{2H+1}}\left|F\left(\frac{\omega}{r}\right)\right|^2 \\ &= \frac{1}{r^{2H+1}}P\left(\frac{\omega}{r}\right). \end{aligned} \tag{3.154}$$

We may therefore write:

$$P(\alpha\omega) = \frac{1}{\alpha^{2H+1}}P(\omega). \tag{3.155}$$

We note that the power at 2ω is 2^{2H+1} times smaller than the power at ω, the power at 3ω is 3^{2H+1} times smaller than the power at ω and so on. We conclude, therefore, that for a fractal:

$$P(\omega) \propto \omega^{-(2H+1)}. \tag{3.156}$$

We define:

$$\beta \equiv 2H + 1 \tag{3.157}$$

Exponent β characterises the spectrum of a fractal and it may be used to derive the fractal dimension. We may write

$$P_f(\omega) = P_f(1)\omega^{-\beta} \Rightarrow \log P_f(\omega) = \log P_f(1) - \beta \log \omega \tag{3.158}$$

where $\omega > 0$.

The power spectral density of the function is plotted as a function of the angular frequency ω in log–log coordinates and the slope of the line $(-\beta)$ is computed. The fractal dimension then is derived by using Equations (3.157) and (3.63):

$$D = 2 - H = 2 - \frac{\beta - 1}{2} = \frac{5 - \beta}{2}. \tag{3.159}$$

This relationship holds for fractal curves. In the general case we must use Equation (3.65). If D_T is the topological dimension of the space we are working, the scaling property of the Fourier transform in pair (3.149) will insert factor $1/r^{D_T}$ (instead of simply $1/r$) on the right-hand side of the pair. This will result in factor $1/r^{2H+2D_T}$ (instead of $1/r^{2H+2}$) on the right-hand side of (3.151). Finally, this will be reduced to $1/r^{2H+D_T}$ when the limits of the ranges of all the independent variables are taken in (3.154). So, in the general case,

$$P(\omega) \propto \omega^{-\beta} \qquad \text{with } \beta = 2H + D_T \tag{3.160}$$

where the relationship between fractal dimension D, topological dimension D_T, Hurst exponent H and parameter β is:

$$D = D_T + 1 - H = D_T + 1 - \frac{\beta - D_T}{2} \Rightarrow D = \frac{3D_T + 2 - \beta}{2}. \tag{3.161}$$

Note that Equation (3.160) is valid if the multidimensional fractal is isotropic, i.e. scaling any one of its independent variables by r results in scaling the function by r^H. In other words it assumes that the power spectrum of the fractal is the same along all directions.

Box 3.13 What is the autocorrelation function of a fractal?

Accepting that a fractal function is not stationary, we cannot define its autocorrelation function. However, as it has stationary increments, we can define the autocorrelation function of its increments, as we have done in Box 3.9. Nevertheless, having defined the generalised power spectrum of a fractal in Box 3.12, we can work out the scaling properties of its autocorrelation function too. According to the Wiener–Khinchine theorem, (see Example 3.109), the Fourier transform of the autocorrelation function of a function is the power spectrum of the function, or the inverse Fourier transform of the power spectrum is the autocorrelation function. So, we

(Continued)

Box 3.13 (Continued)

may write for the autocorrelation function $\rho(x)$ and the generalised power spectrum $P(\omega)$:

$$\rho(x) = \int_{-\infty}^{+\infty} P(\omega) \mathrm{e}^{\mathrm{j}\omega x} \mathrm{d}\omega. \tag{3.162}$$

The scaled version of $\rho(x)$ is:

$$\rho(rx) = \int_{-\infty}^{+\infty} P(\omega) \mathrm{e}^{\mathrm{j}\omega r x} \mathrm{d}\omega. \tag{3.163}$$

Let us define a new variable $z \equiv r\omega$:

$$\rho(rx) = \int_{-\infty}^{+\infty} P\left(\frac{z}{r}\right) \mathrm{e}^{\mathrm{j}zx} \frac{\mathrm{d}z}{r}. \tag{3.164}$$

From the scaling property of the generalised power spectrum of a fractal, Equation (3.155), we may write:

$$P(z) = \frac{1}{\alpha^{2H+1}} P\left(\frac{z}{\alpha}\right) \Rightarrow P\left(\frac{z}{r}\right) = r^{2H+1} P(z). \tag{3.165}$$

Substituting from (3.165) into (3.164), we obtain:

$$\rho(rx) = r^{2H} \int_{-\infty}^{+\infty} P(z) \mathrm{e}^{\mathrm{j}zx} \mathrm{d}z \Rightarrow \rho(rx) = r^{2H} \rho(x). \tag{3.166}$$

This is the scaling property of the generalised autocorrelation for a fractal. It effectively shows that the autocorrelation function for shifts r times larger is r^{2H} stronger than the autocorrelation for the original shift.

Box 3.14 Construction of a 2D fractal

Step 0: Select the Hurst exponent H to be between 0 and 1 and the variance of the samples, σ^2. Assuming that the values of the pixels will be in the range $[0, 1]$, select σ^2 appropriately. Select also the size, $(2N+1) \times (2N+1)$, of the image you wish to construct.

Step 1: Create an empty array of size $(2N+1) \times (2N+1)$. Call it P. For $-N \leq k \leq N$ and $0 \leq l \leq N$, draw random numbers uniformly distributed in the range $[0^\circ, 360^\circ)$ to assign values to elements $P(k, l)$.

Step 2: For $-N \leq k \leq N$ and $-N \leq l < 0$ set $P(k, l) = -P(-k, -l)$.

Step 3: Create an array F of size $(2N+1) \times (2N+1)$. Assign to its elements values according to:

$$F(k, l) = \left[\left(\frac{2\pi k}{2N+1} \right)^2 + \left(\frac{2\pi l}{2N+1} \right)^2 \right]^{-0.5(H+1)}. \tag{3.167}$$

Step 4: Create two arrays of size $(2N+1) \times (2N+1)$. Call them R and I. Assign to them values as follows:

$$R(k, l) = \sigma F(k, l) \cos[P(k, l)] \qquad I(k, l) = \sigma F(k, l) \sin[P(k, l)]. \tag{3.168}$$

Step 5: Use matrices R and I as the real and the imaginary part of the DFT of the image you want to construct and take the inverse DFT:

$$f(i,j) = \sum_{k=-N}^{N} \sum_{l=-N}^{N} (R(k,l) + \mathrm{j}I(k,l))\mathrm{e}^{2\pi\mathrm{j}\frac{ki+lj}{2N+1}}. \tag{3.169}$$

Step 6: As the values of f will be real and may also be negative, they have to be mapped to the range $[0, 255]$ to be visualised as an image. First we scale them to be in the range $[-128, 127]$, as follows.

Identify the minimum and the maximum values of f. Call them a and b, respectively.

Obviously, $a < 0$ and $b > 0$.

Identify $c = \max\{|a|, b\}$.

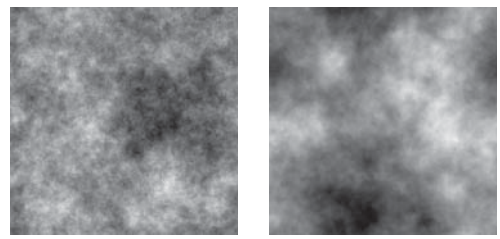
Scale all values of f and round them to the nearest integer, to form $\tilde{f}$, as follows.

$$\tilde{f}(i,j) = \left\lfloor \frac{f(i,j)}{c} \times 128 + 0.5 \right\rfloor. \tag{3.170}$$

Add 128 to all values of $\tilde{f}$ to produce the fractal image.

Figure 3.48 shows two fractal images constructed with this algorithm for $N = 128$ and $H = 0.3$ and 0.7.

Figure 3.48 Synthesising 2D fractals of size 257×257. Source: Maria Petrou.



$H = 0.3$ $H = 0.7$

Example B3.53

Compute the fractal dimension of the fractals constructed in Box 3.14, using the box-counting method of Box 3.2 and Equation (3.158). Compare the results with the true fractal dimensions of the images.

These images are of size 257×257. *First, we crop them to size* 256×256, *as the box counting method is appropriate only for images of size a power of 2. Then we follow the same steps as in example 3.41. Figure 3.49 shows the pairs of points* $(\ln 2^n, \ln N(n))$ *computed for these images. We can see that they do not form a straight line for all values of n. In fact as n increases, and the boxes become smaller, the fractality of the pattern becomes more and more dominated by the digitisation noise, and the image stops behaving like a fractal. One has to be able to identify the values of n over which the straight line may be fitted. We use for this the following algorithm.*

(Continued)

Example B3.53 (Continued)

Step 0: *Select a value for the correlation coefficient r that will determine the quality of fitting we demand. The selected value should be in the range* $0.9 \leq r < 1$. *For constructed fractals, where we expect a very good fractal behaviour, r may be as high as 0.9995. For real images, where we know that the fitting is bound not to be good, r may be considerably lower.*

Step 1: *Set* $n_{\max}$ *equal to the largest value of n.*

Step 2: *Fit the pairs of points with a straight line using all values of n up to and including* $n_{\max}$.

Step 3: *Compute the correlation coefficient* $r_{n_{\max}}$ *to assess the quality of the fitting.*

Step 4: *If* $r_{n_{\max}} \geq r$, *exit the algorithm with the slope of the fitted line as the estimated fractal dimension of the image.*

Step 5: *If* $r_{n_{\max}} < r$, *set* $n_{\max} = n_{\max} - 1$ *and go to Step 2.*

It is possible that if r has been set too high, the algorithm will never exit via step 4, but it will fail to find a satisfactory fitting. A lower value of r should then be used.

In Figure 3.49 we show the fitted lines to the data points for $r = 0.9995$ *for the two constructed fractals, as well as the ideal lines with slopes the true fractal dimensions. This figure shows how sensitive the method of estimating the fractal dimension is. A very slight change in the slope, caused by the inclusion of a single extra point may damage the slope significantly. This is not only for this method, but rather an inherited sensitivity of the model, as it involves exponentials that are very sensitive to small perturbations.*

Table 3.7 shows the fractal dimensions and the correlation coefficients with each range of values of n for $H = 0.3$ *and 0.7.*

The estimations for four different directions in the spectra: horizontal, vertical, and two diagonals for $H = 0.3$ *and 0.7 are shown in Table 3.8.*

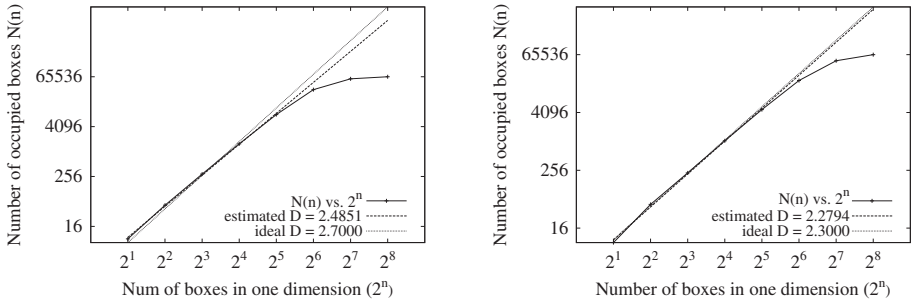


Figure 3.49 The pairs of points $(\ln 2^n, \ln N(n))$ fitted with a straight line for each of the two images. In each case the ideal line, with slope the fractal dimension we used to construct the fractal is also plotted.

Table 3.7 The fractal dimensions and correlation coefficients with each *n*.

$H = 0.3$ 256×256 pixels, 256 gray levels		
n	***N***	***D***
1	8	3.000
2	52	2.850
3	293	2.732
4	1571	2.654
5	8008	2.593
6	31932	2.494
7	58318	2.262
8	65535	2.000
$\varepsilon \to 0$		2.504

$H = 0.7$ 256×256 pixels, 256 gray levels		
n	*N*	*D*
1	8	3.000
2	50	2.822
3	226	2.607
4	1055	2.511
5	4697	2.440
6	18964	2.368
7	48956	2.226
8	65533	2.000
$\varepsilon \to 0$		2.361

Table 3.8 The fractal dimensions for four different directions in the spectra.

	$H = 0.3$		$H = 0.7$	
Direction	β	***D***	β	***D***
Horizontal	2.593	2.704	3.390	2.305
Vertical	2.599	2.701	3.386	2.307
Diagonal 1	2.606	2.697	3.379	2.311
Diagonal 2	2.592	2.704	3.435	2.282

Is fractal dimension a good texture descriptor?

The fractal dimension on its own is not a good texture descriptor, for the following reasons.

(i) Images are not really fractals, i.e. they do not exhibit the same structure at all scales. This can be seen from the plots in Figure 3.45. These curves are supposed to be straight lines, but only parts of them can be fitted well by straight line segments. In general, these curves are more complicated. By describing their shapes with a single number, i.e. the slope, we throw away a lot of the information they contain. This problem is dealt with by the use of the variogram and its models, where the range over which the fractal model applies is also taken into account (see the section on Which parametric models are commonly used to fit the variogram over its Range?).

(ii) In fractal image analysis, often image isotropy is assumed. This problem can be dealt with by computing different fractal dimensions along different orientations.

(iii) The fractal dimension does not capture any texture inhomogeneity. This may be partly dealt with by using extra image features alongside the fractal dimension, like **lacunarity**, or the generalisation of the multifractal spectrum, computed from the generalised fractal dimension for grey images.

(iv) Images are not the products of non-stationary stochastic processes with stationary increments, but often they are non-stationary stochastic processes with non-stationary increments, leading to inhomogeneous appearance. It is obvious, therefore, that one has to use more complex models. This leads us to the adoption of **multifractal** models. Multifractal models, which effectively consider the image consisting of the combination of several fractals and extracts **local** information to characterise the different parts, should not be confused with the multifractal spectrum, which treats the image as a whole and simply tries to capture its inhomogeneities, by producing a single sequence of numbers, usually 5, to be used as its features.

What is lacunarity?

Lacunarity is a measure of non-homogeneity of the data. It measures the "lumpiness" of the data. It is defined in terms of the ratio of the variance over the mean value of the image function. It may be computed for the full image, or locally, inside local windows, in which case its value may be assigned to the central pixel of the window. If computed locally, lacunarity may be used as a feature appropriate to segment non-stationary textures, which will be examined in Chapter 4. Also, the histogram of the local lacunarity values may be used as an image feature.

Any one of the following definitions might be used for an $N \times M$ image (or sub-image), with grey values $I(n,m)$

$$\mathcal{L}_s \equiv \frac{\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M} I(n,m)^2}{\left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M} I(k,l)\right)^2} - 1 \tag{3.171}$$

$$\mathcal{L}_a \equiv \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left|\frac{I(n,m)}{\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M} I(k,l)} - 1\right| \tag{3.172}$$

$$\mathcal{L}_p \equiv \left(\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(\frac{I(n,m)}{\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M} I(k,l)} - 1\right)^p\right)^{\frac{1}{p}} \tag{3.173}$$

where p is some parameter.

Note that if the distribution of grey values is totally uniform, $\mathcal{L}$ is 0. The higher the value of $\mathcal{L}$, the more inhomogeneous the surface.

Example B3.54

Show that $\mathcal{L}_s = \mathcal{L}_2^2$.
We start from the definition of $\mathcal{L}_p$ for $p = 2$ and take its square:

$$\begin{aligned}
\mathcal{L}_2^2 &= \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(\frac{I(n,m)}{\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)} - 1\right)^2 \\
&= \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\frac{\left(I(n,m) - \frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2}{\left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2} \\
&= \frac{1}{\left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2} \times \\
&\quad \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(I(n,m)^2 + \left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2 - \right. \\
&\quad \left. 2I(n,m)\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right) \\
&= \frac{1}{\left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2}\left(\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}I(n,m)^2 + \right. \\
&\quad \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2 - \\
&\quad \left. 2\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}I(n,m)\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right). \qquad (3.174)
\end{aligned}$$

The last term on the right-hand side of the above equation is just the square of the average grey value of the image (the two double sums are identical). The penultimate term includes a double sum over indices n and m, which, however, do not appear in the summand. So, all this double sum yields is a factor NM. This factor NM cancels with the denominator NM, and so the penultimate term is just the summand. So, we get:

$$\begin{aligned}
\mathcal{L}_2^2 &= \frac{1}{\left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2} \times \left(\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}I(n,m)^2 + \left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2 - \right. \\
&\quad \left. 2\left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2\right) \\
&= \frac{1}{\left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2} \times \\
&\quad \left(\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}I(n,m)^2 - \left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M}I(k,l)\right)^2\right). \qquad (3.175)
\end{aligned}$$

(Continued)

Example B3.54 (Continued)

This may be written as:

$$\mathcal{L}_2^2 = \frac{\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M} I(n,m)^2}{\left(\frac{1}{NM}\sum_{k=1}^{N}\sum_{l=1}^{M} I(k,l)\right)^2} - 1 = \mathcal{L}_s. \tag{3.176}$$

Example 3.55

Compute the lacunarity measures for the image in Figure 3.50.

0	**1**	**2**	**0**
3	**1**	**0**	**2**
1	**3**	**0**	**3**
2	**1**	**1**	**0**

Figure 3.50 A 2-bit image.

We note that we have 16 pixels in the image ($NM = 16$), 5 of them have value 0, 5 have value 1, 3 have value 2 and 3 have value 3. The sum of all grey values is 20, and the sum of their squares is 44. Then:

$$\mathcal{L}_s = \frac{\frac{44}{16}}{\left(\frac{20}{16}\right)^2} - 1 = 1.76 - 1 = 0.76$$

$$\mathcal{L}_a = \frac{1}{16}\left[\left|\frac{0}{\frac{20}{16}} - 1\right| \times 5 + \left|\frac{1}{\frac{20}{16}} - 1\right| \times 5 + \left|\frac{2}{\frac{20}{16}} - 1\right| \times 3 + \left|\frac{3}{\frac{20}{16}} - 1\right| \times 3\right]$$

$$= \frac{1}{16}\left[5 + \left|\frac{16}{20} - 1\right| \times 5 + \left|\frac{32}{20} - 1\right| \times 3 + \left|\frac{48}{20} - 1\right| \times 3\right] = 0.75$$

$$\mathcal{L}_2 = \sqrt{\frac{1}{16}\left[\left(\frac{0}{\frac{20}{16}} - 1\right)^2 \times 5 + \left(\frac{1}{\frac{20}{16}} - 1\right)^2 \times 5 + \left(\frac{2}{\frac{20}{16}} - 1\right)^2 \times 3 + \left(\frac{3}{\frac{20}{16}} - 1\right)^2 \times 3\right]}$$

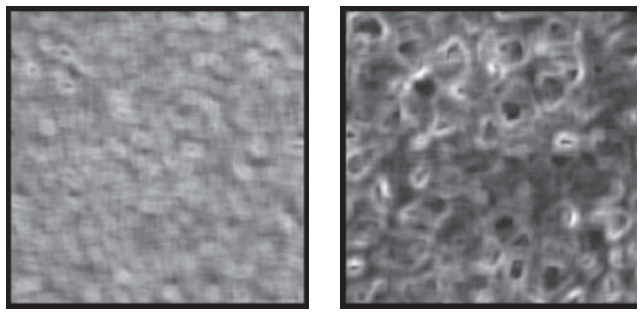
$$= \sqrt{\frac{1}{16}\left[5 + \left(\frac{16}{20} - 1\right)^2 \times 5 + \left(\frac{32}{20} - 1\right)^2 \times 3 + \left(\frac{48}{20} - 1\right)^2 \times 3\right]}$$

$$= \sqrt{\frac{1}{16}\left[5 + \frac{2864}{400}\right]} = \sqrt{\frac{1}{16} \times \frac{4864}{400}} = \sqrt{0.76} = 0.8718. \tag{3.177}$$

Example 3.56

Use an 11×11 window to scan images 3.1c and 3.1d. Inside each window compute the lacunarity using formula (3.172) and assign it to the central pixel of the window. At the end, scale the results of both images in the same scale, $[0, 255]$ in order to visualise them as images. We use the same scale for both images in order to be able to make comparisons between the two. Plot also the histograms of the unscaled lacunarity values you computed.

We compute the lacunarity values using 11×11 windows displayed as images, (a) food, (b) plastic as shown in Figure 3.51. We also plot the histograms of the lacunarity values for each image as shown in Figure 3.52.



(a) (b)

Figure 3.51 Lacunarity values using 11×11 windows displayed as images. (a) Food. (b) Plastic. Source: Maria Petrou.

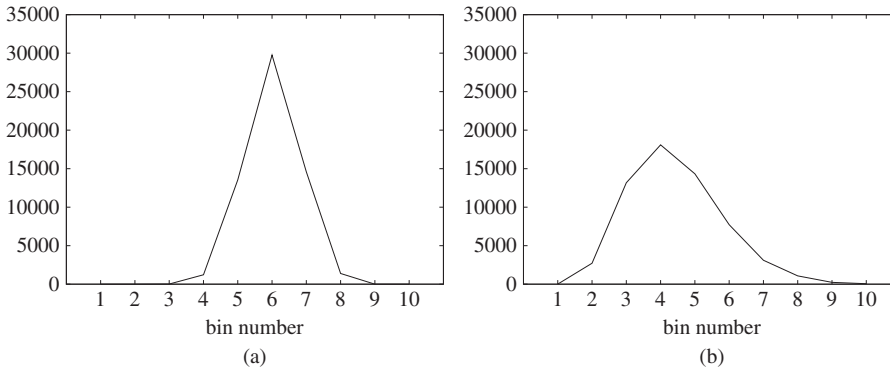


Figure 3.52 Histograms of the lacunarity values. (a) Food. (b) Plastic. Both histograms have been constructed using identical bins, so direct comparison between the two is possible.

How can we compute the multifractal spectrum of a grey image?

Remember that in the box-counting method we simply checked whether a box is occupied or not in order to compute the fractal dimension of a binary shape. For the generalised fractal dimension of a binary shape, we counted the number of points of the shape inside each box. For a grey image now, instead of counting the number of points, we also have to account for the value of each point. So, people assign to each box something they may call "strength" of the signal, or "mass" of the signal. The natural extrapolation of the previous approaches is to sum the grey values of the pixels inside the box. However, in some cases, people measure the contrast of the pixels inside the box. So, to define the generalised fractal dimension of a grey image, we cover the image plane with boxes of size $\varepsilon \times \varepsilon$. Inside each box we compute a "strength", or a "measure". Let us call the strength of box i, n_i. Let us sum all the strengths of all the boxes we have used to cover the image, and call this sum N_ε. Then we may assign to box i the value $p_i \equiv n_i/N_\varepsilon$, which indicates the probability of finding strength n_i inside a box of size $\varepsilon \times \varepsilon$. Then the generalised fractal dimension of the grey image is defined as in the case of the binary shape by formulae (3.41), (3.42) and (3.43).

The multifractal spectrum is the sequence of numbers we obtain if we let q take various values. Usually, $q = -2, -1, 0, 1, 2$. Note: ε actually is a fraction of the image size, i.e. in these calculations, the image is assumed to have size 1×1 and ε takes values $1/2$, $1/4$, etc.

Example 3.57

Calculate the multifractal spectra for the images in Figure 3.1 using as measure the contrast of the signal inside each box.

Each image is of size $M \times M$, where $M = 256$. Let us cover them with boxes of size 64×64, 32×32, 16×16, 8×8 and 4×4, corresponding to values of ε $1/4$, $1/8$, $1/16$, $1/32$ and $1/64$, respectively. There will be 16, 64, 256, 1024 and 4096 boxes, respectively. Inside each box, we shall define the strength of the signal as follows.

Find the minimum, b, and the maximum, a, grey value inside the box. The strength of the signal inside the box is $(a - b)/(M\varepsilon)$. That is, for box i, the strength is:

$$n_{i;\varepsilon} = \frac{a_i - b_i}{M\varepsilon}. \tag{3.178}$$

Here, we made explicit the dependence of n_i on ε, in order to clarify that this calculation happens for boxes of the same size. Also, we made explicit the dependence of a and b on the identity of the box i, in order to make clear that this calculation happens for each box for fixed ε. The total strength of the signal then is:

$$N_\varepsilon = \sum_i n_{i;\varepsilon}. \tag{3.179}$$

Then to each box i we assign

$$p_{i;\varepsilon} = \frac{n_{i;\varepsilon}}{N_\varepsilon} \tag{3.180}$$

where again we made explicit the dependence of p_i on the value of ε.

Then, for fixed q, we consider all pairs $\left[\log\varepsilon, \log\left(\sum_i p_{i;\varepsilon}^q\right)\right]$ *and fit them with a least squares error straight line (see Box 3.3):*

$$\log\left(\sum_i p_{i;\varepsilon}^q\right) = A + B\log\varepsilon. \tag{3.181}$$

The slope of this line B, divided by $q-1$ *(when* $q \neq 1$*) is the generalised fractal dimension* D_q *of the image. Doing it for all values of q, we produce the multifractal spectrum of the image.*

For the case of $q = 0$*, the term "non-empty box" refers to the boxes for which* $n_{i;\varepsilon} \neq 0$*. In other words, we exclude the boxes inside which the image is flat. It is possible that due to noise such boxes do not exist. In that case, perhaps some tolerance around the value of 0 might be considered, but even better, we may realise that since* D_0 *is actually the box-counting fractal dimension, we may calculate it directly using the method of Equation (3.13).*

For the case of $q = 1$*, we must use de l'Hospital's rule, as shown in Equation 3.45 to solve the same problem. Table 3.9 shows the multifractal spectra on* D_q *for each value of q.*

Table 3.9 The multifractal spectra for the images using as measure the contrast of the signal inside each box.

Image	$q=-2$	$q=-1$	$q=0$	$q=1$	$q=2$
Cloth 1	2.0496	2.0229	2.0000	1.9851	1.9593
Cloth 2	2.0307	2.0150	2.0000	1.9895	1.9723
Food	2.0715	2.0172	2.0000	1.9924	1.9840
Plastic	2.1204	2.0523	2.0000	1.9725	1.9303
Fabric 1	2.1864	2.0989	2.0000	1.9408	1.8424
Fabric 2	2.0574	2.0273	2.0000	1.9824	1.9546
Beans	2.1320	2.0534	2.0000	1.9728	1.9310
Sand	2.1064	2.0534	2.0000	1.9641	1.9068

Example 3.58

Calculate the multifractal spectra for the images in Figure 3.1 using as measure the sum of the grey values of the pixels inside each box.

In the same way, we calculate the multifractal spectra for the images using as measure the sum of the grey values of the pixels inside each box. Table 3.10 shows the multifractal spectra on D_q *for each value of q.*

(Continued)

Example 3.58 (Continued)

Table 3.10 The multifractal spectra for the images using as measure the sum of the grey values of the pixels inside each box.

Image	$q=-2$	$q=-1$	$q=0$	$q=1$	$q=2$
Cloth 1	2.0314	2.0147	2.0000	1.9946	1.9748
Cloth 2	2.0052	2.0025	2.0000	1.9990	1.9951
Food	2.0185	2.0088	2.0000	1.9980	1.9853
Plastic	2.0379	2.0180	2.0000	1.9937	1.9706
Fabric 1	2.0139	2.0065	2.0000	1.9976	1.9887
Fabric 2	2.0107	2.0053	2.0000	1.9985	1.9894
Beans	2.0963	2.0387	2.0000	1.9928	1.9575
Sand	2.0021	2.0010	2.0000	1.9997	1.9981

What is a multifractal?

A multifractal is a process that is everywhere locally a fractal characterised by a different Hurst exponent H at each locality. To avoid confusion, a function $\alpha(t)$ is introduced, known as the **Holder exponent**, in association with the so-called **Holder regularity** that derives from Equation (3.86).

What is the Holder regularity?

We may interpret Equation (3.86) to imply that at every point t

$$v(t+\Delta t) \sim v(t) + C|\Delta t|^H \tag{3.182}$$

where C is some constant. This is known as **Holder regularity**. In the case of a multifractal, it is written as

$$v(t+\Delta t) \sim v(t) + C|\Delta t|^{\alpha(t)} \tag{3.183}$$

where $\alpha(t)$ is the **Holder exponent**, characterising the multifractal.

What is the difference between a fractal and a multifractal?

A fractal is a non-stationary process with stationary increments. A multifractal has non-stationary increments. The fractal increments obey Equation (3.86), while for multifractals exponent H is replaced by a function $\alpha(t)$ that depends on locality t.

Example B3.59

Use the method of Box 3.10 to construct a 1D multifractal, with Holder exponent $\alpha(t) = 1 - t^2$, for $t \in [0.01, 0.99]$. The multifractal should consist of 99 points.

A multifractal consists of many ordinary fractals that somehow are "tangent" to the multifractal at each location. Let us sample the t parameter using 99 points, starting with $t = 0.01$ and ending with $t = 0.99$, in steps of 0.01. For every value of t we compute the corresponding value of $\alpha(t)$. This is the value of H with which we should construct an ordinary fractal of size 99 samples, using the method of Box 3.10. This way, we shall have 99 ordinary fractals. From each one of them we shall keep only one sample, i.e. from the first one we shall keep the value of its first sample, from the second the value of its second sample, etc. Combining all these values we shall form the multifractal. The result is shown in Figure 3.53. Using a Gaussian white noise with $\sigma = 1$, we obtain a synthetic 1D multifractal in this figure. If we use a different random sequence at each point, we also obtain a different synthetic 1D multifractal as shown in Figure 3.54.

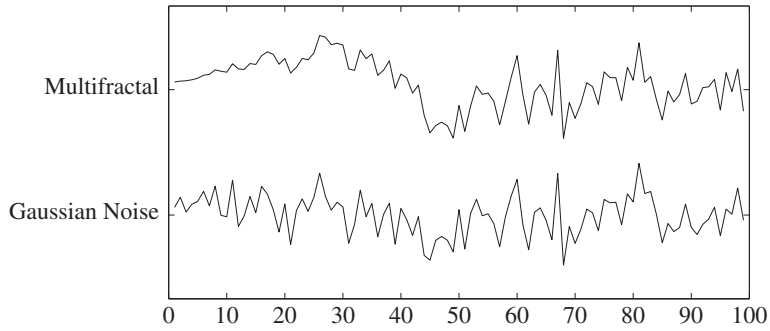


Figure 3.53 Synthetic 1D multifractal

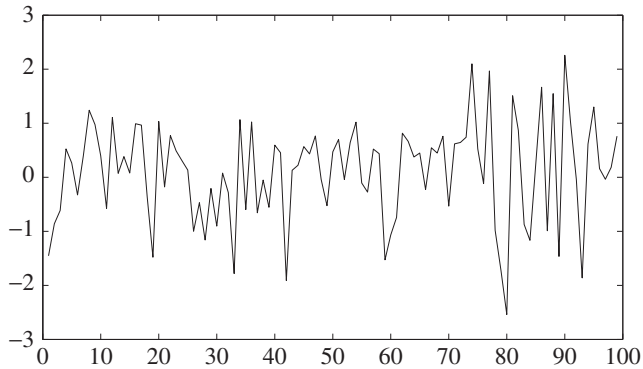


Figure 3.54 Synthetic 1D multifractal using a different random sequence at each point.

Example B3.60

Use the method of Box 3.14 to construct a 2D multifractal, of size 65×65**, with Holder exponent** $\alpha(i,j)$ **that varies across the image according to:**

$$\alpha(2k+1, 2l+1) = \left(1 - \left(\frac{2k+1}{65}\right)^2\right)\left(1 - \left(\frac{2l+1}{65}\right)^2\right) \text{ for } k, l = 0, \dots, 31. \quad (3.184)$$

The value of $\alpha(i,j)$ **at other pixel positions is given by:**

$$\alpha(2k, 2l) = \alpha(2k, 2l-1) = \alpha(2k-1, 2l) = \alpha(2k-1, 2l-1) \text{ for } k, l = 1, \dots, 32. \quad (3.185)$$

You must construct $32^2 = 1024$ **fractal surfaces with Hurst exponent** $H = \alpha(2k+1, 2l+1)$**, given by (3.184). Each surface could be of size** 64×64**. Then from each surface you will retain the values of a single** 2×2 **patch with its top left corner at pixel** $(2k+1, 2l+1)$**. These** 2×2 **patches will form the final multifractal you will construct.**

Using a random field for all fractals, we obtain a synthetic 2D multifractals as shown in Figure 3.55. If we use a different random field at each point, we also obtain a different synthetic 2D multifractal as shown in Figure 3.56.

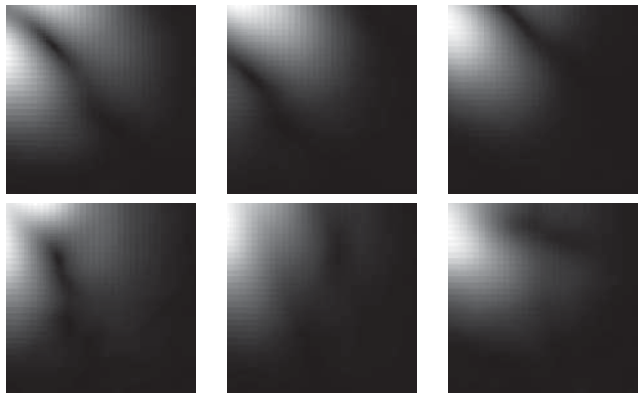


Figure 3.55 Synthetic 2D multifractals. Source: Maria Petrou.

Figure 3.56 Synthetic 2D multifractals using a different random sequence at each point. Source: Maria Petrou.

How can we characterise an image using multifractals?

A multifractal is characterised in the following ways.

(i) With its Holder exponent $\alpha(t)$. However, as this varies with locality, it is not a very efficient way of characterising the fractal because for each pixel it will have a separate value, so we shall have as many values as pixels. Note: as the Holder exponent corresponds to the Hurst dimension H of an ordinary fractal, which in turn corresponds to the fractal dimension D, this method corresponds to the calculation of the local fractal dimension in the vicinity of each pixel.

(ii) A multifractal may also be characterised by its **Hausdorff spectrum** $f_h(\alpha)$. To compute it, we first compute α at the vicinity of every pixel. Then we divide the pixels into subsets, where the elements of each subset have the same value α (within some tolerance $\Delta\alpha$). In other words, we bin the values of α. Then we compute the fractal dimension $f_h(\alpha)$ of all the pixels of each subset. Note that in this process we consider the pixels of each subset as point sets and we do not take into consideration their grey value. This implies that the value of $f_h(\alpha)$ is in the range $[0, 2]$. It is 0 if there is only one pixel in the subset, 1 if the pixels form a line and 2 if all pixels of the image belong to the subset and thus form a plane. If the subset is empty, we set by definition $f_h(\alpha) = -\infty$. Of course, in all other cases, as the pixels of the subset will be irregularly distributed, $f_h(\alpha)$ will have a non-integer value in the range $[0, 2]$. This is known as the **Hausdorff dimension of a set of points**.

(iii) Finally, a multifractal may also be characterised by its **large deviation multifractal spectrum** f_g. The Holder exponent α is computed inside local neighbourhoods of various sizes. The large deviation multifractal spectrum $f_g(\alpha)$ expresses the probability $p(\alpha)$ of finding a box of size parameter n with Holder exponent α:

$$p(\alpha) \equiv n^{f_g(\alpha)-2} \Rightarrow f_g(\alpha) = 2 + \frac{\log p(\alpha)}{\log n}. \tag{3.186}$$

How can we calculate the Holder exponent of each pixel?

There are various approaches.

(i) We may interpret (3.183) to hold literally, and treat $(v(t + \Delta t), v(t))$ as the grey values of two neighbouring pixels (see Example 3.61).

(ii) We may interpret (3.183) more correctly, understanding that it holds on average (expectation values). Then we may use a small window around each pixel, inside which we may consider several pairs of values $(v(t + \Delta t), v(t))$, for several Δt values and compute in the least square error sense (see Box 3.3) a value for α, which will be assigned to the central pixel of the window (see Example 3.62).

Example 3.61

If g is the image, use Equation (3.183) for neighbouring pixels to compute the Holder exponent for each pixel. Then use these Holder exponents to construct the Hausdorff spectrum of the image.

(Continued)

Example 3.61 (Continued)

In terms of grey pixel values, we assume that the following equation holds locally:

$$g(i,j) \simeq g(k,l) + C\sqrt{(i-k)^2+(j-l)^2}^{\alpha}. \tag{3.187}$$

Here C is some constant. Let us apply this relationship to the 3×3 *neighbourhood of a pixel. The four closest neighbours of the pixel are at distance 1 from it and they may be used to estimate constant C:*

$$\begin{aligned} g(i,j+1) &\simeq g(i,j) + C \\ g(i,j-1) &\simeq g(i,j) + C \\ g(i+1,j) &\simeq g(i,j) + C \\ g(i-1,j) &\simeq g(i,j) + C. \end{aligned} \tag{3.188}$$

Therefore:

$$C = \frac{1}{4}\left[g(i,j+1) + g(i,j-1) + g(i+1,j) + g(i-1,j) - 4g(i,j)\right]. \tag{3.189}$$

From the diagonal neighbours we have:

$$\begin{aligned} g(i+1,j+1) &\simeq g(i,j) + C\sqrt{2}^{\alpha} \\ g(i+1,j-1) &\simeq g(i,j) + C\sqrt{2}^{\alpha} \\ g(i-1,j+1) &\simeq g(i,j) + C\sqrt{2}^{\alpha} \\ g(i-1,j-1) &\simeq g(i,j) + C\sqrt{2}^{\alpha}. \end{aligned} \tag{3.190}$$

Therefore:

$$\sqrt{2}^{\alpha} = \frac{1}{4C}\left[g(i+1,j+1) + g(i+1,j-1) + g(i-1,j+1) + g(i-1,j-1) - 4g(i,j)\right] \tag{3.191}$$

Or:

$$\alpha = \frac{1}{\log\sqrt{2}} \log\left|\frac{g(i+1,j+1)+g(i+1,j-1)+g(i-1,j+1)+g(i-1,j-1)-4g(i,j)}{g(i,j+1)+g(i,j-1)+g(i+1,j)+g(i-1,j)-4g(i,j)}\right|. \tag{3.192}$$

So, the algorithm is as follows:

Step 1: *Use (3.192) to work out a value of α for every pixel in the image.*
Step 2: *Quantise the values of α into, say, five bins.*
Step 3: *Identify the pixels that have α values in the same bin. They form a point set.*
Step 4: *Use the algorithm of Box 3.1 to work out the capacity dimension of each of the point sets. You will have one capacity dimension for each bin you used in step 2. The sequence of these numbers constitutes the Hausdorff spectrum of the image.*

Note that Equation 3.192 may result in: $+\infty$, $-\infty$, *and NaN (that is,* $\frac{0}{0}$*). To deal with this, the following is done.*

- *When we have NaN, we assign* $\alpha = 0$*, the same as* $\frac{2}{2}$, $\frac{3}{3}$, *etc.*
- *When we have* $-\infty$*, we use the minimum value of α after discarding* $-\infty$*.*
- *When we have* $+\infty$*, we use the maximum value of α after discarding* $+\infty$*.*

Using the third image in Figure 3.55, we obtained the matrix of alpha values as shown in Figure 3.57.

And Figure 3.58 shows the five points sets from the matrix of alpha values.

Next Table 3.11 shows the fractal dimensions computed for each point set for different values of n.

Figure 3.57 The matrix of alpha values.

Figure 3.58 The five point sets from the matrix of alpha values.

Table 3.11 The five point sets from the matrix of alpha values.

	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$
Point set 1	2.0000	1.9037	1.7859	1.6230	1.4067	1.2263
Point set 2	2.0000	1.9534	1.9093	1.8828	1.8034	1.7233
Point set 3	2.0000	2.0000	1.9183	1.8587	1.7589	1.5657
Point set 4	2.0000	2.0000	1.9527	1.9128	1.8433	1.7319
Point set 5	2.0000	2.0000	1.9443	1.8394	1.6598	1.4206

Example 3.62

If g is the image, use Equation (3.183) in its original form, i.e. (3.86), to compute the Holder exponent for each pixel. Then use these Holder exponents to construct the Hausdorff spectrum of the image.

Write (3.86) in the form:

$$\begin{aligned} &|g(i,j) - g(k,l)| = C\sqrt{(i-k)^2 + (j-l)^2}^{\alpha} \\ \Rightarrow\ &\log|g(i,j) - g(k,l)| = \log C + \frac{\alpha}{2}\log\left[(i-k)^2 + (j-l)^2\right]. \end{aligned} \tag{3.193}$$

(Continued)

Example 3.62 (Continued)

The algorithm is as follows.

Step 1: *Around each pixel* (i,j) *consider a* 5×5 *window. Pair the central pixel with all 24 pixels around it and thus come up with 24 pairs of values* $(\log\left[(i-k)^2+(j-l)^2\right], \log|g(i,j)-g(k,l)|)$.

Step 2: *Fit these pairs of values with a straight line in the least square error sense (see Box 3.3) to identify the value of* $\alpha/2$ *as the slope of the fitted line. Assign this value to pixel* (i,j).

Step 3: *Quantise the values of* α *into, say, five bins.*

Step 4: *Identify the pixels that have* α *values in the same bin. They form a point set.*

Step 5: *Use the algorithm of Box 3.1 to work out the capacity dimension of each of the point sets. You will have one capacity dimension for each bin you used in step 3. The sequence of these numbers constitutes the Hausdorff spectrum of the image.*

The matrix of α *values, the five point sets, and the fractal dimension for each point set, are shown in the following. Note that the image is* 64×64 *and the point sets are* 60×60. *The matrix of alpha values is shown in Figure 3.59.*

And Figure 3.60 shows the five points sets from the matrix of alpha values.

Next Table 3.12 shows the fractal dimensions computed for each point set for different values of n.

Figure 3.59 The matrix of alpha values. Source: Maria Petrou.

Figure 3.60 The five point sets from the matrix of alpha values. Source: Maria Petrou.

Table 3.12 The five point sets from the matrix of alpha values.

	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
Point set 1	2.0000	1.9534	1.8515	1.7692	1.7608	1.7670
Point set 2	2.0000	1.7925	1.7098	1.6646	1.6320	1.5654
Point set 3	2.0000	1.7925	1.7365	1.7145	1.7000	1.6383
Point set 4	1.5850	1.4037	1.4865	1.5271	1.5063	1.4394
Point set 5	1.0000	0.7925	0.9358	1.0000	1.0644	1.0074

Box 3.15 The 1D binomial multifractal

Imagine a histogram of N records binned in 2^k bins. If the shape of this histogram is a binomial multifractal, we expect to find in the first 2^{k-1} bins $N(1-p)$ of the records, and in the second 2^{k-1} bins Np of the records, for all values of k, up to $k = K$. The values of p and K characterise this multifractal model.

Example B3.63

Construct a binomial multifractal distribution with $K = 3$ and $p = 0.8$. If $N = 1000$, how many objects do you expect to find in each bin? This method of constructing a fractal is known as multiplicative cascade.

Let us call the distribution function we are constructing $g(x)$, where $0 \leq x \leq 1$. In the first division ($k = 1$), the range of the independent variable is divided in two, and in the left half the value of $g(x)$ is $g_1 = 0.2$, while in the right half we have $g_2 = 0.8$. This is shown in Figure 3.61a. In the second division ($k = 2$), each one of these ranges is again divided into two halves, and the distribution in the left half will be multiplied with 0.2, while in the right half with 0.8. Thus, in the four bins we have now, $g(x)$ takes the following values:

$g_{11} = 0.2 \times 0.2 = 0.04$
$g_{12} = 0.2 \times 0.8 = 0.16$
$g_{21} = 0.8 \times 0.2 = 0.16$
$g_{22} = 0.8 \times 0.8 = 0.64.$

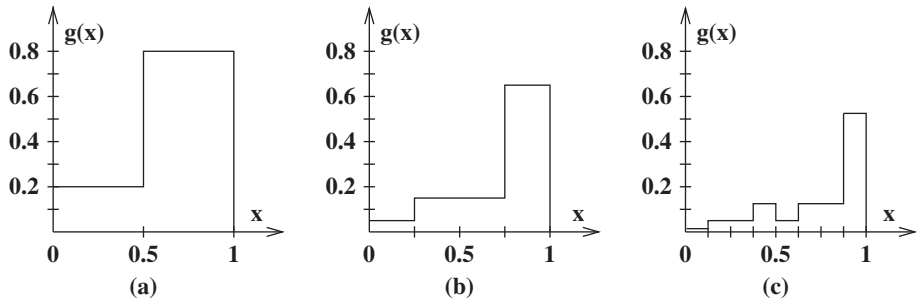


Figure 3.61 Construction of a binomial multifractal distribution with 2^3 bins and $p = 0.8$.

The indices of the bins have been selected to reflect the position of each bin in the two successive divisions. The result is shown in Figure 3.61b. In the third and final division ($k = K = 3$), each one of the bins is again divided into two halves, and the value in the left half is multiplied with 0.2, while in the right half with 0.8. Thus, in the eight final bins, $g(x)$ takes the following values:

$g_{111} = 0.04 \times 0.2 = 0.008$
$g_{112} = 0.04 \times 0.8 = 0.032$
$g_{121} = 0.16 \times 0.2 = 0.032$
$g_{122} = 0.16 \times 0.8 = 0.128$
$g_{211} = 0.16 \times 0.2 = 0.032$
$g_{212} = 0.16 \times 0.8 = 0.128$
$g_{221} = 0.64 \times 0.2 = 0.128$

(Continued)

Example B3.63 (Continued)

$g_{222} = 0.64 \times 0.8 = 0.512$.

This is shown in Figure 3.61c.

If $N = 1000$, we expect to find in the eight bins 8, 32, 32, 128, 32, 128, 128 *and* 512 *objects, from left to right, respectively.*

Example B3.64

Work out how many bins you expect to find with identical number of objects in a binomial multifractal distribution with K divisions and probability p. Verify the formula you derive by checking the validity of the results of Example 3.63.

The number of objects in a bin is N times p^n, where n is the number of times this bin was the right half of a bin of the previous division step, times $(1-p)^m$, where m is the number of times this bin was the left half of a bin of the previous division step. Obviously, $n + m = K$, since we have K divisions in total. So, we may say that the number of objects in a bin is $Np^{K-\alpha}(1-p)^\alpha$, where α may vary from 1 to K. For example, for the leftmost bin, $\alpha = K$, since this bin is in all divisions left, and thus all factors are $1-p$, while for the right-most bin $\alpha = 0$, since this bin is in all divisions on the right, and therefore, all its factors are p. For other values of α, we are going to have $C_\alpha^K = \frac{K!}{\alpha!(K-\alpha)!}$ different bins that will have the same occupancy, since there are as many different combinations of values m and n that sum up to K. In any case, this formula is also valid for the extreme values of α too.

In example 3.63, we had $K = 3$ and $p = 0.8$.

For $\alpha = 0$, there will be $C_0^3 = \frac{3!}{0!3!} = 1$ bin with $1000 \times 0.8^3 \times 0.2^0 = 512$ objects.

For $\alpha = 1$, there will be $C_1^3 = \frac{3!}{1!2!} = 3$ bins with $1000 \times 0.8^2 \times 0.2^1 = 128$ objects.

For $\alpha = 2$, there will be $C_2^3 = \frac{3!}{2!1!} = 3$ bins with $1000 \times 0.8^1 \times 0.2^2 = 32$ objects.

For $\alpha = 3$, there will be $C_3^3 = \frac{3!}{3!0!} = 1$ bin with $1000 \times 0.8^0 \times 0.2^3 = 8$ objects.

These numbers agree with the results of Example 3.63.

Example B3.65

Show that the average number of non-empty bins in a binomial multifractal distribution of N objects and parameters K and p is:

$$F_0 = \sum_\alpha^K C_\alpha^K \left[1 - \left(1 - p^{K-\alpha}(1-p)^\alpha\right)^N\right]. \tag{3.194}$$

When placing the N objects in the bins,
the probability of the first object falling in a particular bin is $p^{K-\alpha}(1-p)^\alpha$;
the probability of the first object not falling in this bin is $1 - p^{K-\alpha}(1-p)^\alpha$;
the probability of none of the N objects falling in this bin is $\left(1 - p^{K-\alpha}(1-p)^\alpha\right)^N$;
the probability of this bin being non-empty is $1 - \left(1 - p^{K-\alpha}(1-p)^\alpha\right)^N$.
Since there are C_α^K bins with the same expected occupancy, determined by the value of α, the average number of non-empty bins F_0 is given by (3.194).

Box 3.16 Extension to 2D of the 1D binomial multifractal

A possible way to extent the definition of the 1D binomial multifractal to 2D is the following. At each step divide the square domain of the image function into four quadrants. Assign to the top left quadrant a probability p_1, to the top right a probability p_2 and to the bottom left a probability p_3. The bottom-right quadrant will receive objects with probability $1 - p_1 - p_2 - p_3$. You carry on dividing each quadrant in the same way, distributing the probabilities in the same way. Figure 3.62 shows the first two steps of this process.

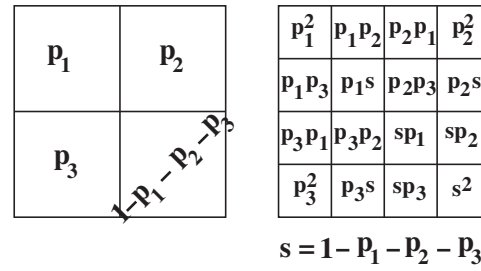


Figure 3.62 The first two stages of creating a 2D binomial multifractal.

At the end you have two options:

(i) You imagine you have N objects to distribute to the cells you have created and distribute them according to the probabilities assigned to each cell, i.e. by multiplying N with the corresponding probability and rounding the results to the nearest integer. If we were to interpret the output as an image, N here would have been the sum of the grey values of all the pixels of the image.

(ii) You scale the probability values you have assigned to the cells in the range $[0, 255]$, rounding the scaled values to the nearest integer.

Option (i) is more appropriate for using the 2D binomial multifractal to model a 2D (joint) histogram, while option (ii) is more appropriate for the creation of a 2D image. Figure 3.63 shows such an image constructed with $p_1 = 0.24$, $p_2 = 0.27$ and $p_3 = 0.29$.

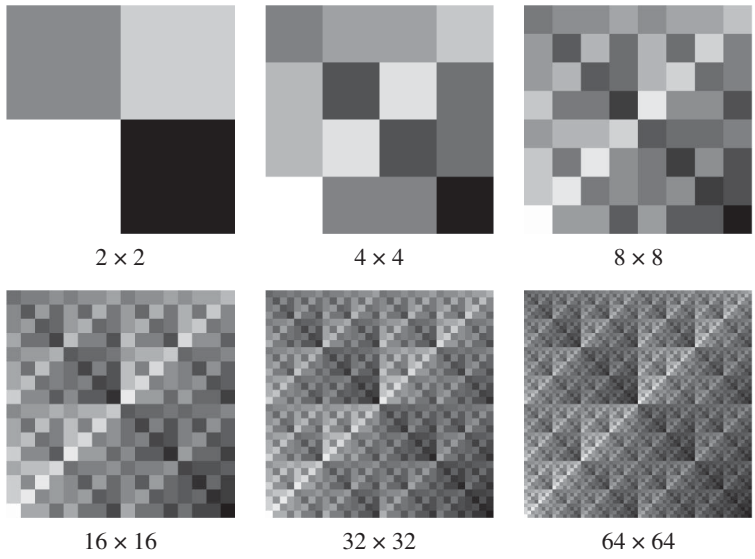


Figure 3.63 Creating a 2D binomial multifractal image.

Are multifractals good image models?

For some images they may be. In particular, multifractals may be used in computer graphics to produce realistically looking surfaces of natural scenes. However, multifractals fit a particular type of model to the data, which may not be appropriate. A far better approach is to let the data guide us to the best model, or not to use any model at all, but the data themselves, an approach known as non-parametric. This leads us to the following approaches:

(i) The variogram (and by extension the autocorrelation function) and the various models that can be used for it, with the fractal model being just one of many.
(ii) The various types of image histograms, which may be modelled with the **Weibull distribution**, that are versatile in shape and scale.
(iii) The various types of co-occurrence matrices, which may be considered as non-parametric representations.

All these approaches are under the general heading of **image statistics**.

3.4 Image Statistics

What is an image statistic?

It is a function that tells us how many pixels have what property. In the calculation of such a function, one may treat the pixels as individuals (**one-point statistics**), as pairs (**two-point statistics**), or even as n-tuples (**many-point statistics**).

What is a one-point statistic?

A one-point statistic is a distribution calculated by considering the pixels as individuals. Examples are the various histograms we construct using the assigned values pixels have, like the **rank-frequency plot**, or the **run length matrix**. Note that the values we use to construct a histogram do not need to be necessarily grey values; they may well be feature values calculated at the vicinity of each pixel and assigned to the pixel. Some histograms may be modelled by the **Weibull distribution**.

What is a two-point statistic?

A two-point statistic is calculated by considering pairs of pixels. Examples of such statistics are the **variogram**, the **normalised correlation function** of the image and the various types of **co-occurrence matrix**. The spatial **relative** position of the paired pixels is fixed when computing the statistic, but the statistic is computed for many such pairs of relative pixel positions.

What is a many-point statistic?

A many-point statistic is calculated by considering n-tuples of pixels. Examples are the **higher order co-occurrence matrices**, but also the **Markov random fields** (see Section 3.6) as well as, at the extreme case, the **Gibbs distribution** (see Section 3.7), in which case n is the number of all image pixels. As in the two-point statistic, the spatial *relative* position of the pixels of an n-tuple is fixed when computing the statistic.

How do we construct features from one-point, two-point or many-point image statistics?

If it is a simple statistic, e.g. a histogram made up from a few bins, the sequence of values of the statistic computed may be used directly as a signature of the texture. If the statistic is more complicated, e.g. Gibbs distribution, we model the corresponding distribution of the statistic and use the parameters of the model as features. An example here is the use of the **Markov parameters** as texture features capturing the distribution of the neighbourhood configurations in an image. Depending on what we do, we call the approach **parametric** (e.g. Markov random fields or Gibbs distributions) or **non-parametric** (e,g. co-occurrence matrices).

What is the histogram of the image?

It is a plot of the number of pixels that have a certain value versus the value, when the values are integer numbers, or against the bin number, when the values are real or integer. For example, if the histogram refers to grey values, which are integer numbers, we may plot the number of pixels that have a certain grey value versus the grey value. Alternatively, we may create bins of a certain width, say width 5, and for the value of the first bin we count together all pixels that have grey value in the range $[0, 4]$, for the second bin all pixels with grey value in the range $[5, 9]$ and so on. If we do not plot integer values, but real numbers, e.g. the gradient magnitude of each pixel, then the pixel values are not integers and the use of bins is compulsory. When the histogram is normalised by dividing its values with the total number of points we plotted and with the bin width, it may be treated as a probability density function.

Can we have rolling bins?

Yes. For example, in bin 1 we may count all pixels with grey value in the range $[0, 4]$, in the second bin all pixels with grey values in the range $[1, 5]$, in the third with $[2, 6]$ etc. Such a histogram counts all pixels as many times as the bin width, in this example 5 times, and so when normalised this has to be taken into account. The resultant histogram is smoother than the one created without the rolling bins. The use of rolling bins is equivalent to spreading the "vote" of each pixel equally to five neighbouring bins of width 1 grey value each. So, the same effect can be achieved by simply smoothing the histogram with a five-samples long averaging window. This naturally leads to the idea of smoothing the histogram with other more appropriate windows, like, for example, a Gaussian window.

Example 3.66

Compute the grey value histograms of images 3.1b, 3.1f, 3.1g and 3.1h. Use bins of grey value width 1. Then smooth the histograms by using an averaging window of five-bins long.

The results are shown in Figure 3.64.

(Continued)

Example 3.66 (Continued)

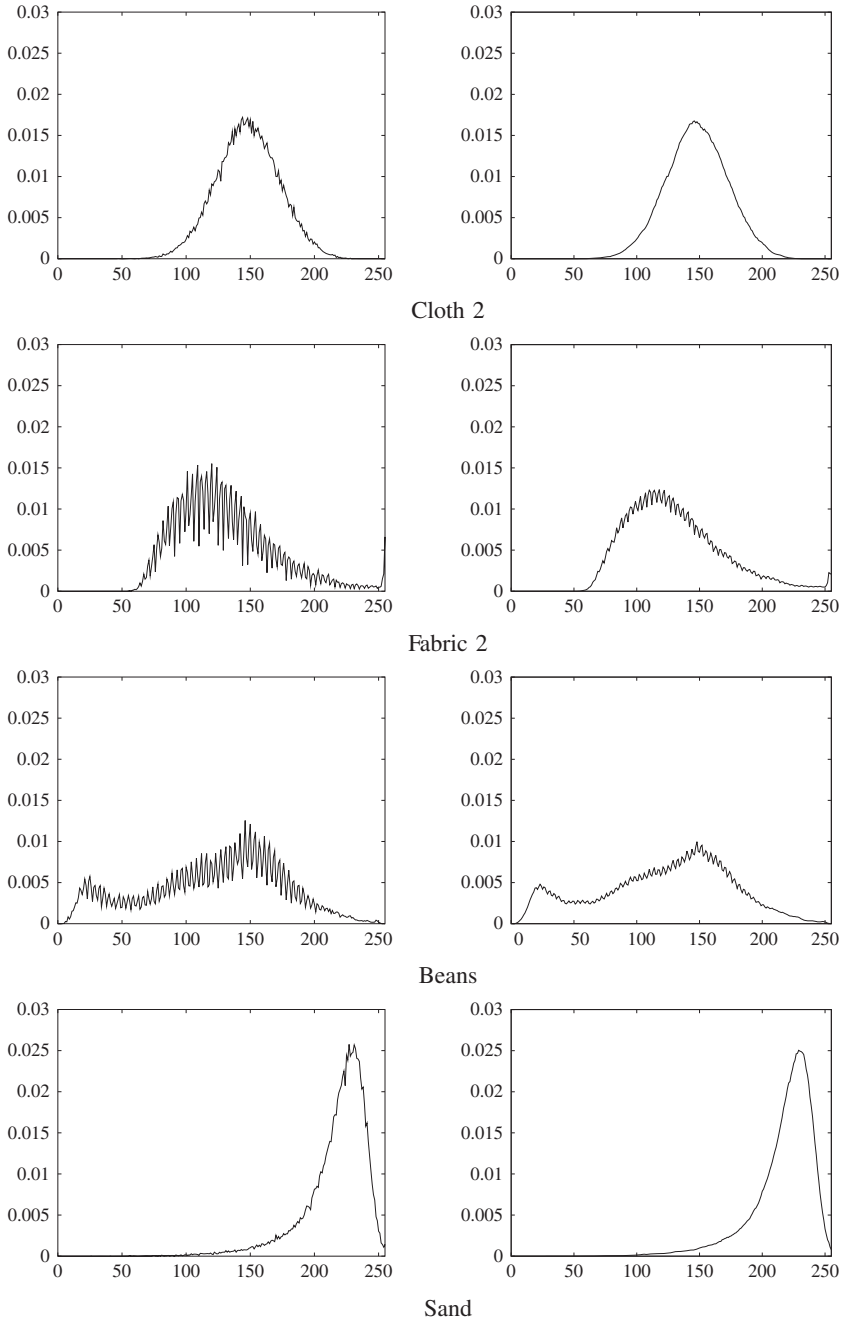


Figure 3.64 Raw (on the left) and smoothed (on the right) probability density function of a single pixel having a certain value (normalised histograms with bin width 1), for some of the textures of Figure 3.1.

Example 3.67

Compute the histograms of the gradient magnitude of all images in Figure 3.1. Use 30 bins, the same for all images.

First we have to compute the gradient magnitude at each pixel position (see the book Image Processing, The Fundamentals [75]). We may use for that a ***Sobel*** *filter: we convolve the image with filter* $(1, 2, 1)$ *horizontally and the result with filter* $(-1, 0, 1)$ *vertically, to work out the vertical component for the gradient vector at each pixel position. Let us call it* δ_y*. Then we convolve the original image vertically with filter* $(1, 2, 1)$ *and the result with filter* $(-1, 0, 1)$ *horizontally, to work out the horizontal component for the gradient vector at each pixel position. Let us call it* δ_x*. Then at each pixel we compute* $\sqrt{\delta_x^2 + \delta_y^2}$*. Finally, we divide each of these numbers by 8 to compute the gradient magnitude at each pixel position. This division is necessary because the smoothing filter* $(1, 2, 1)$ *has weights summing up to 4 and not to 1, and because the differencing filter* $(-1, 0, 1)$ *considers pixels two positions apart and not one to estimate the derivative.*

After we have done this for all images, we find the minimum b and the maximum a value of all gradient magnitudes we computed for all images. This is because we wish to have the same bins for all images, to be able to make direct comparisons. Since we need 30 bins, we define the bin width Δ *as:*

$$\Delta = \frac{\text{Max\_Value} - \text{Min\_Value}}{\text{Number\_of\_Bins}} = \frac{a - b}{30}. \tag{3.195}$$

Then we apply the following algorithm to create the histogram for each image.

Step 1: *Create an array H of 30 elements, enumerated from 0 to 29, all of which have initial value 0. This will be the histogram of the image.*

Step 2: *For each pixel value* $g \neq a$*, compute* $k \equiv \left\lfloor \frac{g-b}{\Delta} \right\rfloor$*. If* $g = a$ *set* $k = 29$*. This step identifies the bin to which this pixel belongs.*

Step 3: *Set* $H(k) = H(k) + 1$*. This step counts the pixel in the correct bin.*

The results are shown in Figure 3.65.

(Continued)

Example 3.67 (Continued)

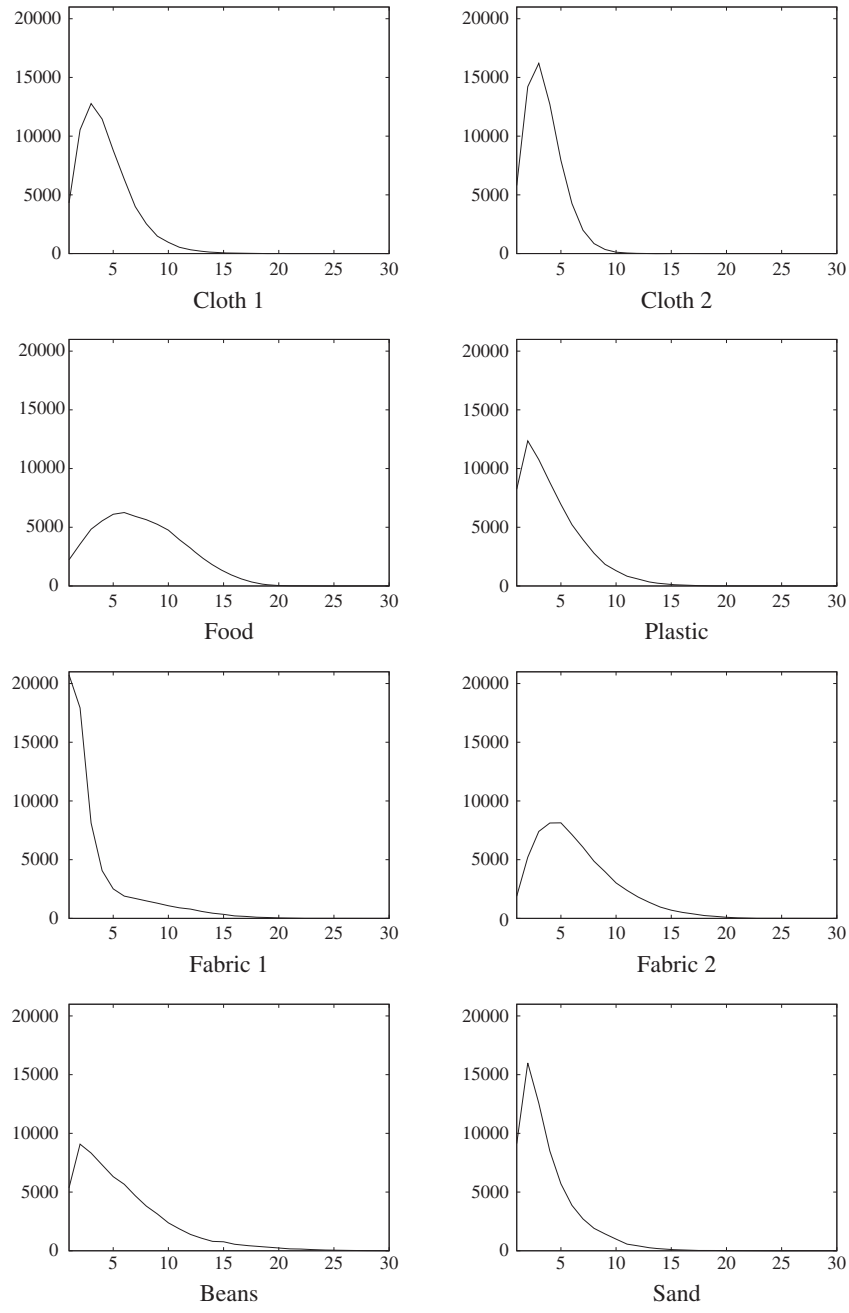


Figure 3.65 Histograms of the gradient magnitude for the textures of Figure 3.1.

Example 3.68

The *orientation histogram* of an image is defined by counting the number of pixels that have gradient vector in the same direction, with some tolerance that defines the width of the bin used for the construction of the histogram. The gradient orientation may be computed as follows. First, we use the method of Example 3.67 to compute the components of the gradient vector $(\delta x, \delta y)$ at each pixel position. From them we compute the orientation as $\theta = \tan^{-1}(\delta y/\delta x)$, taking care of the signs of the numerator and the denominator so the values of θ are in the range $[0^\circ, 360^\circ)$. Figure 3.66 shows the result of this process for a particular image. Compute the orientation histogram of this image and visualise it as a polar function.

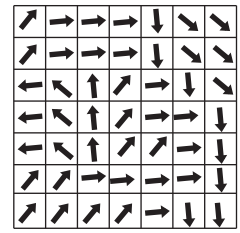


Figure 3.66 The orientations of the gradient vectors computed for each pixel.

We select to use bins of width 45°. *Then we count the number of orientation vectors in the ranges* $[0, 45^\circ)$, $[45^\circ, 90^\circ)$, $[90^\circ, 135^\circ)$, $[135^\circ, 180^\circ)$, $[180^o, 225^\circ)$, $[225^\circ, 270^\circ)$, $[270^\circ, 315^\circ)$ *and* $[315^\circ, 360^\circ)$. *The constructed histogram is shown in Figure 3.67a.*

To visualise the histogram, we consider polar coordinates in the plane, in order to plot the values of the histogram as a function of the orientation. First, we identify the middle value of each bin, i.e. 22.5°, 67.5°, *etc. and we plot a point at radius equal to the value of the histogram for that bin. Then we join the 8 points we plotted with straight lines. The result is shown in Figure 3.67b. A quick glance at this representation immediately makes it obvious that the image, from which the orientation field shown in Figure 3.66 was computed, was anisotropic, with dominant orientation along* $22.5^\circ \pm 22.5^\circ$.

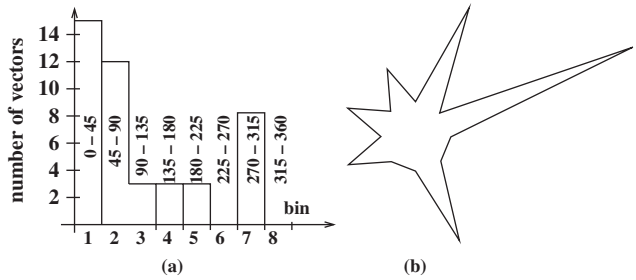


Figure 3.67 (a) The orientation histogram of image 3.66. (b) The orientation histogram visualised as a function of the polar angle.

Example B3.69

When creating histograms, it is very important to use bins with equal capacity. When creating the orientation histogram of 3D (volume) data, the bins represent orientations in 3D space. Show that, in order to create bins of equal solid angle, we must use cylindrical coordinates (R, ϕ, z) and divide the azimuth circle in equal segments and the z axis in equal segments too.

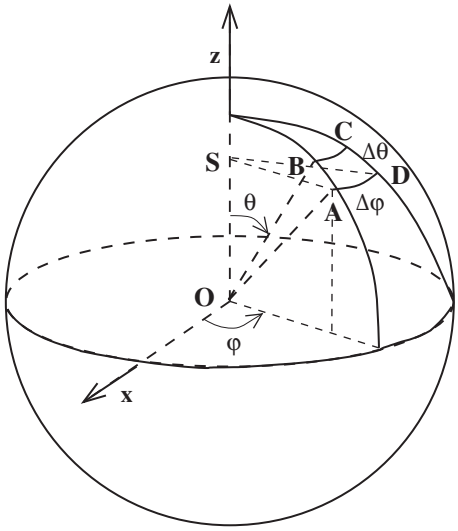


Figure 3.68 The unit sphere and an infinitesimal surface element on it.

Figure 3.68 shows the unit sphere. We know that a solid angle is measured in sterads equal to the area it subtends on the surface of the unit sphere with its centre at the vertex of the angle. We realise, therefore, that we can create bins of equal solid angle if we divide the surface of the unit sphere in spherical patches of equal area. Let us consider an infinitesimal surface element, as shown in Figure 3.68, created by considering two arcs on parallel circles of $\Delta\phi$ degrees, along the azimuth direction, and two arcs of $\Delta\theta$ degrees along two polar circles. These infinitesimal arcs are arcs AB, BC, CD and DA as shown in Figure 3.68. The area of the infinitesimal patch they define is $|AB| \times |AD|$, where $|AB|$ and $|AD|$ are the lengths of the corresponding arcs. Length $|AD|$ is clearly $|SA|\Delta\phi$, since $|SA|$ is the radius of the parallel circle to which this arc belongs. From triangle OAS, this radius is $|SA| = |OA| \sin\theta = \sin\theta$. So, $|AD| = \sin\theta\Delta\phi$. Length $|AB|$ is $|AB| = |OA|\Delta\theta = \Delta\theta$. So, the area of the infinitesimal patch is $\sin\theta\Delta\phi\Delta\theta$. If r is the radius of the sphere, cylindrical coordinate z is related to polar angle θ by $z = r\cos\theta = \cos\theta$. Therefore, $\Delta z = -\sin\theta\Delta\theta$. We may say, therefore, that the infinitesimal patch we considered has area $|\Delta z\Delta\phi|$, where the absolute value is used, because area is always positive. To create a finite patch, we have to integrate this area over a range of z and a range of ϕ. So, if we choose equal ranges in z and in ϕ, we shall create finite patches of equal area on the sphere, which will then be used to define the solid angle bins of the 3D orientation histogram we wish to construct.

Box 3.17 Constructing the 3D orientation histogram of a 3D image

Step 0: Select the number of bins you will use for the z axis. Call it N_z. Select the number of bins you will use for the azimuth angle ϕ. Call it N_ϕ.

Step 1: Compute the bin width for the z axis. The range of values of z is $[-1,1]$. So, set $\Delta z = 2/N_z$. Set $\Delta\phi = 360^\circ/N_\phi$.

Step 2: Set up an array H of size $N_z \times N_\phi$ and initialise all its elements to 0. Assume that the indices (i,j) of its elements will take values from 1 to N_z and from 1 to N_ϕ, respectively.

Step 3: Convolve the image along the x, y and z axis with the 3D filter that is $3 \times 3 \times 3$ in size, with weights along its three planes that are orthogonal to the direction of convolution as shown in Figure 3.69.

Figure 3.69 The generalisation of the Sobel filter to 3D. These are the three panels of weights when the convolution of the image happens from left to right.

−1	−2	−1
−2	−4	−2
−1	−2	−1

0	0	0
0	0	0
0	0	0

1	2	1
2	4	2
1	2	1

The output of each convolution should be divided by 32 to take into consideration the fact that smoothing panels have weights summing up to 16 and the derivatives are estimated by considering voxels two positions apart. The final output of this step will be three volumes, each one containing one component of the gradient vector at each voxel position. Let us call these components δx, δy and δz.

Step 4: For each voxel, compute:

$$O_z = \frac{\delta z}{\sqrt{\delta x^2 + \delta y^2 + \delta z^2}} \qquad \tilde{O}_\phi = \cos^{-1}\left(\frac{|\delta x|}{\sqrt{\delta x^2 + \delta y^2}}\right). \tag{3.196}$$

Note that $\tilde{O}_\phi$ is always going to be in the range $[0^\circ, 90^\circ]$. From $\tilde{O}_\phi$ we can compute O_ϕ, which must take values in the range $[0^\circ, 360^\circ)$, by taking into consideration the sign of δx and δy, as follows.

$$\begin{aligned}
&O_\phi = \tilde{O}_\phi && \text{if } \delta x > 0 \text{ and } \delta y > 0\\
&O_\phi = 180^\circ - \tilde{O}_\phi && \text{if } \delta x < 0 \text{ and } \delta y > 0\\
&O_\phi = 180^\circ + \tilde{O}_\phi && \text{if } \delta x < 0 \text{ and } \delta y < 0\\
&O_\phi = 360^\circ - \tilde{O}_\phi && \text{if } \delta x > 0 \text{ and } \delta y < 0.
\end{aligned} \tag{3.197}$$

Step 5: For each voxel, compute:

$$\tilde{i} = \lfloor \frac{O_z}{\Delta z} \rfloor \qquad j = \lfloor \frac{O_\phi}{\Delta\phi} \rfloor. \tag{3.198}$$

From $\tilde{i}$ we must compute the bin index i, as follows.
If N_z is even, say $N_z = 2k$, $i = \tilde{i} + k$.
If N_z is odd, say $N_z = 2k+1$ and $\delta z > 0$, $i = \tilde{i} + k + 1$.
If N_z is odd, say $N_z = 2k+1$ and $\delta z < 0$, $i = \tilde{i} + k$.

Step 5: Set:

$$H(i,j) = H(i,j) + 1. \tag{3.199}$$

Example B3.70

You want to construct the 3D orientation histogram for some volume data, using 10 bins in the z direction and 12 bins for the azimuth direction. A particular voxel has gradient vector $(15, -12, -8)$**. Work out the bin to which it belongs.**

First we compute the bin width for each component:

$$\Delta z = \frac{2}{10} = 0.2$$
$$\Delta\phi = \frac{360^\circ}{12} = 30^\circ. \tag{3.200}$$

We are given: $\delta x = 15$, $\delta y = -12$, $\delta z = -8$. *We compute:*

$$O_z = \frac{\delta z}{\sqrt{\delta x^2 + \delta y^2 + \delta z^2}} = \frac{-8}{\sqrt{15^2 + (-12)^2 + (-8)^2}} = -0.384$$
$$\tilde{O}_\phi = \cos^{-1}\left(\frac{|\delta x|}{\sqrt{\delta x^2 + \delta y^2}}\right) = \cos^{-1}\left(\frac{15}{\sqrt{15^2 + (-12)^2}}\right) = 38.66^\circ. \tag{3.201}$$

Since $\delta x > 0$ *and* $\delta y < 0$, $O_\phi = 360^\circ - 38.66^\circ = 321.34^\circ$.

Next, we work out:

$$\tilde{i} = \lceil \frac{O_z}{\Delta z} \rceil = \lceil \frac{-0.384}{0.2} \lceil = \lceil -1.92 \rceil = -1$$
$$j = \lceil \frac{O_\phi}{\Delta\phi} \rceil = \lceil \frac{321.34^o}{30^\circ} \lceil = \lceil 10.711 \rceil = 11. \tag{3.202}$$

As N_z *is even, equal to* 2×5, $k = 5$ *and* $i = -1 + 5 = 4$.

Figure 3.70 shows the accumulator array we create to make the histogram. The black dot indicates the bin to which this particular voxel contributes.

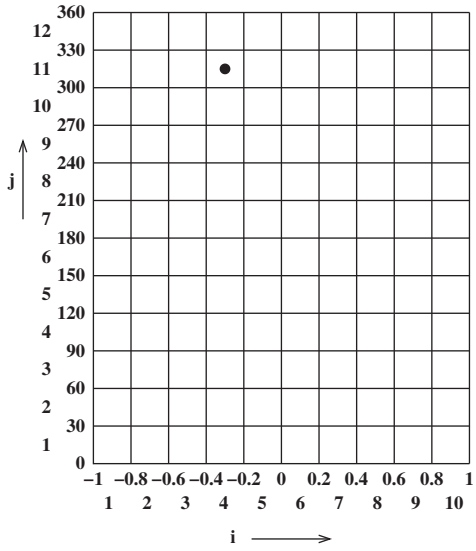


Figure 3.70 The accumulator array we create in order to compute the 3D orientation histogram with 10 bins along the z axis and 12 for the azimuth angle. The numbers next to the lines defining the various cells are the cell boundaries. The integers between the lines indicate the bin indices.

What is the rank-frequency plot?

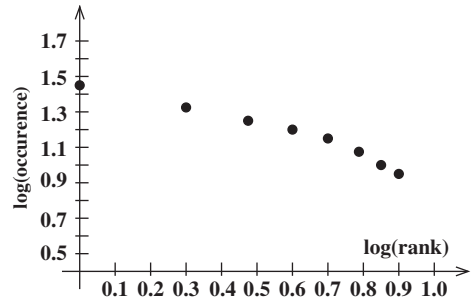
The rank-frequency plot is the plot of the logarithm of the number of pixels that have the same grey value versus the logarithm of the rank of these numbers, in decreasing order.

Example 3.71

Construct the rank-frequency plot for the 3-bit image of Figure 3.71a.

0	3	5	6	6	6	5	7	7	5	6	6
4	5	0	7	3	7	0	7	0	2	1	6
7	7	3	7	3	1	2	1	2	7	7	7
1	6	1	7	2	6	2	7	6	0	7	5
6	7	7	0	4	6	0	1	7	2	1	6
7	6	2	7	2	7	5	3	0	3	4	6
3	4	6	7	4	0	7	4	7	2	1	7
7	0	6	0	7	2	7	2	7	7	4	3
4	6	7	4	1	5	1	0	7	0	7	7
7	7	0	5	6	6	4	3	0	0	0	7
5	2	1	7	7	0	6	6	7	2	7	4
2	1	7	4	7	1	7	3	7	2	2	3

(a)



(b)

Figure 3.71 (a) A 3-bit grey image. (b) Its rank-frequency plot.

Table 3.13 shows the number of pixels for each one of the possible grey values for this image, their rank, in decreasing order of occurrence, and the logarithms of the values needed for the construction of the rank-frequency plot.

Figure 3.71b shows the rank-frequency plot constructed.

Table 3.13 The number of pixels with each one of the grey values and the rank of these numbers in decreasing order. Here $m(r)$ is the number of pixels with grey value with rank r.

Grey value	$m(r)$	$\log m(r)$	r	$\log r$
0	18	1.255	3	0.477
1	13	1.114	5	0.699
2	16	1.204	4	0.602
3	11	1.041	7	0.845
4	12	1.079	6	0.778
5	9	0.954	8	0.903
6	21	1.322	2	0.301
7	44	1.643	1	0.000

What is the run-length matrix?

The run length matrix is a matrix with elements P_{ij}, which return the number of times j consecutive pixels were found to have the same grey value i along a given direction.

Example 3.72

Construct the run-length matrices of the image in Figure 3.72, for the two main directions and along the two diagonal directions.

1	3	3	2	0	0
0	3	2	2	1	0
0	2	2	2	1	1
0	2	3	2	0	1
1	3	3	2	1	0
1	3	3	2	0	0

Figure 3.72 A 2-bit 6 × 6 image.

The four run-length matrices constructed for the four main directions of the image are shown in Figure 3.73.

→

grey value i \ run–length j	1	2	3	4	5	6
0	6	2	0	0	0	0
1	6	1	0	0	0	0
2	5	1	1	0	0	0
3	2	3	0	0	0	0

↓

grey value i \ run–length j	1	2	3	4	5	6
0	3	2	1	0	0	0
1	2	3	0	0	0	0
2	0	2	0	0	0	1
3	1	2	1	0	0	0

↘

grey value i \ run–length j	1	2	3	4	5	6
0	6	2	0	0	0	0
1	4	2	0	0	0	0
2	6	2	0	0	0	0
3	6	1	0	0	0	0

↗

grey value i \ run–length j	1	2	3	4	5	6
0	8	1	0	0	0	0
1	6	1	0	0	0	0
2	4	0	2	0	0	0
3	2	3	0	0	0	0

Figure 3.73 Each matrix is 4 × 6, as there are four different grey values and the maximum number of consecutive pixels we may find with the same grey value is 6. The arrows at the top left of each matrix indicate the direction along which each matrix has been constructed.

How do we compute the run-length matrix in practice?

We compute the run-length matrix in the following way:

Step 0: Set up a binary array S the same size as the image, where you will mark the pixels that already have been counted in some run length. Initialise all elements of the array to 0.

Step 1: Select the direction along which you wish to construct the matrix. If along the horizontal direction, set $i_0 = 1$ and $j_0 = 0$; along the vertical direction, set $i_0 = 0$ and $j_0 = 1$; along the top left to bottom right diagonal, set $i_0 = 1$ and $j_0 = 1$; along the bottom left to top right diagonal, set $i_0 = 1$ and $j_0 = -1$.
Step 2: Create an array P of size $(G + 1) \times N$, where G is the largest grey value in the image and N is the largest linear dimension of the image. Arrange so that index g of the array runs from 0 to G and index n runs from 1 to N. Initialise all elements of the array to 0.
Step 3: For each pixel (i,j) of input image I, which has 0 value in the marking array S:
Step 3.1: Set $k = 1$ and $S(i,j) = 1$.
Step 3.2: If $(i + i_0, j + j_0)$ is not out of the image border go to step 3.3.
Step 3.3: If $I(i,j) = I(i + i_0, j + j_0)$, set $k = k + 1$, $S(i + i_0, j + j_0) = 1$ and go to step 3.5.
Step 3.4: If $I(i,j) \neq I(i + i_0, j + j_0)$, go to step 3.6.
Step 3.5: Set $i = i + i_0$ and $j = j + j_0$ and go to step 3.2.
Step 3.6: Set $P(I(i,j),k) = P(I(i,j),k) + 1$ and go to step 3 to deal with the next unmarked pixel.

The marking array S ensures that a pixel that has been counted in a run length of, say, 5 pixels, will not be counted again as being the origin of a run length of 4 pixels.

How do we compute features from the run-length matrix?

The first step is to normalise all elements of the run-length matrix so they sum up to 1, in order to be able to treat it like a probability density matrix. We set

$$p_{ij} = \frac{P_{ij}}{\sum_{i=0}^{G}\sum_{j=1}^{N} P_{ij}} \tag{3.203}$$

where G is the maximum grey value of the image and N is the maximum linear size of the image.

Then we may compute the following features.

Short run emphasis is the expectation value of the run lengths, putting emphasis on the short ones by taking the inverse of each run length to the power of 2:

$$F_{\text{sre}} \equiv \sum_{i=0}^{G}\sum_{j=1}^{N} \frac{1}{j^2} p_{ij}. \tag{3.204}$$

Long run emphasis is the expectation value of the run lengths, putting emphasis on the long ones by putting them in the power of 2:

$$F_{lre} \equiv \sum_{i=0}^{G}\sum_{j=1}^{N} j^2 p_{ij} \tag{3.205}$$

Grey level non-uniformity is the marginal created by integrating over all runs, raised to the power of 2 to accentuate the different behaviours of the different grey values, and summed up for all grey values:

$$F_{\text{gln}} \equiv \sum_{i=0}^{G}\left(\sum_{j=1}^{N} p_{ij}\right)^2. \tag{3.206}$$

Run length non-uniformity is the marginal created by integrating over all grey values, raised to the power of 2 to accentuate the variety of the encountered run lengths, and summed up for all

run lengths:

$$F_{\text{rln}} \equiv \sum_{j=1}^{N}\left(\sum_{i=0}^{G} p_{ij}\right)^2. \tag{3.207}$$

Run percentage is the number of non-zero elements of matrix P over the total number of possible runs:

$$F_{\text{rp}} \equiv \frac{Number\_of\_p_{ij} \neq 1}{(G+1)N}. \tag{3.208}$$

The definitions of grey level non-uniformity and run length non-uniformity given here are different from those found in the original paper that introduced them, as those were computed from non-normalised matrices. The way we define these features here allows us to interpret them as marginals.

Example 3.73

Compute features (3.204)–(3.207) from the first run-length matrix of Figure 3.73.
The sum of all entries of the horizontal matrix is 27. Then:

$$\begin{aligned} F_{sre} &= \sum_{i=0}^{3}\sum_{j=1}^{6}\frac{1}{j^2}p_{ij} \\ &= \frac{1}{1^2}\times\frac{6}{27}+\frac{1}{1^2}\times\frac{6}{27}+\frac{1}{1^2}\times\frac{5}{27}+\frac{1}{1^2}\times\frac{2}{27} \\ &\quad +\frac{1}{2^2}\times\frac{2}{27}+\frac{1}{2^2}\times\frac{1}{27}+\frac{1}{2^2}\times\frac{1}{27}+\frac{1}{2^2}\times\frac{3}{27}+\frac{1}{3^2}\times\frac{1}{27} \\ &= \frac{1}{27}\times\frac{751}{36}=0.773 \end{aligned} \tag{3.209}$$

$$\begin{aligned} F_{lre} &= \sum_{i=0}^{3}\sum_{j=1}^{6} j^2 p_{ij} \\ &= 1\times\left(\frac{6}{27}+\frac{6}{27}+\frac{5}{27}+\frac{2}{27}\right) \\ &\quad +4\times\left(\frac{2}{27}+\frac{1}{27}+\frac{1}{27}+\frac{3}{27}\right)+9\times\frac{1}{27}=\frac{56}{27}=2.074 \end{aligned} \tag{3.210}$$

$$\begin{aligned} F_{gln} &= \sum_{i=0}^{3}\left(\sum_{j=1}^{6} p_{ij}\right)^2 \\ &= \left(\frac{6}{27}+\frac{2}{27}\right)^2+\left(\frac{6}{27}+\frac{1}{27}\right)^2+ \\ &\quad \left(\frac{5}{27}+\frac{1}{27}+\frac{1}{27}\right)^2+\left(\frac{2}{27}+\frac{3}{27}\right)^2 \\ &= \frac{64}{729}+\frac{49}{729}+\frac{49}{729}+\frac{25}{729}=\frac{187}{729}=0.257 \end{aligned} \tag{3.211}$$

$$F_{rln} = \sum_{j=1}^{6}\left(\sum_{i=0}^{3} p_{ij}\right)^2$$

$$= \left(\frac{6}{27} + \frac{6}{27} + \frac{5}{27} + \frac{2}{27}\right)^2 + \left(\frac{2}{27} + \frac{1}{27} + \frac{1}{27} + \frac{3}{27}\right)^2 + \left(\frac{1}{27}\right)^2$$
$$= \frac{361}{729} + \frac{49}{729} + \frac{1}{729} = \frac{411}{729} = 0.564 \tag{3.212}$$

$$F_{rp} = \frac{1}{24} \times \text{Number\_of\_p}_{ij} \neq 1 = \frac{9}{24} = 0.375. \tag{3.213}$$

What is the variogram of an image?

The variogram of an image I is a function $\gamma(d)$ that returns half the average squared difference in grey values of two pixels that are at a distance d apart:

$$\gamma(d) \equiv \frac{1}{2K} \sum_{\text{all } K \text{ pairs at dist. } d} [I(i,j) - I(i+k, j+l)]^2 \qquad \text{where } \sqrt{k^2 + l^2} = d. \tag{3.214}$$

Note that some people call this function a **semi-variogram** and define the variogram without the 2 in the denominator. A numerical factor plays no role in the use of $\gamma(d)$, so we do not worry about it here. The variogram is related to the image autocorrelation function $R(d)$ through

$$\gamma(d) = \sigma^2 - R(d) - \mu^2 \tag{3.215}$$

where σ^2 is the variance of the pixel values and μ is their mean value.

Example 3.74

The variogram of some data is defined as

$$\gamma(d) \equiv \frac{1}{2} E\{(V_i - V_j)^2\} \qquad \text{for } |i - j| = d \tag{3.216}$$

where E is the expectation operator, and positions i and j are at distance d from each other, while V_i is the value at position i. Prove Equation (3.215).

Let, for the sake of simplicity, index i identify a location (e.g. a pixel in an image) and index $i + d$ identify another location at distance d from i (e.g. another pixel in the image at distance d from the first along any direction). We also remember that the expectation operator means averaging over all pixels, or pairs of pixels, or whatever appears in its argument.

Let us start from the definition of $\gamma(d)$:

$$\gamma(d) \equiv \frac{1}{2} E\{V_i^2 + V_{i+d}^2 - 2V_i V_{i+d}\}$$
$$= \frac{1}{2} E\{V_i^2\} + \frac{1}{2} E\{V_{i+d}^2\} - E\{V_i V_{i+d}\}$$
$$= E\{V_i^2\} - E\{V_i V_{i+d}\}. \tag{3.217}$$

Here we made use of the fact that when averaging over all pixels, it does not matter whether we start from a shifted index or not. Clearly this is correct only if the data are infinite (i.e. index i takes infinite values), or if the data are repeated periodically, so no matter where we start from, as long

(Continued)

Example 3.74 (Continued)

as we consider the same number of samples/pixels, the average, i.e. the output of the expectation operator, will be the same: $E\{V_i^2\} = E\{V_{i+d}^2\}$.

We also have for the variance of the data:

$$\begin{aligned}\sigma^2 &\equiv E\{(V_i - \mu)^2\}\\ &= E\{V_i^2 + \mu^2 - 2V_i\mu\}\\ &= E\{V_i^2\} + \mu^2 - 2\mu E\{V_i\}\\ &= E\{V_i^2\} + \mu^2 - 2\mu^2\\ E\{V_i^2\} &= \sigma^2 - \mu^2. \end{aligned} \tag{3.218}$$

Then, by substitution from (3.218) into (3.217) and recognising that $E\{V_i V_{i+d}\} = R(d)$*, we obtain (3.215).*

Example 3.75

Prove that the variogram of some data $\gamma(d)$ **for shift** d **is related to the autocovariance of the data** $C(d)$ **through**

$$\gamma(d) = \sigma^2 - C(d) - 2\mu^2 \tag{3.219}$$

where μ **is the mean value of the data.**

The autocovariance function of the data is:

$$\begin{aligned}C(d) &\equiv E\{(V_i - \mu)(V_{i+d} - \mu)\}\\ &= E\{V_i V_{i+d} - V_i\mu - V_{i+d}\mu + \mu^2\}\\ &= E\{V_i V_{i+d}\} - \mu E\{V_i\} - \mu E\{V_{i+d}\} + \mu^2\\ &= E\{V_i V_{i+d}\} - \mu^2 - \mu^2 + \mu^2\\ \Rightarrow E\{V_i V_{i+d}\} &= C(d) + \mu^2. \end{aligned} \tag{3.220}$$

Here we made use of the fact that when the expectation operator is applied to individual values it yields the expected value of the data, i.e. μ*, and when the expectation operator is applied to a constant it leaves it unchanged.*

We may substitute now from (3.220) and (3.218) into (3.217) to obtain (3.219).

Example 3.76

Calculate the variogram of the signal shown in Figure 3.8a for $d = 1$, $d = 2$ **and** $d = 3$**.**

Using the entries of table 3.14 we can calculate the values of the variogram:

$$\gamma(1) = \frac{1}{48}(14 \times 0^2 + 2 \times 1^2 + 2 \times 2^2 + 2 \times 3^2 + 4 \times 4^2) = 1.92$$

$$\gamma(2) = \frac{1}{46}(4 \times 0^2 + 4 \times 1^2 + 4 \times 2^2 + 4 \times 3^2 + 7 \times 4^2) = 3.65$$

$$\gamma(3) = \frac{1}{44}(1\times 0^2 + 6\times 1^2 + 4\times 2^2 + 3\times 3^2 + 5\times 4^2 + 2\times 5^2 + 1\times 7^2) = 5.18. \quad (3.221)$$

Table 3.14 Number of pairs of samples with certain absolute difference in value, for different distance d from each other.

Absolute difference	$d = 1$	$d = 2$	$d = 3$
0	14	4	1
1	2	4	6
2	2	4	4
3	2	4	3
4	4	7	5
5	0	0	2
6	0	0	0
7	0	0	1
Total	24	23	22

How do we compute the variogram of an image in practice?

You may use the following simple algorithm, or alternatively we may use precalculated pairs of pixels by considering digital circles (see Example 3.90).

Step 1: Select a value of distance d.

Step 2: For each pixel (i, j) draw a random number ϕ in the range $[0, 90^\circ]$. Then consider the set of coordinates $(i + d\cos\phi, j + d\sin\phi)$. By interpolation, find the grey value at this position. Note that the interpolation you use might be as simple as nearest neighbour interpolation. The restriction of the values of angle ϕ in the first quadrant ensures that you will not count a pair of positions twice.

Step 3: Compute the average square difference of the values of all paired positions you considered. Divide this value by 2. This is $\gamma(d)$.

Step 4: Repeat from step 1 by considering another value of d.

Note that it is possible to consider many angles ϕ to construct several pairs from a particular pixel. In general, however, this is not considered necessary if we have enough pixels to construct reliable statistics by creating a single pair from each pixel. It is, of course possible to fix angle ϕ and construct a different variogram for different orientations in order to capture texture **anisotropy**. It is also worth noting that this method is most appropriate for dense grid data. Often, however, the data we have are sparse, i.e. they are not arranged in a regular grid. This often happens in

geophysics, where the data are collected at positions where data collection is possible and they do not, in general, constitute a rectangular grid. As such data characterise natural structures or physical phenomena, their variation is often characterised by their variogram.

How do we compute the variogram of irregularly sampled data in practice?

Assume that we have data points spread over a grid of size $N \times N$. Assume also that only $M < N^2$ out of these grid positions have known values. These grid positions, with known values, are randomly scattered all over the $N \times N$ grid. We index these points sequentially using index i, $i = 1, 2, \ldots, M$ and assume that the value of point i is V_i, while its grid coordinates are (x_i, y_i). We can compute the variogram of these values using the following algorithm.

Step 1: Create a list of x_i, y_i and V_i of all the points with known values.

Step 2: Compute $\left\lfloor N \times \sqrt{2} \right\rfloor \equiv K$. This is the largest possible distance between any two points with known values which is along the grid diagonal.

Step 3: Create two accumulator arrays $N(k)$ and $S(k)$ for $k = 1, 2, \ldots, K$, and set all their elements to 0.

Step 4: Go through the list of points i, with $i = 1, 2, \ldots, M$, and for each $j > i$, set:

$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$

$k = \lfloor d \rfloor$

$N(k) = N(k) + 1$ and $S(k) = S(k) + (V_i - V_j)^2$.

Step 5: Set: $\gamma(k) = S(k)/(2N(k))$.

Example 3.77

Compute the variogram of the images of Figure 3.1 for $d = 1, 2, 3, 4, 5$**.**

Then subsample each image by randomly keeping only 10% **and** 20% **of its pixels and calculate again their variograms.**

Figure 3.74 shows the results. We observe that even with only 10% *of the pixels, randomly selected from the image, the computed variogram does not look different from that computed by using all the pixels. This is a very important observation, as it means that the variogram of the data on a complete grid may be adequately inferred from sparse data. For textures that are adequately characterised by their second order statistics, one then may assign values to the empty cells of a sparsely populated grid, so that the variogram of the constructed full grid agrees with the variogram of the given values. This method is used in geosciences (see Section 3.8).*

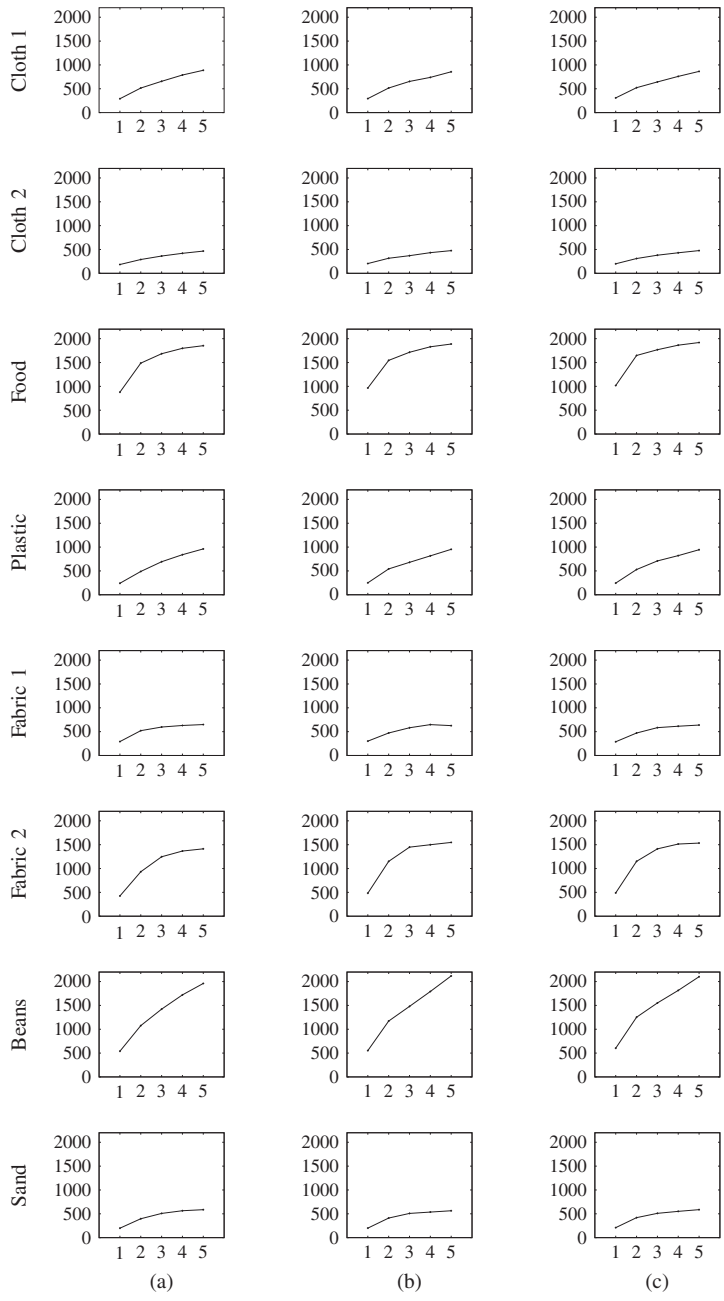


Figure 3.74 Variograms for the images of Figure 3.1. (a) Using the whole image. (b) Using only 10% of the pixels, randomly selected. (c) Using 20% of the pixels, randomly selected.

How can we use the variogram to characterise textures?

A variogram is characterised by three parameters, shown in Figure 3.75, namely the **range**, the **sill** and the **nugget**.

Range is the shift value R_d beyond which the variogram $\gamma(d)$ does not change as d increases.

Sill is the value the variogram attains at the range R_d, which does not change as d increases.

Theoretically, $\gamma(0)$ is expected to be 0. However, when we construct the variogram empirically, due to noise in the measurements or even microstructure in the data, the slope of the points on the left does not often indicate that the data pass through the $(0, 0)$ point. This is known as the **nugget effect**. The intercept of the notional line that fits the data with the vertical axis is known as the nugget of the variogram.

The values of these parameters, as well as the parameters of any model we use to fit the variogram over its range, may be used as features that characterise the texture.

How may we compute automatically the nugget, sill and range of the variogram?

The nugget may be computed by extrapolating a line fitting the data for small values of d until its intercept with the vertical axis is found.

To compute the sill and range, we create a histogram of the values of γ. We then identify the bin with the maximum number of values. This must be the bin that contains all points that correspond to the flat part of the variogram. The average value of this bin is the sill. From all points d_k that had values in the bin with the maximum occupancy, we select the point with the minimum value of d_k. This is the range of the variogram. The algorithm is as follows.

Step 0: Select a width γ_0 of the bins for the values of γ you will use.
Step 1: Set:

$$\gamma_{\min} \equiv \min_d \{\gamma(d)\} \qquad \gamma_{\max} \equiv \max_d \{\gamma(d)\} \qquad d_{\max} \equiv \max\{d\}. \tag{3.222}$$

Step 2: Set $N \equiv \left\lfloor \frac{\gamma_{\max}}{\gamma_0} \right\rfloor + 1$.
Step 3: Create an array A of size N and initialise all its elements to 0. Let its elements be indexed by $k = 1, \ldots, N$.
Step 4: For d from 1 to $d_{\max}$, compute:

$$k = \lceil \frac{\gamma(d)}{\gamma_0} \rceil \qquad A(k) = A(k) + 1. \tag{3.223}$$

Step 5: Find the maximum of $A(k)$, $A_{\max}$, and the corresponding index $k_{\max}$ such that $A(k_{\max}) = A_{\max}$.
Step 6: Create an array B of size $A_{\max}$. Set $l = 1$.
Step 7: For d from 1 to $d_{\max}$: If $(k_{\max} - 1)\gamma_0 \leq \gamma(d) < k_{\max}\gamma_0$, set:

$$B(l) = d \qquad l = l + 1. \tag{3.224}$$

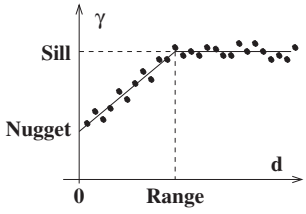


Figure 3.75 The black dots represent data values. The solid line represents the generic shape of the variogram.The value at $d = 0$ is the nugget, the constant value for $d \geq$ range is the sill. We may identify the sill by collapsing the data on the vertical axis and finding the bin with the maximum occupancy.

Step 8: Find the minimum of $B(l)$. This is the range of the variogram.
Step 9: Compute the average value of $\gamma(B(l))$. This is the sill of the variogram.

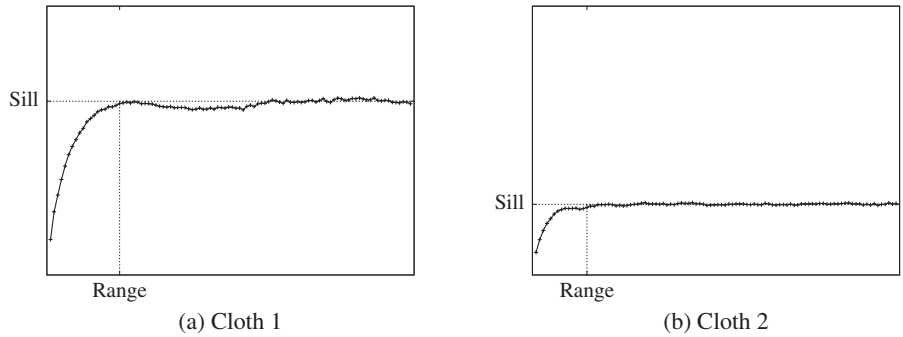
Example 3.78

Compute automatically the range and the sill of the variograms of images 3.1.
We used bin width $\gamma_0 = 50$ and $d_{\max} = 100$. Table 3.15 shows the values of the range and sill of these images computed automatically. We also list the variance of each image. The agreement the variance has with the value of the sill is an indication that the sill and range have been computed correctly. Indeed, beyond the range, the variogram uses the differences of uncorrelated data. As these data values are subtracted, the variance of their difference is twice the variance of the data values themselves. The average squared difference is divided by 2 to produce the variogram value, and, therefore, what the variogram computes beyond the range is the variance of the grey values.

Figure 3.76 shows two indicative results of the variograms of two of the images with the identified values of range and sill marked.

Table 3.15 The range and sill of the variograms of images 3.1, as well as the variance of the grey values of each image.

Image	Range	Sill	Variance
Cloth 1	20	1421.6	1413.6
Cloth 2	15	578.2	592.9
Food	7	1929.2	1941.7
Plastic	16	1423.7	1450.4
Fabric 1	19	875.6	862.5
Fabric 2	10	1469.4	1465.6
Beans	12	2770.1	2767.6
Sand	4	594.7	603.7



(a) Cloth 1 (b) Cloth 2

Figure 3.76 The variograms of the two images in Figure 3.1.

Can we use the variogram for anisotropic textures?

Yes. We may compute the variogram for paired pixels along the same orientation, and thus have a different variogram for different directions of the image. So, the variogram may be expressed as $\gamma(d_1, d_2)$, with d_1 and d_2 being the relative shifts of the pixels along the horizontal and vertical axis, respectively.

How can we model an anisotropic variogram?

Although different models may be used along different directions, often we use a separable function, so that the variogram is modelled as a function of the distance of the paired pixels times a function of the orientation angle. For example, for the case of the fractal model, where $\gamma(d) \sim d^{2H}$, the model becomes $\gamma(d) \sim d^{2H} f(\theta)$, where θ is a function of the orientation angle θ.

Example 3.79

You are given a random field $g(i,j)$, with variogram $\gamma(d_1, d_2)$. Assume that you distort this random field, by shifting the value of each pixel $(i,j)^T \equiv \mathbf{r}$ to a new position $\mathbf{r}' = (i', j')^T$, given by

$$\mathbf{r}' = \begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} \equiv M\mathbf{r}. \tag{3.225}$$

Show that the variogram of the distorted random field is given by $\gamma(M(d_1, d_2)^T)$.
Let us calculate the variogram of the new field $g'(i', j')$ we created. Let us denote by $\mathbf{s}' \equiv (k', l')^T$ another location in field $g'(i', j')$ and by $\mathbf{s} \equiv (k, l)^T$ the corresponding location in field $g(i, j)$. We also note that the arguments of the variogram of $g(i, j)$ represent shift between two locations, so we may write $i - k \equiv d_1$ and $j - l \equiv d_2$. Then:

$$\begin{aligned}
E\{|g'(i', j') - g'(k', l')|^2\} &= E\{|g'(\mathbf{r}') - g'(\mathbf{s}')|^2\} \\
&= E\{|g(M\mathbf{r}) - g(M\mathbf{s})|^2\} \\
&= E\{|g(m_{11}i + m_{12}j, m_{21}i + m_{22}j) - g(m_{11}k + m_{12}l, m_{21}k + m_{22}l)|^2\} \\
&= \gamma(m_{11}i + m_{12}j - m_{11}k - m_{12}l, m_{21}i + m_{22}j - m_{21}k - m_{22}l) \\
&= \gamma(m_{11}(i - k) + m_{12}(j - l), m_{21}(i - k) + m_{22}(j - l)) \\
&= \gamma\left(M(d_1, d_2)^T\right).
\end{aligned} \tag{3.226}$$

Example 3.80

You are given the isotropic fractal images shown in Box 3.14. Construct anisotropic fractal images from them, with the help of the transformation matrix

$$M = \begin{pmatrix} \sqrt{1-\alpha}\cos\theta_0 & \sqrt{1-\alpha}\sin\theta_0 \\ -\sqrt{1+\alpha}\sin\theta_0 & \sqrt{1+\alpha}\cos\theta_0 \end{pmatrix} \tag{3.227}$$

where α is a parameter controlling the degree of anisotropy of the created image, with $0 \leq \alpha \leq 1$, so that for $\alpha = 0$ the new image is also isotropic, while for $\alpha = 1$ the new image has maximum anisotropy, and θ_0 is the dominant direction of anisotropy.

To construct an anisotropic version of the given image, we first create an empty array the same size as the given image. Let us identify its pixels with indices (i', j'). Matrix M tells us how the coordinates (i, j) of the original image are transformed to the coordinates of the new image. This, however, is not what we need. We need to know how the (i', j') coordinates of the new image are transformed to the (i, j) of the original image, so we can read directly the grey value of the original image and assign it to the transformed coordinates. For this purpose, we need the inverse of matrix M, which can easily be calculated. First we calculate the determinant of the matrix:

$$\sqrt{1-\alpha^2}\cos^2\theta_0 + \sqrt{1-\alpha^2}\sin^2\theta_0 = \sqrt{1-\alpha^2}. \tag{3.228}$$

Then, with the help of co-factors, we obtain:

$$M^{-1} = \begin{pmatrix} \frac{1}{\sqrt{1-\alpha}}\cos\theta_0 & -\frac{1}{\sqrt{1+\alpha}}\sin\theta_0 \\ \frac{1}{\sqrt{1-\alpha}}\sin\theta_0 & \frac{1}{\sqrt{1+\alpha}}\cos\theta_0 \end{pmatrix}. \tag{3.229}$$

Then we apply the following algorithm.

Step 1: *For every pixel (i', j') of the new image, work out the corresponding pixel in the original image by multiplying $(i', j')^T$ from the left with matrix M^{-1}, given by (3.229). The coordinates you will find will be real numbers, (x, y).*

Step 2: *Round coordinates (x, y) to the nearest integer, to identify the pixel in the original image (i, j) that corresponds to pixel (i', j') of the new image.*

Step 3: *Read the grey value of pixel (i, j) and assign it to pixel (i', j').*

Note that this algorithm uses nearest neighbour interpolation to assign a grey value to the non-integer position (x, y). Other methods, like bilinear interpolation might be used instead (see the book Image Processing, The Fundamentals [75]).

Figure 3.77 shows anisotropic textures produced from Figure 3.48 using parameters $\theta_0 = 30°, 45°, 60°$ and $\alpha = 0, 0.5$ and 0.9.

(Continued)

Example 3.80 (Continued)

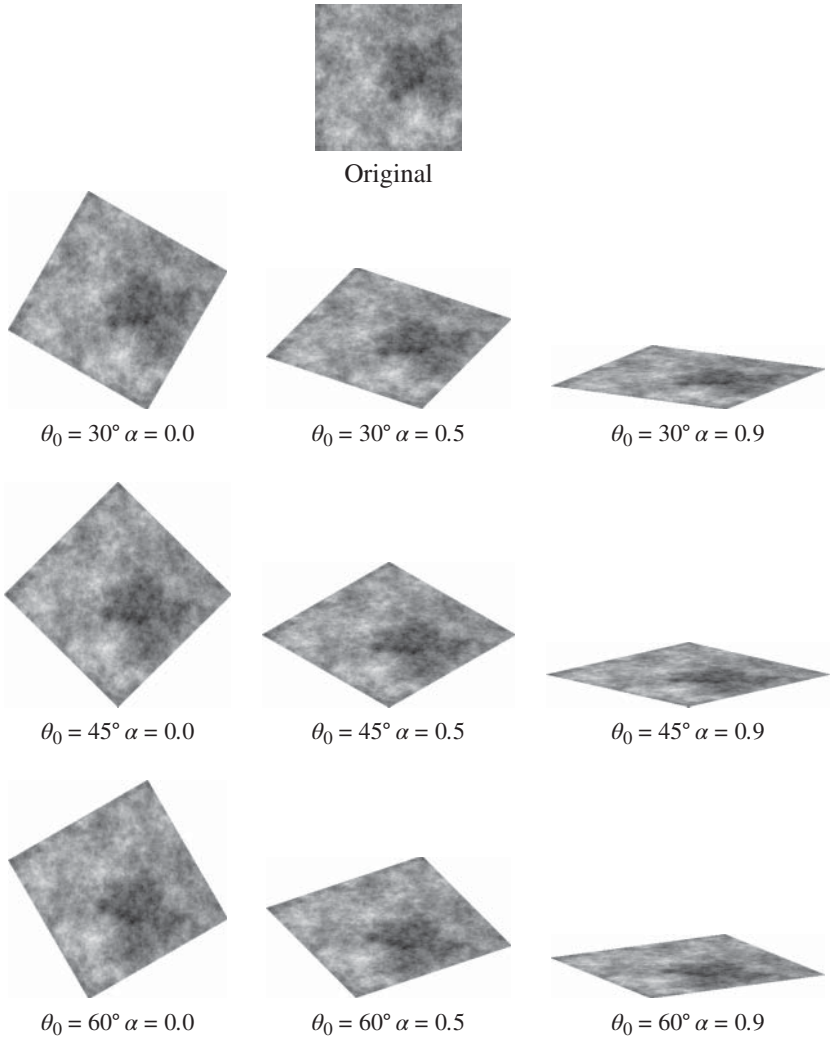


Figure 3.77 Distorted versions of the 2D fractal in Figure 3.48 for $H = 0.3$. The orginal image is 257×257 pixels. Source: Maria Petrou.

Example 3.81

An isotropic fractal image with variogram $\gamma(d_1, d_2) \propto (d_1^2 + d_2^2)^H$, was used to create an anisotropic fractal image with the help of matrix M given by Equation (3.227). Show that the variogram of the distorted image will be proportional to:

$$(d_1^2 + d_2^2)^H \big(1 - \alpha \cos[2(\theta - \theta_0)]\big)^H. \tag{3.230}$$

According to Example 3.79, the variogram of the distorted image will be given by $\gamma(M(d_1,d_2)^T)$. We must first calculate $M(d_1,d_2)^T$ and then substitute it into the formula that gives the variogram of the original image. To be consistent with the isotropic version of the image, we shall use polar coordinates for the displacement vector $(d_1,d_2)^T$, by considering that it consists of a displacement distance d along a direction θ. Then we can write: $(d_1,d_2)^T=(d\cos\theta,d\sin\theta)^T$. So:

$$\begin{aligned}M\begin{pmatrix}d_1\\d_2\end{pmatrix}&=\begin{pmatrix}\sqrt{1-\alpha}\cos\theta_0 & \sqrt{1-\alpha}\sin\theta_0\\ -\sqrt{1+\alpha}\sin\theta_0 & \sqrt{1+\alpha}\cos\theta_0\end{pmatrix}\begin{pmatrix}d\cos\theta\\d\sin\theta\end{pmatrix}\\&=\begin{pmatrix}\sqrt{1-\alpha}\cos\theta_0 d\cos\theta+\sqrt{1-\alpha}\sin\theta_0 d\sin\theta\\ -\sqrt{1+\alpha}\sin\theta_0 d\cos\theta+\sqrt{1+\alpha}\cos\theta_0 d\sin\theta\end{pmatrix}\\&=\begin{pmatrix}d\sqrt{1-\alpha}\cos(\theta-\theta_0)\\ d\sqrt{1+\alpha}\sin(\theta-\theta_0)\end{pmatrix}.\end{aligned}\tag{3.231}$$

Then the variogram of the new image is:

$$\begin{aligned}\gamma(M(d_1,d_2)^T)&\propto\left[\left(d\sqrt{1-\alpha}\cos(\theta-\theta_0)\right)^2+\left(d\sqrt{1+\alpha}\sin(\theta-\theta_0)\right)^2\right]^H\\&=\left[d^2(1-\alpha)\cos^2(\theta-\theta_0)+d^2(1+\alpha)\sin^2(\theta-\theta_0)\right]^H\\&=\left[d^2\cos^2(\theta-\theta_0)-d^2\alpha\cos^2(\theta-\theta_0)+d^2\sin^2(\theta-\theta_0)+d^2\alpha\sin^2(\theta-\theta_0)\right]^H\\&=\left[d^2-d^2\alpha\left[\cos^2(\theta-\theta_0)-\sin^2(\theta-\theta_0)\right]\right]^H\\&=d^{2H}\left[1-\alpha\cos[2(\theta-\theta_0)]\right]^H.\end{aligned}\tag{3.232}$$

Can we use the autocorrelation function itself to characterise a texture?

In general we do not use the autocorrelation function, but the normalised autocorrelation function, defined as

$$\rho(x,y)=\frac{\frac{1}{(N_i-|x|)(N_j-|y|)}\sum_i\sum_j I(i,j)I(i+x,j+y)}{\frac{1}{N_iN_j}\sum_{i=1}^{N_i}\sum_{j=1}^{N_j}I(i,j)^2}\tag{3.233}$$

where $I(i,j)$ is the grey value of pixel (i,j) and the image is assumed to be of size $N_i\times N_j$. The summation over i and j in the numerator is over all legitimate values of these coordinates, depending on the sign of the shifts x and y:

		Sum i		Sum j	
		From	To	From	To
$x\geq 0$	$y\geq 0$	1	N_i-x	1	N_j-y
$x\geq 0$	$y<0$	1	N_i-x	$1-y$	N_j
$x<0$	$y\geq 0$	$1-x$	N_i	1	N_j-y
$x<0$	$y<0$	$1-x$	N_i	$1-y$	N_j

Note that $\rho(0,0)=1$. This definition of the autocorrelation function does **not** obey the Wiener–Kinchine theorem, i.e. its Fourier transform is **not** the power spectrum of the image (see

Example 3.109). This is because for the theorem to be obeyed, the image has to be assumed to be repeated ad infinitum in all directions, while in the present definition we do not make such an assumption.

Another important point to note about the above definition is that although we say we are computing the autocorrelation function, in practice we must remove the mean of each image before we apply formula (3.233). So, we are actually computing the auto-covariance function, and **not** the autocorrelation. This is because we are interested in genuine high-frequency variations of the data and not any drift that might be caused, for example, by spatially variable illumination. Figure 3.78 shows four texture images, and, next to each one, the corresponding auto-covariance function appropriately scaled to be presented as a surface. Figure 3.79 shows the auto-covariance functions of two other images and below each one the corresponding autocorrelation function. The effect of failing to remove the mean is very clear.

The normalised autocorrelation function as defined above may be used in three ways to characterise a texture:

- By using it directly as a signature
- By inferring the periodicity of the texture from it
- By extracting parametric features from it.

How can we use the autocorrelation function directly for texture characterisation?

We can use $\rho(x, y)$ directly as representing the texture and compare it point by point with the autocorrelation function of another texture to see how similar the two textures are. The comparison may be done by computing, for example, the sum of the squares of the differences, or the correlation coefficient between the two functions, or some other measure of similarity.

Alternatively, one may use $\rho_x(x) \equiv \sum_y \rho(x, y)$ and $\rho_y(y) \equiv \sum_x \rho(x, y)$ as the texture signatures. It is expected that if a texture is isotropic, these two signatures will be, to a large extent, identical. However, when one is dealing with stochastic phenomena, the word "identical" has to be used with caution. For example, Figure 3.80 shows these two signatures for two random (and isotropic-looking) textures. It is very difficult to say that the $\rho_x(x)$ and $\rho_y(y)$ marginal signatures of the auto-covariance function of each texture look the same, apart from some general characteristics: the signatures of the micro-texture have higher density of peaks than the signatures of the macro-texture. For textures that have some periodicity, and in particular periodicity along two orthogonal directions, like the textures shown in Figure 3.81, one might have expected the two signatures to look the same. The plots in Figure 3.81 show that this is not the case: a slight misalignment of the direction of repetition of the pattern with one of the axes totally destroys the periodic nature of one of the signatures.

Figure 3.82 shows two macro-textures, a directional periodic and an isotropic random. What characterises these signatures is the lack of much detail in their shapes. They make a clear statement on the type of texture they are called to characterise. In particular, the signature along the y axis for the fabric texture captures very accurately the periodicity of that texture along that axis, and indicates that we may use the auto-covariance function to infer the periodicity of the texture pattern.

How can we infer the periodicity of a texture from the autocorrelation function?

If a texture has a repeated primitive pattern, every time the shift of the autocorrelation function is equal to the repetition shift, $\rho(x, y)$ will have a peak. The number of peaks we observe may be used

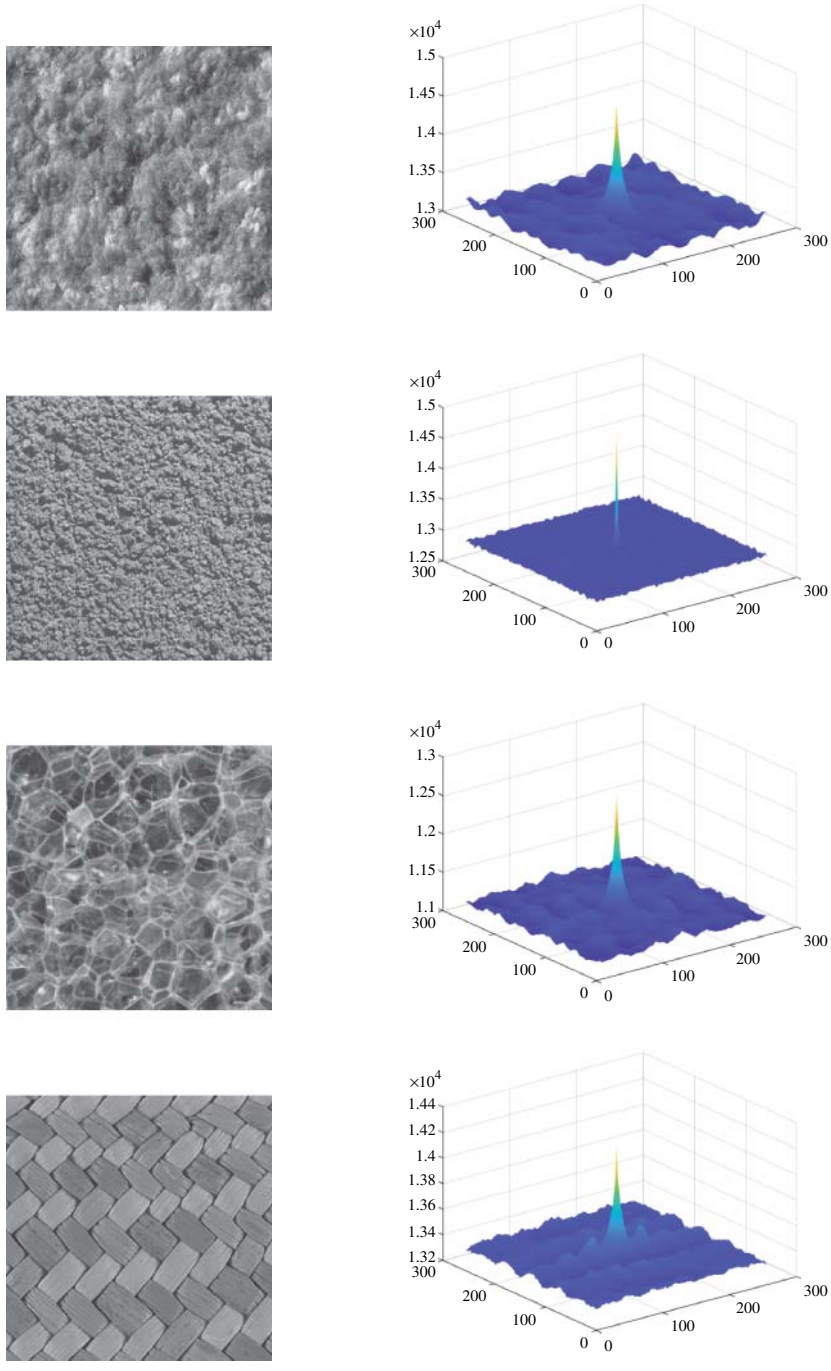


Figure 3.78 Normalised auto-covariance function (the autocorrelation function after having the mean of each image removed) for images cloth 1, food, plastic and fabric 1, from top to bottom, respectively.

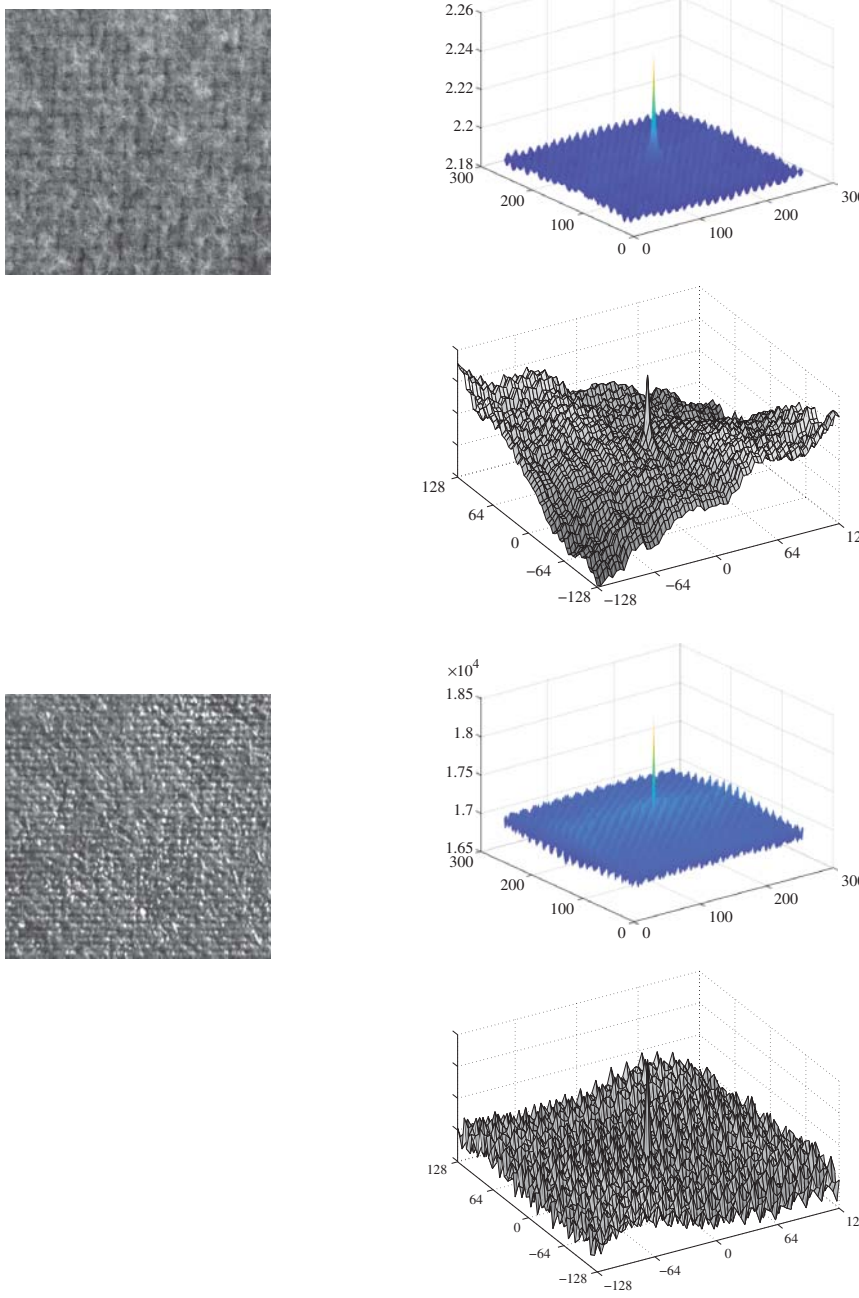
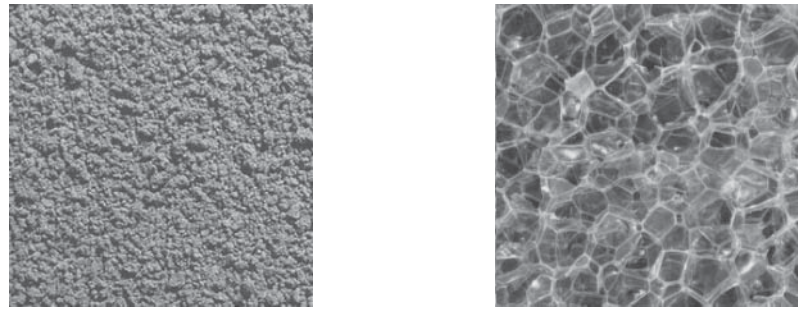
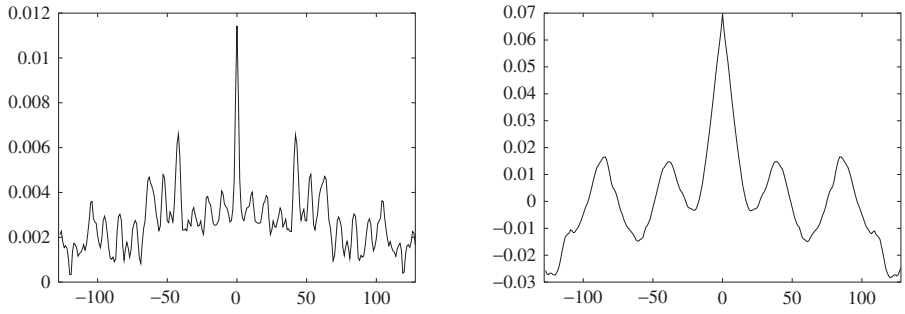


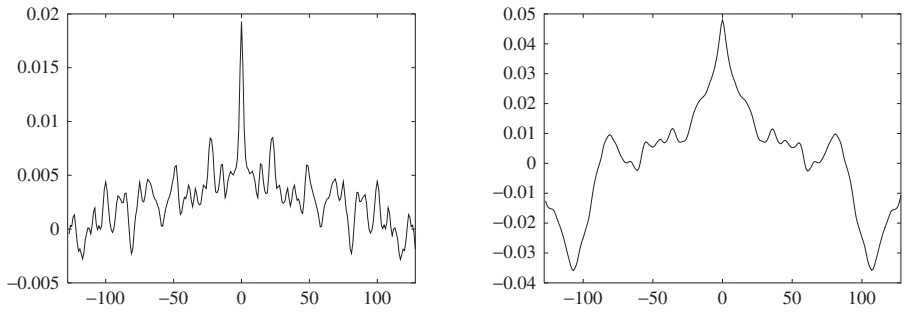
Figure 3.79 Normalised auto-covariance (top) and autocorrelation functions (below them) for cloth 2 and fabric 2, from top to bottom, respectively.



Original images



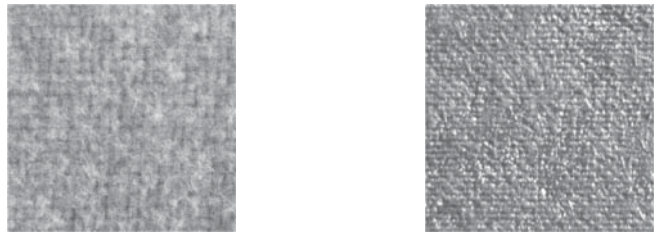
$\rho_x(x)$ signatures



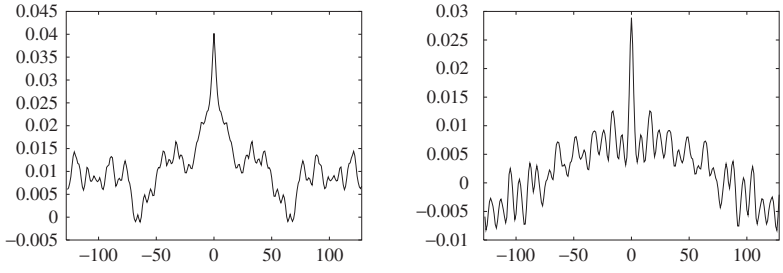
$\rho_y(y)$ signatures

Figure 3.80 Marginals of the auto-covariance function for images food and plastic.

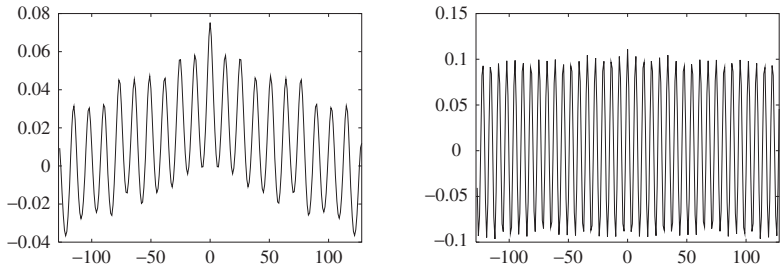
to characterise the texture, and from their positions one may infer the periodicity of the texture. Table 3.16 lists the number of peaks observed in each marginal of $\rho(x, y)$, for all example images of Figure 3.1. By simply looking at these numbers we may say that, for example, the bean texture is a macro-texture (few peaks) isotropic (more or less the same number of peaks in both signatures) with an **effective** periodicity of $T_{\text{eff}} = 256/12 \simeq 21$. T_{eff} was computed by observing that we have an average of 12 $(= (13 + 11)/2)$ peaks in a dimension of 256 (the image is 256×256 in size). So, we may say for this texture that it is macroscopic, isotropic, with blobs of linear size of about 21 pixels. We may make a similar inference for the texture plastic (Figure 3.80). The values of the



Original images



$\rho_x(x)$ signatures



$\rho_y(y)$ signatures

Figure 3.81 Marginals of the auto-covariance function for images cloth 2 and fabric 2.

central peaks of the two marginals, also listed in Table 3.16, may be used as discriminating features between these two textures.

Using the same logic, texture fabric 1 (Figure 3.82) appears to be an anisotropic texture, with more or less double periodicity along one of the axes. We may infer that it is a macroscopic texture made up of elongated blobs that along the x axis have size roughly $256/11 \simeq 23$ and along the y axis $256/5 \simeq 51$. This assessment is roughly correct. However, when it comes to cloth 2, this logic leads to the wrong conclusions: according to the number of peaks along the two signatures, cloth 2 (Figure 3.81) appears to be an anisotropic micro-texture. From the actual signatures themselves, cloth 2 appears anisotropic, perfectly periodic along the y axis, with random fluctuations along the x axis. Neither of these assessments is correct. The reason for this failure is the fact that we are dealing with an anisotropic micro-texture, and the use of two marginals only to characterise the auto-covariance function is not enough: for a better characterisation, the marginal of the auto-covariance function has to be computed along several orientations. The orientations which coincide with the directions of periodicity will present signatures like $\rho_y(y)$ of Figure 3.81 from

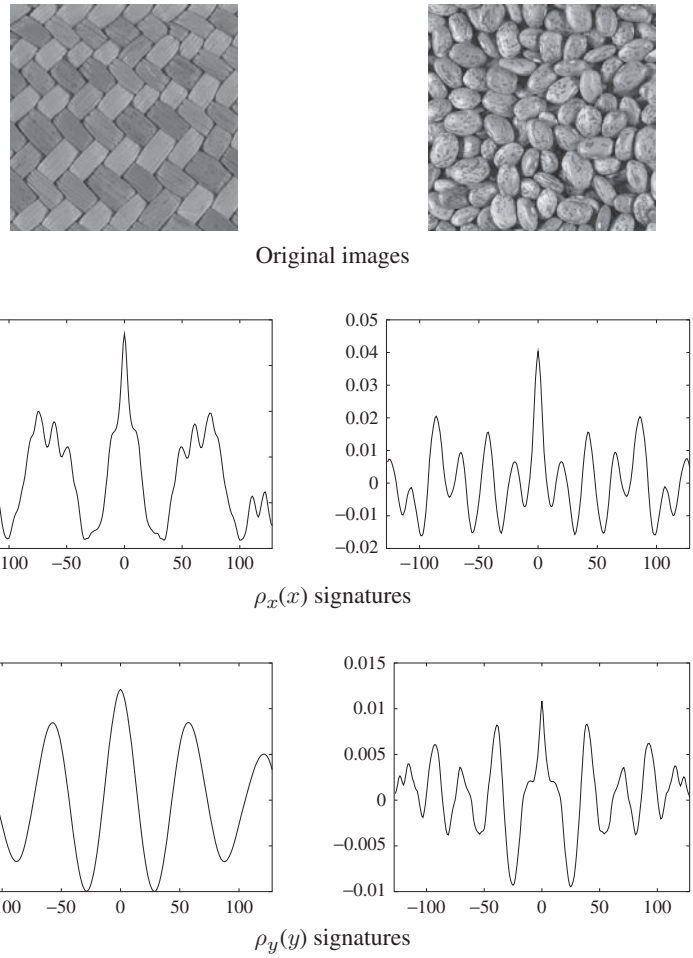


Figure 3.82 Marginals of the auto-covariance function for images fabric 1 and beans.

which one can easily infer the periodicity as being $256/19 \simeq 13$ pixels. Along all other orientations we shall be getting signatures like $\rho_x(x)$ of the same figure.

We can then see that to infer periodicity, we must compute the standard deviation of the distances of the successive peaks we identify: when there is periodicity this spread is expected to be very low, while when there is no periodicity the spread is expected to be high. The values of these standard deviations when **all** peaks are counted, i.e. without any curve smoothing taking place, are given in the sixth and seventh columns of Table 3.16. We note that the correlation with observed periodicity is not very good. This is because we have not taken into consideration the effective breadth of each peak: a standard deviation of the distances between peaks is only meaningful in terms of the mean distance between peaks, i.e. the mean size of the peaks. We may have an estimate of this "effective" peak size if we divide the size of the image, along each direction considered, with the number of peaks observed in the corresponding marginal. So, effective_peak_size=image_size/number_of_peaks. We must then divide each standard deviation with the corresponding effective peak size. As all images we use are of the same size, this is equivalent to multiplying the standard deviations with the corresponding number of observed

Table 3.16 Features derived from the profiles of the autocorrelation functions of the texture images of Figure 3.1.

Image	$\rho_x(0)$	$\rho_y(0)$	Peaks $\rho_x(x)$	Peaks $\rho_y(y)$	$\sigma_{\rho_x(x)}$	$\sigma_{\rho_y(y)}$	$\sigma_{N\rho_x(x)}$	$\sigma_{N\rho_y(y)}$	C
Cloth 1	0.042	0.043	13	9	10.1	16.6	131	149	N
Cloth 2	0.040	0.075	35	19	2.5	0.4	88	8	SP
Food	0.011	0.019	41	39	2.4	3.9	98	152	N
Plastic	0.070	0.048	9	11	12.3	10.1	111	111	N
Fabric 1	0.018	0.121	11	5	15.7	3.5	173	18	SP
Fabric 2	0.029	0.111	31	37	1.3	0.4	40	15	P
Beans	0.040	0.011	13	11	1.3	10.5	17	116	SP
Sand	0.014	0.023	35	33	3.1	2.4	109	79	N

peaks. The result is given in columns 8 and 9 of Table 3.16, under the headings $\sigma_{N\rho_x(x)}$ and $\sigma_{N\rho_y(y)}$, with suffix N standing for "normalised". On the basis of these numbers, we may characterise each texture as periodic, if both numbers are low, semi-periodic, if only one number is low indicating periodicity along one direction only, and non-periodic, if both numbers are high. These classes are indicated by letters P, SP and N respectively, given in the last column of the table, under the heading C. We can see that these characterisations agree with the appearance of these textures.

A note of caution: one should be careful when one uses terms like "low" and "high". Such terms have only relative meaning, and it is in this spirit that they are used here.

How can we extract parametric features from image statistics?

By fitting various parametric models to them. For many-point statistics we may use statistical models, like the Markov random fields, where the parameters involved are known as **Markov parameters**, or Gibbs distributions, where the parameters involved are known as **local potentials**. For the one- or two-point statistics we may use functional models, e.g. first or second order polynomials. The way of computing the parameters of some such models is described in Boxes 3.18–3.20. Alternatively, more specialised models, like the Weibull distribution for the histogram or the **generalised Zipf distribution** for the rank-frequency histogram may be used. In all cases the texture features are the parameters of the model used.

Box 3.18 Least square error fitting of a second order surface in 2D

Let us assume that we have a function with known values ρ_i at points (x_i, y_i). We assume that we can fit the following function to these data:

$$\rho(x,y) = Ax^2 + By^2 + Cxy + Dx + Fy + G. \tag{3.234}$$

The total error of fitting this function to the data is:

$$E = \sum_i (\rho(x_i, y_i) - \rho_i)^2 = \sum_i (Ax_i^2 + By_i^2 + Cx_iy_i + Dx_i + Fy_i + G - \rho_i)^2. \tag{3.235}$$

We must choose parameters A, B, C, D, F and G so that this error is minimum. We take the partial derivative of this error with respect to each one of the parameters in turn, and set it to zero, to produce a system of linear equations for the unknown parameters:

$$\begin{aligned}
\frac{\partial E}{\partial A} &= 0 \Rightarrow 2\sum_i (Ax_i^2 + By_i^2 + Cx_iy_i + Dx_i + Fy_i + G - \rho_i)x_i^2 = 0\\
\frac{\partial E}{\partial B} &= 0 \Rightarrow 2\sum_i (Ax_i^2 + By_i^2 + Cx_iy_i + Dx_i + Fy_i + G - \rho_i)yi^2 = 0\\
\frac{\partial E}{\partial C} &= 0 \Rightarrow 2\sum_i (Ax_i^2 + By_i^2 + Cx_iy_i + Dx_i + Fy_i + G - \rho_i)x_iyi = 0\\
\frac{\partial E}{\partial D} &= 0 \Rightarrow 2\sum_i (Ax_i^2 + By_i^2 + Cx_iy_i + Dx_i + Fy_i + G - \rho_i)x_i = 0\\
\frac{\partial E}{\partial F} &= 0 \Rightarrow 2\sum_i (Ax_i^2 + By_i^2 + Cx_iy_i + Dx_i + Fy_i + G - \rho_i)y_i = 0\\
\frac{\partial E}{\partial G} &= 0 \Rightarrow 2\sum_i (Ax_i^2 + By_i^2 + Cx_iy_i + Dx_i + Fy_i + G - \rho_i).
\end{aligned} \tag{3.236}$$

So, the system of linear equations we have to solve is:

$$\begin{aligned}
A\sum_i x_i^4 + B\sum_i y_i^2x_i^2 + C\sum_i x_i^3y_i + D\sum_i x_i^3 + F\sum_i y_ix_i^2 + G\sum_i x_i^2 &= \sum_i \rho_i x_i^2\\
A\sum_i x_i^2y_i^2 + B\sum_i y_i^4 + C\sum_i x_iy_i^3 + D\sum_i x_iy_i^2 + F\sum_i y_i^3 + G\sum_i y_i^2 &= \sum_i \rho_i y_i^2\\
A\sum_i x_i^3y_i + B\sum_i x_iy_i^3 + C\sum_i x_i^2y_i^2 + D\sum_i x_i^2y_i + F\sum_i x_iy_i^2 + G\sum_i x_iy_i &= \sum_i \rho_i x_iy_i\\
A\sum_i x_i^3 + B\sum_i x_iy_i^2 + C\sum_i x_i^2y_i + D\sum_i x_i^2 + F\sum_i x_iy_i + G\sum_i x_i &= \sum_i \rho_i x_i\\
A\sum_i x_i^2y_i + B\sum_i y_i^3 + C\sum_i x_iy_i^2 + D\sum_i x_iy_i + F\sum_i y_i^2 + G\sum_i y_i &= \sum_i \rho_i y_i\\
A\sum_i x_i^2 + B\sum_i y_i^2 + C\sum_i x_iy_i + D\sum_i x_i + F\sum_i y_i + G\sum_i 1 &= \sum_i \rho_i.
\end{aligned}$$

This is a system of six linear equations for six unknowns and it may be solved to yield the values of A, B, C, D, F and G.

Box 3.19 Least square error fitting of a second order curve in 1D

Let us assume that we have a 1D function with values ρ_i at points x_i. Let us also assume that we wish to fit these data points with a parabola, i.e. with a curve with equation:

$$\rho(x) = Hx^2 + Jx + K. \tag{3.237}$$

(Continued)

Box 3.19 (Continued)

We must select parameters H, J and K so that the fitting square error is minimum:

$$E = \sum_i (\rho(x_i) - \rho_i)^2 = \sum_i (Hx_i^2 + Jx_i + K - \rho_i)^2. \tag{3.238}$$

By setting the first partial derivatives of E with respect to the unknowns equal to zero, we obtain the following system of linear equations:

$$\begin{aligned}
\frac{\partial E}{\partial H} = 0 &\Rightarrow 2\sum_i (Hx_i^2 + Jx_i + K - \rho_i)x_i^2 = 0 \\
\frac{\partial E}{\partial J} = 0 &\Rightarrow 2\sum_i (Hx_i^2 + Jx_i + K - \rho_i)x_i = 0 \\
\frac{\partial E}{\partial K} = 0 &\Rightarrow 2\sum_i (Hx_i^2 + Jx_i + K - \rho_i) = 0.
\end{aligned} \tag{3.239}$$

So, the system of linear equations we have to solve is:

$$\begin{aligned}
H\sum_i x_i^4 + J\sum_i x_i^3 + K\sum_i x_i^2 &= \sum_i \rho_i x_i^2 \\
H\sum_i x_i^3 + J\sum_i x_i^2 + K\sum_i x_i &= \sum_i \rho_i x_i \\
H\sum_i x_i^2 + J\sum_i x_i + K\sum_i 1 &= \sum_i \rho_i.
\end{aligned} \tag{3.240}$$

This is a system of three linear equations for three unknowns that may be solved easily to yield the values of parameters H, J and K.

Box 3.20 Least square error fitting of an exponential function to a set of data

Assume that we have a set of points (d_i, ρ_i) and we wish to fit them with a function of the form:

$$\rho(d) = Me^{-Sd} \Rightarrow \ln \rho(d) = \ln M - Sd \tag{3.241}$$

Our problem is to estimate the values of M and S so the curve fits the data in the least square error sense.

Taking the logarithm of the function allows us to deal with a linear fitting problem instead of a non-linear one. Let us call for simplicity $L \equiv \ln M$. The total fitting error then is:

$$E \equiv \sum_i (L - Sd_i - \ln \rho_i)^2. \tag{3.242}$$

Partial differentiation leads to:

$$\begin{aligned}
\frac{\partial E}{\partial L} = 0 &\Rightarrow 2\sum_i (L - Sd_i - \ln \rho_i) = 0 \\
\frac{\partial E}{\partial S} = 0 &\Rightarrow 2\sum_i (L - Sd_i - \ln \rho_i)d_i = 0.
\end{aligned} \tag{3.243}$$

Then L (and therefore M) and S may be estimated by solving the following system of linear equations:

$$\begin{aligned} L\sum_i 1 - S\sum_i d_i &= \sum_i \ln \rho_i \\ L\sum_i d_i - S\sum_i d_i^2 &= \sum_i d_i \ln \rho_i. \end{aligned} \tag{3.244}$$

What is the generalised Zipf distribution?

If $m(r)$ is the frequency with which a particular grey value is observed in the image and r is its rank, when these frequencies are arranged in decreasing order, then the generalised Zipf distribution is:

$$m(r) = \frac{C}{r^\theta} \Rightarrow \log m(r) = \log C - \theta \log r. \tag{3.245}$$

In other words, the generalised Zipf distribution fits a straight line in the rank-frequency plot. The parameters of this line, namely $\log C$ and θ may be used as the texture features. If $\theta = 1$, we have the **Zipf distribution**.

Example 3.82

Calculate the Zipf parameters for the image of Figure 3.71a.
All we have to do is to fit a straight line using least squares error (see Box 3.3) to the pairs of values given by the third and fifth column of Table 3.13. The equation of such a line is $y = 1.6001 - 0.6925x$. *So,* $C = 39.8199$ *and* $\theta = 0.6925$.

Box 3.21 What is the relationship of the Zipf distribution and fractals?

To answer this question first we manipulate a little the Zipf distribution given by equation (3.245):

$$m(r) = \frac{C}{r^\theta} \Rightarrow m(r)^{\frac{1}{\theta}} = \frac{C^{\frac{1}{\theta}}}{r}. \tag{3.246}$$

If we call $C^{1/\theta} \equiv A$ and $1/\theta \equiv s$, this expression becomes:

$$m(r)^s = \frac{A}{r} \Rightarrow r = \frac{A}{m(r)^s}. \tag{3.247}$$

We may express this equation as:

$$\text{Observation\_of\_decreasing\_importance} \propto \text{Frequency\_of\_occurrence}^{-s}. \tag{3.248}$$

Let us compare this with Equation (3.156), with the substitution (3.157), which may be written as:

$$\text{Power\_in\_spatial\_detail} \propto \text{Spatial\_frequency}^{-\beta} \tag{3.249}$$

We note that in both cases we have a power law that relates the structure of the data with the frequency with which this structure occurs. These are special cases of a much broader category of models, known as the **power law** models.

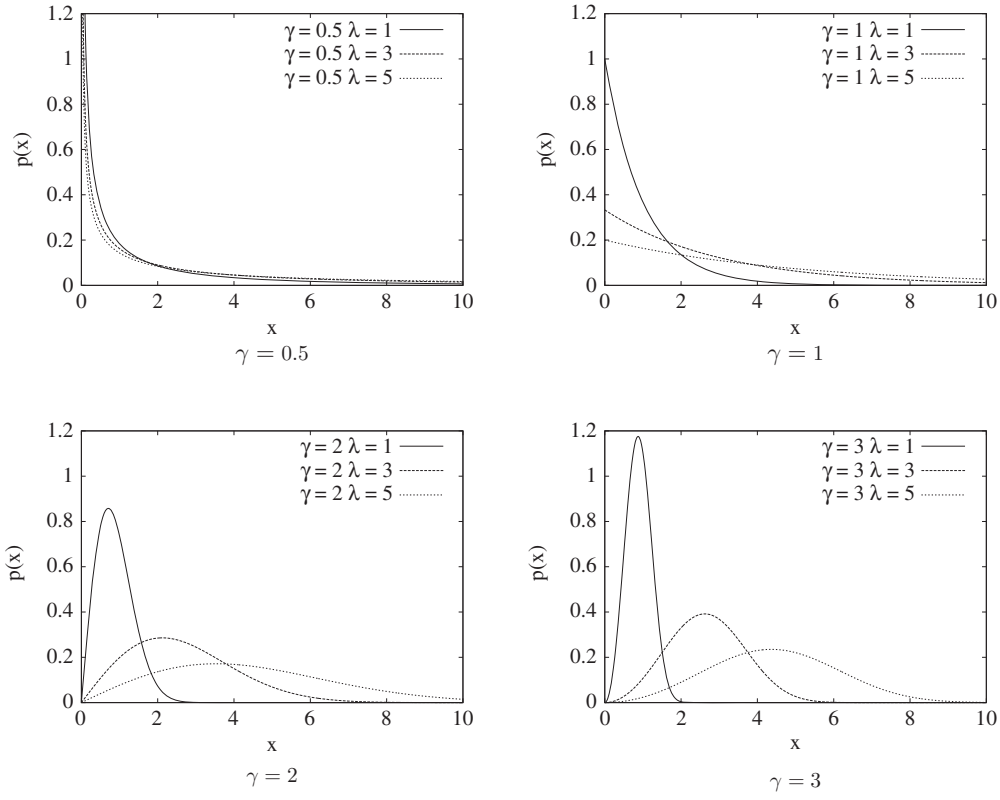


Figure 3.83 Plots of the Weibull distribution for several values of parameters γ and λ.

What is the Weibull distribution?

The Weibull distribution is defined for non-negative arguments only, as

$$f(x) \equiv \frac{\gamma}{\lambda}\left(\frac{x}{\lambda}\right)^{\gamma-1} \mathrm{e}^{-\left(\frac{x}{\lambda}\right)^{\gamma}} \tag{3.250}$$

where $\gamma > 0$ is called the **shape parameter** and $\lambda > 0$ is called the **scale parameter**. An alternative definition is:

$$p(x) \equiv \frac{1}{\beta}\left(\frac{x}{\beta}\right)^{\gamma-1} \mathrm{e}^{-\frac{1}{\gamma}\left(\frac{x}{\beta}\right)^{\gamma}}. \tag{3.251}$$

Some plots of the Weibull distribution for various values of its parameters are shown in Figure 3.83.

Example 3.83

Show that the two definitions of the Weibull distribution, given by (3.250) and (3.251), are equivalent.

By comparing the exponents of the two equations, we realise that $\gamma\beta^{\gamma} = \lambda^{\gamma}$. *We verify that the two definitions then are equivalent by substituting in (3.250)* λ *from* $\lambda = \gamma^{\frac{1}{\gamma}}\beta$.

Example 3.84

Show that the Weibull distribution integrates to 1, and so it may be considered as a probability density function.

$$\begin{aligned}\int_0^\infty f(x)\mathrm{d}x &= \frac{\gamma}{\lambda}\int_0^\infty \left(\frac{x}{\lambda}\right)^{\gamma-1} \mathrm{e}^{-\left(\frac{x}{\lambda}\right)^\gamma}\mathrm{d}x \\ &= \int_0^\infty \mathrm{d}\left(-\mathrm{e}^{-\left(\frac{x}{\lambda}\right)^\gamma}\right) \\ &= -\mathrm{e}^{-\left(\frac{x}{\lambda}\right)^\gamma}\Big|_0^\infty = 1. \end{aligned} \tag{3.252}$$

Example 3.85

Assume that the Weibull distribution is a probability density function, according to which N objects, characterised by their x value, are distributed in bins of width Δx. Calculate the number of expected objects in a bin that accumulates objects with value in the range $[x, x+\Delta x)$.

Let us calculate first the cumulative distribution of $f(x)$.

$$\begin{aligned}F(\alpha) \equiv \int_0^\alpha f(x)\mathrm{d}x &= \frac{\gamma}{\lambda}\int_0^\alpha \left(\frac{x}{\lambda}\right)^{\gamma-1} \mathrm{e}^{-\left(\frac{x}{\lambda}\right)^\gamma}\mathrm{d}x \\ &= \int_0^\alpha \mathrm{d}\left(-\mathrm{e}^{-\left(\frac{x}{\lambda}\right)^\gamma}\right) \\ &= -\mathrm{e}^{-\left(\frac{x}{\lambda}\right)^\gamma}\Big|_0^\alpha = 1-\mathrm{e}^{-\left(\frac{\alpha}{\lambda}\right)^\gamma}. \end{aligned} \tag{3.253}$$

Then the probability assigned to a bin with values in the range $[x, x+\Delta x)$ will be given by the difference of the two values of the cumulative distribution:

$$p_{\mathrm{bin}}(x) = F(x+\Delta x) - F(x) = 1 - \mathrm{e}^{-\left(\frac{x+\Delta x}{\lambda}\right)^\gamma} - \left(1-\mathrm{e}^{-\left(\frac{x}{\lambda}\right)^\gamma}\right) = \mathrm{e}^{-\left(\frac{x}{\lambda}\right)^\gamma} - \mathrm{e}^{-\left(\frac{x+\Delta x}{\lambda}\right)^\gamma}. \tag{3.254}$$

We have to multiply this with N to work out the number of objects we expect to find in this bin.

How can we estimate the parameters of the Weibull distribution from a set of data?

There are two ways.

(i) Assume that the value of the data histogram in bin i is $H(i)$ and that bin i contains objects with values in the range $[x, x+\Delta x)$. The Weibull distribution predicts, for the same bin, value $W(i)$, worked out as $Np_{\mathrm{bin}}(x)$ (see Example 3.85), where N is the total number of binned

objects and $p_{\rm bin}(x)$ is computed from (3.254). Then we may select values for γ and λ so that $C \equiv (H(i) - W(i))^2$ is minimised. The minimisation of this cost function is not trivial, as its derivatives with respect to the unknowns are not first order polynomials in the unknowns. The minimisation may be done by a method like simulated annealing (see the book Image Processing, The Fundamentals for such an application of simulated annealing [75]).

(ii) We may use the so-called **Weibull plot** method. Consider the cumulative distribution given by (3.253) and re-arrange it as follows:

$$\begin{aligned} F(\alpha) &= 1 - e^{-\left(\frac{\alpha}{\lambda}\right)^{\gamma}} \\ \Leftrightarrow \mathrm{e}^{-\left(\frac{\alpha}{\lambda}\right)^{\gamma}} &= 1 - F(\alpha) \\ \Leftrightarrow -\left(\frac{\alpha}{\lambda}\right)^{\gamma} &= \ln\left[1 - F(\alpha)\right] \\ \Leftrightarrow \left(\frac{\alpha}{\lambda}\right)^{\gamma} &= -\ln\left[1 - F(\alpha)\right] \\ \Leftrightarrow \gamma \ln\alpha - \gamma\ln\lambda &= \ln\left(-\ln\left[1 - F(\alpha)\right]\right). \end{aligned} \tag{3.255}$$

The algorithm then is as follows.

Step 1: Construct the histogram you wish to model, and normalise it by dividing all its values with the total number of pixels you used and with the bin width. Call it $H(i)$, where $i = 1, \ldots, M$, with M being the total number of bins.

Step 2: Construct the cumulative histogram as $F(j) = \sum_{i=1}^{j} H(i)$.

Step 3: To every j value corresponds an α_j value, which is the **upper** limit of the range of values accumulated in bin j.

Step 4: Use least square error fitting (see Box 3.3) with pairs $\left(\ln\alpha_j, \ln\left(-\ln\left[1 - F(j)\right]\right)\right)$.

Step 5: The slope of the fitted line is the value of γ.

Step 6: The value of the intercept of the line is $B = -\gamma\ln\lambda$. From the knowledge of the intercept and the value of γ, work out the value of λ as $\lambda = \mathrm{e}^{-\frac{B}{\gamma}}$.

It has been shown theoretically, that the γ parameter of the Weibull model of the grey level histogram of a fractal image is related to the fractal dimension D by

$$D = \gamma + 1. \tag{3.256}$$

Example 3.86

Fit the histograms you computed in Example 3.67 with the Weibull distribution. Comment on the applicability of the model.

Figure 3.84 shows the results of the fittings using gradient histograms.

The Weibull distribution is appropriate for modelling the statistics of random textures. It is therefore inappropriate for modelling histograms that correspond to deterministic textures or are bimodal (i.e. they have two or more peaks), like the one shown in Figure 3.64 of the bean image.

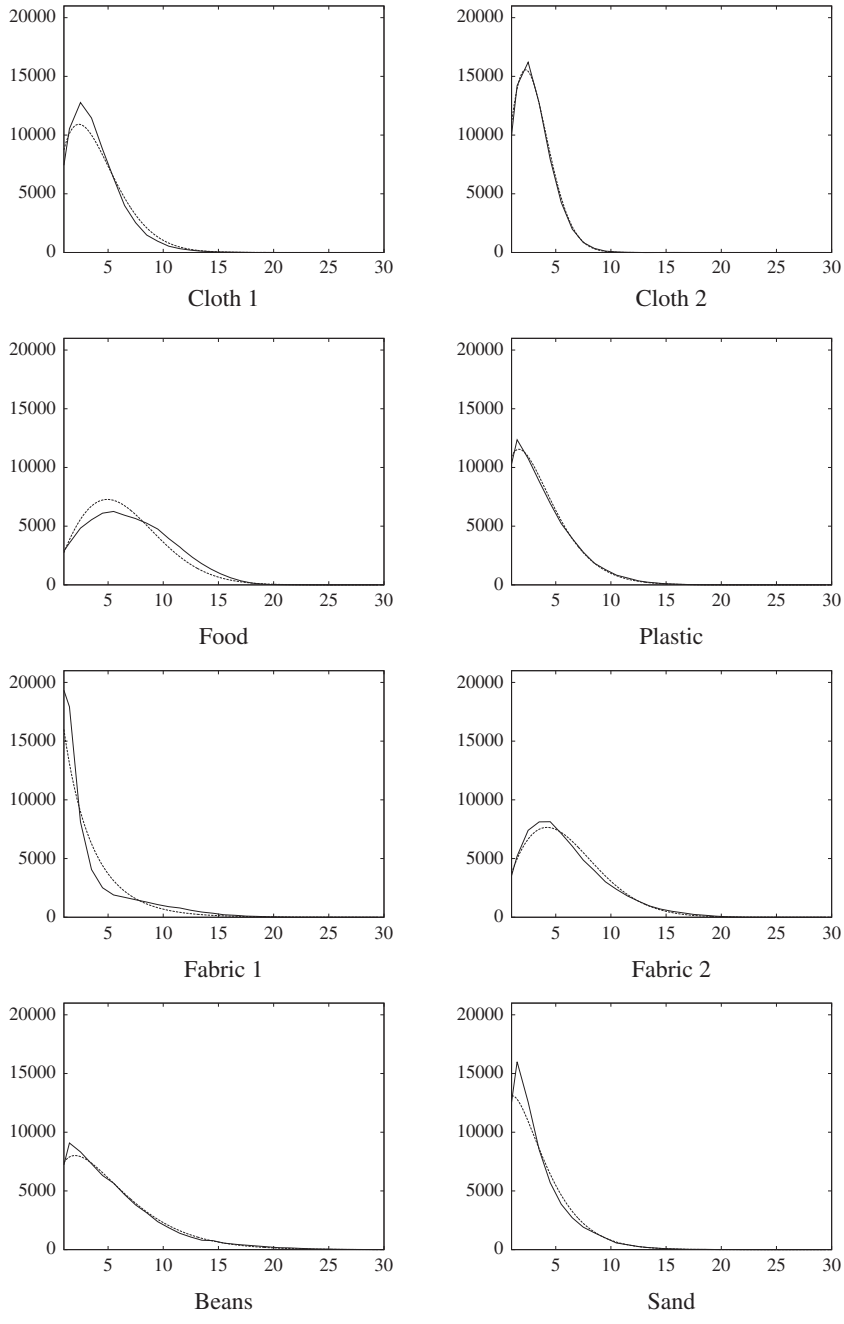


Figure 3.84 Fitting the histograms of the gradient magnitude for the textures of Figure 3.1 with the Weibull distribution.

Which parametric models are commonly used to fit the variogram over its range?

The following models have been used in various applications. **The exponential model**

$$\gamma(d) \equiv \begin{cases} C_0 & \text{for} \quad d = 0 \\ C_0 + C_1 \left(1 - \exp\left(-\frac{|d|}{a}\right)\right) & \text{for} \quad 0 < d < \text{range} \\ C_S & \text{for} \quad d \geq \text{range} \end{cases} \tag{3.257}$$

The Gaussian model

$$\gamma(d) \equiv \begin{cases} C_0 & \text{for} \quad d = 0 \\ C_0 + C_1 \left(1 - \exp\left(-\frac{d^2}{a}\right)\right) & \text{for} \quad 0 < d < \text{range} \\ C_S & \text{for} \quad d \geq \text{range} \end{cases} \tag{3.258}$$

The spherical model

$$\gamma(d) \equiv \begin{cases} C_0 & \text{for} \quad d = 0 \\ C_0 + C_1 \left(\frac{3}{2}\frac{d}{a} - \frac{1}{2}\left(\frac{d}{a}\right)^3\right) & \text{for} \quad 0 < d < \text{range} \\ C_S & \text{for} \quad d \geq \text{range} \end{cases} \tag{3.259}$$

The linear model

$$\gamma(d) \equiv \begin{cases} C_0 & \text{for} \quad d = 0 \\ C_0 + C_1 \frac{d}{a} & \text{for} \quad 0 < d < \text{range} \\ C_S & \text{for} \quad d \geq \text{range} \end{cases} \tag{3.260}$$

The fractal model

$$\gamma(d) = \begin{cases} C_0 & \text{for} \quad d = 0 \\ C_0 + C_1 d^{2H} & \text{for} \quad 0 < d \leq \text{range} \\ C_S & \text{for} \quad d \geq \text{range} \end{cases} \tag{3.261}$$

In these formulae, a, C_1 and H are the model parameters. Obviously, in each case, the sill value C_S is equal to the value of the middle branch of the equation for $d = \text{range}$.

Example 3.87

Fit the fractal model to the variograms of the images of Figure 3.1 knowing the range of values over which the model fits.

We computed the range for each one of these images in Example 3.78. So, we shall fit the fractal model for $0 < d \leq \text{range}$.

Let us call the variogram computed from the data $\gamma(d)$ *and the model we wish to fit* $\tilde{\gamma}(d) = C^2 d^{2H}$. *Let us take the logarithm of both sides of the model:*

$$\log \tilde{\gamma}(d) = 2 \log C + 2H \log d \equiv \gamma_0 + 2H \log d. \tag{3.262}$$

Here we defined $\gamma_0 \equiv 2\log C$ for the sake of simplicity. We wish to fit the model to the data in the least square error sense. We follow the algorithm in Box 3.3 with

$$x_i \leftrightarrow \log d_i$$
$$y_i \leftrightarrow \log \gamma(d_i)$$
$$A \leftrightarrow 2H$$
$$B \leftrightarrow \gamma_0$$

where d_i are the values of distance d for which we know the value of the variogram.

So, the algorithm that fits the fractal model to the data is:

Step 1: *Compute the variogram $\gamma(d_i)$ of the data, using the algorithm in the section How do we compute the variogram of an image in practice?, for values d_i of d, for $i = 1, 2, \ldots, K$, where K is the total number of d values we use that are less than or equal to the range.*
Step 2: *Compute $x_i \equiv \log d_i$ and $y_i \equiv \log \gamma(d_i)$.*
Step 3: *Compute: $S \equiv \sum_i x_i$, $T \equiv \sum_i x_i^2$, $E \equiv \sum_i y_i$ and $F \equiv \sum_i x_i y_i$.*
Step 4: *Compute $G \equiv KT - S^2$*
Step 5: *Compute $\gamma_0 = (ET - FS)/G$ and $H = (KF - SE)/(2G)$.*

Table 3.17 shows the results for all the images.

Table 3.17 Fitting the fractal model to the variograms of the texture images in Figure 3.1 for $0 < d \leq$ range.

Image	Range	γ_0	H
Cloth 1	20	2.5681	0.2435
Cloth 2	25	2.3542	0.1883
Food	7	3.0004	0.1896
Plastic	16	2.5079	0.2940
Fabric 1	19	2.5885	0.1395
Fabric 2	10	2.7690	0.2328
Beans	12	2.8113	0.3179
Sand	4	2.3205	0.3850

How can we fit automatically a model to a variogram?

This algorithm tries to find automatically the range of shifts over which the model we have selected fits the data with a goodness of fit better than a certain threshold. As a measure of goodness of fit it uses the correlation coefficient.

Step 0: Select a threshold r_0, which is in the range $[0.5, 1)$.
Step 1: Fit the model to the variogram data for the full range of d values.
Step 2: Set $k_{\max} = K$, where K is the total number of distances d_k used to construct the variogram.

Step 3: Compute:

$$A \equiv \sum_{k=1}^{k_{\max}} \log d_k \log \gamma(d_k)$$
$$B \equiv \sum_{k=1}^{k_{\max}} \log d_k$$
$$C \equiv \sum_{k=1}^{k_{\max}} \log \gamma(d_k)$$
$$D \equiv \sum_{k=1}^{k_{\max}} [\log d_k]^2$$
$$E \equiv \sum_{k=1}^{k_{\max}} [\log \gamma(d_k)]^2. \tag{3.263}$$

Step 4: Compute the correlation coefficient of the fit, defined as:

$$r \equiv \frac{k_{\max} A - BC}{\sqrt{(k_{\max} D - B^2)(k_{\max} E - C^2)}}. \tag{3.264}$$

If the model fits well, r tends to 1 and if the model does not fit the data well, r tends to 0.

Step 5: If $r \geq r_0$, we accept that the model fits the data well and go to step 7.

Step 6: If $r < r_0$, the model does not fit the data well. We set $k_{\max} = k_{\max}/2$ and go to step 3.

Step 7: If we reach this point with $k_{\max} = K$, we exit the algorithm accepting that the model fits the data well for the full range of variogram shifts.

Step 8: If $k_{\max} \neq K$, in the previous iteration r was smaller than r_0, but now r is above r_0. We wish to find the maximum possible value of index k for which the model fits the data, i.e. for which $r \geq r_0$. We set:

$$k_T \equiv 2k_{\max} \qquad k_{\max} = k_{\max} + 1. \tag{3.265}$$

Step 9: We fit the model to the data in the range $[1, k_{\max}]$ and compute r.

Step 10: If $r < r_0$, we fit the model to the data in the range $[1, k_{\max} - 1]$ and exit the algorithm.

Step 11: If $r \geq r_0$, we set $k_{\max} = k_{\max} + 1$.

Step 12: If $k_{\max} < k_T$, we go to step 9.

Step 13: If $k_{\max} \geq k_T$, we fit the model to the data in the range $[1, k_{\max} - 1]$ and exit the algorithm.

This algorithm may be used for any model we select for the variogram.

Example 3.88

Apply the above algorithm to the images of Figure 3.1 for the fractal model and check whether the algorithm will fit the same model as the one identified in Example 3.87.

The results are shown in Table 3.18. We must compare these results with those of Table 3.17. Given that the range of applying the model was selected by using bins of the values of the variogram, something that by itself is a little gross, the first thing to notice is that the agreement between the range computed in Example 3.78, and reproduced in Table 3.17, agrees remarkably well with the upper bound of the interval over which the model is fitted. In addition, the values of γ_0 agree to

within one or two decimal places. The agreement in the value of H (and by implication in the value of the fractal dimension that is 3 − H) is not as good, but still within 10–20% accuracy, apart from the two fabric textures that are nearly deterministic patterns. This should not surprise us, as H is really an exponent and exponential models are very sensitive and non-linear to perturbations.

Table 3.18 Best fitting of the fractal model to the variograms of the texture images of Figure 3.1 using $r_0 = 0.95$.

Image	Interval fitted	γ_0	H
Cloth 1	[1, 28]	2.6165	0.2076
Cloth 2	[1, 14]	2.3467	0.1956
Food	[1, 5]	2.9780	0.2317
Plastic	[1, 20]	2.5428	0.2640
Fabric 1	[1, 5]	2.5108	0.2485
Fabric 2	[1, 5]	2.6763	0.3850
Beans	[1, 19]	2.8774	0.2557
Sand	[1, 6]	2.3545	0.3082

Can we use non-parametric descriptions of texture?

Yes. For example, we may use the various moments of the various histograms we construct. Alternatively, we may use features constructed from the co-occurrence matrices and the run length matrices.

What is a histogram moment?

The q moment of a histogram with bins $H(i)$ is defined as

$$M_q \equiv \sum_i i^q h(i) \tag{3.266}$$

where $h(i)$ is the normalised value of bin i, obtained by dividing $H(i)$ by the bin width and the sum of all bin values.

What is the co-occurrence matrix?

It is a discrete function representing the probability of finding a pair of pixels in a certain relative position having a certain pair of grey values. It is a matrix, because it is defined over pairs of discrete grey values.

What are the generalised co-occurrence matrices?

These are matrices that express the co-occurrence of characteristics other than the grey values of pixels. They may be created by considering pixel attributes in addition to grey values. Successful algorithms have been produced by considering co-occurrence matrices that measure the frequency of occurrence of pairs of pixels, in a particular relative position from each other, having a particular pair of grey values, a particular pair of magnitudes of the local gradient and a particular relative orientation of their gradient vectors. Such co-occurrence matrices also capture local shape information of the image function. One, however, should be careful in dealing with such high-dimensionality matrices: the resolution with which these quantities are measured should be reduced to produce matrices of a manageable size. For example, if each of the above-mentioned quantities is quantised using 10 discrete values only, the co-occurrence matrix that results will have $10 \times 10 \times 10 \times 10 \times 10 = 10^5$ elements. Relative abundances of some of these elements have been shown to make good texture features.

What is a high order co-occurrence matrix?

A high order co-occurrence matrix may be defined by considering n-tuples as opposed to just pairs of pixels. It may be constructed by considering other pixel characteristics than their grey values. When considering n-tuples of pixels, the co-occurrence matrix corresponds to a higher-order joint probability density function.

How is a co-occurrence matrix defined?

There are two ways in which we may define a co-occurrence matrix. We may define it to be directional and we may define it to be rotationally invariant. For the construction of a rotationally invariant co-occurrence matrix, we consider all pairs of pixels that are at a fixed distance d from each other, irrespective of the relative orientation the line that joins them forms with the reference direction of the image. Such matrices are parametrised by the distance d only. We may have as many matrices as different distances we choose to use. So, we may write

$$C(k,l;d) \equiv \sum_i \sum_j \sum_{\hat{n}} \delta\left(k - g(i,j)\right) \delta\left(l - g((i,j) + d\hat{n})\right) \tag{3.267}$$

where $\hat{n}$ is the unit vector pointing in a chosen direction, $g(i,j)$ is the grey value of pixel (i,j), $g((i,j) + d\hat{n})$ is the grey value of another pixel that is at distance d from pixel (i,j) and at the orientation defined by unit vector $\hat{n}$, and $C(k,l;d)$ is the total number of pairs of pixels at distance d from each other identified in the image, such that the first one has grey value k and the second has grey value l. In the above expression, $\delta(a-b)$ takes value 1 if $a = b$ and it takes value 0 if $a \neq b$.

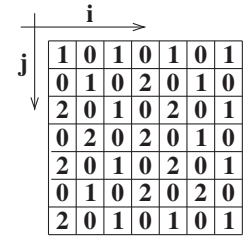
Alternatively, we may define the co-occurrence matrix so that it captures directional information. In this case we consider only pairs of pixels that are at a certain distance apart, and at the same time the line that connects them forms a certain angle with the reference direction of the image. These matrices are parametrised by two parameters, the distance d and the angle ϕ with respect to the reference direction. Again, we may choose to have as many such matrices as different values of d and ϕ we wish to consider. In this case, we may formally define the co-occurrence matrix as:

$$C(k,l;d,\phi) = \sum_i \sum_j \delta\left(k - g(i,j)\right) \delta\left(l - g(i + d\cos\phi, j + d\sin\phi)\right). \tag{3.268}$$

Example 3.89

From the image of Figure 3.85, compute the co-occurrence matrix for the following four relative positions: next door neighbours along the horizontal direction, next door neighbours along the vertical direction, two positions in the direction of the main diagonal once removed, and pairs of positions expressed as (i, j) and $(i + 2, j + 1)$.

Figure 3.85 An image with three grey levels.



As the image contains three grey levels only, the co-occurrence matrices will be 3×3*. To construct them we count the number of pairs of pixels we find at the given relative positions with a certain pair of grey values. If we divide each co-occurrence matrix with the number of pairs of pixels used to construct it, we may treat them as probability density functions. The results are shown in Figure 3.86.*

	0	1	2
0	0	14	7
1	11	0	0
2	10	0	0

	0	1	2
0	0	11	10
1	12	0	0
2	9	0	0

	0	1	2
0	12	0	0
1	0	3	3
2	0	5	2

	0	1	2
0	0	9	6
1	7	0	0
2	8	0	0

	0	1	2
0	0	.33	.17
1	.26	0	0
2	.24	0	0

(a)

	0	1	2
0	0	.26	.24
1	.29	0	0
2	.21	0	0

(b)

	0	1	2
0	.48	0	0
1	0	.12	.12
2	0	.20	.08

(c)

	0	1	2
0	0	.30	.20
1	.23	0	0
2	.27	0	0

(d)

Figure 3.86 Top row, co-occurrence matrices. Bottom row their normalised versions. The pairs of pixels used in each case are (a) next door neighbours along the horizontal direction, (b) next door neighbours along the vertical direction, (c) neighbours in the direction of the main diagonal leaving a gap in between, and (d) at positions expressed as (i, j) and $(i + 2, j + 1)$.

Example 3.90

Construct digital circles with radii $1, 2, \ldots, 8$ **pixels.**

A circle is defined as the locus of points at a fixed distance from the centre. Since the representation of each circle has to be made by pixels, it is not possible to apply the definition of the circle as we know it for continuous spaces. Instead, we must check the distance of each pixel from the central

(Continued)

Example 3.90 (Continued)

pixel, which is assumed to be at coordinate position $(0, 0)$*, and then decide whether the pixel belongs to the perimeter of the circle with radius* r *or not, according to whether this distance* d *is in the range* $[r - 0.5, r + 0.5)$ *or not. For example, a pixel at position* $(7, 2)$ *with respect to the centre is at distance* $\sqrt{7^2 + 2^2} = \sqrt{53} = 7.3$ *from it. This number rounds to 7, so this pixel belongs to a digital circle with radius 7. Figure 3.87 shows the digital circles with radii* $1, 2, \ldots, 8$*. Note that the larger the radius, the more the digital circle resembles a continuous circle. The centre of each circle is marked with an* $\times$ *and it is assumed to be at position* $(0, 0)$*. The coordinate directions are marked with the coordinate axes drawn.*

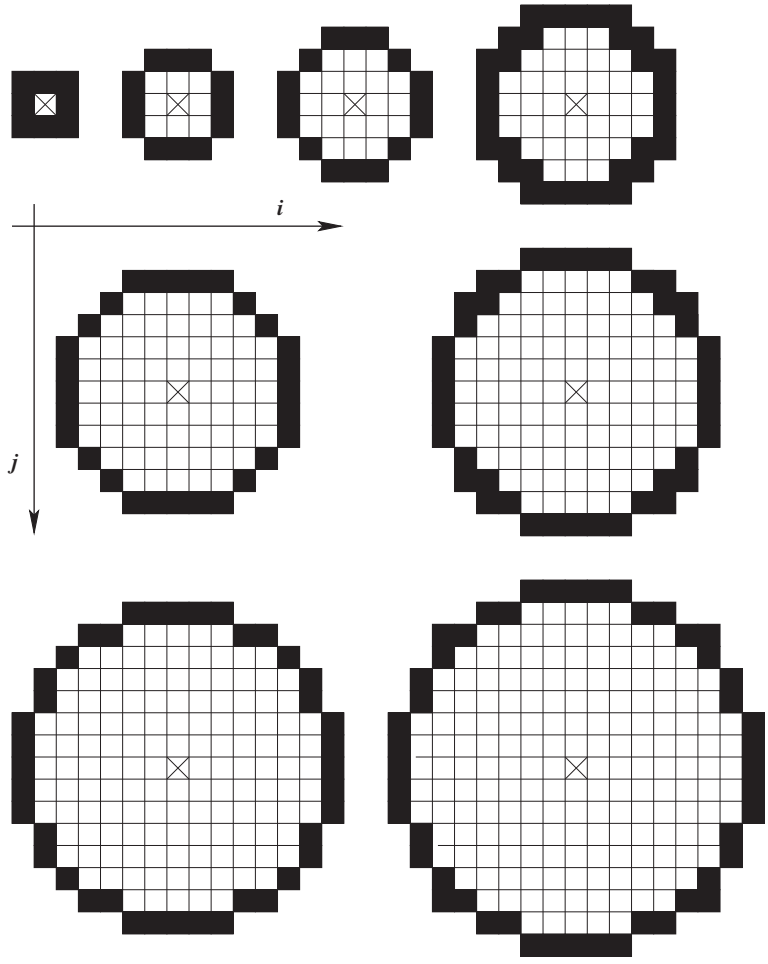


Figure 3.87 Digital circles with radii $1, 2, \ldots, 8$, from top to bottom and left to right, respectively. The centre of each circle is at position $(0, 0)$ and the directions of the coordinate axes used are also indicated.

Example 3.91

If the centre of a digital circle is at position $(0, 0)$, identify the coordinates of the pixels that constitute each circle with radius $1, 2, \ldots, 8$ pixels. Use the coordinate directions marked in Figure 3.87.

Table 3.19 lists the coordinates of the pixels that constitute each circle in Figure 3.87.

Table 3.19 The coordinates of the pixels that make up the digital circles in Figure 3.87. The centre of the coordinate system in each case is assumed to be at the centre of each circle.

$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = 7$	$r = 8$	
(−1,0)	(−2,0)	(−3,0)	(−4,0)	(−5,0)	(−6,0)	(−7,0)	(−8,0)	(−4,−7)
(−1,1)	(−2,1)	(−3,1)	(−4,1)	(−5,1)	(−6,1)	(−7,1)	(−8,1)	(−5,−6)
(0,1)	(−1,2)	(−2,2)	(−4,2)	(−5,2)	(−6,2)	(−7,2)	(−8,2)	(−6,−6)
(1,1)	(0,2)	(−1,3)	(−3,2)	(−4,3)	(−5,3)	(−6,3)	(−7,3)	(−6,−5)
(1,0)	(1,2)	(0,3)	(−3,3)	(−3,4)	(−5,4)	(−6,4)	(−7,4)	(−7,−4)
(1,−1)	(2,1)	(1,3)	(−2,3)	(−2,5)	(−4,4)	(−5,5)	(−6,5)	(−7,−3)
(0,−1)	(2,0)	(2,2)	(−2,4)	(−1,5)	(−4,5)	(−4,6)	(−6,6)	(−8,−2)
(−1,−1)	(2,−1)	(3,1)	(−1,4)	(0,5)	(−3,5)	(−3,6)	(−5,6)	(−8,−1)
	(1,−2)	(3,0)	(0,4)	(1,5)	(−2,6)	(−2,7)	(−4,7)	
	(0,−2)	(3,−1)	(1,4)	(2,5)	(−1,6)	(−1,7)	(−3,7)	
	(−1,−2)	(2,−2)	(2,4)	(3,4)	(0,6)	(0,7)	(−2,8)	
	(−2,−1)	(1,−3)	(2,3)	(4,3)	(1,6)	(1,7)	(−1,8)	
		(0,−3)	(3,3)	(5,2)	(2,6)	(2,7)	(0,8)	
		(−1,−3)	(3,2)	(5,1)	(3,5)	(3,6)	(1,8)	
		(−2,−2)	(4,2)	(5,0)	(4,5)	(4,6)	(2,8)	
		(−3,−1)	(4,1)	(5,−1)	(4,4)	(5,5)	(3,7)	
			(4,0)	(5,−2)	(5,4)	(6,4)	(4,7)	
			(4,−1)	(4,−3)	(5,3)	(6,3)	(5,6)	
			(4,−2)	(3,−4)	(6,2)	(7,2)	(6,6)	
			(3,−2)	(2,−5)	(6,1)	(7,1)	(6,5)	
			(3,−3)	(1,−5)	(6,0)	(7,0)	(7,4)	
			(2,−3)	(0,−5)	(6,−1)	(7,−1)	(7,3)	
			(2,−4)	(−1,−5)	(6,−2)	(7,−2)	(8,2)	
			(1,−4)	(−2,−5)	(5,−3)	(6,−3)	(8,1)	

(Continued)

Example 3.91 (Continued)

Table 3.19 (Continued)

$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$
			(0,−4)	(−3,−4)	(5,−4)	(6,−4)	(8,0)
			(−1,−4)	(−4,−3)	(4,−4)	(5,−5)	(8,−1)
			(−2,−4)	(−5,−2)	(4,−5)	(4,−6)	(8,−2)
			(−2,−3)	(−5,−1)	(3,−5)	(3,−6)	(7,−3)
			(−3,−3)		(2,−6)	(2,−7)	(7,−4)
			(−3,−2)		(1,−6)	(1,−7)	(6,−5)
			(−4,−2)		(0,−6)	(0,−7)	(6,−6)
			(−4,−1)		(−1,−6)	(−1,−7)	(5,−6)
					(−2,−6)	(−2,−7)	(4,−7)
					(−3,−5)	(−3,−6)	(3,−7)
					(−4,−5)	(−4,−6)	(2,−8)
					(−4,−4)	(−5,−5)	(1,−8)
					(−5,−4)	(−6,−4)	(0,−8)
					(−5,−3)	(−6,−3)	(−1,−8)
					(−6,−2)	(−7,−2)	(−2,−8)
					(−6,−1)	(−7,−1)	(−3,−7)

Example 3.92

Assume that an image contains one of the circles in Figure 3.87 and that it is scanned in a raster way, from top left to bottom right, following a scanning line like the one defined in Figure 2.29. This creates a sequence of the pixels. Identify the coordinates of the pixels that are on the perimeter of each circle and they come after the pixel in the centre of the circle.

Since the sequence we create scans the image line by line from left to right, after we meet the centre of the circle, the first black pixel we meet is the one on the right of the centre and in the same row as the centre. The next is on the left of the centre and one row below it, and so on. Table 3.20 lists the coordinates of all those pixels that come after the central pixel in the case of each circle.

Table 3.20 The coordinates of the pixels that will appear after the centre of the circle in a raster scan of an image, with the digital circles of Figure 3.87. The centre of the coordinate system in each case is assumed to be at the centre of each circle.

$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = 7$	$r = 8$
(−1,1)	(−2,1)	(−3,1)	(−4,1)	(−5,1)	(−6,1)	(−7,1)	(−8,1)
(0,1)	(−1,2)	(−2,2)	(−4,2)	(−5,2)	(−6,2)	(−7,2)	(−8,2)
(1,1)	(0,2)	(−1,3)	(−3,2)	(−4,3)	(−5,3)	(−6,3)	(−7,3)
(1,0)	(1,2)	(0,3)	(−3,3)	(−3,4)	(−5,4)	(−6,4)	(−7,4)
	(2,1)	(1,3)	(−2,3)	(−2,5)	(−4,4)	(−5,5)	(−6,5)
	(2,0)	(2,2)	(−2,4)	(−1,5)	(−4,5)	(−4,6)	(−6,6)
		(3,1)	(−1,4)	(0,5)	(−3,5)	(−3,6)	(−5,6)
		(3,0)	(0,4)	(1,5)	(−2,6)	(−2,7)	(−4,7)
			(1,4)	(2,5)	(−1,6)	(−1,7)	(−3,7)
			(2,4)	(3,4)	(0,6)	(0,7)	(−2,8)
			(2,3)	(4,3)	(1,6)	(1,7)	(−1,8)
			(3,3)	(5,2)	(2,6)	(2,7)	(0,8)
			(3,2)	(5,1)	(3,5)	(3,6)	(1,8)
			(4,2)	(5,0)	(4,5)	(4,6)	(2,8)
			(4,1)		(4,4)	(5,5)	(3,7)
			(4,0)		(5,4)	(6,4)	(4,7)
					(5,3)	(6,3)	(5,6)
					(6,2)	(7,2)	(6,6)
					(6,1)	(7,1)	(6,5)
					(6,0)	(7,0)	(7,4)
							(7,3)
							(8,2)
							(8,1)
							(8,0)

How do we compute the co-occurrence matrix in practice?

In practice we do not compute distances between pixels as a matter of routine, because this is very computationally expensive. For the rotationally invariant matrix we create lists of pixels that, given a starting pixel, all are within distance $d \pm 0.5$ from it. We then simply consider these lists as digital masks that are used to scan the image. For example, if we wish to create the co-occurrence matrix for $d = 2$, we consider each pixel of the image in turn. Let us say that the coordinates of the current pixel are (i_0, j_0). Then we consider all pairs this pixel forms with pixels at positions $(i_0 + k, j_0 + l)$ where (k, l) take all values under the heading $r = 2$ of Table 3.20. The purpose of considering only part of the full digital circle is for pairs of pixels to be considered only once.

So, to compute the rotationally invariant co-occurrence matrix $C(g_1, g_2; d)$ of an 8-bit image $g(i, j)$ for distance d, we apply the following algorithm.

Step 1: Create an array $C_d(g_1, g_2)$ of size $G \times G$, where G is the numbers of grey levels of the image, and initialise all its elements to 0.

Step 2: Scan the image once. At each position (i,j), update the following elements of the array

$$C_d\,(g(i,j), g(i+k, j+l)) = C_d\,(g(i,j), g(i+k, j+l)) + 1 \tag{3.269}$$

$$C_d\,(g(i+k, j+l), g(i,j)) = C_d\,(g(i+k, j+l), g(i,j)) + 1 \tag{3.270}$$

where, for d up to 8, (k,l) takes all pairs of values that appear in Table 3.20 under the heading $r = d$. For larger values of d one must create first the corresponding list of pixel coordinates.

Note that we update two elements of matrix $C_d(g_1, g_2)$ because rotational symmetry means that we do not distinguish between pairs of grey values (g_1, g_2) and (g_2, g_1). Alternatively, we may use only half of array $C_d(g_1, g_2)$, say the top-right triangular half, and increment only one element, namely element $(\min(g_1, g_2), \max(g_1, g_2))$.

For pixels that are near the left, right and bottom border of the image, we consider only those values of (k,l) for which positions $(i+k, j+l)$ are within the image boundaries.

If we wish to create a directional co-occurrence matrix, then we simply use a single pair of values (k,l) in Equation (3.269) which are the same for all pixels (i,j) in the image. We need an intermediate step for the algorithm, where for a given orientation ϕ and distance d, we compute the shift (k,l).

Step 0: Decide upon the distance d and the direction ϕ along which you wish to compute the co-occurrence matrix. Note that ϕ could be in the range $[0, 360°)$.

Step 1.5: Set:

$$k = \lfloor d\cos\phi + 0.5 \rfloor \qquad l = \lfloor d\sin\phi + 0.5 \rfloor. \tag{3.271}$$

The co-occurrence matrix we create this way is of size 256×256 for an 8-bit image. If we do not wish to have such a large matrix, we may reduce the number of separate grey levels in the image. For example, we may map the 256 grey levels to only G by using:

$$g(i,j)^{\text{new}} = \left\lfloor \frac{g(i,j)^{\text{old}}}{255} \times (G-1) + 0.5 \right\rfloor. \tag{3.272}$$

How can we recognise textures with the help of the co-occurrence matrix?

Co-occurrence matrices are very rich representations of an image. At the same time they are very bulky. One may use directly some of the elements of co-occurrence matrices to characterise a texture, particularly for the cases where some reduction in the number of grey values has already been applied. In particular, **ratios** of elements of the co-occurrence matrix have been shown to be good texture descriptors. This is because they capture the **relative** abundance of certain image characteristics. Which elements should be used to create characteristic relative abundances is a matter of training the algorithm, i.e. it is a matter of trial and error.

The classical approach, however, is to compute certain characteristics of the co-occurrence matrix. The co-occurrence matrix, after all, corresponds to a joint probability density function and one can characterise probability density functions by computing a few statistics from them. The most commonly used features computed from the co-occurrence matrix are listed below. In all cases, the co-occurrence matrix has been normalised by dividing all its elements with the total number of pairs of pixels considered, so that it really is a joint probability density function

$$p(m,n) = \frac{1}{\text{All\_pairs\_of\_pixels\_used}} C_d(m,n). \tag{3.273}$$

The features used then are defined as follows:

- Energy:

$$\sum_{m=0}^{G-1}\sum_{n=0}^{G-1} p(m,n)^2. \tag{3.274}$$

- Entropy:

$$-\sum_{m=0}^{G-1}\sum_{n=0}^{G-1} p(m,n)\ln p(m,n). \tag{3.275}$$

- Contrast:

$$\frac{1}{(G-1)^2}\sum_{m=0}^{G-1}\sum_{n=0}^{G-1}(m-n)^2 p(m,n). \tag{3.276}$$

- Correlation:

$$\frac{\sum_{m=0}^{G-1}\sum_{n=0}^{G-1} mnp(m,n) - \mu_x\mu_y}{\sigma_x\sigma_y}. \tag{3.277}$$

Here:

$$\mu_x = \sum_{m=0}^{G-1} m\sum_{n=0}^{G-1} p(m,n) \qquad \mu_y = \sum_{n=0}^{G-1} n\sum_{m=0}^{G-1} p(m,n) \tag{3.278}$$

$$\sigma_x^2 = \sum_{m=0}^{G-1}(m-\mu_x)^2\sum_{n=0}^{G-1} p(m,n) \qquad \sigma_y^2 = \sum_{n=0}^{G-1}(n-\mu_y)^2\sum_{m=0}^{G-1} p(m,n). \tag{3.279}$$

- Homogeneity:

$$\sum_{m=0}^{G-1}\sum_{n=0}^{G-1}\frac{p(m,n)}{1+|m-n|}. \tag{3.280}$$

In all these definitions, G is the total number of grey levels we use. For an 8-bit image $G = 256$.

Example 3.93

Compute the features from co-occurrence matrix 3.86c.

Energy:

$$\begin{aligned} E &= \sum_{m=0}^{2}\sum_{n=0}^{2} p(m,n)^2 \\ &= 0.48^2 + 0.12^2 + 0.12^2 + 0.2^2 + 0.08^2 = 0.3056. \end{aligned} \tag{3.281}$$

Entropy:

$$\begin{aligned} H &= -\sum_{m=0}^{2}\sum_{n=0}^{2} p(m,n)\ln p(m,n) \\ &= -0.48\ln 0.48 - 0.12\ln 0.12 - 0.12\ln 0.12 - 0.2\ln 0.2 - 0.08\ln 0.08 \\ &= 1.385. \end{aligned} \tag{3.282}$$

(Continued)

Example 3.93 (Continued)

Contrast:

$$C = \frac{1}{2^2}\sum_{m=0}^{2}\sum_{n=0}^{2}(m-n)^2 p(m,n)$$
$$= \frac{1}{4}\times\left[(1-2)^2\times 0.2 + (2-1)^2\times 0.12\right] = \frac{1}{4}\times 0.32 = 0.08. \tag{3.283}$$

Correlation:

$$\mu_x = \sum_{m=0}^{2} m\sum_{n=0}^{2} p(m,n) = 0\times 0.48 + 1\times 0.32 + 2\times 0.20 = 0.72$$
$$\mu_y = \sum_{n=0}^{2} n\sum_{m=0}^{2} p(m,n) = 0\times 0.48 + 1\times 0.24 + 2\times 0.28 = 0.80$$
$$\sigma_x^2 = \sum_{m=0}^{2}(m-0.72)^2\sum_{n=0}^{2} p(m,n)$$
$$= (0-0.72)^2\times 0.48 + (1-0.72)^2\times 0.32 + (2-0.72)^2\times 0.2 = 0.6016$$
$$\sigma_y^2 = \sum_{n=0}^{2}(n-0.8)^2\sum_{m=0}^{2} p(m,n)$$
$$= (0-0.8)^2\times 0.48 + (1-0.8)^2\times 0.24 + (2-0.8)^2\times 0.28 = 0.72$$
$$R = \frac{\sum_{m=0}^{2}\sum_{n=0}^{2} mnp(m,n) - \mu_x\mu_y}{\sigma_x\sigma_y}$$
$$= \frac{1\times 1\times 0.12 + 1\times 2\times 0.12 + 1\times 2\times 0.2 + 2\times 2\times 0.08 - 0.72\times 0.8}{\sqrt{0.6016}\times\sqrt{0.72}}$$
$$= \frac{0.504}{0.658} = 0.766. \tag{3.284}$$

Homogeneity:

$$U = \sum_{m=0}^{2}\sum_{n=0}^{2}\frac{p(m,n)}{1+|m-n|}$$
$$= \frac{0.48}{1+|0-0|} + \frac{0.12}{1+|1-1|} + \frac{0.12}{1+|1-2|} + \frac{0.2}{1+|2-1|} + \frac{0.08}{1+|2-2|} = 0.84.$$

How can we choose the parameters of the co-occurrence matrix?

Macro-textures are textures with large primitive patterns. Therefore, large values of d would be most appropriate for capturing the repetitiveness of the pattern. **Micro-textures**, on the contrary, are textures with very fine detail. Small values of d would be most appropriate. Rotationally invariant co-occurrence matrices are most appropriate for isotropic textures. For textures that show different behaviour along different directions, we should use several co-occurrence matrices along a few different angles and compute features from all of them.

Example 3.94

Compute the rotationally invariant co-occurrence matrices for image 3.1g for $d = 2$, $d = 6$ and $d = 16$ for 256 grey levels and for 16 grey levels. Present the matrices as grey images with their values scaled and rounded to the nearest integer, in the range $[0, 255]$**.**

The results are shown in Figure 3.88. These matrices are symmetric.

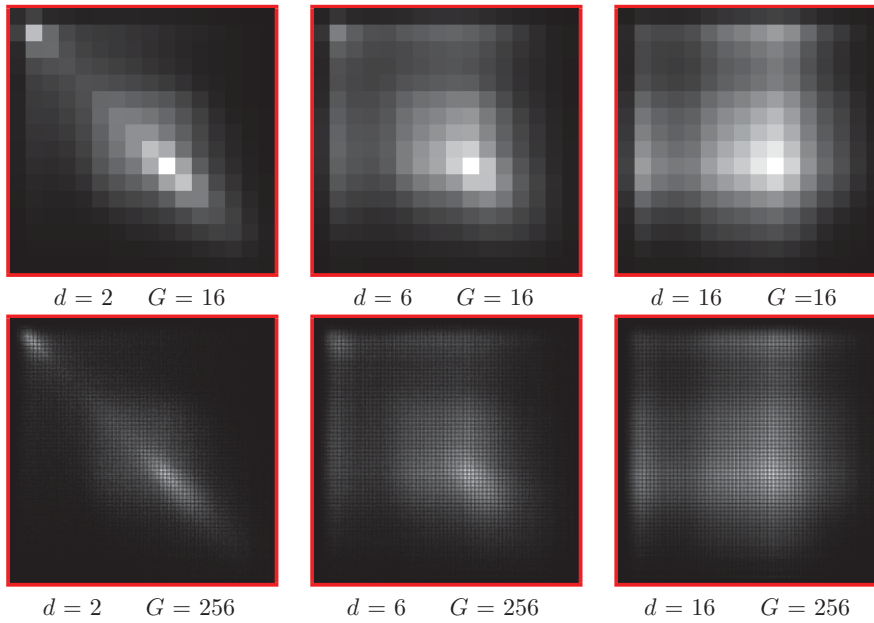


Figure 3.88 Rotationally invariant co-occurrence matrices for the image beans. Source: Maria Petrou.

Example 3.95

Compute the co-occurrence matrices for image 3.1e for $\phi = 45°$, $\phi = 135°$ and $\phi = 225°$, for 256 grey levels and for 16 grey levels. Present the matrices as grey images with their values scaled and rounded to the nearest integer, in the range $[0, 255]$**.**

The results are shown in Figure 3.89. To visualise better the detailed structure of the matrix, the grey values have been produced by appropriate scaling the values $\log(x + 1)$ *where x is the entry of the co-occurence matrix before normalisation by division with the sum of its elements.*

This is a rather anisotropic texture and one can see that the directional co-occurrence matrices constructed are different from each other.

(Continued)

Example 3.95 (Continued)

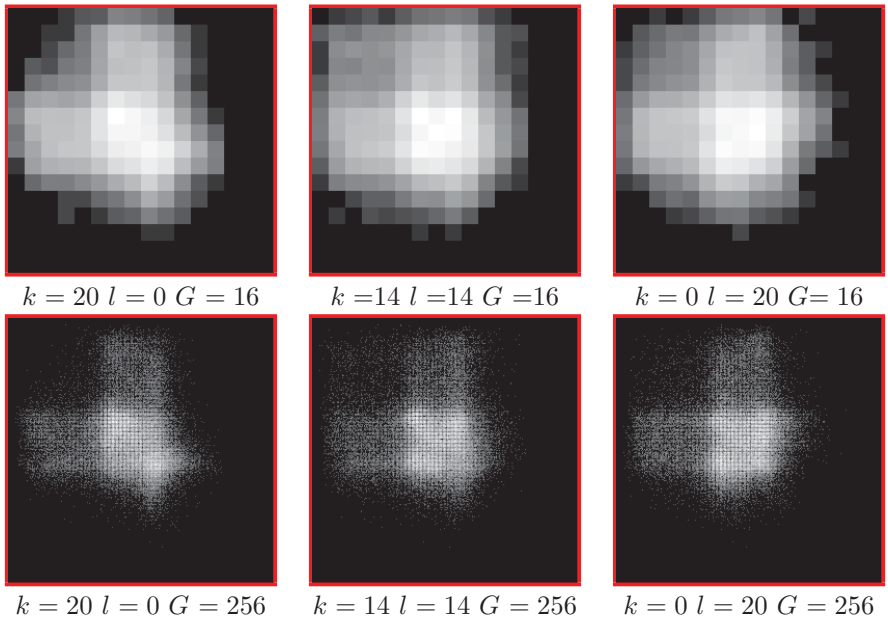


Figure 3.89 Directional co-occurrence matrices for the image fabric 1. Source: Maria Petrou.

Example 3.96

Use co-occurrence matrices to infer the illumination direction under which image 3.1c was captured. Tolerate an error of $\pm 22.5°$.

When a rough surface is illuminated from a direction other than the zenith direction, even if the surface is isotropic, the image created is anisotropic. Along the direction of illumination there will be more interchanges of dark and bright spots than along all other directions. Since we are allowed to work out the illumination direction with tolerance $\pm 22.5°$, we can consider the rotationally sensitive co-occurrence matrices constructed for pairs of pixels along the two main directions and the two main diagonals, and compute the contrast feature from them. The direction with the maximum value of the contrast feature is the direction of illumination. We use distance $d = 1$ for such a calculation, as we wish to capture sharp transitions from bright to dark caused by shadows created on the rough surface. The values of the contrast for this image are given in Table 3.21.

We note that the maximum contrast is for the direction along the secondary diagonal, i.e. from bottom left to top right. So, the lighting source when this image was captured was either towards the bottom left of the image or towards the top right, i.e. either at $225°$ from the horizontal image axis or at $45°$. To distinguish which of the two is correct, we must consider also the sign of the change when we compute the contrast feature from the co-occurrence matrix. In other words, in factor $(m - n)^2$ in formula 3.283 it matters whether the first or the second pixel of the pair had the

value m or n. So, we may compute again contrast features, but now using the top and the bottom triangle of the co-occurrence matrix separately.

From the upper triangle we shall have all the transitions from low to high value, as in an ordinary matrix representation, element C_{mn} belongs to the upper triangle for $n > m$. So, if m is the grey value of the first pixel of the pair and n the grey value of the second, element C_{mn} represents transitions from dark to bright. The new contrast values for the 8 directions are given in Table 3.22.

We conclude that the illumination was from the top right side of the image. This indeed is confirmed by inspecting the original image.

Table 3.21 Contrast value for the four main directions for the image in Figure 3.1c.

Horizontal	Main diagonal	Vertical	Secondary diagonal
0.0222	0.0313	0.0273	0.0407

Table 3.22 Contrast values for 8 different directions for the image in Figure 3.1c.

Direction	Contrast value
From right to left	0.0115
From bottom right to top left	0.0150
From bottom to top	0.0129
From bottom left to top right	0.0194
From left to right	0.0107
From top left to bottom right	0.0163
From top to bottom	0.0144
From top right to bottom left	0.0213

3.5 Texture Features from the Fourier Transform

How can we use the Fourier transform to construct texture features?

We can use the power spectrum to infer the structural characteristics of a texture. When using the power spectrum of an image we must have the following points in mind.

(i) If the image is assumed to be repeated ad infinitum in both directions, the power spectrum is the Fourier transform of the autocorrelation function of the image (**Wiener–Khinchine** theorem, see Example 3.109). So, just like the autocorrelation function, the power spectrum may be used to infer the periodicity of a texture.

(ii) The power spectrum of an image may easily be altered by interference that may not be obvious to the naked eye (see Examples 3.97 and 3.107).
(iii) The calculation of the DFT assumes that the image is repeated ad infinitum in all directions. This creates false peaks in the power spectrum (see Example 3.103). This may be mitigated by multiplying the signal or image with a mask that tampers its values around the borders, so the discontinuities due to the implicit repetition are reduced (see Examples 3.104 and 3.105).
(iv) The power spectrum looses a lot of important information concerning the structure of the image conveyed by the phase component (see Example 3.110).

How is the power spectrum defined?

The power spectrum is defined as the square magnitude of the Fourier transform

$$P(u,v) = F_{\mathrm{R}}(u,v)^2 + F_{\mathrm{I}}(u,v)^2 \tag{3.285}$$

where $F_{\mathrm{R}}(u,v)$ and $F_{\mathrm{I}}(u,v)$ are the real and the imaginary parts of the Fourier transform of the image respectively, and u and v are the frequencies along the x and y axes of the image, respectively. If it is divided with the number of pixels in the image, it is called a **periodogram**.

In practice, since we are interested in characterising textures, we ignore the peak at frequency $(0,0)$, as this corresponds to the overall average brightness of the image.

Figure 3.90 shows the power spectra for four of the textures in Figure 3.1 without the value at $(0,0)$. These images were created from the logarithmic values of the spectra computed according to Equation 3.293 and scaled according to Equation 3.294.

How can we infer the periodicity of a texture from its power spectrum?

The peaks of the power spectrum indicate basic repetition frequencies in the image. As the power spectrum is a 2D function, to work out periodicities displayed along a particular axis we have to sum up the values of the power spectrum along directions orthogonal to the axis. The obvious directions we may consider are the image axes, and so we may define $P_u(u) \equiv \sum_v P(u,v)$ and $P_v(v) \equiv$

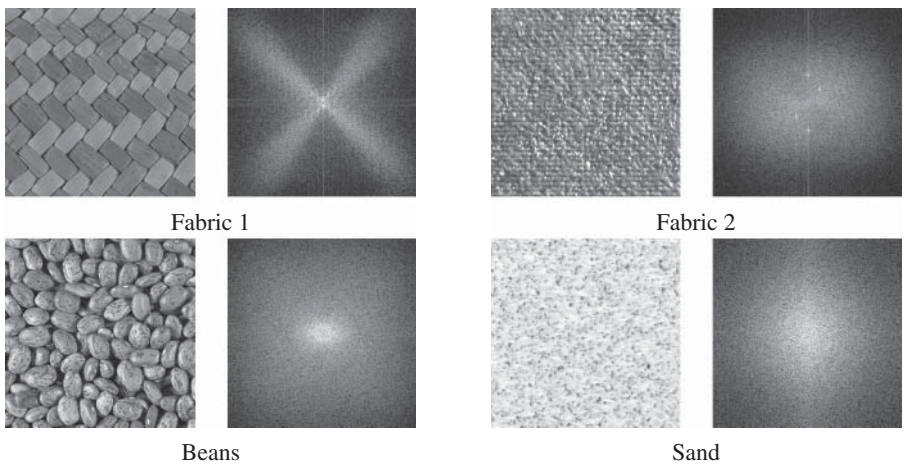


Figure 3.90 Power spectra for four of the textures of Figure 3.1. The spectra images were created with the help of Equations (3.293) and (3.294), after the direct component was omitted.

Table 3.23 The five strongest peaks in the signatures of the spectra of the images fabric 1 and fabric 2.

	Fabric 1				Fabric 2			
Order	**Position**	**Value**	***D***	**ΔB**	**Position**	**Value**	***D***	**ΔB**
1	$P_u(4)$	11699.5	9003.7	9	$P_v(16)$	15012.8	11834.0	4
2	$P_v(7)$	4635.4	3356.2	6	$P_u(38)$	14601.6	13254.6	10
3	$P_v(15)$	2717.8	1585.0	8	$P_u(1)$	11176.9	4341.7	2
4	$P_u(10)$	2535.9	379.6	2	$P_u(18)$	10805.5	8057.5	10
5	$P_u(12)$	2125.5	145.4	2	$P_v(32)$	6987.2	5201.9	8

Table 3.24 The five strongest peaks in the signatures of the spectra of the images beans and sand.

	Beans				Sand			
Order	**Position**	**Value**	***D***	**ΔB**	**Position**	**Value**	***D***	**ΔB**
1	$P_u(2)$	24199.7	3335.6	2	$P_v(2)$	3098.6	423.7	3
2	$P_v(7)$	22397.0	10439.4	3	$P_v(6)$	2835.9	506.3	4
3	$P_u(4)$	21719.0	3934.1	3	$P_v(9)$	2547.0	357.8	2
4	$P_u(7)$	21209.1	6045.7	3	$P_u(9)$	2499.7	827.0	6
5	$P_v(10)$	18802.4	7679.4	3	$P_u(3)$	2291.2	452.1	5

$\sum_u P(u, v)$. If the periodicity of the image is perfectly aligned with any of these axes, $P_u(u)$ and $P_v(v)$ will have clear peaks in the right frequencies.

Tables 3.23 and 3.24 show the five highest peaks identified for each of the textures in Figure 3.90, identified in the corresponding $P_u(u)$ and $P_v(v)$ of each spectrum. Figure 3.91 shows the plots of $P_u(u)$ and $P_v(v)$. We may see that the peaks of fabric 2 are clearly distinct from the background and indicate the presence of some repeated pattern along the two orthogonal directions. The continuum over which these peaks sit shows that the pattern is noisy, as indeed is the case. None of the other spectral signatures shows clearly distinct peaks, except perhaps fabric 1, which shows some peaks at low frequencies (large repeated pattern), as indeed is the case. These peaks are at positions 4 and 7 along the two axes, indicating a repeated primitive of size $256/4 \simeq 64$ along one axis and $256/7 \simeq 37$ along the other axis. In a similar way, we can see that the repeated patterns for fabric 2 must be smaller, as their periodicities manifest themselves in higher frequencies.

To be able to tell whether the identified peaks are of any significance, we must find a way to express how distinct they are from their surrounding values. One may do that by moving away from each peak in both directions until a local minimum is reached. Let us say that the value of the peak is A and the values at the two surrounding minima are B_1 and B_2. We define then the "distinctiveness" of the peak as

$$D \equiv A - \frac{B_1 + B_2}{2} \tag{3.286}$$

and its "breadth" as

$$\Delta B = |\text{Position of } B_2 - \text{Position of } B_1|. \tag{3.287}$$

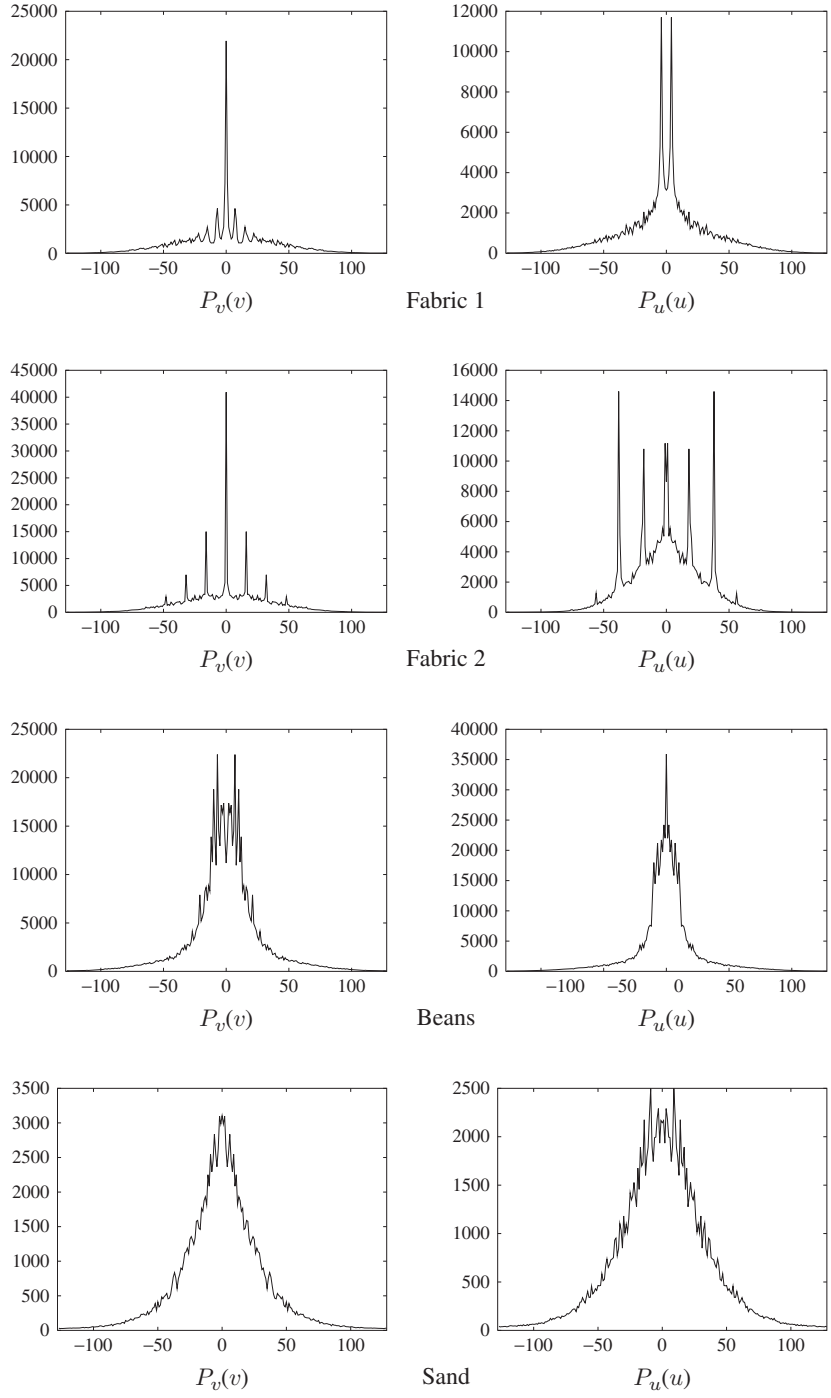


Figure 3.91 Power spectral signatures for fabric 1, fabric 2, beans and sand.

Figure 3.92 The phase spectra of the first three images of Figure 3.1 with their values scaled in the range [0, 255] to allow their representation as images. Source: Maria Petrou.

Tables 3.23 and 3.24 give the distinctiveness and breadth of each of the five strongest peaks for each spectrum. We can clearly see that the peaks which really indicate some periodicity are those which are the most distinct.

What is the phase and magnitude of the Fourier transform?

The magnitude of the Fourier transform is defined as

$$M(u,v) = \sqrt{F_{\mathrm{R}}(u,v)^2 + F_{\mathrm{I}}(u,v)^2} = \sqrt{P(u,v)}. \tag{3.288}$$

The phase $\Phi(u,v)$, which takes values in the range $[-\pi,\pi)$ or $[0,2\pi)$, is defined by

$$\sin\Phi(u,v) = \frac{F_{\mathrm{I}}(u,v)}{M(u,v)} \quad \text{and} \quad \cos\Phi(u,v) = \frac{F_{\mathrm{R}}(u,v)}{M(u,v)}. \tag{3.289}$$

Figure 3.92 shows the phases of the first three images of Figure 3.1. We may observe that these representations do not appear to have any recognisable or useful structure.

Example 3.97

Compute the power spectrum of image 3.93a.
Let us assume that the image is of size $M \times N$, and let us call the image function $f(k,l)$. Then we apply the following algorithm.

Step 1: *Compute the Fourier transform of the image:*

$$F(m,n) = \frac{1}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f(k,l) e^{-2\pi \mathrm{j}\left[\frac{km}{M} + \frac{ln}{N}\right]}. \tag{3.290}$$

Step 2: *Express it as a real and an imaginary part:*

$$F(m,n) = F_{\mathrm{R}}(m,n) + \mathrm{j}F_{\mathrm{I}}(m,n). \tag{3.291}$$

Step 3: *Compute the power spectrum by multiplying $F(m,n)$ with its complex conjugate:*

$$P_f(m,n) = F(m,n)F^*(m,n) = F_{\mathrm{R}}^2(m,n) + F_{\mathrm{I}}^2(m,n). \tag{3.292}$$

(Continued)

Example 3.97 (Continued)

Visualisation of this function is a problem, because its dynamic range is very large. As a consequence, it is difficult to present it as a digital image with grey values in the range $[0, 255]$*. To overcome this problem we need the following steps.*

Step 4: *Compute:*

$$\tilde{P}_f(m,n) \equiv \log(1 + P_f(m,n)). \tag{3.293}$$

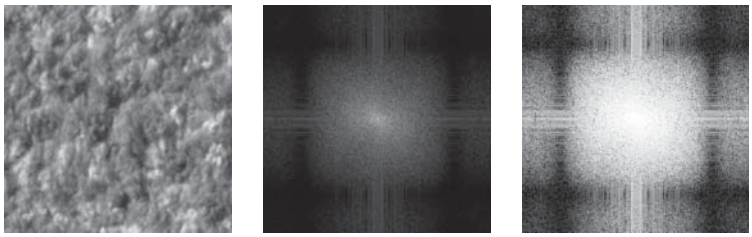
Step 5: *Scale* $\tilde{P}_f(m,n)$ *and round it to the nearest integer, in order to plot it as an image*

$$\tilde{P}_f(m,n)_{\text{new\_value}} = \left\lfloor \frac{\tilde{P}_f(m,n)_{\text{old\_value}} - \tilde{P}_f(m,n)_{\text{min\_value}}}{\tilde{P}_f(m,n)_{\text{max\_value}} - \tilde{P}_f(m,n)_{\text{min\_value}}} \times 255 + 0.5 \right\rfloor \tag{3.294}$$

where $\lfloor\ \rfloor$ *means taking the integer part, and* $\tilde{P}_f(m,n)_{\text{min\_value}}$ *and* $\tilde{P}_f(m,n)_{\text{max\_value}}$ *are the minimum and the maximum values of* $\tilde{P}_f(m,n)$*, respectively.*

Note that function $\tilde{P}_f(m,n)$ *is only created to help us visualise the power spectrum, and any subsequent calculations involving the spectrum have to be done using the raw values of* $P_f(m,n)$*.*

Figure 3.93a shows the original image and Figure 3.93b its power spectrum. If we notice carefully, the power spectrum appears to have some vertical and horizontal structure that does not seem to correspond to any visible periodicity of the kind in the original image. In panel 3.93c we can see the same power spectrum after histogram equalisation. Strong blockiness and some vertical and horizontal structure are very obvious now. These must be due to some artifacts present in the original image which are not discernible by the naked eye.



(a) (b) (c)

Figure 3.93 Computing the power spectrum of an image. (a) Original image. (b) Fourier spectrum of the image. Source: Maria Petrou. (c) The Fourier spectrum after histogram equalisation. Source: Maria Petrou.

Example 3.98

Derive the discrete Fourier transform of a discrete function of length $2N + 1$ **defined as follows:**

$$f(i) = \begin{cases} A & \textbf{if } -M \le i \le M, \\ 0 & \textbf{if } -N \le i < -M \text{ or } M < i \le N. \end{cases} \tag{3.295}$$

Plot the result for $A = 1, M = 3$ **and** $N = 7$**.**

The discrete Fourier transform of this function is defined as:

$$F(k) = \frac{1}{\sqrt{2N+1}} \sum_{i=-M}^{M} A e^{-j\frac{2\pi k i}{2N+1}}. \tag{3.296}$$

If we change summation variable from i to $\tilde{i} \equiv i + M$, we obtain:

$$F(k) = \frac{A}{\sqrt{2N+1}} \sum_{\tilde{i}=0}^{2M} e^{-j\frac{2\pi k(\tilde{i}-M)}{2N+1}} = \frac{A}{\sqrt{2N+1}} e^{j\frac{2\pi k M}{2N+1}} \sum_{\tilde{i}=0}^{2M} e^{-j\frac{2\pi k \tilde{i}}{2N+1}}. \tag{3.297}$$

The sum on the right-hand side of this expression may be computed by applying the geometric progression formula

$$\sum_{t=0}^{T} a_0 q^t = a_0 \frac{q^{T+1} - 1}{q - 1} \tag{3.298}$$

for $a_0 = 1$, $q = e^{-j\frac{2\pi k}{2N+1}}$ and $T = 2M$:

$$F(k) = \frac{A}{\sqrt{2N+1}} e^{j\frac{2\pi k M}{2N+1}} \frac{e^{-j\frac{2\pi k(2M+1)}{2N+1}} - 1}{e^{-j\frac{2\pi k}{2N+1}} - 1}. \tag{3.299}$$

This expression may be further simplified as follows:

$$F(k) = \frac{A}{\sqrt{2N+1}} e^{j\frac{2\pi k M}{2N+1}} \frac{e^{-j\frac{\pi k(2M+1)}{2N+1}} \left(e^{-j\frac{\pi k(2M+1)}{2N+1}} - e^{j\frac{\pi k(2M+1)}{2N+1}} \right)}{e^{-j\frac{\pi k}{2N+1}} \left(e^{-j\frac{\pi k}{2N+1}} - e^{j\frac{\pi k}{2N+1}} \right)}$$

$$F(k) = \frac{A}{\sqrt{2N+1}} e^{j\frac{\pi k M}{2N+1}(2M-2M-1+1)} \frac{-2j \sin\frac{\pi k(2M+1)}{2N+1}}{-2j \sin\frac{\pi k}{2N+1}}$$

$$F(k) = \frac{A}{\sqrt{2N+1}} \frac{\sin\frac{\pi k(2M+1)}{2N+1}}{\sin\frac{\pi k}{2N+1}}. \tag{3.300}$$

To plot this function for $A = 1$, $M = 3$ and $N = 7$, we must compute its values for $k = 0, \pm 1, \pm 2, \ldots, \pm 7$. We notice that for $k = 0$ both numerator and denominator become 0, so we must apply de L'Hospital's rule:

$$F(0) = \frac{A}{\sqrt{2N+1}} \lim_{k \to 0} \frac{\frac{\partial}{\partial k}\left(\sin\frac{\pi k(2M+1)}{2N+1} \right)}{\frac{\partial}{\partial k}\left(\sin\frac{\pi k}{2N+1} \right)}$$

$$F(0) = \frac{A}{\sqrt{2N+1}} \lim_{k \to 0} \frac{\frac{\pi(2M+1)}{2N+1} \cos\frac{\pi k(2M+1)}{2N+1}}{\frac{\pi}{2N+1} \cos\frac{\pi k}{2N+1}}$$

$$F(0) = \frac{A}{\sqrt{2N+1}}(2M+1). \tag{3.301}$$

(Continued)

Example 3.98 (Continued)

Figure 3.94a shows the original function $f(i)$, while Figure 3.94b shows $F(k)$ plotted as a function of k for the parameter values specified.

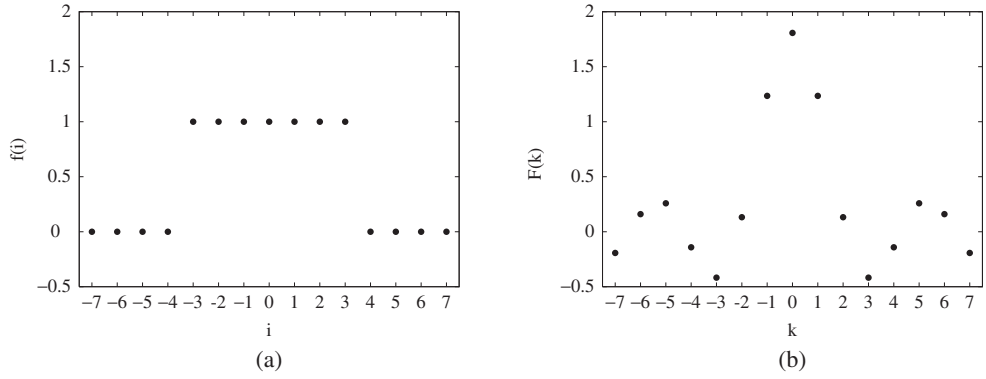


Figure 3.94 (a)A 1D digital signal. (b) The discrete Fourier transform of the signal.

Example B3.99

Show that the Fourier transform of a Gaussian is also a Gaussian.
Consider a Gaussian function

$$g(x) \equiv \mathrm{e}^{-\frac{x^2}{2\sigma^2}} \tag{3.302}$$

and take its Fourier transform:

$$G(\omega) = \int_{-\infty}^{\infty} g(x)\mathrm{e}^{-\mathrm{j}\omega x}\mathrm{d}x = \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{x^2}{2\sigma^2}}\mathrm{e}^{-\mathrm{j}\omega x}\mathrm{d}x = \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{x^2+\mathrm{j}2\sigma^2\omega x}{2\sigma^2}}\mathrm{d}x. \tag{3.303}$$

We may complete the square in the exponent on the right-hand side of this expression to obtain:

$$G(\omega) = \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{x^2+\mathrm{j}2\sigma^2\omega x+\mathrm{j}^2\sigma^4\omega^2-\mathrm{j}^2\sigma^4\omega^2}{2\sigma^2}}\mathrm{d}x = \mathrm{e}^{-\frac{-\mathrm{j}^2\sigma^4\omega^2}{2\sigma^2}}\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x+\mathrm{j}\sigma^2\omega)^2}{2\sigma^2}}\mathrm{d}x. \tag{3.304}$$

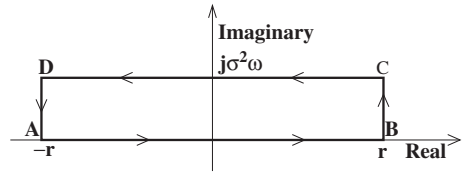
We observe that $\mathrm{j}^2 = -1$. *So:*

$$G(\omega) = \mathrm{e}^{-\frac{\sigma^4\omega^2}{2\sigma^2}}\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x+\mathrm{j}\sigma^2\omega)^2}{2\sigma^2}}\mathrm{d}x. \tag{3.305}$$

The integral we have to compute is that of a complex variable along the real axis. We may use for that contour integration (see Box 3.22). We select the contour shown in Figure 3.95, with $r \to \infty$. *The integrand we wish to integrate along this contour is* $f(z) \equiv \mathrm{e}^{-\frac{z^2}{2\sigma^2}}$. *This function has no poles inside the contour, so its integral along the contour is* 0*:*

$$\int_A^B f(z)\mathrm{d}z + \int_B^C f(z)\mathrm{d}z + \int_C^D f(z)\mathrm{d}z + \int_D^A f(z)\mathrm{d}z = 0. \tag{3.306}$$

Figure 3.95 The contour along which we are going to integrate.



Let us consider each term separately.

$$\int_A^B f(z)\mathrm{d}z = \int_{-r}^{r} \mathrm{e}^{-\frac{x^2}{2\sigma^2}}\mathrm{d}x = \sqrt{2}\sigma \int_{-r/(\sigma\sqrt{2}}^{r/(\sigma\sqrt{2}} \mathrm{e}^{-y^2}\mathrm{d}y. \tag{3.307}$$

For $r \to \infty$ this integral is given by (3.78) as being equal to $\sigma\sqrt{2}\sqrt{\pi}$.

$$\begin{aligned}\int_B^C f(z)\mathrm{d}z &= \int_0^{\sigma^2\omega} \mathrm{e}^{-\frac{(r+\mathrm{j}y)^2}{2\sigma^2}}\mathrm{d}y \\ &= \int_0^{\sigma^2\omega} \mathrm{e}^{-\frac{r^2+2\mathrm{j}ry-y^2}{2\sigma^2}}\mathrm{d}y \\ &= \mathrm{e}^{-\frac{r^2}{2\sigma^2}} \int_0^{\sigma^2\omega} \mathrm{e}^{-\frac{2\mathrm{j}ry-y^2}{2\sigma^2}}\mathrm{d}y.\end{aligned} \tag{3.308}$$

As r tends to ∞, this integral vanishes. The same happens for the integral from D to A. The remaining integral is:

$$\int_C^D f(z)\mathrm{d}z = \int_r^{-r} \mathrm{e}^{-\frac{(x+\mathrm{j}\sigma^2\omega)^2}{2\sigma^2}}\mathrm{d}x = -\int_{-r}^{r} \mathrm{e}^{-\frac{(x+\mathrm{j}\sigma^2\omega)^2}{2\sigma^2}}\mathrm{d}x. \tag{3.309}$$

This is the integral we wish to calculate, if we let r go to ∞. So, if we substitute in (3.306), we obtain:

$$\sigma\sqrt{2\pi} - \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x+\mathrm{j}\sigma^2\omega)^2}{2\sigma^2}}\mathrm{d}x = 0. \tag{3.310}$$

Finally:

$$G(\omega) = \sqrt{2\pi}\sigma \mathrm{e}^{-\frac{\sigma^2\omega^2}{2}} \tag{3.311}$$

This shows that the Fourier transform of a Gaussian is a Gaussian with amplitude $\sqrt{2\pi}\sigma$ times the amplitude of the original Gaussian and variance Σ^2 equal to the inverse of the variance of the original Gaussian: $\Sigma^2 = 1/\sigma^2$.

Box 3.22 Contour integration

Several integrals may be computed with the help of contour integration in the complex plane using the **residue theorem**. The residue of a function $f(z)$ of a complex variable z at point z_0 is defined with the help of function $\phi(z)$

$$\phi(z) \equiv (z - z_0)^m f(z) \tag{3.312}$$

(Continued)

Box 3.22 (Continued)

where m is a positive integer. If $\phi(z_0) \neq 0$, function $f(z)$ is said to have a pole of order m at z_0. Then the residue of $f(z)$ at z_0 is computed as

$$B_0 \equiv \frac{\phi^{(m-1)}(z_0)}{(m-1)!} \tag{3.313}$$

where we remember that $0! = 1$ and $\phi^{(m-1)}(z_0)$ is the $m-1$ derivative of $\phi(z)$, with respect to z, computed at z_0, with $\phi^{(0)}(z_0)$ being the value of the function itself.

Once we have identified the poles of the integrand of the integral we wish to compute, we then select a closed contour along which we shall perform the integration. The residue theorem states that the integral of a function with poles at points $z_0, z_1, \ldots, z_n$, enclosed by the contour, is given by:

$$\int_{\text{contour}} f(z)\mathrm{d}z = 2\pi\mathrm{j}(B_0 + B_1 + \cdots + B_n) \tag{3.314}$$

where $B_0, B_1, \ldots, B_n$ are the residues of the integrand at the corresponding poles.

Figure 3.96 shows some typical contours which might be constructed for the calculation of some integrals.

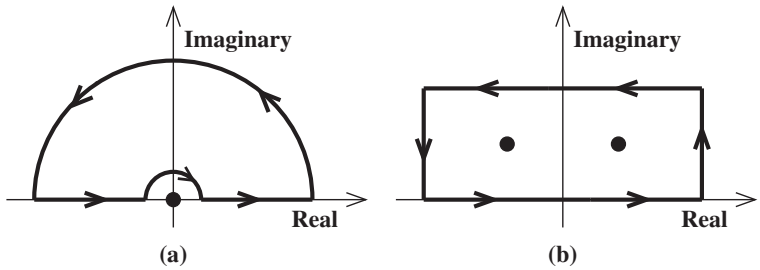


Figure 3.96 The black dots indicate poles of the integrand. In (a) the sum of the integrals along the four branches of the contour (the two semicircles and the two straight line parts) is 0, as the pole is outside the contour. Nevertheless, the pole influences the value of the integral, as, in order to avoid it, we have included the small semicircle that is constructed to have radius $\varepsilon \to 0$. In (b) the sum of the integrals along the four straight line branches of the contour is equal to the sum of the residues at the two poles inside the contour.

Example B3.100

Show that the discrete Fourier transform of a digital Gaussian is not a Gaussian.

Let us consider a Gaussian function of length $2N+1$ and standard deviation σ defined as:

$$g(i) \equiv \mathrm{e}^{-\frac{i^2}{2\sigma^2}} \qquad \text{for } i = -N, -(N-1), \ldots, 0, \ldots, (N-1), N. \tag{3.315}$$

The discrete Fourier transform of this function is given by:

$$G(k) = \frac{1}{\sqrt{2N+1}} \sum_{i=-N}^{N} g(i)\mathrm{e}^{-\mathrm{j}\frac{2\pi ik}{2N+1}} \Rightarrow$$

$$G(k)=\frac{1}{\sqrt{2N+1}}\sum_{i=-N}^{N}\mathrm{e}^{-\frac{i^2}{2\sigma^2}}\mathrm{e}^{-\mathrm{j}\frac{2\pi ik}{2N+1}}\Rightarrow$$

$$G(k)=\frac{1}{\sqrt{2N+1}}\sum_{i=-N}^{N}\mathrm{e}^{-\frac{i^2(2N+1)+\mathrm{j}2\pi ik2\sigma^2}{2\sigma^2(2N+1)}}\Rightarrow$$

$$G(k)=\frac{1}{\sqrt{2N+1}}\sum_{i=-N}^{N}\mathrm{e}^{-\frac{(\sqrt{2N+1}i)^2+2(\sqrt{2N+1}i)\left(\frac{1}{\sqrt{2N+1}}2\pi\mathrm{j}k\sigma^2\right)+\frac{4\pi^2\mathrm{j}^2k^2\sigma^4}{2N+1}-\frac{4\pi^2\mathrm{j}^2k^2\sigma^4}{2N+1}}{2\sigma^2(2N+1)}}\Rightarrow$$

$$G(k)=\frac{1}{\sqrt{2N+1}}\mathrm{e}^{\frac{4\pi^2\mathrm{j}^2k^2\sigma^4}{2\sigma^2(2N+1)^2}}\sum_{i=-N}^{N}\mathrm{e}^{-\frac{(\sqrt{2N+1}i+\frac{1}{\sqrt{2N+1}}2\pi\mathrm{j}k\sigma^2)^2}{2\sigma^2(2N+1)}}\Rightarrow$$

$$G(k)=\frac{1}{\sqrt{2N+1}}\mathrm{e}^{-\left(\frac{2\pi k}{2N+1}\right)^2\frac{\sigma^2}{2}}\sum_{i=-N}^{N}\mathrm{e}^{-\frac{((2N+1)i+2\pi\mathrm{j}k\sigma^2)^2}{2\sigma^2(2N+1)^2}}. \tag{3.316}$$

We notice that the second factor on the right-hand side of (3.316) is a Gaussian of the form $\mathrm{e}^{-\frac{\omega^2}{2\Sigma^2}}$ *where* $\omega\equiv\frac{2\pi k}{2N+1}$ *and* $\Sigma^2\equiv\frac{1}{\sigma^2}$. *However, the sum on the right depends on k, i.e. it makes the amplitude of the Gaussian depend on the frequency k. We cannot remove this dependency by changing the variable of summation, as we did in the case of example 3.99, because the limits of the summation are finite and if we change summation variable, the dependency on k will manifest itself in the limits of the summation instead of being explicit in the summand. This shows that the discrete Fourier transform of function (3.315) does not have the same functional form as the original function.*

Example B3.101

Compare the DFT of a digital Gaussian with the FT of the Gaussian function.

The discrete Fourier transform of $g(i)$ *defined by equation (3.315) may be written as:*

$$G(k)=\frac{1}{\sqrt{2N+1}}\sum_{i=-N}^{N}g(i)\cos\frac{2\pi ik}{2N+1}-\mathrm{j}\frac{1}{\sqrt{2N+1}}\sum_{i=-N}^{N}g(i)\sin\frac{2\pi ik}{2N+1}. \tag{3.317}$$

First we shall show that the imaginary part of $G(k)$ *is zero. Let us consider the sum:*

$$\sum_{i=-N}^{N}g(i)\sin\frac{2\pi ik}{2N+1}. \tag{3.318}$$

Obviously, $g(i)$ *is a symmetric function, i.e.* $g(-i)=g(i)$, *while sine is antisymmetric, so the summand in (3.318) is an antisymmetric function summed over pairs of positive and negative indices. All terms will cancel in pairs, making this sum zero. So, we may say that:*

$$G(k)=\frac{1}{\sqrt{2N+1}}\sum_{i=-N}^{N}\mathrm{e}^{-\frac{i^2}{2\sigma^2}}\cos\frac{2\pi ik}{2N+1}. \tag{3.319}$$

(Continued)

Example B3.101 (Continued)

In Figure 3.97a we plot $G(k)$ as a function of k for $N = 7$ and $\sigma = 3.5$. In the same graph we also plot the continuous Fourier transform of the continuous Gaussian function with the same value of σ, as given by Equation (3.311), appropriately scaled: the definition of the discrete Fourier transform we used, as given by Equation (3.316), includes a factor $1/\sqrt{2N+1}$ that in the continuous domain would have corresponded to a factor $1/\sqrt{2\pi}$, which, however, was not included in definition (3.303) of the continuous Fourier transform we used. So, in Figure 3.97 we plot $G(\omega)/\sqrt{2\pi}$, instead of just $G(\omega)$. The horizontal axis in these figures is index k. For each value of k we compute the corresponding value of ω, which is $\omega = 2\pi k/(2N+1)$, and use it to compute the value of $G(\omega)$. Figures 3.97b and 3.97c show the corresponding graphs for $N = 9$ and $N = 12$ respectively.

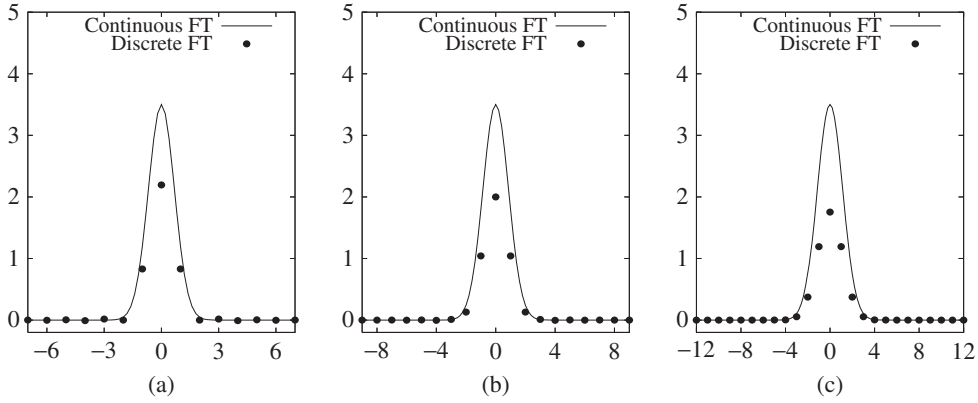


Figure 3.97 Continuous and discrete Fourier transforms of a Gaussian. (a) $N = 7$. (b) $N = 9$. (c) $N = 12$.

Example 3.102

Compute the power spectrum of the 1D digital signals

$$f_1(i) = \sin\frac{2\pi i}{4} \qquad \text{for } i = 0, 1, \ldots, 15 \tag{3.320}$$

and

$$f_2(i) = \sin\frac{2\pi i}{8} \qquad \text{for } i = 0, 1, \ldots, 15. \tag{3.321}$$

Figure 3.98 shows the digital signals and their corresponding power spectra. Note that each signal contains exactly one frequency, and therefore, only one peak appears in the first half of its power spectrum at the expected location.

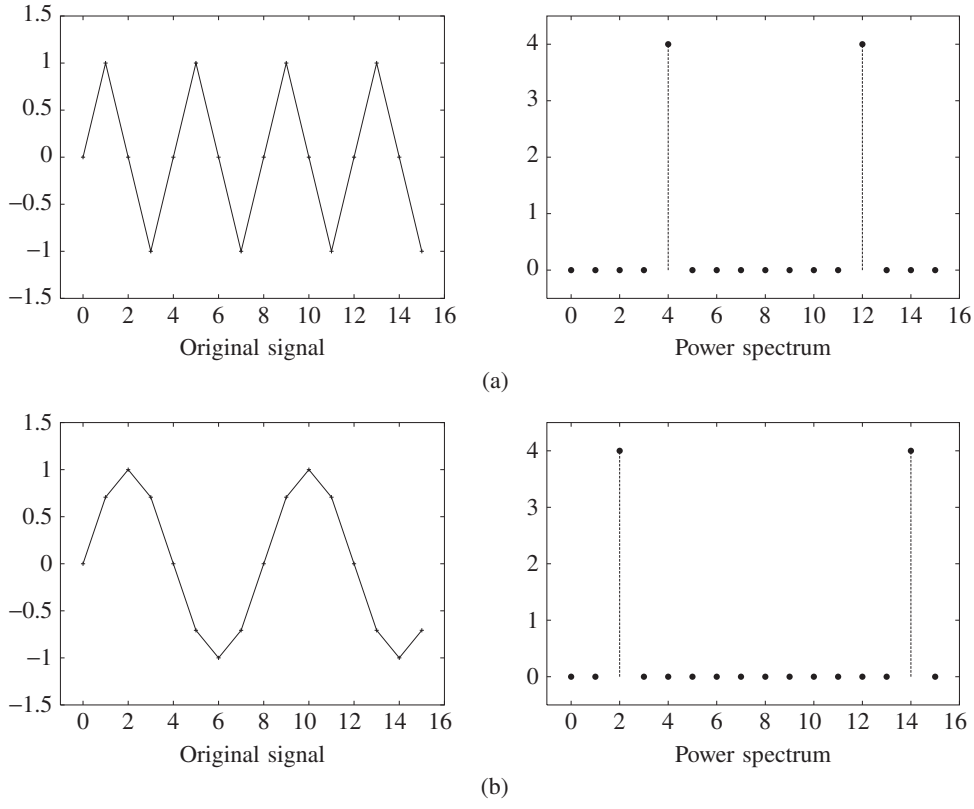


Figure 3.98 Two digital signals and their corresponding power spectra. (a) $f_1(i) = \sin\frac{2\pi i}{4}$. (b) $f_2(i) = \sin\frac{2\pi i}{8}$.

Example 3.103

Consider the following 1D digital signal:

$$f(i) = \sin\frac{2\pi i}{4} \qquad \text{for } i = 0, 1, \ldots, 16. \tag{3.322}$$

Compute its power spectrum, compare it with the power spectrum of signal $f_1(i)$ of Example 3.102 and comment.

The second column of Table 3.25 presents the values of this signal for $i = 0, 1, 2, \ldots, 16$. The Fourier transform of this signal is given by:

$$F(k) = \frac{1}{\sqrt{17}}\sum_{i=0}^{16} f(i)\mathrm{e}^{-\mathrm{j}\frac{2\pi ki}{17}} \qquad \text{for } k = 0, \ldots, 16. \tag{3.323}$$

(Continued)

Example 3.103 (Continued)

Table 3.25 Second column: values of function $f(i)$. Third column: a Gaussian window. Fourth column: the result of multiplying $f(i)$ with the Gaussian window. Fifth column: $T(i)$ window given by equation (3.326). Last column: the result of multiplying $f(i)$ with $T(i)$.

i	$f(i)$	$G(i)$	$f(i)G(i)$	$T(i)$	$f(i)T(i)$
0	0.000000	0.009999	0.000000	0.000000	0.000000
1	1.000000	0.029424	0.029424	0.761594	0.761594
2	0.000000	0.074983	0.000000	0.964028	0.000000
3	−1.000000	0.165472	−0.165472	1.000000	−1.000000
4	0.000000	0.316216	0.000000	1.000000	−0.000000
5	1.000000	0.523289	0.523289	1.000000	1.000000
6	0.000000	0.749888	0.000000	1.000000	0.000000
7	−1.000000	0.930570	−0.930570	1.000000	−1.000000
8	0.000000	1.000000	0.000000	1.000000	0.000000
9	1.000000	0.930570	0.930570	1.000000	1.000000
10	0.000000	0.749888	0.000000	1.000000	0.000000
11	−1.000000	0.523289	−0.523289	1.000000	−1.000000
12	0.000000	0.316216	0.000000	1.000000	0.000000
13	1.000000	0.165472	0.165472	1.000000	1.000000
14	0.000000	0.074983	0.000000	0.964028	0.000000
15	−1.000000	0.029424	−0.029424	0.761594	−0.761594
16	0.000000	0.009999	0.000000	0.000000	0.000000

We may separate the real and the imaginary parts in the above expression to write:

$$F(k) = \frac{1}{\sqrt{17}} \sum_{i=0}^{16} f(i) \cos \frac{2\pi ki}{17} - \mathrm{j} \frac{1}{\sqrt{17}} \sum_{i=0}^{16} f(i) \sin \frac{2\pi ki}{17}. \tag{3.324}$$

The power spectrum computed from $F(k)$ is shown in the second column of Table 3.26. For the time being, ignore the other two columns of this table.

Table 3.26 Values of the power spectra of $f(i)$, $f(i) \times G(i)$, and $f(i) \times T(i)$. $G(i)$ is a Gaussian mask the same size as the signal, used in order to taper the edges of the signal. $T(i)$ is used for the same purpose but it is flat away from the edges, so it preserves the shape of the signal better.

	Power spectrum of		
k	$f(i)$	$f(i)G(i)$	$f(i)T(i)$
0	0.000000	0.000000	0.000000
1	0.002284	0.000105	0.000191
2	0.014056	0.017783	0.000253
3	0.082049	0.500836	0.032783
4	3.135978	2.066701	3.148183
5	0.500200	1.294236	0.626852
6	0.129792	0.118448	0.128415
7	0.075281	0.001889	0.043126
8	0.060361	0.000000	0.020199
9	0.060361	0.000000	0.020199
10	0.075281	0.001889	0.043126
11	0.129792	0.118448	0.128415
12	0.500200	1.294236	0.626852
13	3.135978	2.066701	3.148183
14	0.082049	0.500836	0.032783
15	0.014056	0.017783	0.000253
16	0.002284	0.000105	0.000191

In Figure 3.99a we plot the original signal and in Figure 3.99b its power spectrum. The power spectrum is intentionally plotted out of scale to show the power that appears at frequencies other than those of the main peaks. From (3.322) we can see that the original continuous signal was a sinusoid with frequency $\pi/2$. However, the power spectrum we computed from it indicates non-zero energy in other frequencies, due to the discontinuity between the start and the end point of the signal. In addition, the two main peaks appear at indices $k_1 = 4$ and $k_2 = 13$, which correspond to frequencies $\omega_1 = 2\pi 4/17 = 0.47\pi$ and $\omega_2 = 2\pi 13/17 = 1.53\pi$ (i.e. -0.47π) which are shifted away from the $\pm 0.5\pi$ frequencies of the continuous signal.

(Continued)

Example 3.103 (Continued)

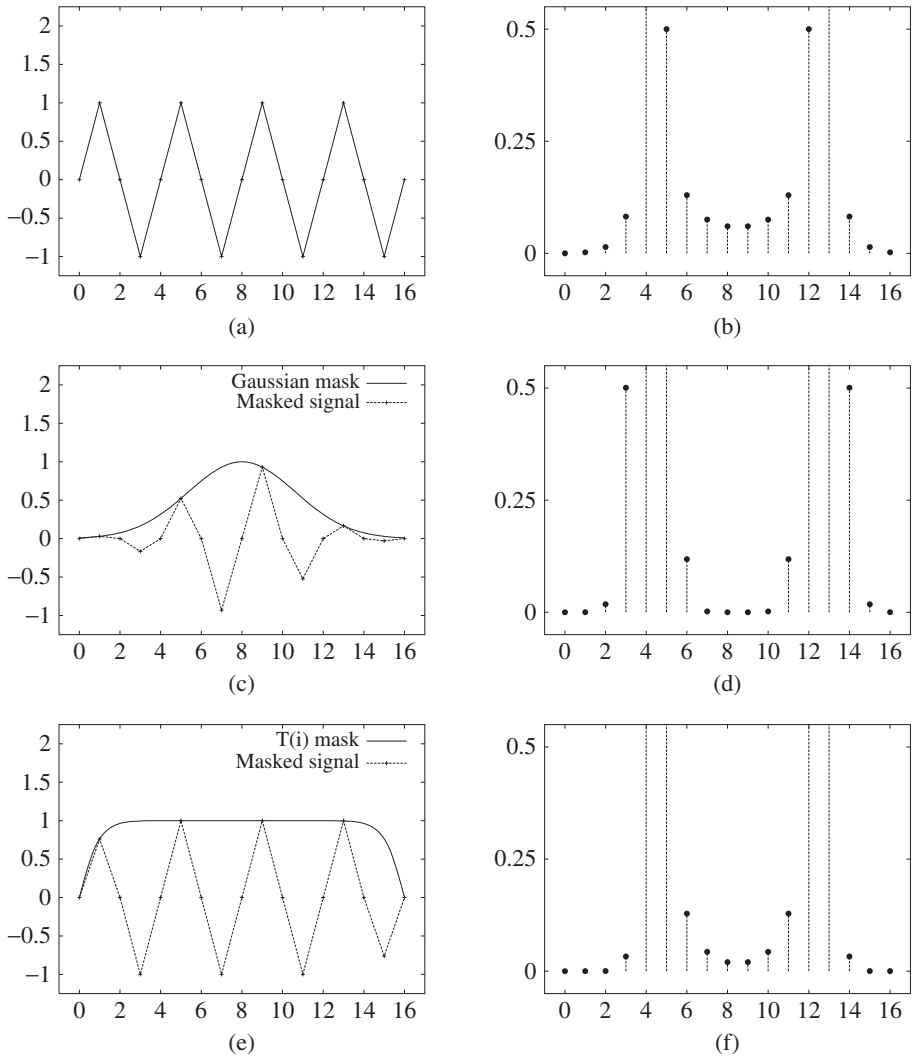


Figure 3.99 (a) A digital signal. (b) The power spectrum of the digital signal. (c) The signal multiplied with a Gaussian mask. (d) The power spectrum of the signal on the left. (e) The signal multiplied with the mask given by Equation (3.326). (f) The power spectrum of the signal on the left. These power spectra contain some peaks that go beyond the vertical scale of the graphs. The reason we plot them this way is because we wish to pay attention to the power distributed at frequencies other than the main peaks.

The difference between Figures 3.98 and 3.99b is due to the size of the signal in each case. In the first case the size of the signal is a multiple of the signal period (see Equation (3.320)), so the repetition of the signal assumed by the discrete Fourier transform does not introduce any new frequencies. Here, however, the size of the signal is not a multiple of the signal period, so non-zero energy appears at several frequencies due to the interference between the natural frequency of the signal and the size of the signal. The peaks are spread, and their maxima are shifted away from their true locations.

Example 3.104

Multiply the digital signal of Equation (3.322) with a digital Gaussian function point by point and compute its power spectrum. Choose the standard deviation σ of the Gaussian so that the discontinuity due to its truncation is 0.01. Normalise the result so that the original signal and the new signal have the same total energy.

Let us first define the Gaussian mask we need:

$$G(i) = \mathrm{e}^{-\frac{(i-8)^2}{2\sigma^2}}. \tag{3.325}$$

If we wish to have a maximum discontinuity of 0.01 at the edges of the mask, we must choose the σ *of the Gaussian so that* $G(0) = 0.01$, *i.e.* $0.01 = \mathrm{e}^{-\frac{(-8)^2}{2\sigma^2}} \Rightarrow \ln 0.01 = -\frac{64}{2\sigma^2} \Rightarrow \sigma = \sqrt{-\frac{64}{2\ln 0.01}} = 2.636$.

The third column of Table 3.25 gives the values of this mask and the fourth column the values of the signal after it is multiplied with the mask point by point.

Next, we compute the power spectrum of the masked signal. However, since we do not wish the new signal to have less overall energy than the original one, we scale the values of its power spectrum by the ratio of the total energy of the original signal over the total energy of the new signal. By summing the squares of the values of the second column of Table 3.25 we conclude that the original signal had energy 8. By summing the squares of the values of the fourth column of Table 3.25 we conclude that the new signal has energy 2.336077. *Therefore, we multiply the values of the spectrum of the new signal with* $8/2.336077 = 3.424544$. *This way we obtain the values presented in the penultimate column of Table 3.26. From these numbers we can see that the use of the Gaussian mask reduced the energy in frequencies away from the true frequency of the original signal, while it preserved the two signal peaks. Figure 3.99c shows the profile of the mask and the masked signal, and Figure 3.99d shows the power spectrum of the masked signal. The vertical scale has intentionally been chosen so that the real peaks are drawn out of scale while we see the changes in the energy distribution in the remaining frequencies. We can see now that the true spectral content of the signal has been preserved while the energy in other frequencies has been largely suppressed.*

Example 3.105

Multiply point by point the digital signal of Equation (3.322) with a digital $T(i)$ function defined as

$$T(i) \equiv \begin{cases} \tanh i & \text{for } i = 0, 1, 2 \\ 1 & \text{for } 3 \leq i \leq I-3 \\ \tanh(I-i) & \text{for } i = I-2, I-1, I \end{cases} \tag{3.326}$$

where I is the maximum value of i. Compute the power spectrum of the resultant signal. Normalise the result so that the original signal and the new signal have the same total energy.

In this case parameter I in definition (3.326) is $I = 16$. *The last two columns of Table 3.25 show the mask, and the signal multiplied with this mask point by point, respectively. The total energy of the masked signal is* 7.160051. *To preserve, therefore, the total energy of the signal, we must multiply the values of its power spectrum with* $8/7.160051 = 1.117311$. *The last column of Table 3.26 gives the power spectrum of the masked signal.*

Figures 3.99e and 3.99f show the masked signal and its scaled power spectrum, in a scale that allows us to appreciate the suppression of the energy in frequencies outside the main two peaks. We note that in comparison with the Gaussian mask, this mask preserves more of the energy of the original signal.

If we fit the values of each spectrum of Table 3.26 around the peak at 4 with a parabola, we may identify the exact location of the peak and from that infer the frequency of the continuous signal. It turns out that the peak is at points 4.037, 4.170 *and* 4.053 *for the three spectra, i.e. that of the unmasked signal, that of the signal masked with the Gaussian and that of the signal masked with the* $T(i)$ *filter, respectively, yielding frequencies* $2\pi 4.037/17 = 0.475\pi$, $2\pi 4.17/17 = 0.491\pi$ *and* $2\pi 4.053/17 = 0.477\pi$, *respectively, instead of the correct* 0.5π. *This shows that the Gaussian mask, even though it requires higher energy correction than the* $T(i)$ *mask, has superior performance in terms of yielding a more accurate estimation of the true frequency of the underlying signal.*

Example 3.106

Draw three cycles of the signal that is actually seen by the discrete Fourier transform in Examples 3.103, 3.104 and 3.105. Use them to explain how the use of the mask in the last two examples allows the reduction of the energy in frequencies outside the peaks formed by the main frequency present in the signal.

The discrete Fourier transform assumes that the signal is repeated ad infinitum outside its range of definition. So, the Fourier transform of the original signal actually refers to the signal shown in Figure 3.100a, while the Fourier transform of the signal multiplied with the Gaussian mask refers to the signal shown in Figure 3.100b. The Fourier transform of the signal multiplied with mask $T(i)$ *refers to the signal shown in Figure 3.100c.*

In all three cases only three cycles of the signal are shown, but the signal is assumed to continue ad infinitum in both directions. We can see that in all three cases we cannot avoid the periodicity

due to the finite length of the original signal. However, in both masked cases the transitions at the seams of the repeated segments are reduced in size, suppressing their power.

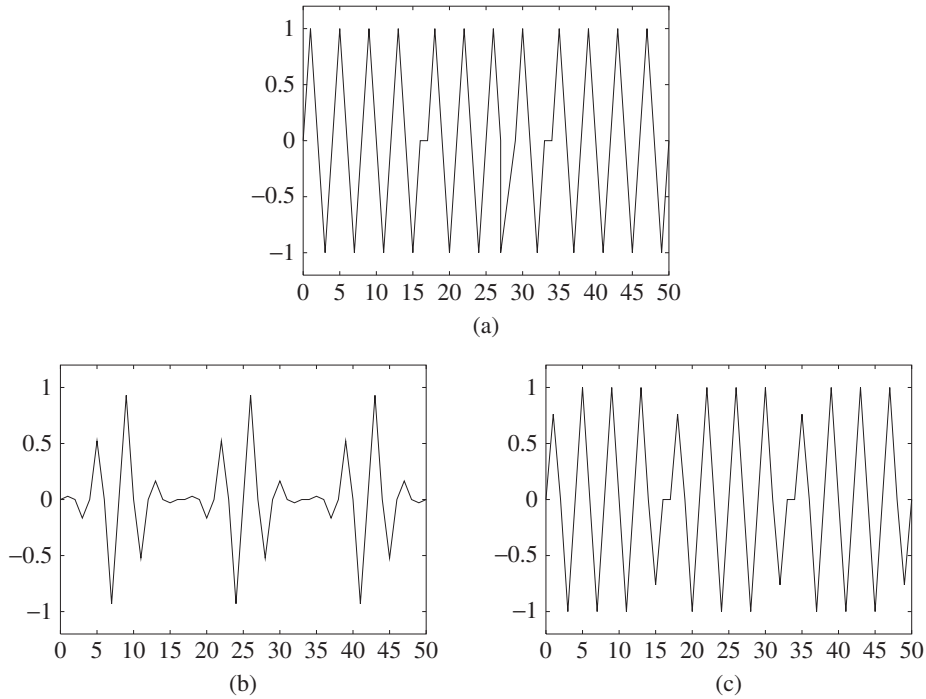


Figure 3.100 (a) Digital signal (3.322) repeated three times. (b) The same signal, multiplied with a Gaussian mask, repeated three times. (c) The same signal, multiplied with the $T(i)$ mask, repeated three times.

Example B3.107

In view of the results of examples 3.98 and 3.104 interpret the power spectrum computed in Example 3.97, shown in Figure 3.93c.

The vertical and horizontal lines at low frequencies are due to the discontinuities of the image at its edges: the discrete Fourier transform assumes that the image is repeated ad infinitum. This implies that the left-most edge of the image is brought into contact with the right-most edge and the top with its bottom. If the image contains macro-texture like this particular one, this wrapping round of the image borders creates discontinuities that manifest themselves as vertical and horizontal frequencies. We may eliminate them by tapering the edges of the image by multiplying it with a Gaussian window. This particular image is of size 256×256 *and the Gaussian window we must use must be centred at pixel* $(128.5, 128.5)$ *and have standard deviation* $\sigma = 59.4$*, so that its*

(Continued)

Example B3.107 (Continued)

discontinuity at point $i = 1, j = 128.5$ is only 0.1. It is defined according to the following formula:

$$G(i,j) \equiv e^{-((i-128.5)^2+(j-128.5)^2)/(2\sigma^2)} \quad \text{for } i = 1,\dots,256 \quad \text{and } j = 1,\dots,256. \tag{3.327}$$

This window is shown in Figure 3.101a. The windowed image is shown in 3.101b. The power spectrum of the windowed image is shown in 3.101c and the histogram equalised version of it in 3.101d.

We notice that the vertical and horizontal lines of the power spectrum have disappeared, but the blocky structure remains. This blocky structure is also artificial as it implies a repetition of the image with period smaller than that of the whole image. In fact, the size of the central block is 172×172. Such an effect can only be created by applying a rectangular convolution filter on the image. The power spectrum of the image then must have been multiplied with the Fourier transform of this filter. It is possible that such a filter was applied by the scanner with which this image was digitised. The impulse response of a rectangular filter has been computed in Example 3.98. From the position of the edges of the blocky structure we can infer the size of the filter used to scan the image. From Equation (3.300) we can see that the Fourier transform of such a filter becomes zero when $k = (2N+1)/(2M+1)$. Since for our image $2N+1 = 256$, and the first zero of the Fourier transform happens when $k = 172/2 = 86$, we infer that $2M+1 = 2.98$ and therefore the size of the unknown filter must have been 3×3. This image, therefore, cannot be used for any further analysis as it has been distorted. A correct (undistorted) version of it will be used for the subsequent examples.

In all cases we wish to use the power spectrum of an image to work out the image characteristics, we shall use the Gaussian window to avoid the effects of the wrap-round borders. Alternatively, we could use the 2D version of the window defined by Equation (3.326). It is also worth noting that before we use an image we must check it carefully for any distortions it might have been subjected to, as these distortions may be detrimental to what we wish to compute from it.

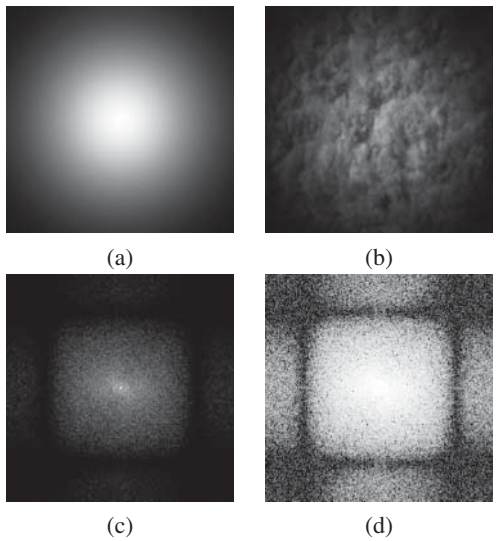


Figure 3.101 Computing the power spectrum of an image using a Gaussian window. (a) A Gaussian mask. (b) The image multiplied with the Gaussian mask. (c) The Fourier spectrum of the masked image. (d) The Fourier spectrum of the masked image after histogram equalisation. This spectrum has to be compared with the spectrum of the same image shown in 3.93c to observe that the vertical and horizontal artifacts have been eliminated by using the Gaussian window. Source: Maria Petrou.

Example 3.108

Compute the fractal dimension of image 3.1d using Equations (3.158) and (3.161).

First we have to compute the power spectrum of the image using the process defined in Example 3.104 and the Gaussian window defined in Example 3.107 as the image is of size 256×256*. The result for image 3.102a is shown in Figure 3.102b.*

We need a single cross-section of this power spectrum from which to compute the fractal dimension. Let us define some polar coordinates (ω, θ) *in the frequency space* (m, n)*:*

$$m \equiv \omega \cos\theta \qquad n \equiv \omega \sin\theta. \tag{3.328}$$

One may choose a value of θ *for which to plot the cross-section of the computed power spectrum, for values of* ω*:* $1, 2, 3, 4, \ldots, \min(M, N)$*. For the selected value of* θ *and for each value of* ω *we can produce a pair of values* $(\tilde{m}, \tilde{n})$*, for which we need the corresponding value of the power spectrum* $P_f(\tilde{m}, \tilde{n})$*. The pair* $(\tilde{m}, \tilde{n})$*, however, will not, in general, be an integer pair, so the value* $P_f(\tilde{m}, \tilde{n})$ *will correspond to an inter-grid position and it will have to be computed by interpolation. This is a bad idea. We should avoid interpolating spectra, as interpolation tacitly assumes that the interpolated function is smooth, while spectra often contain spikes, even when they correspond to smooth images. It is far better to consider the values of* (m, n) *we have, and work out the corresponding* (ω, θ) *values:*

$$\omega = \sqrt{m^2 + n^2} \qquad \theta = \tan^{-1}\left(\frac{n}{m}\right). \tag{3.329}$$

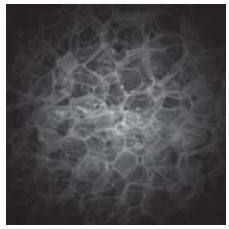
Here we must use a DFT where the dc component is in the middle of the image, and m and n are allowed to take values in the range $[-128, 127]$*. The sign of m and n should be used, so that* θ *takes values in the range* $[0, 360°)$*.*

Then we usually take slices of the power spectrum along various directions and compute parameter β*. Often people average the values obtained along the various directions. Of course, when we select an orientation, we have to allow for some tolerance in the values of* θ *we worked out. Say we select direction* θ_0*. Then we may consider all points we have with* $\theta_0 - \Delta\theta < \theta < \theta_0 + \Delta\theta$*, where* $\Delta\theta$ *is some tolerance we are prepared to accept. We can plot those points versus* ω*, to visualise the power spectrum of the image along direction* θ_0*, and we may fit the pairs of values we have* $(\log\omega, \log[P(\omega; \theta_0)])$ *with the least squares method, to work out the slope of the fitted curve* $\beta(\theta_0)$*, and from it the fractal dimension along* θ_0*, from Equation (3.161) with* $D_T = 2$*. We may also use all points we have together, irrespective of their* θ *value.*

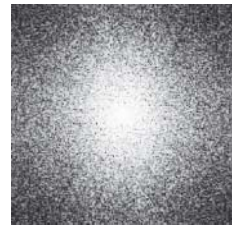
In Figures 3.102c–3.102e we plot some of the cross-sections we computed for this image for $\theta_0 = 0°$*,* $\theta_0 = 45°$ *and* $\theta_0 = 90°$ *with* $\Delta\theta = 1°$*. In 3.102f we plot all points, irrespective of the value of* θ*. By fitting the pairs of points in each panel with a straight line using LSE, we worked out the following fractal dimensions for this image. From the power spectrums as shown in Figures 3.102, by calculating* β*, the results of the fractal dimension are shown in Table 3.27.*

(Continued)

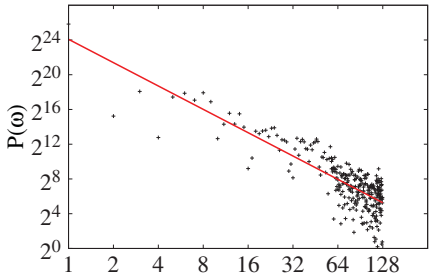
Example 3.108 (Continued)



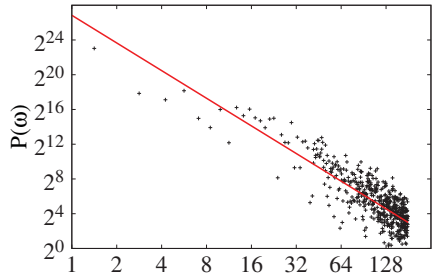
(a) Original image multiplied with a Gaussian mask



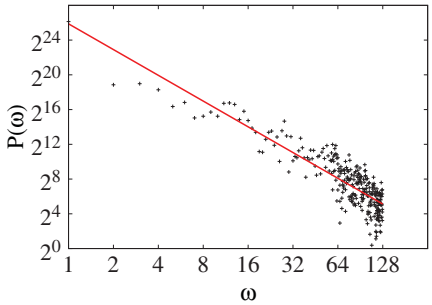
(b) Histogram equalised Fourier spectrum



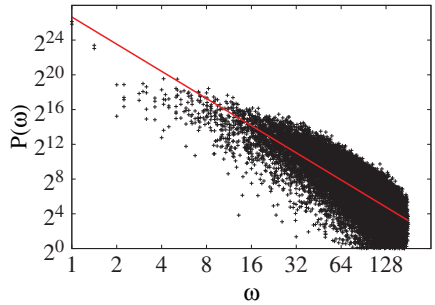
(c) Cross-section for $\theta = 0°$



(d) Cross-section for $\theta = 45°$



(e) Cross-section for $\theta = 90°$



(f) Whole spectrum

Figure 3.102 Computing the fractal dimension of an image from its power spectrum. (a) Original image (plastic) multiplied with the Gaussian mask. Source: Maria Petrou. (b) Fourier spectrum of the image after histogram equalisation. Source: Maria Petrou. (c)–(e) Cross-sections of the power spectrum for $\theta = 0°, 45°$ and $90°$, respectively. (f) Using all orientations together.

Table 3.27 The results of the fractal dimension D by calculating β.

θ	β	D
0°	2.687	2.656
45°	3.183	2.409
90°	2.964	2.518
All points	3.127	2.436

Example B3.109

You are given a real digital signal $f(i)$ of length $2N+1$. Prove that its power spectrum is equal to the discrete Fourier transform of its autocorrelation function.

For the sake of this proof we define the discrete Fourier transform of a signal as:

$$F(k) = \frac{1}{2N+1}\sum_{i=-N}^{N} f(i)\mathrm{e}^{-\mathrm{j}\frac{2\pi ki}{2N+1}}. \tag{3.330}$$

The power spectrum of the signal then is:

$$P(k) \equiv |F(k)|^2 = F(k)F^*(k) = \frac{1}{(2N+1)^2}\sum_{i=-N}^{N} f(i)\mathrm{e}^{-\mathrm{j}\frac{2\pi ki}{2N+1}}\sum_{l=-N}^{N} f^*(l)\mathrm{e}^{\mathrm{j}\frac{2\pi kl}{2N+1}}. \tag{3.331}$$

Here we have changed the dummy variable of the definition of DFT from i to l in the second sum in order to avoid any confusion. Since $f(i)$ is real, it is equal to its complex conjugate, so $f^*(l) = f(l)$, and we may write:

$$P(k) = \frac{1}{(2N+1)^2}\sum_{i=-N}^{N}\sum_{l=-N}^{N} f(i)f(l)\mathrm{e}^{\mathrm{j}\frac{2\pi k(l-i)}{2N+1}}. \tag{3.332}$$

Let us define a new summation variable: $t \equiv i-l \Rightarrow i = l+t$.

Figure 3.103 shows the summation space when the (i,l) or the (t,l) variables are used. Note that the bounding equations $i = N$ and $i = -N$ are transformed into $t = N-l$ and $t = -N-l$, respectively. The power spectrum now is equal to:

$$\begin{aligned} P(k) &= \frac{1}{(2N+1)^2}\sum_{l=-N}^{N}\sum_{t=-N-l}^{N-l} f(l+t)f(l)\mathrm{e}^{-\mathrm{j}\frac{2\pi kt}{2N+1}} \\ &= \frac{1}{(2N+1)^2}\left\{\sum_{l=0}^{N}\sum_{t=-N-l}^{-N-1} + \sum_{l=0}^{N}\sum_{t=-N}^{-1} + \sum_{l=0}^{N}\sum_{t=0}^{N-l} + \right. \\ &\left. \sum_{l=-N}^{-1}\sum_{t=-N-l}^{-1} + \sum_{l=-N}^{-1}\sum_{t=0}^{N} + \sum_{l=-N}^{-1}\sum_{t=N+1}^{N-l}\right\} f(l+t)f(l)\mathrm{e}^{-\mathrm{j}\frac{2\pi kt}{2N+1}}. \end{aligned} \tag{3.333}$$

In the first sum we change the variable of summation from t to $\tilde{t} \equiv t+2N+1 \Rightarrow t = \tilde{t}-2N-1$ so that the limits of summation over $\tilde{t}$ are from $N-l+1$ to N. In the last sum we change variable of summation from t to $\tilde{t}' \equiv t-2N-1 \Rightarrow t = \tilde{t}'+2N+1$ so that the limits of summation over $\tilde{t}'$ are from $-N$ to $-N-l-1$:

$$\begin{aligned} P(k) = \Bigg[&\left\{\sum_{l=0}^{N}\sum_{t=-N}^{-1} + \sum_{l=0}^{N}\sum_{t=0}^{N-l} + \sum_{l=-N}^{-1}\sum_{t=-N-l}^{-1} + \sum_{l=-N}^{-1}\sum_{t=0}^{N}\right\} f(l+t)f(l)\mathrm{e}^{-\mathrm{j}\frac{2\pi kt}{2N+1}} \\ &+\sum_{l=0}^{N}\sum_{\tilde{t}=N-l+1}^{N} f(l+\tilde{t}-2N-1)f(l)\mathrm{e}^{-\mathrm{j}\frac{2\pi k(\tilde{t}-2N-1)}{2N+1}} \\ &+\sum_{l=-N}^{-1}\sum_{\tilde{t}'=-N}^{-N-l-1} f(l+\tilde{t}'+2N+1)f(l)\mathrm{e}^{-\mathrm{j}\frac{2\pi k(\tilde{t}'+2N+1)}{2N+1}}\Bigg] \frac{1}{(2N+1)^2}. \end{aligned} \tag{3.334}$$

(Continued)

Example B3.109 (Continued)

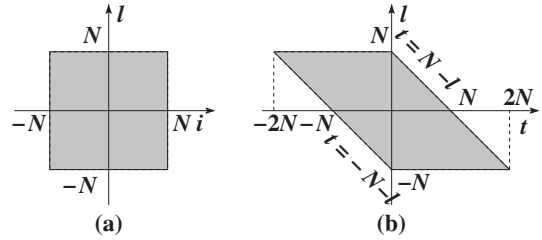


Figure 3.103 (a) The shaded region indicates the range of indices i and l over which we sum in order to compute the power spectrum of the signal. (b) The region over which we sum when we change indices of summation to t and l.

Since for the computation of DFT, the signal is assumed to be repeated ad infinitum with periodicity equal to its size, $f(l+\tilde{t}-2N-1)=f(l+\tilde{t})$ and $f(l+\tilde{t}'+2N+1)=f(l+\tilde{t}')$. In addition,

$$e^{-j\frac{2\pi k(\tilde{t}-2N-1)}{2N+1}}=e^{-j\frac{2\pi k\tilde{t}}{2N+1}}e^{j\frac{2\pi k(2N+1)}{2N+1}}=e^{-j\frac{2\pi k\tilde{t}}{2N+1}}e^{j2\pi k}=e^{-j\frac{2\pi k\tilde{t}}{2N+1}} \tag{3.335}$$

and similarly:

$$e^{-j\frac{2\pi k(\tilde{t}'+2N+1)}{2N+1}}=e^{-j\frac{2\pi k\tilde{t}'}{2N+1}}. \tag{3.336}$$

Finally, we may rename the dummy summation variables $\tilde{t}$ and $\tilde{t}'$ as t without confusion. Then upon substitution into (3.334), we obtain:

$$P(k)=\frac{1}{(2N+1)^2}\left\{\sum_{l=0}^{N}\sum_{t=-N}^{-1}+\sum_{l=0}^{N}\sum_{t=0}^{N-l}+\sum_{l=-N}^{-1}\sum_{t=-N-l}^{-1}+\sum_{l=-N}^{-1}\sum_{t=0}^{N}+\sum_{l=0}^{N}\sum_{t=N-l+1}^{N}+\sum_{l=-N}^{-1}\sum_{t=-N}^{-N-l-1}\right\}f(l+t)f(l)e^{-j\frac{2\pi kt}{2N+1}}. \tag{3.337}$$

We combine the first, second and fifth sums, as well as the third, fourth and sixth sums together, to obtain:

$$\begin{aligned}P(k)&=\frac{1}{(2N+1)^2}\left\{\sum_{l=0}^{N}\sum_{t=-N}^{N}+\sum_{l=-N}^{-1}\sum_{t=-N}^{N}\right\}f(l+t)f(l)e^{-j\frac{2\pi kt}{2N+1}}\\&=\frac{1}{(2N+1)^2}\sum_{l=-N}^{N}\sum_{t=-N}^{N}f(l+t)f(l)e^{-j\frac{2\pi kt}{2N+1}}\\&=\frac{1}{2N+1}\sum_{t=-N}^{N}\underbrace{\frac{1}{2N+1}\sum_{l=-N}^{N}f(l+t)f(l)e^{-j\frac{2\pi kt}{2N+1}}}_{R(t)}.\end{aligned} \tag{3.338}$$

We recognise the quantity identified by the underbrace as being the autocorrelation function $R(t)$ of the original signal. Then:

$$P(k)=\frac{1}{2N+1}\sum_{t=-N}^{N}R(t)e^{-j\frac{2\pi kt}{2N+1}} \tag{3.339}$$

which shows that the DFT of the autocorrelation function of the signal is equal to its power spectrum. This result is known as the **Wiener–Khinchine** *theorem.*

Does the phase of the Fourier transform convey any useful information?

It has repeatedly been demonstrated that the phase of the Fourier transform contains more information than the magnitude. See Examples 3.111 and 3.112.

Example 3.110

Consider the original image of Figure 3.104. Compute its Fourier transform and from this its magnitude, using formula (3.288), and its phase, using formulae (3.289).

From the phase and magnitude of an image we can recover the real and imaginary parts of its Fourier transform using:

$$\begin{aligned} F_{\mathrm{R}}(u,v) &= M(u,v)\cos\Phi(u,v) \\ F_{\mathrm{I}}(u,v) &= M(u,v)\sin\Phi(u,v). \end{aligned} \tag{3.340}$$

Apply the above formulae to compute the real and imaginary parts of the Fourier transform using the original values of $M(u,v)$, but instead of $\Phi(u,v)$, use $\Phi_{new\_1}(u,v) = \Phi(u,v) + 30°$, $\Phi_{\mathrm{new\_2}}(u,v) = \Phi(u,v) + 60°$, and $\Phi_{\mathrm{new\_3}}(u,v) = \Phi(u,v) + 90°$. In each case use the real and imaginary parts as input to the inverse Fourier transform to construct an image. What is the relationship of these images with the original one?

The results are shown in Figure 3.104. The original image was a real image with no imaginary part. Note how the added angle to the phase influences the real and imaginary parts of the inverse Fourier transform. For example, adding 90° exchanges the real and imaginary parts of the original image.

To work out the relationship between these outputs and the original one, we start from the reconstruction formula for image $f(m,n)$ from its DFT $F(u,v)$. The image is assumed to be of size $M \times N$. Then

$$f(m,n) = \frac{1}{MN}\sum_{u,v} F(u,v)\mathrm{e}^{\mathrm{j}2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)} = \frac{1}{MN}\sum_{u,v} M(u,v)e^{\mathrm{j}\Phi(u,v)}\mathrm{e}^{\mathrm{j}2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)} \tag{3.341}$$

where we expressed the Fourier transform in terms of its magnitude and phase. Suppose now that we add and subtract a number θ from the phase of the DFT:

$$\begin{aligned} f(m,n) &= \frac{1}{MN}\sum_{u,v} M(u,v)\mathrm{e}^{\mathrm{j}(\Phi(u,v)+\theta)}\mathrm{e}^{\mathrm{j}2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)}\mathrm{e}^{-\mathrm{j}\theta} \\ &= \Bigg[\underbrace{\frac{1}{MN}\sum_{u,v} M(u,v)\cos\left(\Phi(u,v)+\theta+\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right)}_{\tilde{F}_{\mathrm{R}}(m,n)} + \\ &\quad \underbrace{\mathrm{j}\frac{1}{MN}\sum_{u,v} M(u,v)\sin\left(\Phi(u,v)+\theta+\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right)}_{\tilde{F}_{\mathrm{I}}(m,n)}\Bigg]\mathrm{e}^{-\mathrm{j}\theta} \\ &\equiv \left[\tilde{F}_{\mathrm{R}}(m,n) + \mathrm{j}\tilde{F}_{\mathrm{I}}(m,n)\right]\mathrm{e}^{-\mathrm{j}\theta}. \end{aligned} \tag{3.342}$$

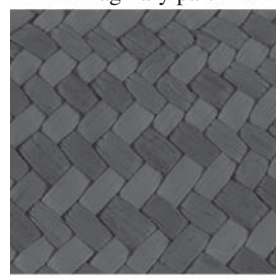
(Continued)

Example 3.110 (Continued)

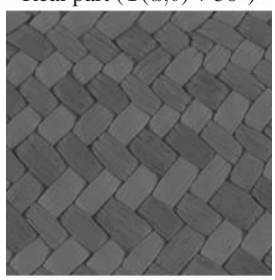
Original image

Imaginary part

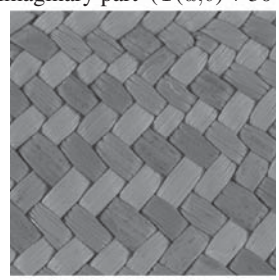
Real part $(\Phi(u,v)+30°)$



Imaginary part $(\Phi(u,v)+30°)$

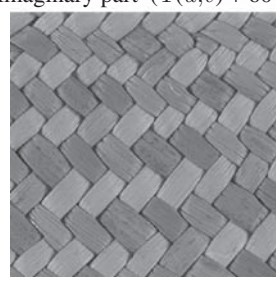


Real part $(\Phi(u,v)+60°)$



Imaginary part $(\Phi(u,v)+60°)$

Real part $(\Phi(u,v)+90°)$



Imaginary part $(\Phi(u,v)+90°)$

Figure 3.104 Real and imaginary parts of the inverse Fourier transform after adding 0°, 30°, 60° and 90° to the phase of the image.

The reconstruction we obtain after we add the constant to the phase of the DFT consists of a real $\tilde{F}_{\mathrm{R}}(m,n)$ *and an imaginary image* $\tilde{F}_{\mathrm{I}}(m,n)$*. From the above expression we see that:*

$$f(m,n) = \tilde{F}_{\mathrm{R}}(m,n)\cos\theta + \tilde{F}_{\mathrm{I}}(m,n)\sin\theta + \mathrm{j}\left(-\tilde{F}_{\mathrm{R}}(m,n)\sin\theta + \tilde{F}_{\mathrm{I}}(m,n)\cos\theta\right). \tag{3.343}$$

Since $f(m,n)$ is real, we deduce that:

$$\tilde{F}_{\mathrm{R}}(m,n)\cos\theta + \tilde{F}_{\mathrm{I}}(m,n)\sin\theta = f(m,n)$$
$$-\tilde{F}_{\mathrm{R}}(m,n)\sin\theta + \tilde{F}_{\mathrm{I}}(m,n)\cos\theta = 0. \tag{3.344}$$

These equations may be solved for $\tilde{F}_{\mathrm{R}}(m,n)$ and $\tilde{F}_{\mathrm{I}}(m,n)$:

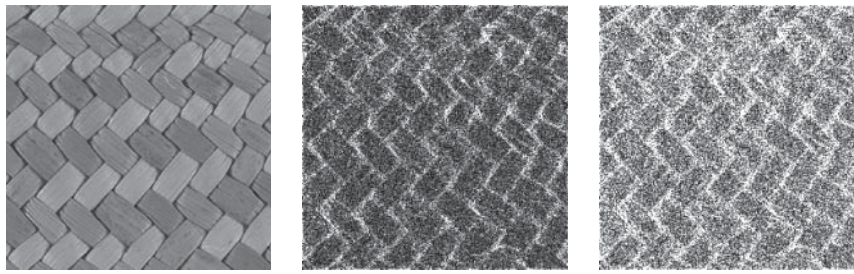
$$\tilde{F}_{\mathrm{R}}(m,n) = f(m,n)\cos\theta \qquad \tilde{F}_{\mathrm{I}}(m,n) = f(m,n)\sin\theta \tag{3.345}$$

Since θ is a constant, we see that the addition of a constant to the phase of the DFT of an image creates a complex image in the spatial domain with real and imaginary parts that are dimmed versions of the original image.

Example 3.111

In the case of Example 3.110 compute the mean grey value g_m of the image and set $M(u,v)_{\mathrm{new\_1}} = g_m$ for every pair of values (u,v). Use this $M(u,v)_{\mathrm{new\_1}}$ with the original values of $\Phi(u,v)$ in Equations (3.340) to recover the real and imaginary parts of the Fourier transform. Then take the inverse Fourier transform to obtain an image. Repeat this for $M(u,v)_{new\_2} = \min\{1.5g_m, 255\}$. Comment on the relationship of the recovered images with the original one and the role played by the fixed value of $M(u,v)$.

The results are shown in Figure 3.105. It can be seen that, although the magnitude information has been totally destroyed, the constructed images still have some similarity with the original one. Increasing the chosen value of $M(u,v)$ simply makes the reconstructed image brighter.



Original Image $M\ (u,v)_{\mathrm{new\_1}}$ $M\ (u,v)_{\mathrm{new\_2}}$

Figure 3.105 Image reconstruction using a fixed value for its Fourier transform magnitude.

Example 3.112

In the case of example 3.110 set $\Phi_{\text{new\_1}}(u,v) = 30°$ for every pair of values (u,v) and use the original values of $M(u,v)$ in Equations (3.340) to recover the real and imaginary parts of the Fourier transform. Then take the inverse Fourier transform to obtain an image. Repeat this for $\Phi_{\text{new\_2}}(u,v) = 90°$. Comment on the relationship of the recovered images with the original one.

The results are shown in Figure 3.106. Note that, although the original values of the magnitude have been used, there is no similarity between the reconstructed images and the original one.

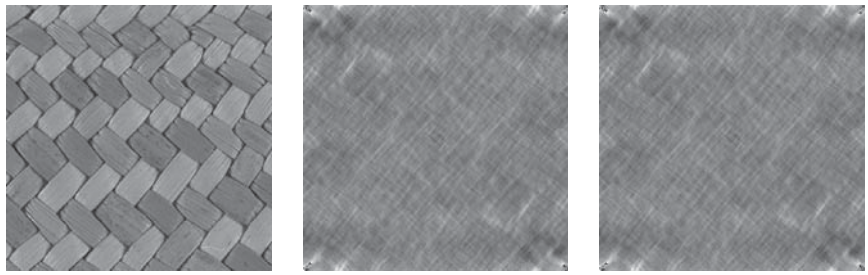


Figure 3.106 Image reconstruction using a fixed value for its Fourier transform phase. The grey values of the reconstructed images are proportional to the magnitude of the reconstructed complex values. For $\Phi_{\text{new\_2}}(u,v) = 90°$ the reconstructed image has only an imaginary part. For $\Phi_{\text{new\_1}}(u,v) = 30°$ the reconstructed image has real and imaginary parts, both of which look pretty similar to the magnitude that is displayed here.

Example 3.113

Consider the original images beans and Costas shown in Figure 3.90 and 1.1, respectively. Compute their Fourier transforms. Call the magnitude and phase of the transform $M_B(u,v)$, $M_C(u,v)$, and $\Phi_B(u,v)$ and $\Phi_C(u,v)$ for the beans and Costas images, respectively. Define

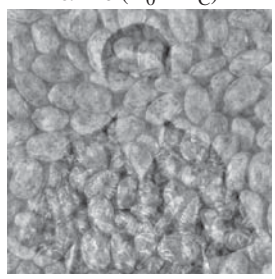
$$\Phi_0(u,v;\alpha) \equiv \alpha\Phi_B(u,v) + (1-\alpha)\Phi_C(u,v) \tag{3.346}$$

where $0 \leq \alpha \leq 1$. By using $M_B(u,v)$ and $\Phi_0(u,v;\alpha)$ for $\alpha = 0.75, 0.5, 0.25$ and 0, compute the real and imaginary parts of the Fourier transform and use the inverse Fourier transform to construct images. Comment on the resemblance these images have with the original images.

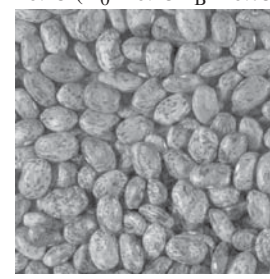
The results are shown in Figure 3.107. Although the magnitude used in all cases is from the beans texture, it is clear that α strongly influences the final result, that is, the appearance of the final result is dominated by the image the phase of that enters with the highest weight in (3.346).

$\alpha = 0$ ($\Phi_0 = \Phi_C$)

$\alpha = 0.25$ ($\Phi_0 = 0.25\Phi_B + 0.75\Phi_C$)



$\alpha = 0.5$ ($\Phi_0 = 0.5\Phi_B + 0.5\Phi_C$)



$\alpha = 0.75$ ($\Phi_0 = 0.75\Phi_B + 025\Phi_C$)

Figure 3.107 Inverse Fourier transform using $M_B(u, v)$ and $\Phi_0(u, v)$ for different values of α. Source: Maria Petrou. For $\alpha = 0$ the phase used in the inverse Fourier transform is from the Costas image and that is clearly visible here, although the magnitude is from the beans image. When the phase of both images enters in equal proportions ($\alpha = 0.5$), we have a transparency effect, as the two images exist one on top of the other.

Since the phase conveys more information for a pattern than its power spectrum, why dońt we use the phase to describe textures?

When we use a numerical quantity to describe something, we require the values of this quantity to change only slightly when the differences between two "somethings" is small. This is not the case for the phase values of a signal, because we express them by angles modulo 360°, creating strong discontinuities in the values near the edges of the range; something with phase 359° is not expected to be very different from something with phase 2°, and yet their phases differ by as much as 357°!

Is it possible to compute from the image phase a function the value of which changes only due to genuine image changes?

Yes. This process is called **phase unwrapping**. We can imagine that the phase we compute with the Fourier transform is the modulo 360° of a quantity that varies continuously. Figure 3.108 shows how this quantity takes sequential values in an image and how we just map it always on a circle. The process of phase unwrapping tries to recover the original quantity from the computed one, by adding 360° to the computed phase every time it makes a sudden jump.

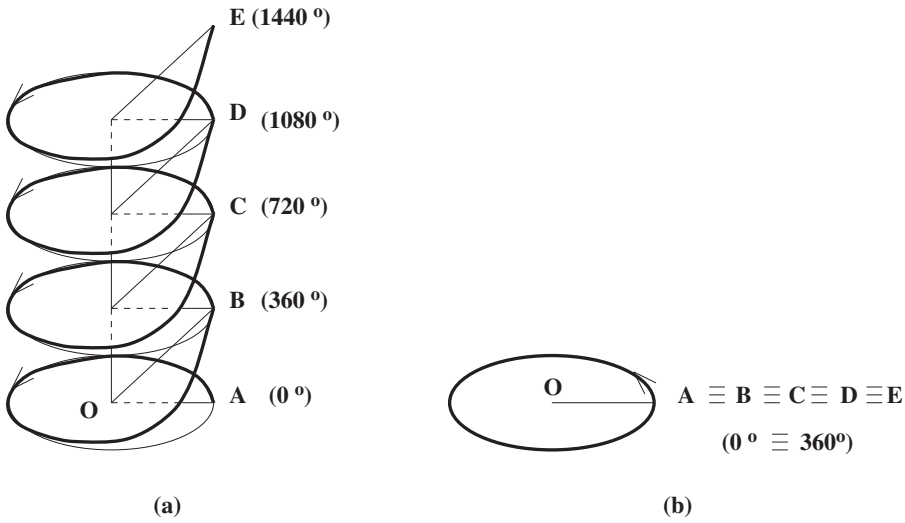


Figure 3.108 (a) Imagine that you start from point *A* and you move along the thick line measuring your orientation anti-clockwise with respect to reference direction *OA*. When the angle you measure becomes 360°, imagine that it carries on increasing by taking values around the circle at the second level, and so on. It is as if a cut has been made along line *OA* so that, when you reach point *A* after you have completed a circle, you can move along some other dimension of space, onto another circle, where a similar cut has been made to allow you to move to another circle and so on. (b) Here all these extra circles are shown collapsed on the original circle, so that when you move along the thick line you always retrace the same circle and your measuring apparatus is set back to 0 every time you pass through point *A*. In phase unwrapping we have situation (b) and we are trying to recover situation (a).

How do we perform phase unwrapping?

Let us consider a signal, the true phase of which is shown by the curve in the top panel of Figure 3.109. The black dots indicate the places where this continuous function is sampled. In reality we do not have these points. We have the points shown in the second panel of the same figure, because we compute the phase using the inverse tangent function which wraps it in the range $(-180°, 180°]$. So, our problem is to start from the points in the second panel and recover the points in the top panel. This may be achieved by differentiating the phase and then integrating it as demonstrated next. The third panel of the figure shows the first derivative of the sequence of points in the second panel:

$$\Delta\phi(n) \equiv \phi(n) - \phi(n-1) \qquad \text{for } n = 1, \ldots, N. \tag{3.347}$$

We can see that at the points where the phase was wrapped round this difference has high absolute value. To recover then the unwrapped phase $\Phi(n)$, we define first a sequence of corrections $c(n)$

$$c(n) = \begin{cases} 0 & \text{when } |\Delta\phi(n)| < T \text{ or when } n = 0 \\ -360 & \text{when } \Delta\phi(n) \geq +T \\ +360 & \text{when } \Delta\phi(n) \leq -T \end{cases} \tag{3.348}$$

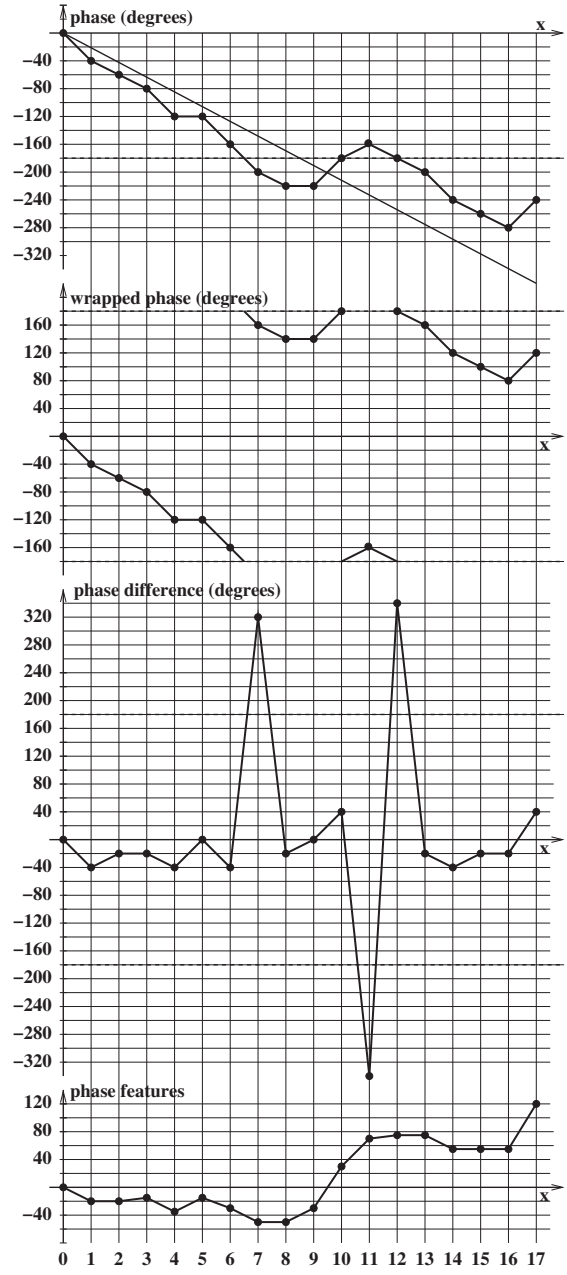


Figure 3.109 At the top the true phase of a signal. Below it, the phase we compute from the real and imaginary parts of the Fourier transform of the signal. Below, the first difference of the numbers plotted in the second panel. At the bottom, the sequence of numbers plotted from the unwrapped phase after the linear drift has been removed. This sequence is intrinsic to the signal and it may be used to characterise it.

where T is some positive threshold. Let us choose it for this example to be $T = 100$. We can easily see that for our example the correcting sequence $c(n)$ is:

$$0, 0, 0, 0, 0, 0, 0, -360, 0, 0, 0, +360, -360, 0, 0, 0, 0, 0.$$

Then the corrected sequence is recovered by performing integration from left to right and step by step along the differentiated sequence:

$$\Phi(0) = 0 + c(0)$$

$$\begin{aligned}\Phi(1) &= \Phi(0) + \Delta\phi(1) + c(1)\\ \Phi(2) &= \Phi(1) + \Delta\phi(2) + c(2)\\ \Phi(3) &= \Phi(2) + \Delta\phi(3) + c(3)\\ &\dots .\end{aligned} \tag{3.349}$$

We notice that at the end the created sequence will have a total added corrective term $C(N) = \sum_{n=0}^{N} c(n) = -360$. To identify the sequence that genuinely expresses signal changes, we remove from the result this drift by fitting a straight line between points $(0, 0)$ and $(N, C(N))$. The equation of such a line is:

$$\hat{\Phi}(n) = \frac{n}{N} C(N). \tag{3.350}$$

Then the sequence which is of interest to us is:

$$g(n) = \Phi(n) - \hat{\Phi}(n) = \Phi(n) - \frac{n}{N} C(N). \tag{3.351}$$

The drift line is the straight line plotted in the top panel of Figure 3.109. We then find the difference between this line and the unwrapped phase values. These are the values that characterise the signal, which for this particular example are:

$$0, -19, -18, -16, -35, -14, -33, -52, -51, -29, 32, 73, 74, 75, 56, 58, 59, 120.$$

This sequence is plotted in the bottom panel of Figure 3.109.

What are the drawbacks of the simple phase unwrapping algorithm?

The phase unwrapping algorithm described above has several drawbacks.

- If the signal becomes zero at some position, its phase is undefined and there is no mechanism in the algorithm allowing one to deal with that.
- The signal has to be sampled very densely, so that its genuine phase changes much more slowly than the changes due to the discontinuity jumps. This may allow one to choose a threshold T reliably.
- The algorithm propagates the integrated value by incrementing the value at the first sample. This means that any error committed at any stage influences all subsequent samples.

We shall see in Chapter 4 that instead of using such an algorithm to define phase features, we find other indirect ways to encode phase information (see Section 4.4).

Example 3.114

Consider the textures of Figure 3.1. Pick up two horizontal lines from each texture, and by treating them as 1D signals, phase unwrap them and produce for each line the characterising sequence of numbers. Plot the 16 sequences, with the two that came from the same texture one below the other.

The results are shown in Figure 3.110. We can see that lines that came from the same texture are reasonably similar to each other, and sufficiently distinct from the lines of other textures, with the only exception perhaps the lines that correspond to the image sand.

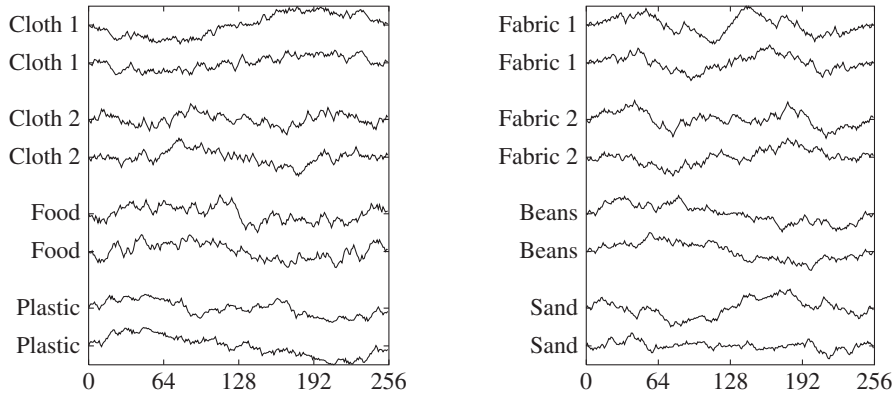


Figure 3.110 Phase-unwrapped sequences obtained for two horizontal lines of the texture images of Figure 3.1.

How do we perform phase-unwrapping in 2D?

In 2D we can work out the phase function from one point to another along many different paths. This is schematically demonstrated in Figure 3.111. All paths should yield the same phase for the final point, but of course this usually does not happen. At the end, the unwrapped phase of the final point may be computed as the average of the results of all paths followed, or by considering the phase that most paths yielded. In some schemes one may backtrack along paths to check whether the original point with the correct starting phase may be recovered, when starting from the estimated phase of the final point, and accordingly correct the estimation of the phase of the final point.

Example B3.115

Identify all the paths that lead from pixel A to pixel Q in the grid shown in Figure 3.111, when you are allowed to move only from left to right and from top to bottom. How many do you expect there to be?

Let us consider an $N \times M$ grid such that pixel A is at its top left corner and pixel Q at its bottom right. As we are allowed to move only from left to right and from top to bottom when we try to go from A to Q, it is obvious that any path we follow will consist of $N + M - 2$ pixels. Let us call

(Continued)

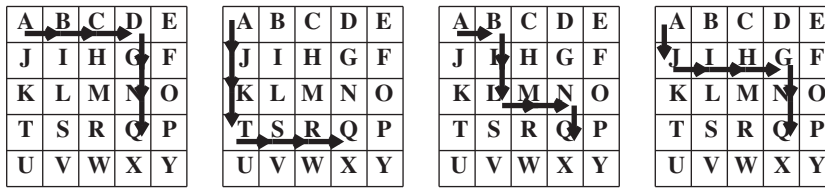


Figure 3.111 Four of the possible paths that lead from pixel A to pixel Q.

Example B3.115 (Continued)

the horizontal "steps" we perform when tracing such a path by letter R and the vertical steps by letter B. The sequence of steps that make up a path will be a string of $M-1$ Rs and $N-1$ Bs of total length $N+M-2$. The question then is: how many strings of total length $N+M-2$ can we make that contain $M-1$ Rs and $N-1$ Bs? This is a combinatorial problem and the answer is:

$$Number\_of\_paths = \frac{(N+M-2)!}{(N-1)!(M-1)!}. \tag{3.352}$$

Applying this formula to our problem, we have: $N = M = 4$. Then we work out that there are 20 possible paths from A to Q. These are shown in Figure 3.112

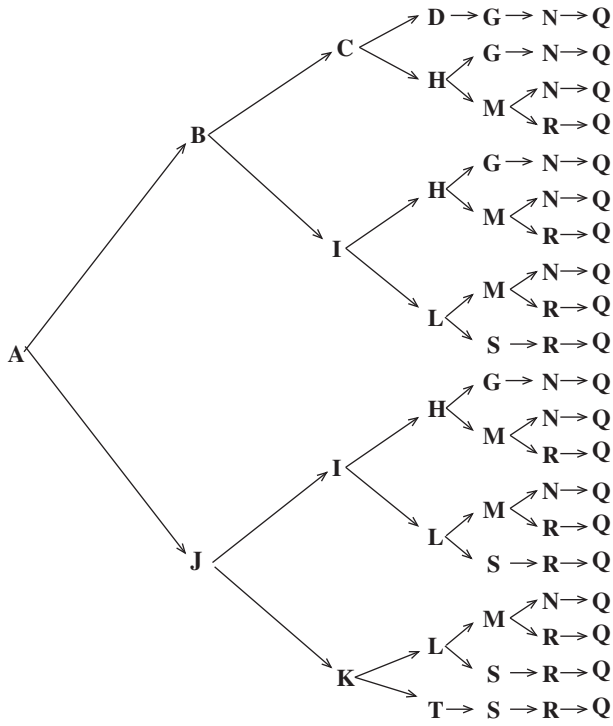


Figure 3.112 All paths that lead from A to Q when we can only move from left to right and from top to bottom in the grid shown in Figure 3.111.

The creation of binary slices by thresholding is equivalent to re-binning the pixels into wider histogram bins and marking the pixels that belong to each bin in a separate slice. This leads to the **slice transform** of the image.

What is the slice transform of an image

It is the representation of an image as a linear combination of the binary slices from which it consists. If in one slice we mark with 1 all pixels that have a particular grey value and with 0 all pixels

$$\begin{bmatrix}0&0&2\\0&1&2\\1&2&3\end{bmatrix} = 0\times\begin{bmatrix}1&1&0\\1&0&0\\0&0&0\end{bmatrix} + 1\times\begin{bmatrix}0&0&0\\0&1&0\\1&0&0\end{bmatrix} + 2\times\begin{bmatrix}0&0&1\\0&0&1\\0&1&0\end{bmatrix} + 3\times\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix}$$

Figure 3.113 A 2-bit image may be written as the linear superposition of its binary slices.

that do not have this value, the binary slices we create may be thought of as an orthogonal basis. The image then may be written as the sum of these binary slices, with each one multiplied with the corresponding grey value. This is shown schematically in Figure 3.113. See Box 3.23 for further details on the slice transform.

Box 3.23 The slice transform

Consider that we write the binary slices shown in Figure 3.113 as column vectors, one next to the other. Then we may re-write the expansion shown in Figure 3.113 as:

$$\begin{pmatrix}0\\0\\1\\0\\1\\2\\2\\2\\3\end{pmatrix} = \begin{pmatrix}1&0&0&0\\1&0&0&0\\0&1&0&0\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&1&0\\0&0&1&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}0\\1\\2\\3\end{pmatrix}. \tag{3.353}$$

In general, a column image $\mathbf{g}$, with NM elements, may be written as

$$\mathbf{g} = S\mathbf{h} \tag{3.354}$$

where S is an $NM \times G$ matrix, one column of that identifies with 1 the pixels with a particular grey value and with 0 all other pixels, and $\mathbf{h}$ is the list of the G grey values of the image, each one corresponding to one column of matrix S.

Now, let us imagine that we bin the grey values of the image into H different bins, and let us say that we consider that each bin corresponds to its middle grey value, rounded to the nearest integer. For example, if we have an 8-bit image and we use bins of width 10, the first bin with values in the range $[0, 9)$ will correspond to grey value 4, the second bin, with values in the range $[10, 19)$ will correspond to grey value 14, and so on. We may construct matrix S now so that each of its columns identifies the pixels that belong to a particular bin. Then the slice transform of the image approximates the image by representing it as the linear superposition of the binary slices that correspond to the histogram bins, each slice multiplied with the representative grey value of the bin:

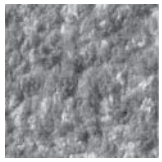
$$\mathbf{g} \simeq S\mathbf{h}. \tag{3.355}$$

Here now $\mathbf{h}$ is the column vector of the histogram bins representative grey values. For example, for the slices of Figure 3.2, the histogram bins are $[0, 30)$, $[31, 62)$, $[63, 94)$, $[95, 126)$, $[127, 158)$, $[159, 190)$, $[191, 222)$ and $[223, 255)$. The corresponding grey values may be chosen to be: 15, 46, 78, 110, 142, 174, 206 and 239, respectively. Figure 3.114a shows the

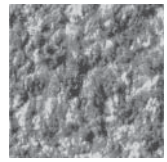
(Continued)

Box 3.23 (Continued)

approximate representation of the original image using these values in (3.355). The bins used in Figure 3.3 are [0, 76), [77, 91), [92, 102), [103, 122), [123, 134), [135, 151) and [152, 255]. The corresponding grey values of the bins may be selected to be: 38, 84, 97, 112, 128, 143 and 203, respectively. Figure 3.114b shows the approximate representation of the original image using these values in (3.355).



(a)



(b)

Figure 3.114 The approximate representation of the original image.

Example 3.116

The tone transform of an image is defined as a transform that takes the grey values of the image and maps them on different grey values, not necessarily in a linear way. The image of Figure 3.2a has been subject to the following tone transform:

$$
\begin{array}{lcl}
[0, 30) & \rightarrow & [220, 256) \\
[31, 62) & \rightarrow & [150, 179) \\
[63, 94) & \rightarrow & [120, 149) \\
[95, 126) & \rightarrow & [90, 119) \\
[127, 158) & \rightarrow & [60, 89) \\
[159, 190) & \rightarrow & [30, 59) \\
[191, 222) & \rightarrow & [0, 29) \\
[223, 255) & \rightarrow & [180, 219)
\end{array}
$$

Show what the image looks like now, using the approximate slice transform.

All we have to do is to use Equation (3.355), with the same S matrix as we had for the original histogram, but now replacing vector $\mathbf{h}$ *with the new vector of representative grey values of the bins:* $(238, 165, 135, 105, 75, 45, 15, 200)^T$. *The result is shown in Figure 3.115.*

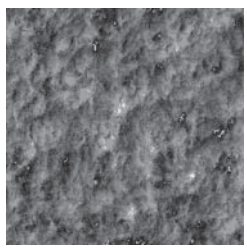


Figure 3.115 The approximate representation of the original image. Source: Maria Petrou.

3.6 Markov Random Fields

What is a Markov random field?

A **Markov random field** (**MRF** for short) is a **random field** that possesses the **Markovian property**: the value of a pixel depends **directly** only on the values of the neighbouring pixels and on no other pixel.

What is a random field?

A random field is created when one performs a random experiment at each location and assigns the outcome of the random experiment to that location.

Which are the neighbouring pixels of a pixel?

This depends on how we define the neighbourhood of a pixel. Figure 3.116 shows some commonly used neighbourhoods for Markov random fields. These neighbourhoods imply spatial proximity of the pixels that influence the value at the central location. This, however, is not necessary. Figure 3.117 shows some more complicated Markov neighbourhoods.

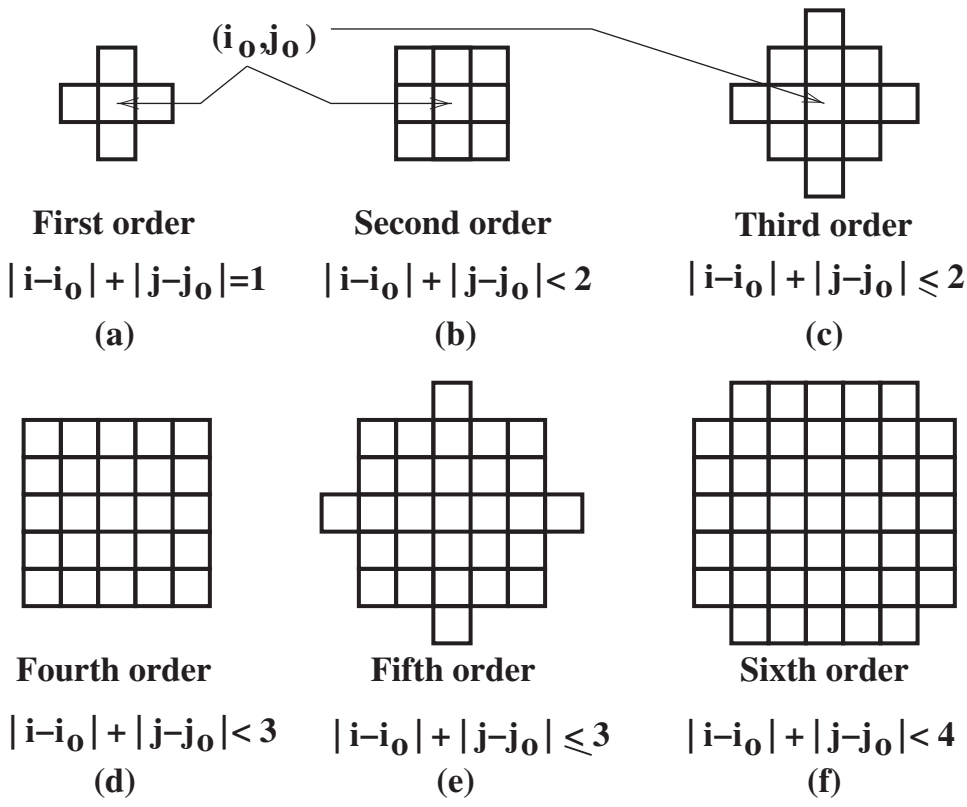


Figure 3.116 Some Markov neighbourhoods and the conditions that have to be satisfied by the coordinates of a pixel (i, j) in order to belong to the neighbourhood of the central pixel (i_0, j_0). The most commonly used ones are those of the first and second order.

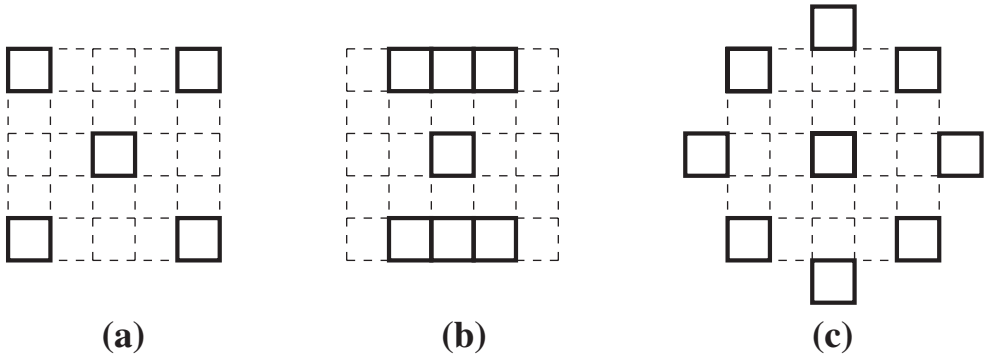


Figure 3.117 Some more exotic Markov neighbourhoods, where the neighbours that influence the value of the pixel at the centre are not spatially adjacent to it.

How can we create a Markov random field?

Consider an empty grid, like the one shown in Figure 3.118. Suppose that we decide to fill in the missing numbers by throwing a coin for each pixel 255 times and counting the number of heads. The "grey" value we assign to each empty grid cell is the number of heads we count for that pixel position. By doing this we create a **random field**.

The probability $p(k)$ of getting k heads in n throws is given by the binomial distribution

$$p(k) = \binom{n}{k} \theta^k (1-\theta)^{n-k} \tag{3.356}$$

where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3.357}$$

and θ is the probability of getting a head in a single throw. For an unbiased coin, $\theta = 1/2$.

Let us imagine now that, having done this process once and filled in all the numbers, we change the coin, so that it becomes biased. We bias it in such a way that when we decide to choose the value assigned to a certain grid location, we take into consideration the values of the neighbouring locations. Let us say, for example, that we want the coin to be such that

$$\frac{p(\text{head}|\text{given\_values\_of\_nbrs})}{p(\text{tail}|\text{given\_values\_of\_nbrs})} = \mathrm{e}^s \tag{3.358}$$

where s is a function of the values of the neighbours. We can work out now the bias of the coin by observing that

$p(\text{head}|\text{given\_values\_of\_nbrs}) = \theta$ and, therefore,

$p(\text{tail}|\text{given\_values\_of\_nbrs}) = 1-\theta$:

$$\frac{\theta}{1-\theta} = \mathrm{e}^s \Rightarrow \theta = \frac{\mathrm{e}^s}{1+\mathrm{e}^s}. \tag{3.359}$$

Using such a coin to decide the new values at the grid positions allows us to create an example of a **Markov random field**.

For the particular example considered here, if we substitute θ from (3.359) into (3.356), we obtain the local conditional probability density function for a particular grey value k to arise, given the values of the neighbouring pixels, as:

$$p(k|\text{given\_values\_of\_nbrs}) = \binom{n}{k} \frac{\mathrm{e}^{ks}}{(1+\mathrm{e}^s)^n}. \tag{3.360}$$

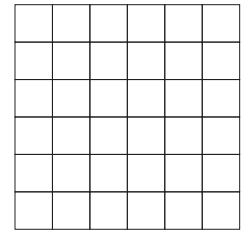


Figure 3.118 An empty grid. We have to choose grey values for these pixels to create first a random field, and from it a Markov random field.

As the outcomes of the random experiment we performed to decide the value of each pixel were governed by the binomial distribution, this model is known as an **auto-binomial MRF** model.

How can we use Markov random fields to characterise textures?

This is a parametric approach where we model the texture as a Markov random field. We have to choose first the type of neighbourhood we shall adopt and then the form of function s in Equation (3.360). Once these are fixed, the parameters on which function s depends characterise the texture. These are called **Markov parameters**.

Let us say, for example, that we adopt the neighbourhood structure of Figure 3.116a and assume that function s has the form

$$s = a(g_l + g_r) + b(g_t + g_b) \tag{3.361}$$

where g_l, g_r, g_t and g_b stand for the grey value of the left, right, top and bottom neighbour of the central pixel, respectively. Parameters a and b in (3.361) are the Markov parameters.

Such models also offer the option to create textures, so they are appropriate for **texture synthesis by analysis**, a technique used in computer graphics.

What is texture synthesis by analysis?

It is a technique by which the parameters of a real image texture are inferred and used to create textures to render images. This technique may also be used to infer how uniquely the estimated parameters characterise the texture.

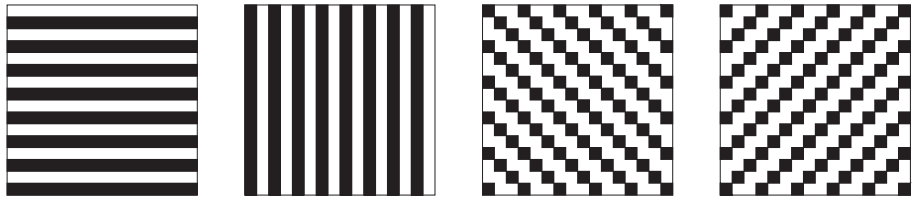
Example 3.117

Show that if a Markov parameter is positive, it encourages a pixel to have a value similar to its neighbours, and if it is negative, it encourages a pixel to have a value dissimilar to its neighbours.

Consider Equations (3.359) and (3.361). For simplicity, assume that in (3.361) $a = b$. Let us also consider that the neighbours have high values. Then if a is positive, e^s is a very high number, and we can see from (3.359) that $\theta \to 1$, which implies that we bias our coin so that it has a very high probability of producing a head in a single throw, thus a very high probability of producing a high number of heads in 255 throws, thus a very high probability of producing a grey value similar to those of the neighbouring pixels. On the contrary, if $a < 0$, e^s would tend to zero for large values of the neighbours, so θ would tend to zero, and our experiment would tend to produce a small number of heads in 255 throws, so the central pixel would most likely have a small grey value, in contrast to its neighbouring pixels, which have been assumed to have high values.

Example 3.118

Infer the qualitative relationships for the Markov parameters that might characterise the textures of Figure 3.119, assuming that the model applied is the one given by Equation (3.360), where s is a linear combination of the values of the eight neighbours of the pixel, identified by the second-order Markov neighbourhood of Figure 3.116b. Assume that in each case each stripe is 1 pixel wide.



(a) (b) (c) (d)

Figure 3.119 Four directional textures.

If we identify each neighbour as left, right, top, bottom, etc., with the corresponding parameters arranged as shown in Figure 3.120, we can easily deduce that for each texture in Figure 3.119 we must have:

Image 3.119a: $a_l = a_r > 0 \quad and \quad a_t = a_b = a_{tl} = a_{br} = a_{tr} = a_{bl} < 0$

Image 3.119b: $a_t = a_b > 0 \quad and \quad a_l = a_r = a_{tl} = a_{br} = a_{tr} = a_{bl} < 0$

Image 3.119c: $a_{tl} = a_{br} > 0 \quad and \quad a_{tr} = a_{bl} = a_t = a_b = a_l = a_r < 0$

Image 3.119d: $a_{tr} = a_{bl} > 0 \quad and \quad a_{tl} = a_{br} = a_t = a_b = a_l = a_r < 0$

	a_{tl}	a_t	a_{tr}		
	a_l	?	a_r		
	a_{bl}	a_b	a_{br}		

Figure 3.120 The parameters and pixels that influence the value of the central pixel in the case of a second-order Markov neighbourhood.

Example 3.119

Infer the missing value of the empty pixel in the images of Figure 3.121.

1	0	1	0	1	0	1
0	1	0	1	0	1	0
1	0	1	0		0	1
0	1	0	1	0	1	0
1	0	1	0	1	0	1
0	1	0	1	0	1	0
1	0	1	0	1	0	1

(a)

1	0	1	0	1	0	1
0	1	0	1	0	1	0
2	0	1	0		0	1
0	2	0	2	0	1	0
2	0	1	0	2	0	1
0	1	0	2	0	2	0
2	0	1	0	1	0	1

(b)

Figure 3.121 Context allows us to determine the missing values.

The first image appears to contain a deterministic texture, where four 0s in a cross-like shape imply the value of 1 in the middle. So, in 3.121a the missing value is 1. In 3.121b, we see that four 0s in a cross-like shape imply the value of 1 in 6 out of 10 times, and they imply the value of 2 in 4 out of 10 times. So, for the second texture we may say that the neighbours imply that the missing value is 1 with probability 0.6 and 2 with probability 0.4.

How can we apply the Markov model to create textures?

Applying directly formula (3.360) to choose the grey values of the pixels is not practicable: the formula contains very large numbers which will easily overflow any computer, particularly if we want to have grey values in the range [0, 255]. We shall see later in this chapter how we create textures in practice. Here, however, in order to demonstrate the formula we shall apply it for simplified cases where we consider grids of size 64×64 and pixels that take only nine distinct grey values, i.e. they take integer values in the range $[0, 8]$. We shall create textures using the neighbourhoods shown in Figures 3.116a,b,c and 3.117 with the Markov model given by Equation (3.360) with function s being a linear combination of the values of the neighbours of the pixel. The parameter value with which the grey value of each neighbour will have to be multiplied is shown in Figure 3.122. We shall assume that each image is repeated ad infinitum in all directions, so that even the border pixels have well defined neighbourhoods.

We start by creating a random field of size 64×64 where the grey value at each position is drawn from the probability density function given by (3.356) with $n = 8$ and $\theta = 0.5$:

$$p(k) = \frac{8!}{k!(8-k)!}\frac{1}{2^8} = \frac{157.5}{k!(8-k)!}. \tag{3.362}$$

We have to use the following algorithm that uses the process described in Box 2.2.

Step 1: Create a look-up table with the values of k_1 and $\mathcal{P}(k \leq k_1)$:

$$P(k_1) \equiv \mathcal{P}(k \leq k_1) = \sum_{k=0}^{k_1} p(k) = 157.5 \sum_{k=0}^{k_1} \frac{1}{k!(8-k)!}. \tag{3.363}$$

Step 2: Draw random numbers uniformly distributed in the range $[0, 1]$ and, treating them as values of $P(k_1)$, use this look-up table to identify the corresponding number k_1 which is thus drawn according to the probability density function (3.362). Draw 64×64 numbers, one for each pixel.

Figure 3.123 shows a random field created this way. To visualise this field, we multiplied the grey value of each pixel with 255/8 and rounded it to the nearest integer. However, for the subsequent stages of the process, we kept the numbers in the range $[0, 8]$ to avoid overflows.

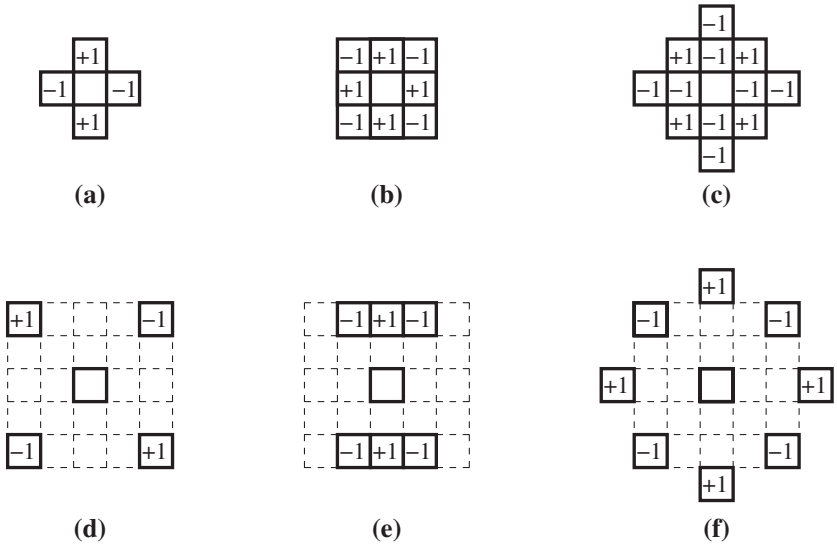


Figure 3.122 Some Markov neighbourhoods. The numbers indicate the Markov parameters that determine the way the corresponding pixels influence the value of the pixel in the middle.

Figure 3.123 A random field created using a binomial distribution with $n = 8$ and $\theta = 0.5$. Source: Maria Petrou.

From this random field we want to create a texture with the neighbourhood structure and the Markov parameters shown in 3.122a. This neighbourhood structure implies that for pixel (i, j), function s has the form:

$$s = g_{i,j-1} + g_{i,j+1} - g_{i-1,j} - g_{i+1,j} \tag{3.364}$$

where g_{ij} is the grey value of pixel (i, j).

Step 1: Create a random field.
Step 2: Visit each pixel in turn and compute function s for it, using the values its neighbours have in the random field created in the previous step.
Step 3: From the value of s compute the value of θ, using Equation (3.359).
Step 4: Compute the form of the probability density function appropriate for this pixel.
Step 5: Create the look-up table for this probability density function.
Step 6: Draw the new value for this pixel, using uniformly distributed numbers and the look up table.

We have to be very careful, however, which pixels we update in one go. If we update all pixels simultaneously, we may, for example, update the value of a pixel to make it more similar to its

horizontal neighbours, but then we update the values of the horizontal neighbours according to their own neighbours, and as a result we may destroy any similarity we might have achieved in the previous update. It is appropriate, therefore, to update in one pass only the pixels that are *not* neighbours of each other, and leave the remaining pixels with their old values. In the next pass, we update the next lot of pixels that are *not* neighbours of each other, and so on, until all sets of pixels have been assigned updated values. Then we start again updating once more all pixels, in waves of sets of pixels that are not neighbours of each other. The division of an image into sets of pixels that are not neighbours of each other is called **coding** or **colouring**. The coding of an image depends on the neighbourhood structure we have adopted. Figure 3.124 shows the codings appropriate for the example neighbourhoods of Figures 3.116a,b,c and 3.117 and for an image of size 16×16. Pixels that belong to the same coding are marked with the same symbol and they may have their values updated simultaneously.

Figure 3.125 shows the results obtained for the neighbourhoods and values shown in Figure 3.122, when all pixels in the image are updated simultaneously. Figure 3.126 shows the results obtained for the same neighbourhoods and values of the Markov parameters when the pixels are updated in sets of pixels with non-overlapping neighbourhoods. In the case of neighbourhood 3.122a, for example, we update first half of the pixels, those marked by an open circle in the top left panel of Figure 3.124, and then we update the remaining pixels in the next pass.

Can we apply the method discussed in the previous section to create images with 256 grey levels?

It is known that a binomial distribution may be approximated with a Gaussian if n is very large and $n\theta(1-\theta)$ is also much larger than 1. Then Equation (3.356) may be replaced with:

$$p(k) \simeq \frac{1}{\sqrt{2\pi n\theta(1-\theta)}} e^{-\frac{(k-n\theta)^2}{2n\theta(1-\theta)}}. \tag{3.365}$$

This approximation effectively means that drawing random numbers according to the probability density function (3.356) may be replaced by drawing random number from a Gaussian distribution with mean $n\theta$ and standard deviation $\sqrt{n\theta(1-\theta)}$, provided that n and $n\theta(1-\theta)$ are much larger than 1.

If we allow 256 different grey levels in the image, we can create a random field using this approximation with $n = 255$ and $\theta = 0.5$. Such a field is shown in Figure 3.127. Naturally, since $\theta = 0.5$, the grey level values of this image cluster around the value 128, and so this image has very little contrast.

Figure 3.128 shows the images we obtain after 1, 5, 20 and 50 iterations of the image, using the Gaussian approximation with 256 grey values and updating at each pass only the pixels that do not influence each other. (An iteration consists of as many passes as are necessary for all pixels to have their values updated.)

The images in Figure 3.128 should be compared with those of Figure 3.126. We see that, although we started from a totally uncorrelated field, different patterns are gradually emerging according to the model we impose on it, and they are similar with those of Figure 3.126, except in the cases that correspond to the neighbourhoods 3.122c and 3.122e. We also observe that, although we allowed 256 grey levels here, the patterns we get appear almost binary. If we see the form function s takes (e.g. Equation (3.364)), we see that for grey values in the range $[0, 255]$, s could be a very high number. In fact, s can be positive or negative, with a very high absolute value, for a large range of grey values of the neighbours of a pixel. Once s becomes a large positive number, θ is almost 1, and

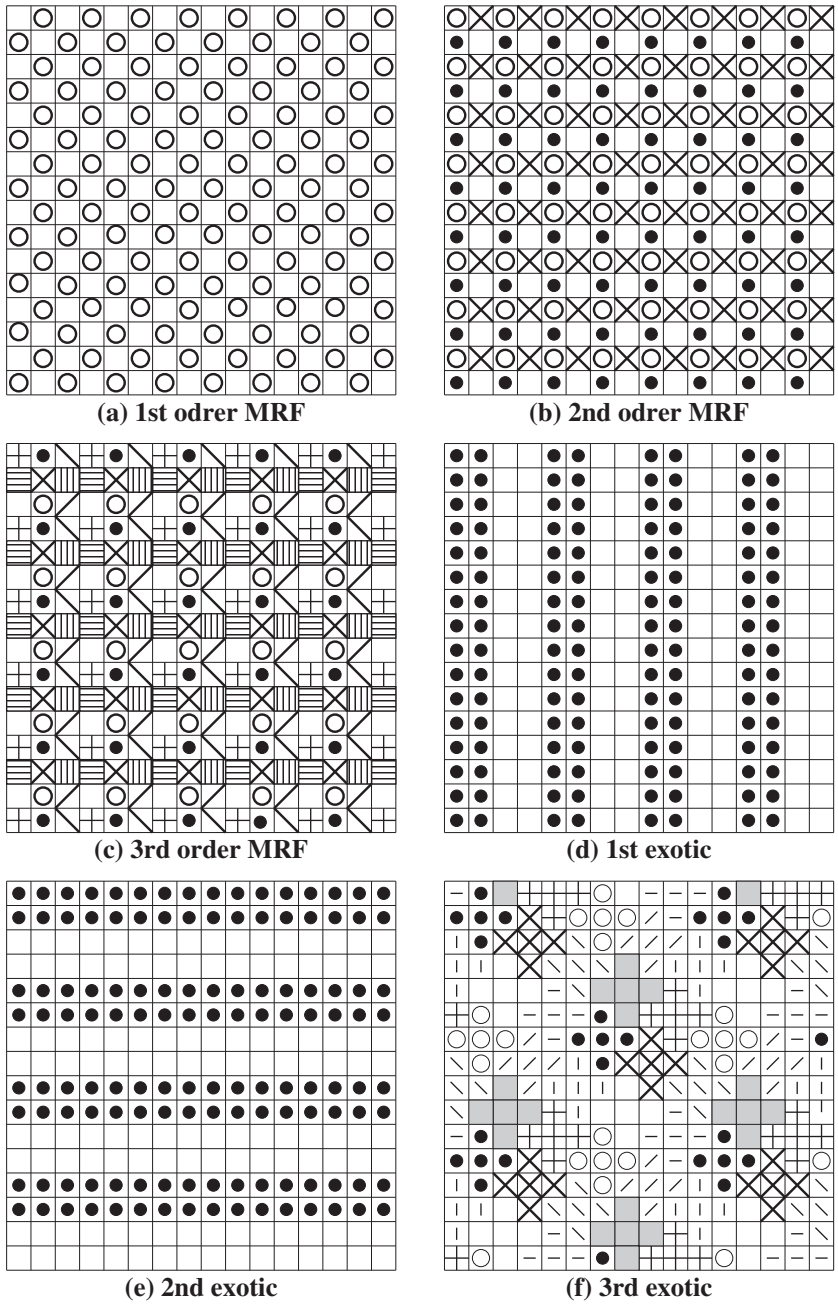


Figure 3.124 Codings of a 16×16 image when the Markov neighbourhoods of figures 3.116a,b,c and 3.117 are adopted, from left to right and top to bottom, respectively. In each case, pixels flagged by the same symbol may have their values updated together. Note that a coding is not necessarily unique for a particular neighbourhood structure. For example, coding (e) may also be used for neighbourhood 3.117a instead of coding (d).

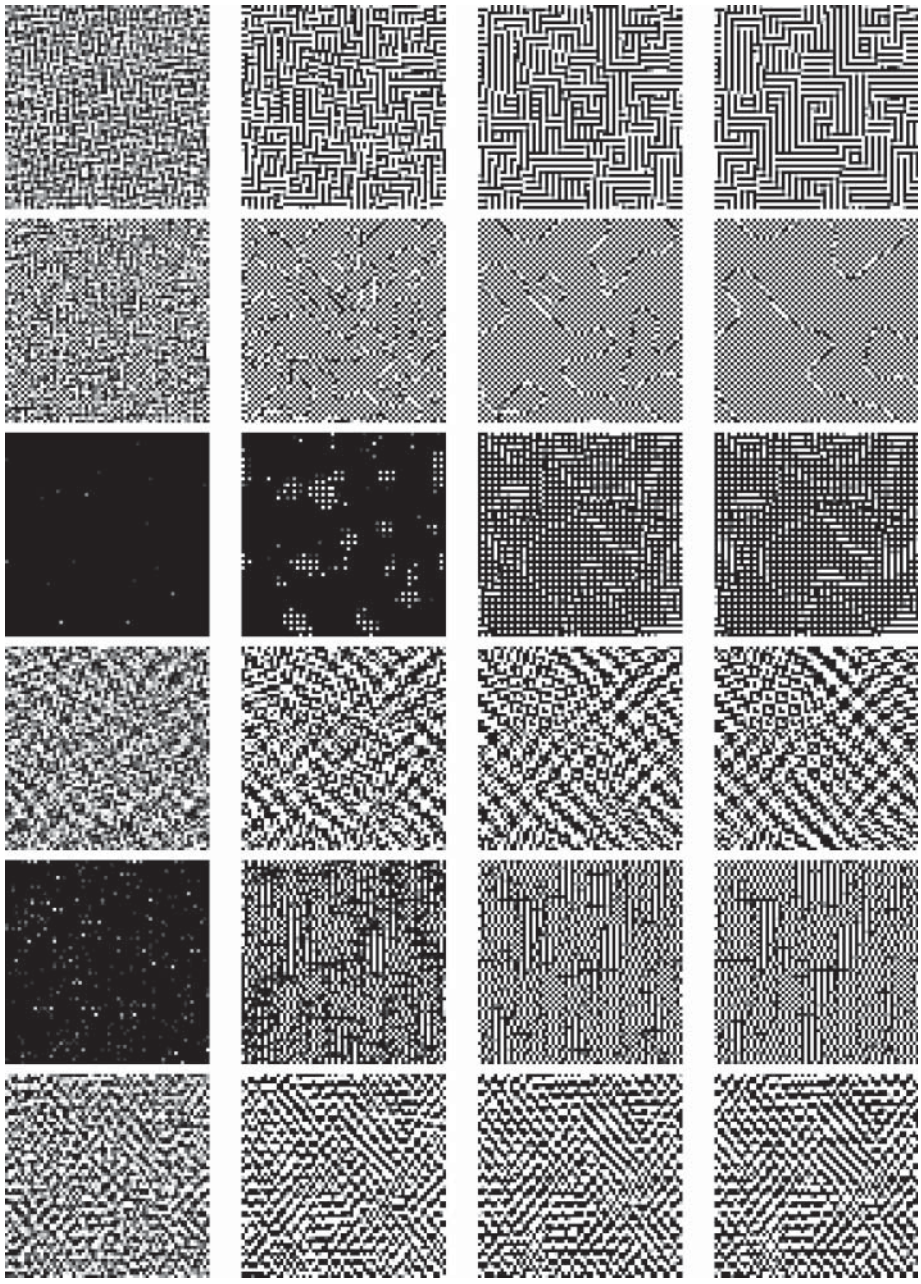


Figure 3.125 Creating textures using the Markov model with a binomial distribution, for the neighbourhoods defined in Figure 3.122 used sequentially from top to bottom, respectively. In all cases the starting configuration was the one shown in Figure 3.123. These results are **incorrect** as all pixels were updated simultaneously at each iteration, and the updates of pixels which influence each other often cancel each other out and prevent the target pattern from emerging. Source: Maria Petrou.

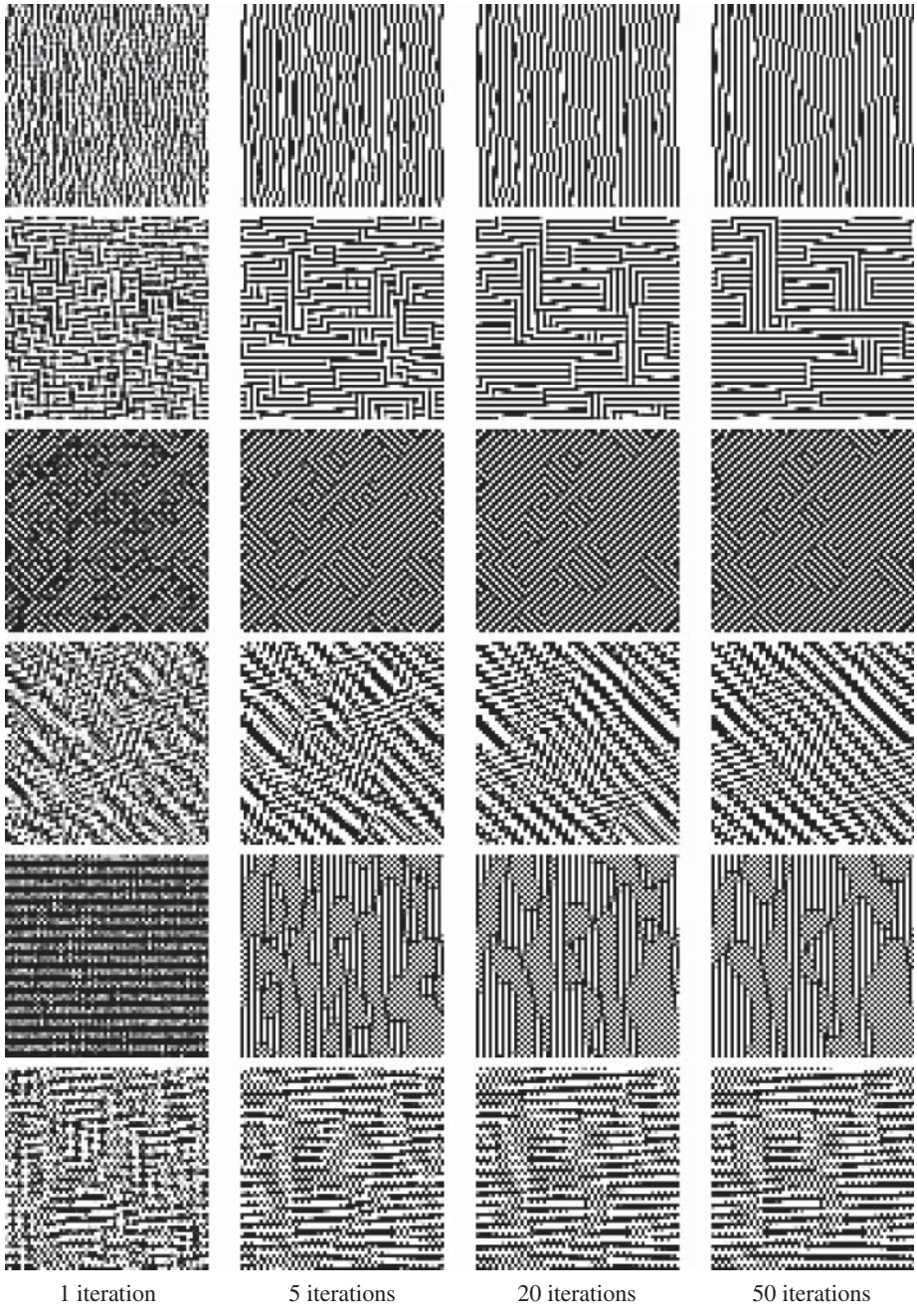


Figure 3.126 Creating textures using the Markov model with a binomial distribution for the neighbourhoods defined in Figure 3.122 used sequentially from top to bottom, respectively. Source: Maria Petrou. In all cases the starting configuration was the random field shown in 3.123. These results are **correct** and as expected. At each pass only pixels that do not influence each other had their values updated.

Figure 3.127 A random field created using the approximation of a binomial distribution with a Gaussian distribution with $n = 255$ and $\theta = 0.5$. Source: Maria Petrou.

once s becomes a large negative number, θ is almost 0. This means that the Gaussian approximation we use breaks down, as the condition that $n\theta(1-\theta)$ must be large is violated. It also means that the "coin" we use to conduct our random experiment is very easily biased towards producing almost always heads, or towards producing almost always tails. This explains the binary nature of the constructed images.

The two neighbourhoods that produce the most unexpected results of all are the only two in Figure 3.122 where the positive and negative weights do not sum up to 0. This implies that, right from the beginning, the value of s is a very high positive or negative number, since it is the result of adding, for example, four numbers in the range [0, 255] and subtracting from them eight numbers in the same range, for the case of neighbourhood 3.122c. Such numbers are bound to polarise e^s to be close to 0. The problem arises from the use of e as basis of the exponential model adopted. If the basis used were a number close to 1, it would not be easy to polarise θ to be either 1 or 0. Say we use as basis a number α. We can see that α^s may be written as $e^{s \ln \alpha}$, where $\ln \alpha$ could be considered as a scale factor. So, the same effect may be achieved by using integer numbers not in the range [0, 255], but integer numbers clustered closer to 1. This interpretation allows us to move away from the "flipping a coin" auto-binomial model we used to exemplify Markov random fields.

Another remedy would be to make sure that the positive and negative influences a pixel accepts are more balanced, by replacing neighbourhoods 3.122c and 3.122e with those shown in Figure 3.129. The images constructed with these Markov parameters are shown in Figure 3.130. The images constructed with the same Markov parameters but with basis $\alpha = 1.008$ (instead of $e \simeq 2.71828$) are shown in Figure 3.131. We can see that these images look much more similar to the images in 3.126.

When our problem is such that it requires the use of approximation (3.365), it becomes evident that the auto-binomial model is no longer adequate. Then it is better to use instead the **auto-normal Markov random field** model.

What is the auto-normal Markov random field model?

This is the model which assumes that: the value of each pixel is the outcome of a random experiment (thus the use of term "random field"); the samples of the outputs of the random experiment performed to decide each pixel value are drawn from a Gaussian distribution (thus the use of term "normal"); the parameters of this Gaussian are functions of the values of the neighbouring pixels (thus the term "Markov"). This model is also known as the **Gaussian Markov random field**.

A simple version of such a model assumes that the standard deviation of the Gaussian is constant and independent of the values of the neighbours. So, the probability density function from which

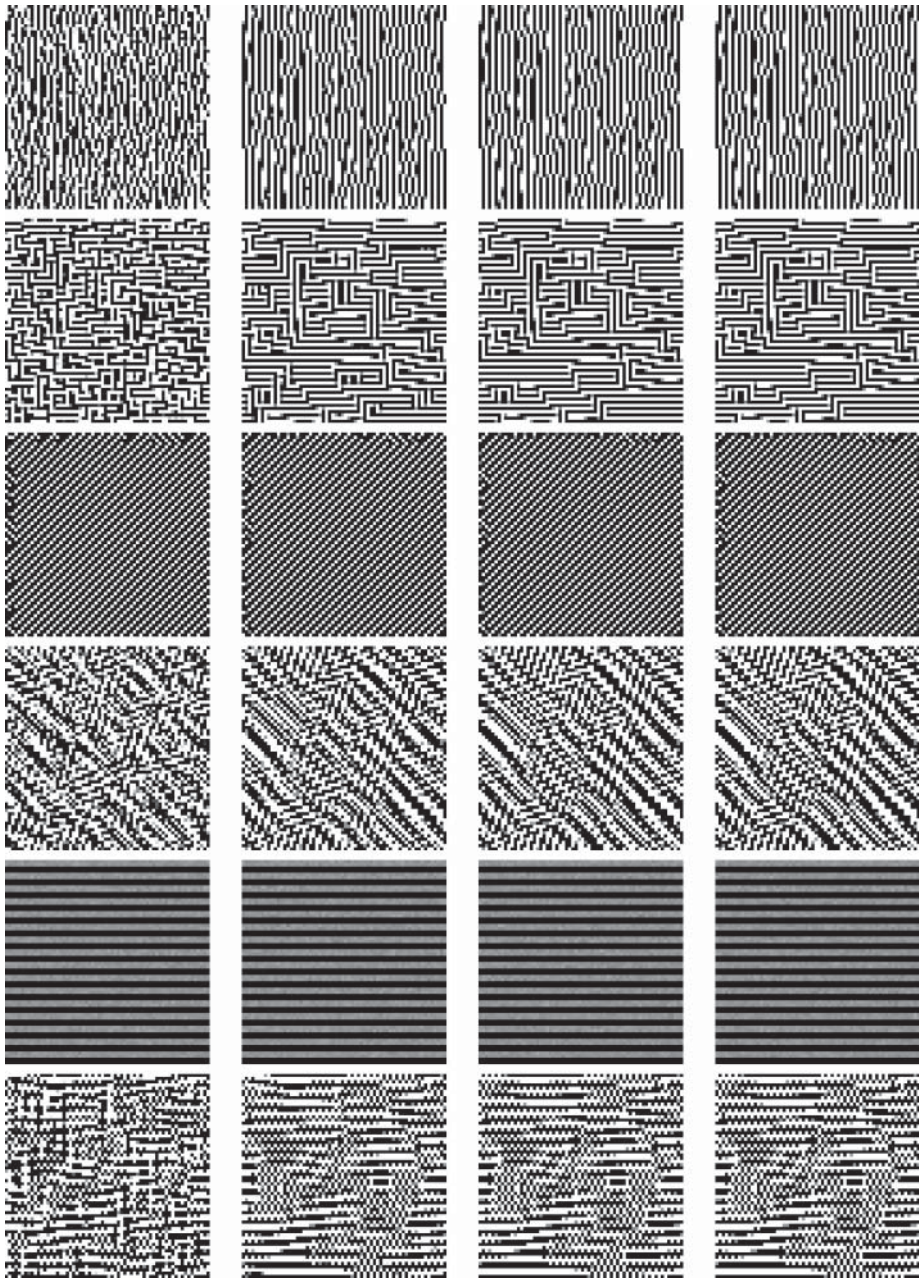


Figure 3.128 Creating textures using the Markov model and approximating the binomial distribution with a Gaussian, for the neighbourhoods defined in Figure 3.122 used sequentially from top to bottom, respectively. Source: Maria Petrou. In all cases the starting configuration was the random field shown in 3.127. For the third and fifth neighbourhood models we note that the created images immediately freeze in configurations that are clearly wrong.

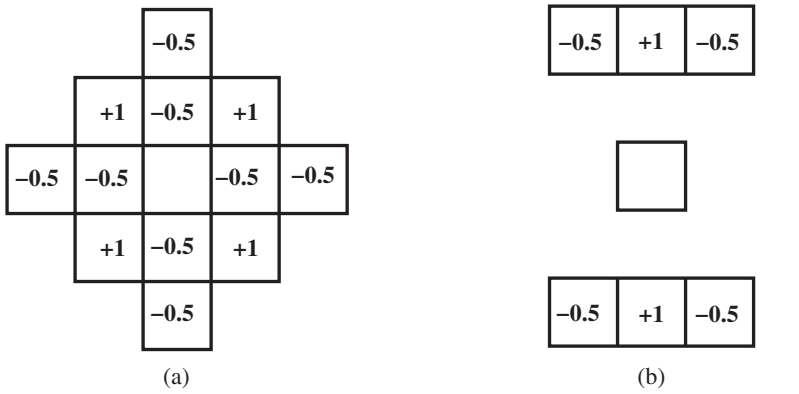


Figure 3.129 Two neighbourhoods with Markov parameters that retain the balance between positive and negative influences to the central pixel.

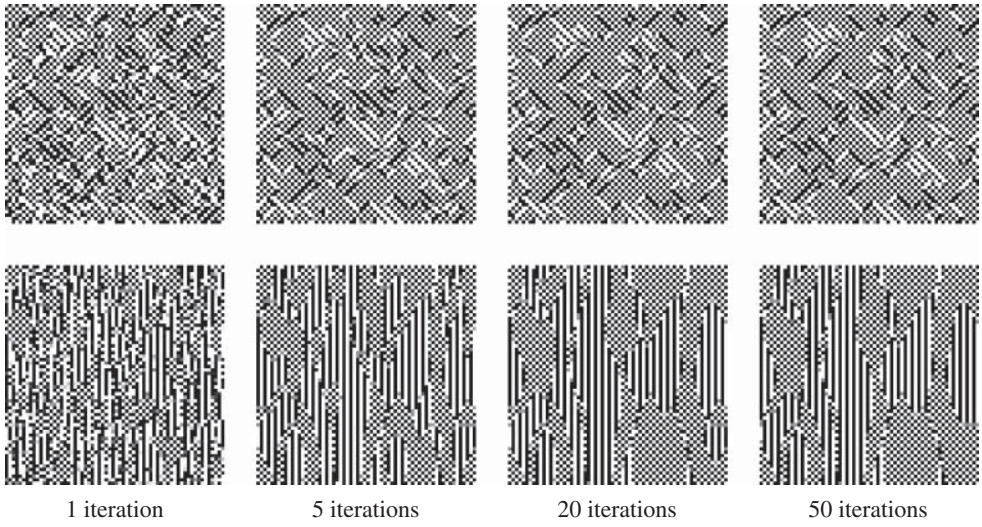


Figure 3.130 Creating textures using the Markov model and approximating the binomial distribution with a Gaussian, for the neighbourhoods defined in Figure 3.129, used sequentially from top to bottom, respectively. Source: Maria Petrou. In both cases the starting configuration was the one shown in Figure 3.127.

the new values of a pixel are drawn has the form

$$p(g_{ij}|g_{i'j'},(i',j')\in N_{ij})=\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{\left(g_{ij}-\sum_{l=1}^{L}a_l g_{i'j';l}\right)^2}{2\sigma^2}\right\} \tag{3.366}$$

where $p(g_{ij}|g_{i'j'},(i',j')\in N_{ij})$ is the probability of pixel (i,j) having grey value g_{ij}, given the values of its neighbours, L is the total number of pixels in the neighbourhood N_{ij} of pixel (i,j) which influence its value, a_l is the parameter with which a neighbour influences the value of (i,j), and $g_{i'j';l}$ is the value of the pixel at the corresponding position. If we assume that pixels in symmetric positions about pixel (i,j) have identical parameters, then instead of $g_{i'j';l}$ being the grey value of a single pixel, it will be the sum of the values of the two pixels that are in symmetric positions about (i,j) and that

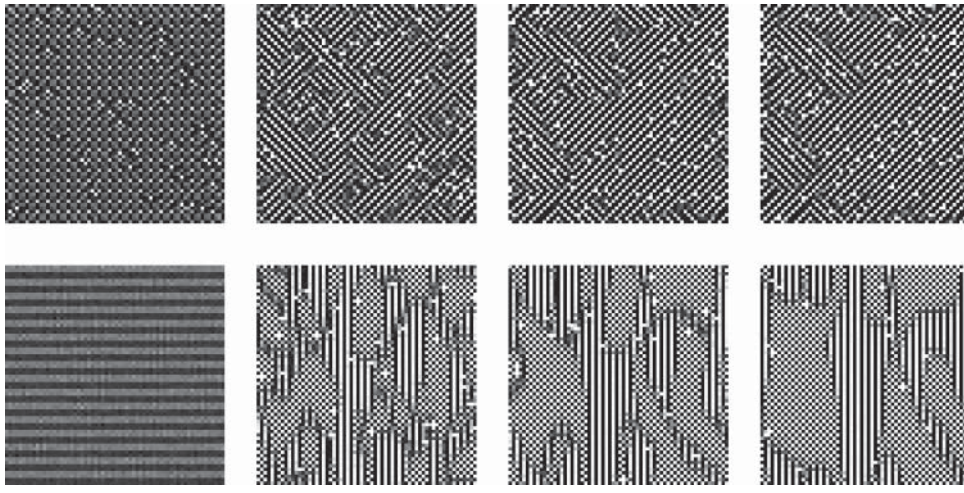


Figure 3.131 Creating textures using the Markov model and approximating the binomial distribution with a Gaussian, for the neighbourhood defined in Figures 3.122c and 3.122e, from top to bottom, respectively, but changing the basis of the exponential to $\alpha = 1.008$. In both cases the starting configuration was the random field shown in 3.127. Source: Maria Petrou.

influence the value of (i, j) with identical parameters. Then L is not the total number of pixels in the neighbourhood of (i, j), but rather the total number of **different** Markov parameters we have. In addition, we have parameter σ, which is also a model parameter.

Figure 3.132 shows some textures created by using this model on the random field shown in Figure 3.127 with the neighbourhood structures of Figures 3.116a,b,c and 3.117, and Markov parameters shown in Figures 3.122a,b,d,f and 3.129. The value of σ was chosen to be 8, which corresponds to a variance of 64 which is 256×0.25, i.e. the variance of the Gaussian approximation of the binomial model for $n = 256$ and $\theta = 0.5$. Of course, we again use the codings of Figure 3.124. The results shown in Figure 3.132 should be compared with those shown in Figures 3.128 and 3.130.

We note that the realisation of the model with neighbourhood structure 3.122b is rather different from that shown in 3.128, in spite of the balance of the negative and positive influences a pixel receives from its neighbours. This is because this model simultaneously encourages vertical and horizontal lines, and, depending on the starting configuration, we expect to obtain sometimes an image with vertical lines, sometimes with horizontal lines, and sometimes with patches of vertical and horizontal lines. When such self-contradictory models are used, the outcome is not always predictable and it is possible to get different results by changing, for example, the order in which the different sets of coded pixels are used.

Note that the difference with the auto-binomial model is that the binomial distribution returns discrete values, while the normal distribution returns continuous values. The approximation that led from the auto-binomial model to the auto-normal model becomes possible when the possible outcomes of the associated random experiment become so many that one may assume that the corresponding random variable becomes a continuous variable. Thus, the binomial distribution is no longer appropriate, while the Gaussian becomes relevant. Of course, we know that grey values will always be discrete rather than continuous, but we tacitly agree to round the values drawn from the normal distribution to the nearest integer in order to convert them to grey values.

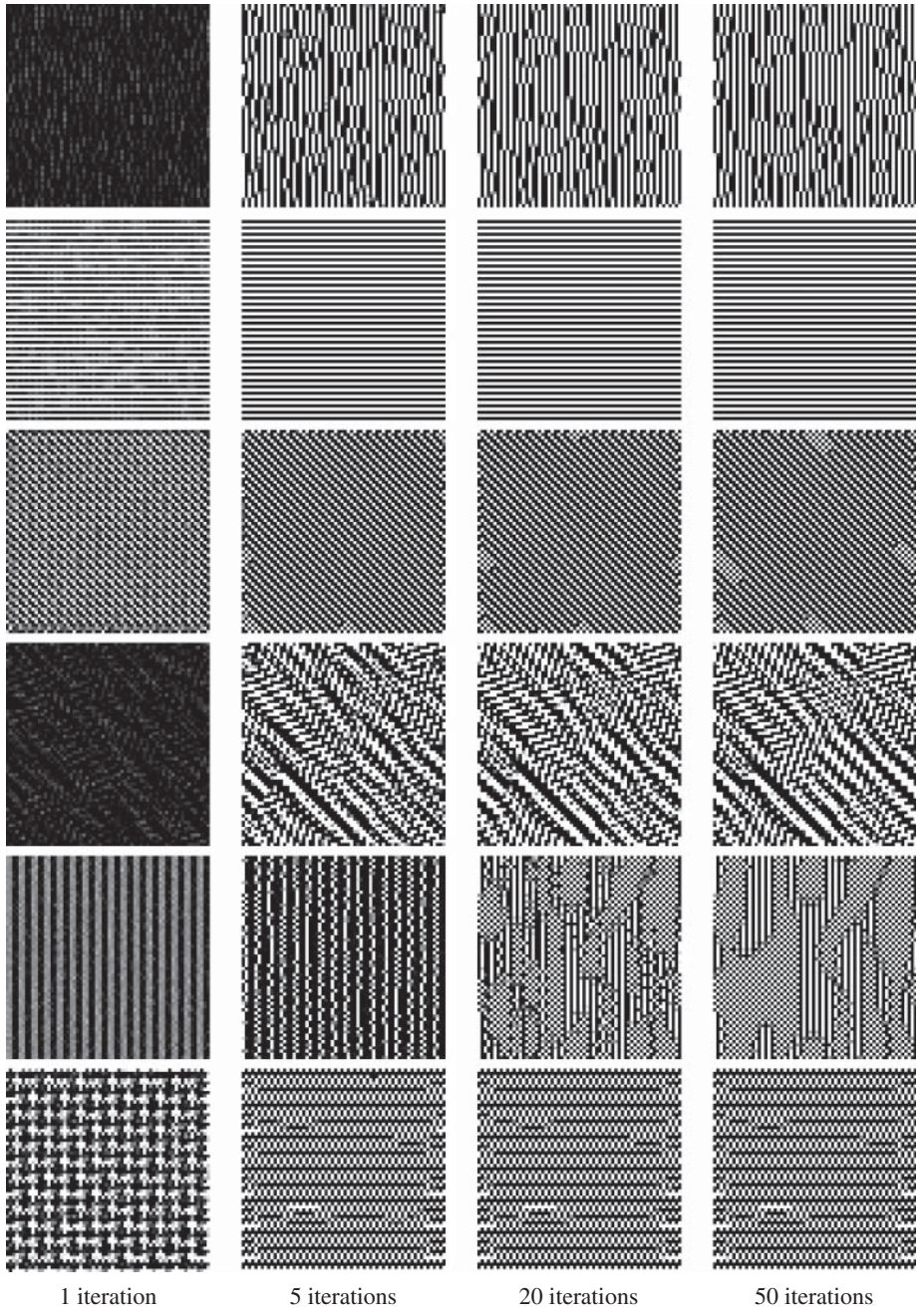


Figure 3.132 Creating textures using the Markov model with a normal distribution, for the neighbourhoods defined in Figure 3.122 used sequentially from top to bottom, respectively, but replacing neighbourhoods (c) and (e) with the ones defined in Figure 3.129. In all cases the starting configuration was the random field shown in 3.127. Source: Maria Petrou.

How can we estimate the Markov parameters of a texture?

We can do that by using a **maximum likelihood estimation**, **MLE** for short, or **least square error estimation**, **LSE** for short.

What is maximum likelihood estimation?

Assume that we have a probability density function that depends on some parameters. To make this dependence explicit, we write $p(x|\Omega)$ where Ω is a vector made up from the various parameters that appear in $p(x)$.

Let us first consider the case when $p(x)$ depends only on one parameter, say ω, and we observe one value of variable x, say $x = x_1$. Can we deduce anything about the value of parameter ω? We **decide** that ω should be such that the probability of observing the value x_1 is maximal. This is a **maximum likelihood** approach. It is schematically demonstrated in Figure 3.133 where ω is assumed to be allowed to take one of only three possible values.

If we have more than one observations for x, say we have points $x_1, x_2, \ldots, x_N$, then we compute the **likelihood** $p(Data|\Omega)$ of this combination of points to arise, given that they are assumed independent. The likelihood is defined as:

$$p(Data|\Omega) \equiv \prod_{k=1}^{N} p(x_k|\Omega). \tag{3.367}$$

We then choose Ω that maximises the likelihood $p(\text{Data}|\Omega)$.

Often in practice we use the **log-likelihood** instead of the likelihood.

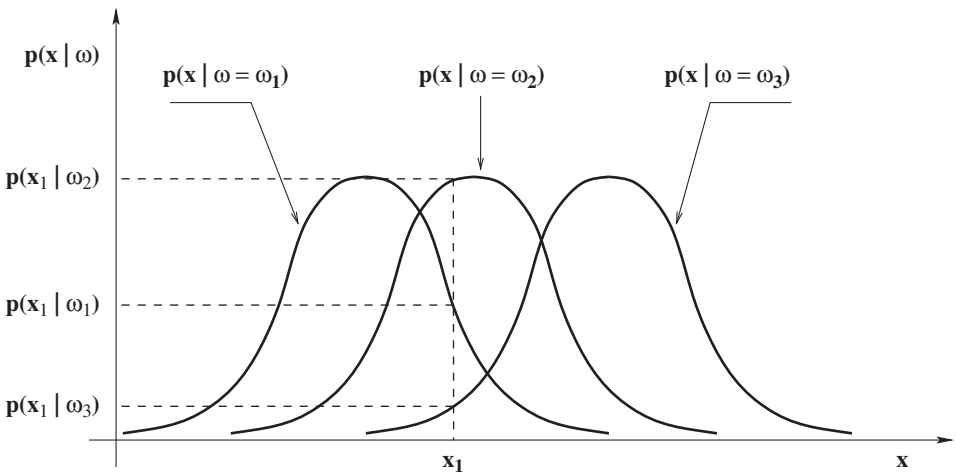


Figure 3.133 We have a different probability density function for each value of parameter ω. Each one of these probability density functions assigns a different probability to the value x_1 to arise. If we observe value x_1 and use a maximum likelihood estimation for parameter ω, we shall estimate ω as being equal to ω_2, since this value maximises the probability of observing value x_1.

What is the log-likelihood?

The **log-likelihood** is defined as

$$l(\text{Data}|\Omega) \equiv \sum_{k=1}^{N} \ln p(x_k|\Omega) \tag{3.368}$$

where N is the total number of independent observations and $p(x_k|\Omega)$ is the probability of value x_k to arise, given the parameter values Ω. Sometimes, because the computation of $p(x_k|\Omega)$ may cause underflows, instead of maximising the likelihood $p(\text{Data}|\Omega)$, we maximise the log-likelihood. Note that either we maximise a function or its logarithm makes no difference to the result, except that the log-likelihood $l(\text{Data}|\Omega)$ tends to have a more blunt maximum than the likelihood $p(\text{Data}|\Omega)$.

Box 3.24 What is the relationship between a maximum likelihood estimation and a Bayesian estimation?

The estimation problem we have may be expressed as follows, according to the Bayesian estimation theory. Given a set of observed data values $\{x_1, x_2, \dots, x_N\}$, which come from the same population with probability density function $p(x_i|\Omega)$, what are the most probable values of parameters Ω? In other words, choose Ω so that we maximise the posterior probability $p(\Omega|\text{Data})$. By applying Bayesṫheorem we have

$$p(\Omega|\text{Data}) = \frac{p(\text{Data}|\Omega)p(\Omega)}{p(\text{Data})} \tag{3.369}$$

where $p(\Omega)$ is the prior probability of the particular configuration of parameter values to arise and $p(\text{Data})$ is the prior probability of the data to arise.

Prior probability $p(\text{Data})$ does not depend on the unknown Ω, so it makes no difference to the solution of the problem and it may be ignored. The Bayesian estimator then should choose values of Ω so that the numerator of (3.369), i.e. $p(\text{Data}|\Omega)p(\Omega)$ is maximised.

Now the maximum likelihood estimator chooses Ω so that $p(\text{Data}|\Omega)$ is maximised.

If we have no prior information about the parameters of function $p(\Omega)$, we may assume that all parameter values are equally likely, i.e. we may assume that $p(\Omega)$ is a constant. Then the maximum of $p(Data|\Omega)p(\Omega)$ is the same as the maximum of $p(\text{Data}|\Omega)$, i.e. the solution of the Bayesian estimator is the same as the solution of the maximum likelihood estimator.

How can we apply the maximum likelihood estimation to estimate the parameters of a Markov random field?

First we have to extract from the image a set of independent observations. Since the value of a pixel depends on the values of its neighbours, we must choose pixels that are not neighbours of each other. So, we must again use the coding of the image as we did for the creation of textures. Pixels that belong to the same code and may be used together are marked with the same symbol in Figure 3.124. These pixels are not neighbours of each other and so they constitute independent observations.

We then use the pixels that are flagged by the same symbol and compute the value of the probability density function $p(x_k|\Omega)$ for each one of them, given the values of its neighbours and a chosen

set of model parameter values Ω. We then multiply all these quantities to produce the likelihood $p(\text{Data}|\Omega)$ of Equation (3.367). We repeat the process for various values of Ω and choose the one that maximises $p(\text{Data}|\Omega)$.

We may repeat the procedure for the pixels flagged by all other symbols in the coding. Each set of pixels will result in a different set of values for the Markov parameters, although ideally they should all result in exactly the same values. We adopt the average values as the parameters that characterise the texture. These are the **Markov texture features**. This method is extremely slow in practice and it can only be used for the simplest of cases.

How do we know which parameter values to try when we apply the maximum likelihood estimation to estimate the Markov parameters?

People often express the values of the Markov parameters as random variables themselves, distributed according to some probability density function. The parameters of the distributions of the parameters are called **super-parameters**.

For example, one may assume that parameter a in Equation (3.361) is a random number drawn from a Gaussian probability density function with mean μ_a and standard deviation σ_a. Parameters μ_a and σ_a are the super-parameters for this case. Their values are specified from some prior knowledge concerning the type of image we are dealing with.

Example 3.120

Use maximum likelihood estimation to compute the parameters of the artificial texture with nine distinct grey levels shown in the first line of Figure 3.126 using the neighbourhoods of Figures 3.116a and 3.117a.

This texture was created with parameter values $a = -1$ and $b = 1$ as they appear in Equation (3.361). To estimate these parameters from a given image, we shall apply the maximum likelihood estimation using the two sets of pixels identified by the two codes in Figure 3.124a separately. Each set of pixels will yield a different estimate of the parameters. The accepted estimate will be the average of the two estimates.

The maximum likelihood estimation relies on us sampling several possible values of the parameters. Let us make the assumption that both parameters are uniformly distributed in the range $[-\alpha, \alpha]$. Let us choose 1.5 as the value of super-parameter α. Let us also decide to sample the range $[-1.5, 1.5]$ uniformly in steps of 0.25. Then there are 169 possible distinct pairs of values of parameters (a, b) we must try in order to choose the one that maximises the probability of observing the pattern we observe.

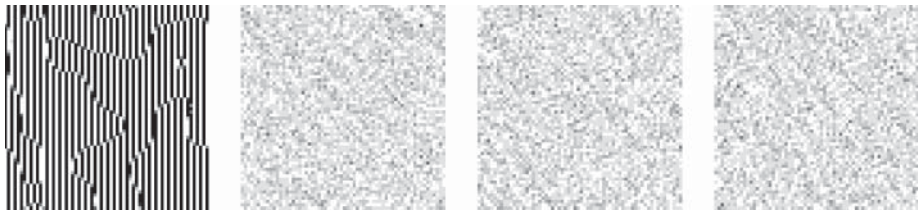
It turned out that several pairs of parameter values maximised the likelihood. These pairs of parameters (a, b) were: $(-0.25, 0.25)$, $(-0.5, 0.5)$, $(-0.75, 0.75)$, $(-1.0, 1.0)$, $(-1.25, 1.25)$ and $(-1.5, 1.5)$, that is, $a = -b$ for all values of $a < 0$ that were considered. The results were exactly the same for both sets of pixels considered (identified by the two codes).

Then we applied neighbourhood 3.117a and used the two sets of pixels identified by the coding of Figure 3.124d, in order to obtain two estimates for each of the two parameters of this model. These two estimates were subsequently averaged. Note that this model is totally wrong for this texture. We used again the same super-parameters and we identified that the values of the parameters that maximised the likelihood were $a = 1.25$ (for the top-left and bottom-right neighbours) and $b = -0.5$ (for the top-right and bottom-left neighbours).

Example 3.121

Use the parameters you estimated in Example 3.120 for the neighbourhood structure 3.117a to re-synthesise the texture. How similar is this texture to the one from which the parameters were originally estimated?

The textures obtained using the parameters estimated in Example 3.120 for the neighbourhood structure in Figure 3.117a are shown in Figure 3.134. We can see that this texture looks totally different from the one in the first row of Figure 3.126 since it was created using a different model. This example shows how significant is to be able to select the right neighbourhood for the Markov model with which to model a given texture.



Original 5 iterations 20 iterations 50 iterations

Figure 3.134 An original texture and texture created using the parameters estimated from it, in Example 3.120, assuming the wrong type of neighbourhood. Source: Maria Petrou.

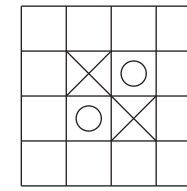
Example 3.122

Use the maximum likelihood estimation to compute the parameters of the textures of Figure 3.135a, assuming a first order Markov neighbourhood and the auto-binomial Markov random field model given by Equation (3.360) with function s given by (3.361). Consider the following options for the parameter values and select the one that maximises the likelihood of the image: $(a, b) = (-1, 1)$ **and** $(a, b) = (1, -1)$**.**

Figure 3.135 (a) A 2-bit 4×4 image. (b) The two codings keeping only the pixels that have full neighbourhoods.

2	**1**	**3**	**0**
3	**0**	**3**	**0**
3	**1**	**2**	**1**
2	**0**	**2**	**1**

(a)



(b)

Figure 3.135b shows the codings of the 2×2 image that remains after we ignore the pixels that do not have complete Markov neighbourhood. pixels marked with the same symbol belong to the same coding as they are not neighbours of each other when we adopt the first order Markov neighbourhood. Table 3.28 lists the values related to the pixels of the coding marked with crosses. We remember that $s \equiv a(g_l + g_r) + b(g_t + g_b)$.

(Continued)

Example 3.122 (Continued)

Table 3.28 The data used for the estimation of the best set of Markov parameters from the first coding of the image.

k	g_l	g_r	g_t	g_b	s for $(-1,1)$	p_k	s for $(1,-1)$	p_k
0	3	3	1	1	−4	0.930	4	10^{-7}
2	1	1	3	2	3	0.012	−3	0.012

The entries in the column under p_k were computed from:

$$p_k = \binom{n}{k} \frac{e^{ks}}{(1+e^s)^n} = \frac{4!}{(4-k)!k!} \frac{e^{k\left(a(g_l+g_r)+b(g_t+g_b)\right)}}{\left(1+e^{a(g_l+g_r)+b(g_t+g_b)}\right)^4}. \tag{3.370}$$

For example, for the first set of parameters, i.e. $a=-1$ and $b=1$, we have:

$$p_0 = \frac{4!}{4!0!} \frac{1}{\left(1+e^{-4}\right)^4} \qquad p_2 = \frac{4!}{2!2!} \frac{e^6}{\left(1+e^3\right)^4}. \tag{3.371}$$

The likelihood value for the first set of parameters is $0.930 \times 0.012 = 0.01116$, while the likelihood value for the second set of parameters is $10^{-7} \times 0.012 \simeq 10^{-9}$. Therefore, according to the first coding, the parameter values that maximise the likelihood are $a=-1$ and $b=1$.

Table 3.29 lists the values related to the pixels of the coding marked with open circles.

Again the second coding indicates that the set of parameters that maximises the product $p_3 p_1$, which is the likelihood, is $a=-1$ and $b=1$. This set of parameters seems correct, as it corresponds to the case of horizontal neighbours being dissimilar and vertical neighbours similar, and the image we have shows the presence of vertical lines.

Table 3.29 The data used for the estimation of the best set of Markov parameters from the second coding of the image.

k	g_l	g_r	g_t	g_b	s for $(-1,1)$	p_k	s for $(1,-1)$	p_k
3	0	0	3	2	5	0.026	−5	10^{-7}
1	3	2	0	0	−5	0.026	5	10^{-7}

Example 3.123

Use the maximum likelihood estimation to compute the parameters of the textures of Figure 3.1 assuming that the first order Markov neighbourhood (shown in Figure 3.116a applies.

First of all, to make the problem tractable, we re-quantise the grey levels of these textures to only eight. We may do that by taking each 8-bit pixel value and keeping only its first three significant bits. This way all pixels will have grey values in the range $[0, 7]$. *This is equivalent to applying the method demonstrated at the beginning of this chapter in Figure 3.2. This method may not be very appropriate as it may totally destroy some low-contrast patterns. So, it is better to use the method demonstrated in Figure 3.3, which results in all eight grey levels being equally represented in the reduced image.*

Then we apply the maximum likelihood estimation to the images that result, using the same super-parameter assumptions as in Example 3.120.

The results for the parameters of each image are presented in Table 3.30. Each set of pixels of the coding produced exactly the same parameter values.

Note that on the basis of these parameters the textures are not easily distinguishable.

In the third column of Table 3.30 we give the parameters computed for neighbourhood 3.117a. There were again two parameters to be estimated, referring to the influence of pixels in symmetric positions about the central pixel. We used the coding of Figure 3.124d. In both cases we used the same super-parameters as in Example 3.120. If instead of using basis e we used basis $\alpha = 1.1$, *we obtained the parameters shown in Table 3.31 for the two types of the assumed neighbourhood. We can see that the two sets of parameters do not differ from each other by a simple scaling (by* $\ln 1.1 = 0.095310$*), in spite of the fact that changing basis amounts to simply scaling the parameters.*

Table 3.30 Parameters estimated for the textures in Figure 3.1 using the MLE method, and the neighbourhood structures shown in Figures 3.116a and 3.117a.

	Neighbourhood 3.116a		Neighbourhood 3.117a	
Image	***a***	***b***	***a***	***b***
Cloth 1	−0.25	0.25	0.25	−0.25
Cloth 2	−0.25	0.25	−0.25	0.25
Food	0.25	−0.25	−0.25	0.25
Plastic	0.25	−0.25	0.25	−0.25
Fabric 1	−0.25	0.25	0.25	−0.25
Fabric 2	−0.25	0.25	−0.25	0.25
Beans	−0.25	0.25	0.00	0.00
Sand	0.25	−0.25	0.25	−0.25

(Continued)

Example 3.123 (Continued)

Table 3.31 Parameters estimated for the textures in Figure 3.1 using the maximum likelihood estimation method, and the neighbourhood structures shown in Figures 3.116a and 3.117a. For the exponential function basis $\alpha = 1.1$ was used instead of e.

	Neighbourhood 3.116a		Neighbourhood 3.117a	
Image	***a***	***b***	***a***	***b***
Cloth 1	−0.50	1.00	0.50	−0.25
Cloth 2	−0.25	0.50	−0.25	0.25
Food	0.50	−0.25	−0.50	0.25
Plastic	0.25	0.25	0.50	−0.25
Fabric 1	−0.25	0.50	0.50	−0.25
Fabric 2	−0.25	0.75	−0.75	0.50
Beans	−0.25	0.75	−0.25	0.25
Sand	1.50	−1.25	0.25	−0.25

How can we estimate the Markov parameters using the least square error estimation method?

We shall demonstrate this method by applying it to model (3.366) for the second-order Markov neighbourhood shown in Figure 3.116b.

There are eight pixels in this case that influence the value of a pixel. However, we do not expect neighbours in symmetric positions about the centre to play different roles in the value of the central pixel, so we may assume that pixels in diametrically opposite positions about the central position enter into the equation with identical coefficients. This means that we have four parameters a_l to estimate plus parameter σ, which characterises the random experiment performed in each location to assign a value to the pixel. (Remember that when we introduced Markov random fields we talked about flipping a coin n times to decide the value of a pixel. In a Gaussian Markov random field, instead of flipping a coin, one draws a random number from a Gaussian distribution with mean value that depends on the values of the neighbours and with variance some value that is also characteristic of the process.) So, all in all, we have to estimate five parameters.

We re-write here Equation (3.366) in a more suitable form

$$p(g_{ij}|g_{i'j'},(i',j')\in N_{ij}) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{\left(g_{ij}-\sum_{l=1}^{4}a_l s_{ij;l}\right)^2}{2\sigma^2}\right\} \tag{3.372}$$

where we used

$$\begin{aligned} s_{ij;1} &\equiv g_{i-1,j} + g_{i+1,j} \\ s_{ij;2} &\equiv g_{i,j-1} + g_{i,j+1} \\ s_{ij;3} &\equiv g_{i-1,j-1} + g_{i+1,j+1} \\ s_{ij;4} &\equiv g_{i-1,j+1} + g_{i+1,j-1} \end{aligned} \tag{3.373}$$

with g_{ij} being the grey value of pixel (i,j).

We have one such equation for each pixel. **In the least square error estimation, we assume that each pixel takes its most probable value allowed by all other pixels.** In the absence of any constraints, each pixel (i,j) will take value $g_{ij} = \sum_{l=1}^{4} a_l s_{ij;l}$ which maximises the probability (3.372). Clearly, this will not be possible, and it can only be achieved within the limits of uncertainty expressed by parameter σ. We may, therefore, for an image of size $N \times M$, estimate parameters a_l in such a way that the system of NM equations

$$g_{ij} = \sum_{l=1}^{4} a_l s_{ij;l} \tag{3.374}$$

is satisfied as much as possible, with the variance of the errors of imbalance between the left and the right side of each equation being σ^2.

The values of the parameters estimated from these equations are given by:

$$\sigma^2 = \frac{1}{(N-2)(M-2)} \sum_{ij} \left[g_{ij} - \sum_{l=1}^{4} a_l s_{ij;l} \right]^2 \tag{3.375}$$

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \left\{ \sum_{ij} \begin{bmatrix} s_{ij;1}^2 & s_{ij;1}s_{ij;2} & s_{ij;1}s_{ij;3} & s_{ij;1}s_{ij;4} \\ s_{ij;2}s_{ij;1} & s_{ij;2}^2 & s_{ij;2}s_{ij;3} & s_{ij;2}s_{ij;4} \\ s_{ij;3}s_{ij;1} & s_{ij;3}s_{ij;2} & s_{ij;3}^2 & s_{ij;3}s_{ij;4} \\ s_{ij;4}s_{ij;1} & s_{ij;4}s_{ij;2} & s_{ij;4}s_{ij;3} & s_{ij;4}^2 \end{bmatrix} \right\}^{-1} \sum_{ij} g_{ij} \begin{pmatrix} s_{ij;1} \\ s_{ij;2} \\ s_{ij;3} \\ s_{ij;4} \end{pmatrix}. \tag{3.376}$$

Equation (3.375) is the unbiased estimation of the variance of the errors with which the estimated values of the parameters allow the satisfaction of equations (3.374). For the derivation of these equations see Box 3.25.

Box 3.25 Least square parameter estimation for the Markov random field parameters

Equation (3.374) expresses a system of NM linear equations with four unknowns. It is an overdetermined system, so it may be solved in the least square error sense. Let us write it first in matrix form

$$G = SA \tag{3.377}$$

where G is an $(NM) \times 1$ vector made up from the pixel grey values,

$$G \equiv \begin{pmatrix} g_{11} \\ g_{12} \\ \vdots \\ g_{ij} \\ \vdots \\ g_{NM} \end{pmatrix} \tag{3.378}$$

(Continued)

Box 3.25 (Continued)

S is a $(NM) \times 4$ matrix made up from the values of the neighbours of each pixel,

$$S \equiv \begin{pmatrix} s_{11;1} & s_{11;2} & s_{11;3} & s_{11;4} \\ s_{12;1} & s_{12;2} & s_{12;3} & s_{12;4} \\ \vdots & \vdots & \vdots & \vdots \\ s_{ij;1} & s_{ij;2} & s_{ij;3} & s_{ij;4} \\ \vdots & \vdots & \vdots & \vdots \\ s_{NM;1} & s_{NM;2} & s_{NM;3} & s_{NM;4} \end{pmatrix} \tag{3.379}$$

and A is a 4×1 vector of the values of the unknown parameters a_l:

$$A \equiv \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} \tag{3.380}$$

As S is not a square matrix, we cannot take its inverse to solve system (3.377). However, if we multiply both sides of (3.377) from the left with S^T, we shall have on the right-hand side matrix $S^T S$ which is 4×4, i.e. square, and so allows us to take its inverse:

$$S^T G = S^T S A \Rightarrow A = \left(S^T S\right)^{-1} S^T G. \tag{3.381}$$

Matrix $\left(S^T S\right)^{-1} S^T$ is called the **pseudo-inverse** of matrix S and when used to solve an overdetermined system of linear equations yields the least square error solution of the system.

If we substitute the result of Example 3.124 for $S^T G$ and the result of example 3.125 for $S^T S$, we shall obtain the formula given by Equation (3.376) for the estimation of parameters a_l.

Example B3.124

If matrices G and S are defined by Equations (3.378) and (3.379), respectively, show that

$$S^T G = \sum_{ij} g_{ij} \begin{pmatrix} s_{ij;1} \\ s_{ij;2} \\ s_{ij;3} \\ s_{ij;4} \end{pmatrix}. \tag{3.382}$$

We start by computing explicitly the left-hand side of the above equation and then show that it is equal to the right-hand side:

$$S^T G = \begin{pmatrix} s_{11;1} & s_{12;1} & \cdots & s_{NM;1} \\ s_{11;2} & s_{12;2} & \cdots & s_{NM;2} \\ s_{11;3} & s_{12;3} & \cdots & s_{NM;3} \\ s_{11;4} & s_{12;4} & \cdots & s_{NM;4} \end{pmatrix} \begin{pmatrix} g_{11} \\ g_{12} \\ \vdots \\ g_{NM} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{ij} s_{ij;1} g_{ij} \\ \sum_{ij} s_{ij;2} g_{ij} \\ \sum_{ij} s_{ij;3} g_{ij} \\ \sum_{ij} s_{ij;4} g_{ij} \end{pmatrix}$$

$$= \sum_{ij} g_{ij} \begin{pmatrix} s_{ij;1} \\ s_{ij;2} \\ s_{ij;3} \\ s_{ij;4} \end{pmatrix}. \tag{3.383}$$

Example B3.125

If matrix S is defined by Equation (3.379), show that

$$S^T S = \sum_{ij} \left[\begin{pmatrix} s_{ij;1} \\ s_{ij;2} \\ s_{ij;3} \\ s_{ij;4} \end{pmatrix} \begin{pmatrix} s_{ij;1} & s_{ij;2} & s_{ij;3} & s_{ij;4} \end{pmatrix} \right]. \tag{3.384}$$

We start by directly computing the left-hand side of the above equation and show that it is equal to the right-hand side:

$$S^T S = \begin{pmatrix} s_{11;1} & s_{12;1} & \cdots & s_{NM;1} \\ s_{11;2} & s_{12;2} & \cdots & s_{NM;2} \\ s_{11;3} & s_{12;3} & \cdots & s_{NM;3} \\ s_{11;4} & s_{12;4} & \cdots & s_{NM;4} \end{pmatrix} \begin{pmatrix} s_{11;1} & s_{11;2} & s_{11;3} & s_{11;4} \\ s_{12;1} & s_{12;2} & s_{12;3} & s_{12;4} \\ \vdots & \vdots & \vdots & \vdots \\ s_{NM;1} & s_{NM;2} & s_{NM;3} & s_{NM;4} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{ij} s_{ij;1}^2 & \sum_{ij} s_{ij;1} s_{ij;2} & \sum_{ij} s_{ij;1} s_{ij;3} & \sum_{ij} s_{ij;1} s_{ij;4} \\ \sum_{ij} s_{ij;2} s_{ij;1} & \sum_{ij} s_{ij;2}^2 & \sum_{ij} s_{ij;2} s_{ij;3} & \sum_{ij} s_{ij;2} s_{ij;4} \\ \sum_{ij} s_{ij;3} s_{ij;1} & \sum_{ij} s_{ij;3} s_{ij;2} & \sum_{ij} s_{ij;3}^2 & \sum_{ij} s_{ij;3} s_{ij;4} \\ \sum_{ij} s_{ij;4} s_{ij;1} & \sum_{ij} s_{ij;4} s_{ij;2} & \sum_{ij} s_{ij;4} s_{ij;3} & \sum_{ij} s_{ij;4}^2 \end{pmatrix}$$

$$= \sum_{ij} \left[\begin{pmatrix} s_{ij;1} \\ s_{ij;2} \\ s_{ij;3} \\ s_{ij;4} \end{pmatrix} \begin{pmatrix} s_{ij;1} & s_{ij;2} & s_{ij;3} & s_{ij;4} \end{pmatrix} \right]. \tag{3.385}$$

Example 3.126

Use the least square error approach to compute the parameters of the texture images of Figure 3.1 using the second order Markov neighbourhood shown in Figure 3.116b.
We assume that pixels in symmetric positions about the central pixel influence its value with the same parameter. We have, therefore, five independent parameters to estimate. For the chosen neighbourhood the coding of Figure 3.124b identifies four sets of pixels. The values of the five parameters computed from each set of pixels separately as well as their average values are shown in Table 3.32. Parameters a_{tb}, a_{lr}, a_{tlbr} and a_{trbl} refer to the way the top and bottom, left and right, top-left and bottom-right, and top-right and bottom-left neighbours influence the central pixel.

Table 3.32 Estimated MRF parameters using the LSE method for the images in Figure 3.1. Different sets of pixels refer to different codes in the coding used.

Image	Set	a_{tb}	a_{lr}	a_{tlbr}	a_{trbl}	σ
Cloth 1	Set 1	0.410	0.291	−0.137	−0.064	13.097
	Set 2	0.406	0.300	−0.143	−0.062	13.153
	Set 3	0.423	0.281	−0.141	−0.062	13.013
	Set 4	0.411	0.294	−0.138	−0.068	13.108
	Mean	0.412	0.292	−0.140	−0.064	13.093
Cloth 2	Set 1	0.381	0.213	−0.007	−0.087	11.678
	Set 2	0.377	0.221	−0.003	−0.095	11.755
	Set 3	0.386	0.218	−0.007	−0.098	11.847
	Set 4	0.375	0.225	−0.012	−0.088	11.846
	Mean	0.380	0.219	−0.007	−0.092	11.781
Food	Set 1	0.353	0.444	−0.099	−0.198	22.693
	Set 2	0.352	0.440	−0.096	−0.195	22.910
	Set 3	0.348	0.451	−0.100	−0.197	22.966
	Set 4	0.353	0.442	−0.101	−0.194	22.893
	Mean	0.352	0.444	−0.099	−0.196	22.866
Plastic	Set 1	0.448	0.437	−0.209	−0.176	9.005
	Set 2	0.452	0.438	−0.222	−0.167	9.116
	Set 3	0.460	0.431	−0.222	−0.168	9.086
	Set 4	0.462	0.431	−0.214	−0.178	9.088
	Mean	0.455	0.434	−0.217	−0.173	9.074
	Set 1	0.246	0.148	0.000	0.107	13.930
	Set 2	0.235	0.185	−0.006	0.088	14.191

Table 3.32 (Continued)

Image	Set	a_{tb}	a_{lr}	a_{tlbr}	a_{trbl}	σ
Fabric 1	Set 3	0.288	0.115	0.009	0.090	14.223
	Set 4	0.252	0.183	−0.024	0.091	14.221
	Mean	0.255	0.158	−0.005	0.094	14.141
Fabric 2	Set 1	0.552	0.493	−0.262	−0.281	7.521
	Set 2	0.551	0.497	−0.262	−0.287	7.782
	Set 3	0.550	0.491	−0.259	−0.284	7.550
	Set 4	0.558	0.495	−0.265	−0.287	7.756
	Mean	0.553	0.494	−0.262	−0.285	7.652
Beans	Set 1	0.464	0.438	−0.196	−0.205	15.313
	Set 2	0.461	0.438	−0.197	−0.200	15.189
	Set 3	0.464	0.424	−0.191	−0.196	15.034
	Set 4	0.467	0.441	−0.197	−0.210	15.382
	Mean	0.464	0.435	−0.195	−0.203	15.229
Sand	Set 1	0.453	0.485	−0.222	−0.216	7.911
	Set 2	0.449	0.485	−0.236	−0.198	8.009
	Set 3	0.457	0.480	−0.236	−0.201	7.933
	Set 4	0.451	0.478	−0.218	−0.211	7.951
	Mean	0.453	0.482	−0.228	−0.207	7.951

Example 3.127

Use the parameters computed in Example 3.126 to re-synthesise the textures.
The results after 50 iterations are shown in Figures 3.136 and 3.137.

We can see that, for micro-textures, the synthesised textures are quite similar to the original ones.

For textures which contain blob-like structures, much bigger than the size of the neighbourhood used, the synthesised textures are very different from the original ones. It is obviously impossible for the neighbourhood we used to capture the characteristics of these textures. The neighbourhoods of the sizes we chose simply do not exhibit stationary statistics over the whole image in

(Continued)

Example 3.127 (Continued)

these cases, since when, one blob meets another blob, the statistics of the local neighbourhood change significantly.

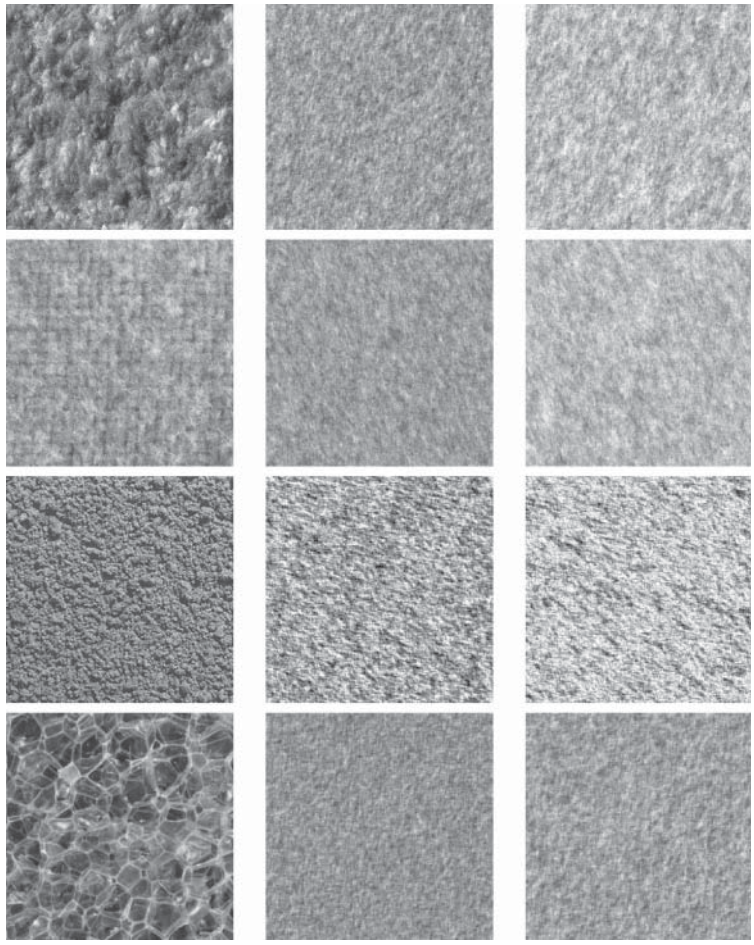


Figure 3.136 Synthesised textures using the parameters estimated by the least square error (LSE) estimation method. Results for images cloth 1, cloth 2, food and plastic, from top to bottom, respectively.

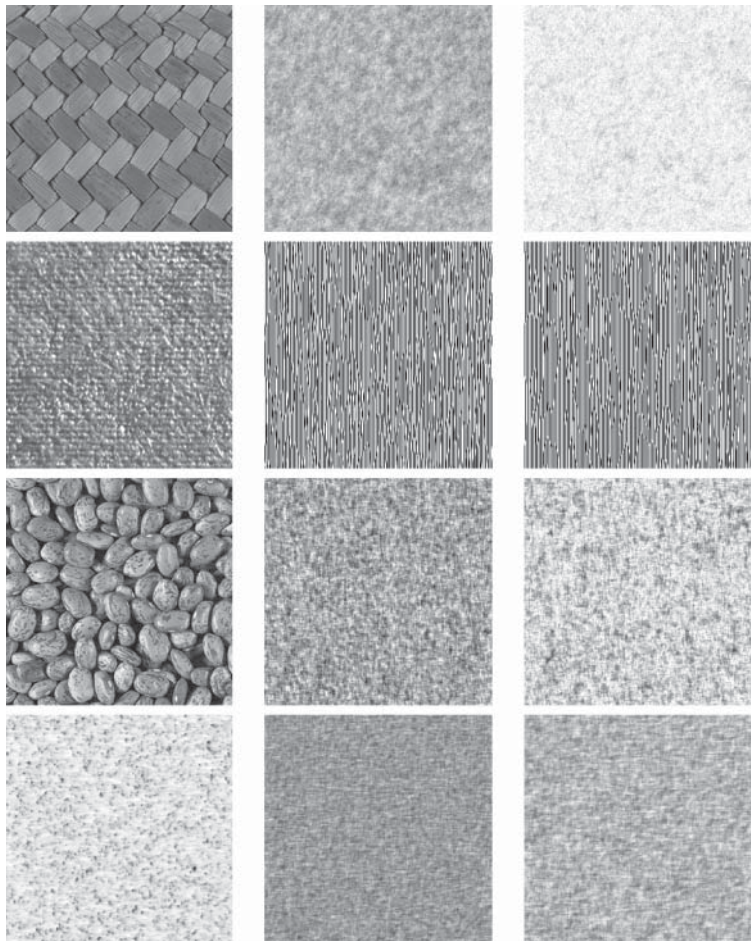


Figure 3.137 Synthesised textures using the parameters estimated by least square error. Results for fabric 1, fabric 2, beans and sand, from top to bottom, respectively.

How can we work out the size of the Markov neighbourhood?

To work out the most relevant size of Markov neighbourhood we should use for modelling a texture with the help of the autocorrelation function of the texture image. However, any neighbourhood of size more than, say, 3×3 pixels is rather difficult to use in practice: it requires the determination of far too many parameters from the sample texture we have. A much better approach to using Markov random field models for textures with large blobs, i.e. macrotextures, is to analyse the texture image into frequency bands with the help of the **Laplacian pyramid**.

What is the Laplacian pyramid?

The Laplacian pyramid is an image representation that returns the value of the sum of the second derivatives of the image, along the coordinate axes, at each pixel location, for a succession of scales. It is constructed as follows.

Step 1: Smooth the input image I_0 with a Gaussian filter, to produce an output image I_1'.
Step 2: Subtract I_1' from I_0 to form array $L_1 = I_0 - I_1'$.
Step 3: Subsample image I_1' by omitting every second row and every second column to produce image I_1, which is half the size of the original image along each dimension.
Step 5: Smooth image I_1 with a Gaussian filter, to produce an output image I_2'.
Step 6: Subtract I_2' from I_1 to form array $L_2 = I_1 - I_2'$.

The above process may be repeated for as many steps as we wish, until the final array L_n is as small as a single pixel. The sequence of image representations we create that way $I_0, I_1, \ldots, I_n$ is known as a **Gaussian pyramid** and the sequence $L_1, L_2, \ldots, L_{n-1}$ that we can create from it is called a **Laplacian pyramid**. The process of constructing these two image representations is shown schematically in Figure 3.138. In Chapter 4, when we discuss wavelets, we shall see in more detail why such a representation is said to analyse the image into a set of frequency bands. Figure 3.139 shows the frequency bands in which the Gaussian and the Laplacian pyramids analyse the image, without any further analysis at this point.

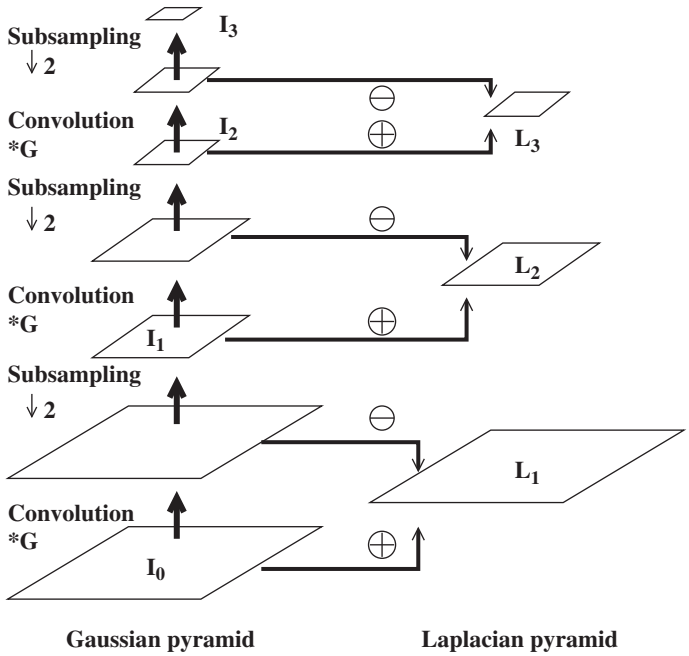


Figure 3.138 The way we construct Gaussian and Laplacian pyramids. I_0 is the input image. G represents the Gaussian filter with which we convolve the image to smooth it.

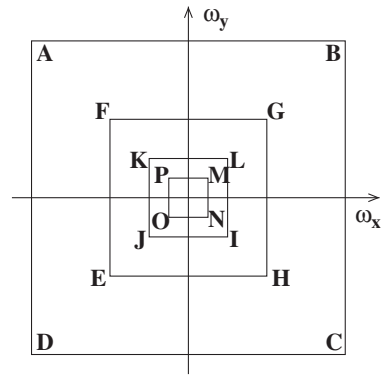


Figure 3.139 The frequency domain of an image, with ω_x and ω_y the frequencies along the x and y axis, respectively. The dc component is at the centre. A four-level Gaussian pyramid represents an image in the three nested frequency bands represented here by rectangles ABCD, FGHE, KLIJ and MNOP. The corresponding Laplacian pyramid consists of three levels, one of which is made up from the frequencies inside the annulus ABCDEFGH, the next one made up from the frequencies inside the annulus EFGHIJKL and the final one made up from the frequencies inside the annulus IJKLMNOP.

Why is the creation of a Laplacian pyramid associated with the application of a Gaussian function at different scales and the subtraction of the results?

Convolution is a linear process. Subtracting the results of two convolutions produced by using two different filters is equivalent to convolving the original signal with the difference of the filters. When the two filters are Gaussians with different standard deviations, their difference looks like the filter of Figure 4.92. A filter with such a shape is known as a **Mexican hat** filter, or **sombrero filter** or **DoG (difference of Gaussians)** filter. Such a filter estimates the second derivative of the signal.

Why may the second derivative of a Gaussian function be used as a filter to estimate the second derivative of a signal?

Differentiation always enhances noise. Before we differentiate a signal we usually smooth it. A Gaussian filter may be used for this purpose. The smoothed signal is then differentiated. Smoothing (by convolution with a low pass filter) is a linear process. Differentiation is also a linear process. The order by which we apply two linear processes does not affect the result we get. So, instead of smoothing the signal, by convolving it with a filter, and differentiating the result, we may differentiate the filter and convolve the signal with the derivative of the filter. So, convolution of a signal with the second derivative of the Gaussian filter will produce the same result as we would have got if we had convolved the signal with a Gaussian to smooth it first, and then differentiated it twice to estimate its second derivative.

Example 3.128

Construct the four-level Gaussian pyramid of the image shown in Figure 3.140a. For the low pass Gaussian filter use 3.140b. Assume that the image is repeated ad infinitum in all directions.

(Continued)

Example 3.128 (Continued)

1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	2	2	2	2	0	1	0	1	0	1	0	1	0	1	0
1	3	3	3	3	3	1	0	1	0	1	0	1	0	1	0
1	2	2	2	2	2	1	0	1	0	3	3	3	0	1	0
1	3	3	3	3	3	1	0	1	2	2	2	2	2	1	0
1	0	2	2	2	2	2	0	1	3	3	3	3	3	1	0
1	0	1	0	1	0	1	0	1	2	2	2	2	2	1	0
1	0	1	0	1	0	1	0	3	3	3	3	3	3	1	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	0	3	3	3	3	1	0	1	0	1	0	1	0	1	0
1	0	2	2	2	2	1	0	1	0	1	2	2	2	1	0
1	3	3	3	3	3	3	0	1	0	3	3	3	3	1	0
1	2	2	2	2	2	2	0	1	0	2	2	2	2	1	0
1	3	3	3	3	3	1	0	1	3	3	3	3	3	1	0
1	0	2	2	2	0	1	0	1	2	2	2	2	2	1	0
1	0	1	0	1	0	1	0	1	3	3	3	3	3	1	0

(a)

1	2	1
2	4	2
1	2	1

(b)

Figure 3.140 (a) A 2-bit 16×16 image. (b) A smoothing Gaussian mask.

We first augment the image into size 18×18 *by repeating at the top its last row, at the bottom its first row, on its left its right-most column and on its right its left-most column. Then we convolve the augmented image with the Gaussian filter given in 3.140b, ignoring border pixels that do not have complete neighbourhoods. The result will be a* 16×16 *image. We note that the elements of the filter sum up to 16, so the output of the convolution should be divided by 16 to yield the smoothed version of the image. The result is shown in Figure 3.141a.*

Image 3.141a is subsequently sub-sampled by omitting its even rows and columns, to produce the 8×8 *image shown in 3.141b.*

Image 3.141b next is augmented by repeating its top row at the bottom and its bottom row at the top, its left column on the right and its right column on the left. Then the 10×10 *augmented image is convolved with the Gaussian mask, omitting from the convolution process a layer of pixels all round. The* 8×8 *output of the convolution, after division by 16, is shown in Figure 3.142a.*

This is sub-sampled by keeping only its odd rows and columns, to produce the 4×4 *image of Figure 3.142b.*

Finally, 3.142b is smoothed and sub-sampled to produce the final layer of the Gaussian pyramid, consisting only of 2×2 *pixels, as shown in Figure 3.143.*

Figures 3.140a, 3.141b, 3.142b and 3.143b constitute the four layers of the Gaussian pyramid.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
1	2	3	3	2	2	1	1	1	1	1	1	1	1	1	1
1	2	3	3	3	2	1	1	1	1	2	2	2	1	1	1
1	2	2	3	3	2	1	1	1	2	2	3	2	2	1	1
1	1	2	2	2	2	1	1	1	2	3	3	3	2	1	1
1	1	1	1	1	1	1	1	1	2	3	3	3	2	1	1
1	1	1	1	1	1	1	1	2	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	2	2	2	2	1	1	1	1	1	1	1	1	1	1
1	1	2	3	3	2	1	1	1	1	1	2	2	2	1	1
1	2	2	3	3	2	2	1	1	1	2	2	3	2	1	1
1	2	3	3	3	2	2	1	1	1	2	3	3	2	1	1
1	2	2	3	2	2	1	1	1	2	2	3	3	2	1	1
1	1	2	2	2	1	1	1	1	2	3	3	3	2	1	1
1	1	1	1	1	1	1	1	1	2	2	2	2	2	1	1

(a) The 16×16 smoothed image

1	1	1	1	1	1	1	1
1	3	2	1	1	1	1	1
1	2	3	1	1	2	2	1
1	1	1	1	1	3	3	1
1	1	1	1	1	1	1	1
1	2	3	1	1	1	2	1
1	3	3	2	1	2	3	1
1	2	2	1	1	3	3	1

(b) The 8×8 sub-sampled image

Figure 3.141 The smoothed and sub-sampled images.

(Continued)

Example 3.128 (Continued)

Figure 3.142 The smoothed image and sub-sampled images.

1	2	1	1	1	1	1	1
1	2	2	1	1	1	1	1
1	2	2	1	1	2	2	1
1	1	1	1	1	2	2	1
1	1	1	1	1	1	2	1
1	2	2	2	1	1	2	1
1	2	2	2	1	2	2	1
1	2	2	1	1	2	2	1

(a) The 8×8 smoothed image

1	1	1	1
1	2	1	2
1	1	1	2
1	2	1	2

(b) The 4×4 sub-sampled image

Figure 3.143 The smoothed and sub-sampled images.

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

(a) The 4×4 smoothed image

1	1
1	1

(b) The 2×2 sub-sampled image

Example 3.129

For the image of Figure 3.140a construct the three-level Laplacian pyramid.

For the first level, we subtract image 3.141a from the original image. The result is shown in Figure 3.144a.

Next, we subtract Figure 3.142a from 3.141b to produce the result shown in Figure 3.144b.

The third level of the pyramid is created by subtracting image 3.143a from 3.142b. Therefore the third and final levels of the pyramid are shown in figures 3.144c and 3.144d, respectively.

0	−1	0	−1	0	−1	0	−1	0	−1	0	−1	0	−1	0	−1
0	0	0	0	0	−1	0	−1	0	−1	0	−1	0	−1	0	−1
0	1	0	0	1	1	0	−1	0	−1	0	−1	0	−1	0	−1
0	0	−1	−1	−1	0	0	−1	0	−1	1	1	1	−1	0	−1
0	1	1	0	0	1	0	−1	0	0	0	−1	0	0	0	−1
0	−1	0	0	0	0	1	−1	0	1	0	0	0	1	0	−1
0	−1	0	−1	0	−1	0	−1	0	0	−1	−1	−1	0	0	−1
0	−1	0	−1	0	−1	0	−1	1	1	1	1	1	1	0	−1
0	−1	0	−1	0	−1	0	−1	0	−1	0	−1	0	−1	0	−1
0	−1	1	1	1	1	0	−1	0	−1	0	−1	0	−1	0	−1
0	−1	0	−1	−1	0	0	−1	0	−1	0	0	0	0	0	−1
0	1	1	0	0	1	1	−1	0	−1	1	1	0	1	0	−1
0	0	−1	−1	−1	0	0	−1	0	−1	0	−1	−1	0	0	−1
0	1	1	0	1	1	0	−1	0	1	1	0	0	1	0	−1
0	−1	0	0	0	−1	0	−1	0	0	−1	−1	−1	0	0	−1
0	−1	0	−1	0	−1	0	−1	0	1	1	1	1	1	0	−1

(a) The first level of the 16×16 subtracted image

0	−1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	1	1	0
0	0	0	0	0	0	−1	0
0	0	1	−1	0	0	0	0
0	1	1	0	0	0	1	0
0	0	0	0	0	1	1	0

(b) The second level of the 8×8 subtracted image

0	0	0	0
0	1	0	1
0	0	0	1
0	1	0	1

(c) The third level of the 4×4 subtracted image

0	0
0	0

(d) The second level of the 4×4 subtracted image

Figure 3.144 The Laplacian pyramid.

How can we use the Laplacian pyramid representation of an image to model it as a Markov field?

The key point of the Laplacian pyramid representation of an image is that each level of the pyramid corresponds to a different frequency band of the image. Then we may use a different Markov random field to model each frequency band.

How can we synthesise a texture if we know the Markov parameters of the levels of its Laplacian pyramid?

The key issue is on the way we recover an image from the knowledge of its Laplacian pyramid. Let us use the following operators to help us represent schematically the construction of the Laplacian pyramid:

$*$ means convolution with the Gaussian mask G.

$\downarrow$ means subsampling by a factor of 2. Let I_0 be the original image, and L_i the ith level of the Laplacian pyramid. Let also I_i' indicate the smoothed version of image I_{i-1}. For a three-level pyramid, we may then write:

$$I_1' = I_0 * G \tag{3.386}$$
$$L_1 = I_0 - I_1' \tag{3.387}$$
$$I_1 = \downarrow I_1' \tag{3.388}$$
$$I_2' = I_1 * G \tag{3.389}$$
$$L_2 = I_1 - I_2' \tag{3.390}$$
$$I_2 = \downarrow I_2' \tag{3.391}$$
$$I_3' = I_2 * G \tag{3.392}$$
$$L_3 = I_2 - I_3' \tag{3.393}$$
$$I_3 = \downarrow I_3'. \tag{3.394}$$

We can clearly see from the above that, if for the moment we neglect the details, and if we add equations (3.387), (3.390) and (3.393), we shall have $L_1 + L_2 + L_3 \sim I_0 - I_3$, as the intermediate terms cancel each other out. So, we must be able to recover I_0 from the knowledge of L_1, L_2, L_3 and I_3.

More precisely, assume that we are given L_1, L_2, L_3 and I_3. We wish to reconstruct the original image I_0. To do that, we must define first an operation that upsamples an image by a factor of 2, so that we have a mechanism to make two different layers of the pyramid of equal size. Such an operation may be applied in two successive steps:

(i) Insert rows and columns of 0s between the existing rows and columns of the image and at its bottom and right, so that the downsampled image is restored to its original size; let us denote this operator by $\uparrow$.
(ii) Blur the augmented with 0s image with the help of the Gaussian kernel.

Let us denote by $\tilde{I}_i$ the result of applying these two operators to image I_i. So:

$$\begin{aligned}
&\tilde{I}_3 = (\uparrow I_3) * G\\
&I_2 \equiv L_3 + \tilde{I}_3\\
&\tilde{I}_2 = (\uparrow I_2) * G\\
&I_1 \equiv L_2 + \tilde{I}_2\\
&\tilde{I}_1 = (\uparrow I_1) * G\\
&I_0 \equiv L_1 + \tilde{I}_1.
\end{aligned} \tag{3.395}$$

Note that I_1, I_2 and I_0 that appear in (3.395) are not quite the same as their corresponding counterparts in (3.386)–(3.394). Nevertheless, we retained the same name to make the correspondence obvious.

Now we know how to reconstruct an image from the knowledge of its Laplacian pyramid and the top most level of its Gaussian pyramid, we are ready to synthesise a texture with the help of the multi-resolution Markov random fields, and a given training sample from which we shall learn the Markov parameters we wish to use for the synthesis.

Step 1: Construct the Laplacian pyramid of the image you have been given for training.
Step 2: Work out the Markov parameters of each layer of the Laplacian pyramid separately, assuming a first or second order Markov neighbourhood.
Step 3: Synthesise a texture for each set of the levels of the Laplacian pyramid, using the estimated Markov parameters.
Step 4: Replace the Laplacian levels of the given image with the synthesised levels and work out the full sized image, which must exhibit similar texture to the given training image.

Is a Markov random field always realisable given that we define it arbitrarily?

No, it is not. A Markov random field is realisable provided it is self-consistent: as each pixel is influenced by its neighbours, and the neighbours by their own neighbours, and so on, it means that not all neighbourhoods we choose and not all model parameters we define lead to a structure that is self-consistent and therefore realisable. Certain conditions have to apply for the field to be self-consistent.

What conditions make a Markov random field self-consistent?

A Markov random field is self-consistent and equivalent to a **Gibbs distribution** if it is below its **critical temperature**, and if its neighbourhood structure can be expressed in terms of **cliques**. The concept of critical temperature will be dealt with at the end of the section on Gibbs distributions (see Example 3.163).

What is a clique in a neighbourhood structure?

A clique is a set of pixels that are neighbours of each other according to the neighbourhood scheme we have adopted. Figure 3.145 shows the cliques that correspond to the neighbourhoods of Figure 3.116a,b,c. Below each neighbourhood structure, we have first the individual pixels: each pixel is neighbour of itself! Then, we have pairs of pixels that are neighbours of each other. The second-order Markov neighbourhood allows also triplets of pixels that are neighbours of each other, as well as a quadruple of such pixels. The third-order Markov neighbourhood allows a few more triplets and quartets of pixels that are neighbours of each other, and in addition a quintet of such pixels.

Example 3.130

Construct the cliques of the neighbourhoods shown in Figure 3.117.
Two pixels form a clique if they are neighbours of each other. In Figure 3.146 we mark with different symbols the neighbourhoods of the neighbours of the central pixel for neighbourhoods (a) and (b) of Figure 3.117. We can then see whether two pixels belong to the neighbourhood of each other and decide whether they form a clique or not. The cliques identified this way, for the

(Continued)

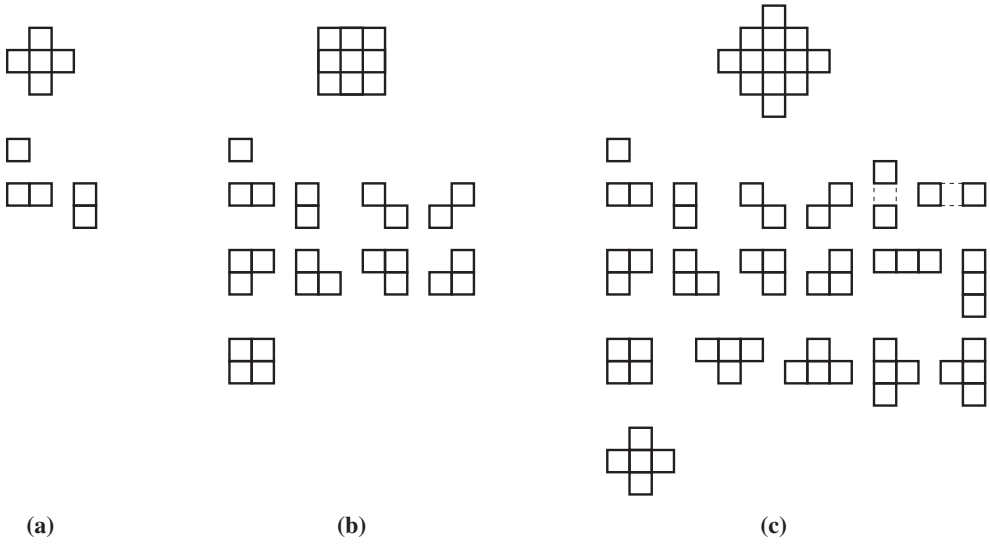


Figure 3.145 The cliques that correspond to the Markov neighbourhoods shown in the first row.

Example 3.130 (Continued)

three neighbourhoods of Figure 3.117, are shown in Figure 3.147. We have three cliques for the first Markov random field neighbourhood, four for the second and five for the third.

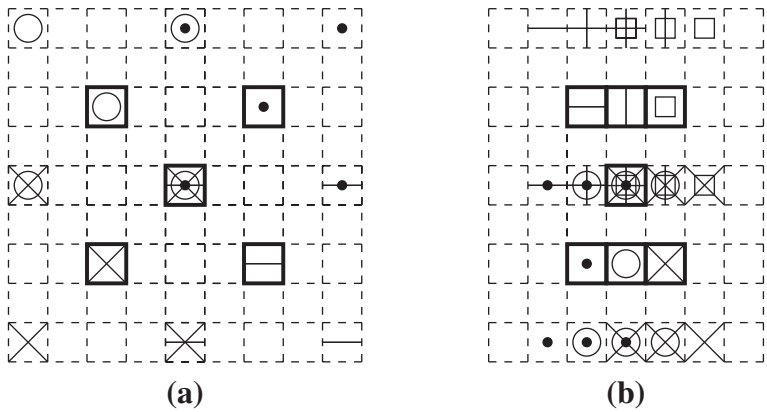


Figure 3.146 We mark with a different symbol the neighbourhood of each pixel in the original neighbourhood.

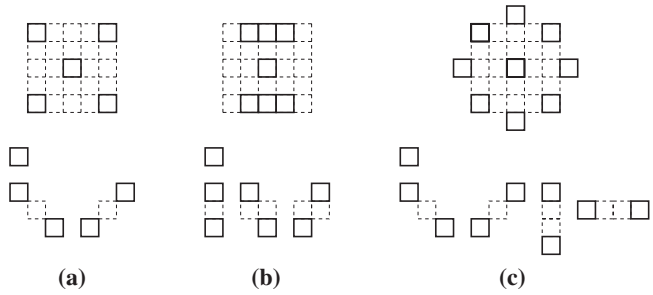


Figure 3.147 The cliques that correspond to the Markov neighbourhoods shown in the first row.

3.7 Gibbs Distributions

What is a Gibbs distribution?

A Gibbs distribution is a function that specifies, in terms of **clique potentials**, the **joint** probability density function of a particular combination X of values to exist over the whole grid. In general it has the form

$$p(X) = \frac{1}{Z} e^{\Sigma_{\text{all\_types\_of\_clique}} \Sigma_{\text{all\_cliques\_of\_this\_type}} x_{i_1} x_{i_2} \ldots x_{i_{c_k}} U_k(x_{i_1} x_{i_2} \ldots x_{i_{c_k}})} \tag{3.396}$$

where c_k is the number of pixels in clique of type k, x_{i_j} is the value of the i_jth pixel in the clique, and $U_k(x_{i_1} x_{i_2} \ldots x_{i_{c_k}})$ is the potential of clique of type k. The normalising constant Z is called the **partition function** and it ensures that $p(X)$ is a probability density, i.e. upon integration over all possible combinations of values X it sums to 1, and its value for any particular combination is in the range $[0, 1]$.

What is a clique potential?

A clique potential is a function of the pixel values that make up the clique. It specifies the interaction between the members of a clique. Its functional form and the numerical parameters on which it depends may be different for different clique types.

Example 3.131

Write the form of the Gibbs distribution for an $N \times N$ image and the neighbourhood structure of Figure 3.145a for the simplest clique potentials you can think of.

There are only three types of clique for such a neighbourhood: singletons, next-door neighbours along the horizontal direction, and next-door neighbours along the vertical direction. Let us call the corresponding potentials for these cliques U_0, U_1 and U_2, respectively. In the simplest case these will be just some numerical constants. Then upon application of (3.396) we have:

$$p(X) = p(x_{11}, x_{12}, \ldots, x_{1N}, x_{21}, \ldots, x_{2N}, \ldots, x_{N1}, \ldots, x_{NN})$$

(Continued)

Example 3.131 (Continued)

$$= \frac{1}{Z} e^{\sum_i \sum_j x_{ij} U_0 + \sum_i \sum_j x_{ij} x_{i+1,j} U_1 + \sum_i \sum_j x_{ij} x_{i,j+1} U_2}. \quad (3.397)$$

In the expression above we have used x_{ij} to indicate the grey value of pixel (i, j).

The partition function Z is given by:

$$Z = \sum_X e^{\sum_i \sum_j x_{ij} U_0 + \sum_i \sum_j x_{ij} x_{i+1,j} U_1 + \sum_i \sum_j x_{ij} x_{i,j+1} U_2}. \quad (3.398)$$

The summation here is over all configurations X, i.e. all possible images of size $N \times N$ with pixel values in the legitimate range.

The summations of the binary cliques in (3.397) involve pixels that might be at the boundary of the image and therefore they may not have neighbours. There are two ways to deal with these pixels: we either omit them from the sums, i.e. we consider only pixels in the image that have neighbours inside the image, or we assume that the image repeats itself periodically in both directions, so that the boundary pixels have as neighbours the wrapped around pixels from the other end of the image. This is perfectly legitimate, since we assume the texture to be stationary, i.e. we assume that the same pattern fills up the whole image.

Example 3.132

Consider an image of size 4×4 with 3-bit pixels, i.e. pixels that take grey values in the range $[0, 7]$. Compute how many different configurations X you will have appearing in the summation on the right-hand side of Equation (3.398).

If we have 16 pixels, and if each pixel can take one of 8 possible values, then we have 8^{16} possible combinations, since each value taken by the first pixel may be combined with each value taken by the second pixel, and each value taken by the third pixel, and so on. This number is 2^{48} which is approximately equal to 281 475 billion configurations!

Example 3.133

How many different binary images of size 3×3 can we have?

A 3×3 image has nine pixels, each of which may take two values. So, the total number of combinations we can have is $2^9 = 512$ different images.

Example B3.134

Assume that pixel $(1, 1)$ in a 3×3 binary image has value 1, and pixel $(3, 2)$ has value 0. The other pixels are allowed to take either value 0 or 1. Draw all the different images that are possible.

Since we have seven free pixels each of which may take any one of two values, we may have $2^7 = 128$ different combinations. These are shown in Figure 3.148.

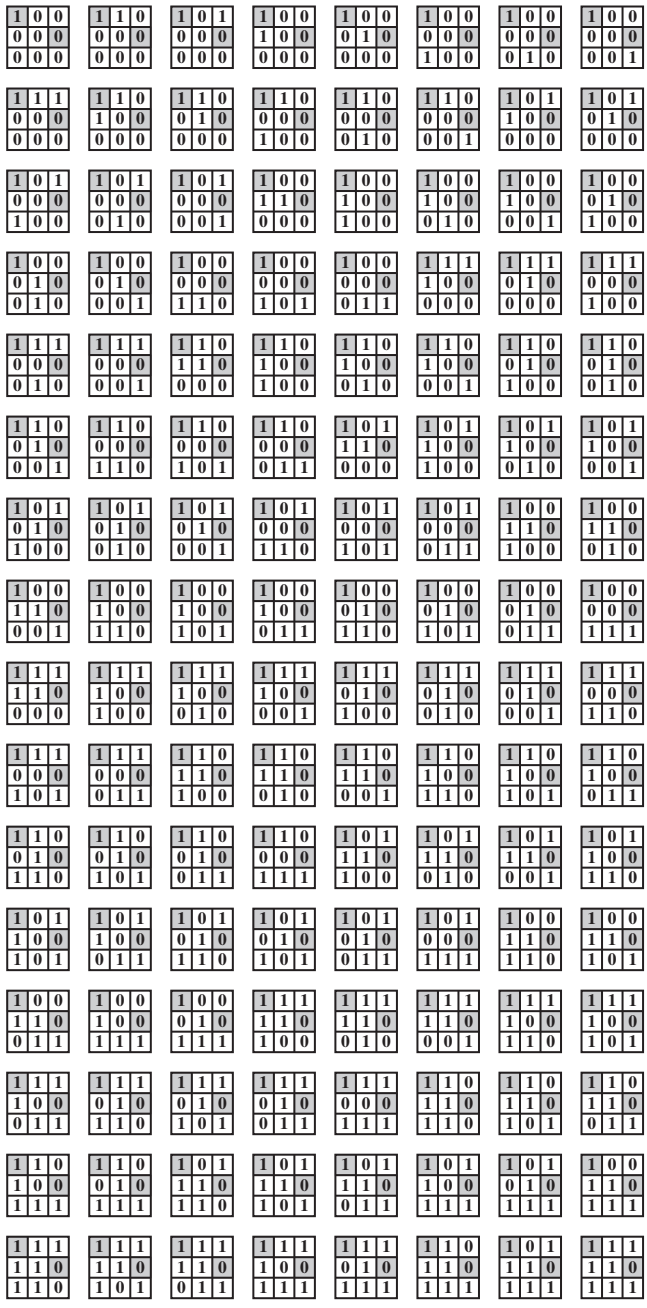


Figure 3.148 All possible binary 3×3 images we can have when the grey pixels carry fixed values.

Grey marks the two pixels with the fixed values. These configurations were constructed according to the following logic.

1. *There is only one configuration with all seven pixels 0.*

(Continued)

Example B3.134 (Continued)

2. *If we replace one of those 0s by 1, there are seven different places where we can put that 1, so there must be seven configurations with six 0s and one 1.*
3. *For every configuration with one 1, we can add another 1 in one of the remaining six positions to create a configuration with two 1s and five 0s. However, to avoid counting the same configuration twice, we follow a sequential order of the free pixels, from top left to bottom right in a raster scan. We may place the second 1 in a position* ***after*** *the first 1, but never before. So, from the first configuration with one 1, we can create six configurations with two 1s, from the second configuration with one 1, we can create five configurations with two 1s, from the third four, and so on. The total number of distinct configurations with two 1s and five 0s we may create is* $6 + 5 + 4 + 3 + 2 + 1 = 21$.
4. *For every configuration with two 1s, we may create a configuration with three 1s by placing the third 1 in one of the remaining five places. However, we shall have multiple configurations, unless we follow the sequential numbering of pixels as above, and avoid placing the third 1 in a position before the two positions already occupied by 1s. As for every one of the 21 configurations with two 1s we have five options to place the third 1, we may have* 21×5 *options in total. However, not all of them will be distinct because one cannot distinguish between a 1 that was there originally and a 1 that was added to make the total number of 1s three. As each one of the 1s could be considered to be the added 1, we must divide this number by 3 to deduce the total number of distinct configurations. This leads to* $21 \times 5/3 = 35$ *configurations with three 1s and four 0s.*
5. *If now we duplicate the configurations we created and make all 1s 0 and all 0s 1, we shall have the remaining configurations as follows: 35 configurations with four 1s and three 0s. 21 configurations with five 1s and two 0s. 7 configurations with six 1s and one 0. 1 configuration with seven 1s.*

Can we have a Markov random field with only singleton cliques?

If a field has only singleton cliques, it may still be a Markov field, in the sense that it may still be described by a Gibbs distribution where, however, the neighbours do not exert any influence on the value of a pixel. Nevertheless, there may still be bias in the "coin" with which we create it, although the coin will not be biased by the values of the neighbours, but by some other "external" to the field factor. This bias may make some configurations more probable than others. This is in contrast to a field which has no Gibbs structure, for which all configurations are equally likely. See Examples 3.135 and 3.136.

Example B3.135

Assume that a 3×3 **binary image is an uncorrelated random field. Compute the joint probability density function for pixel** $(1, 1)$ **to have value 1 and pixel** $(3, 2)$ **to have value 0. Confirm your answer by referring to the answer of Example 3.134 shown in Figure 3.148.**

Since the random field is uncorrelated, the probability of pixel $(1,1)$ *to take value 1 is* $1/2$*, and the probability of pixel* $(3,2)$ *to take value 0 is also* $1/2$*. The joint probability of these two values to arise in these two positions is, therefore,* $1/2 \times 1/2 = 1/4$*.*

Figure 3.148 shows all possible configurations for which pixel $(1,1)$ *has value 1 and pixel* $(3,2)$ *has value 0. One can have three other similar lots of configurations where the fixed values of the two pixels marked by grey are 1 and 1, 0 and 1, or 0 and 0, respectively. Thus, the configurations shown in Figure 3.148 constitute one quarter of all possible configurations. This means that if we pick a* 3×3 *binary image at random, the prior probability of picking an image with pixel* $(1,1)$ *having value 1 and pixel* $(3,2)$ *having value 0 is one in four, which is in agreement with the previous calculation.*

Example B3.136

Assume that a 3×3 **binary image is a Markov random field with cliques consisting of singletons only. The clique potential is** $U_0 = 1$**. Use the Gibbs distribution of Equation (3.397) to compute the joint probability density function of pixel** $(1,1)$ **to have value 1 and pixel** $(3,2)$ **to have value 0. Compare your answer with that of Example 3.135.**

In Equation (3.397) we set $U_0 = 1$*,* $U_1 = U_2 = 0$*. Then the joint probability density function of any* 3×3 *binary image is given by*

$$p(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = \frac{1}{Z} e^{\sum_{ij} x_{ij}} \tag{3.399}$$

where

$$Z = \sum_{\text{all\_configurations}} \mathrm{e}^{\sum_{ij} x_{ij}}. \tag{3.400}$$

Let us compute Z first. The summat is equal to e^M *where M is the number of 1s in each configuration. Let us make use of Figure 3.148 and work out the number of 1s in each possible configuration, by assigning different values to the grey pixels and counting each time the 1s in each configuration we create.*

For $x_{11} = 0$ *and* $x_{32} = 0$ *we have*

1 *configuration with zero* 1s

7 *configurations with one* 1

21 *configurations with two* 1s

35 *configurations with three* 1s

35 *configurations with four* 1s

21 *configurations with five* 1s

7 *configurations with six* 1s

1 *configuration with seven* 1s.

(Continued)

Example B3.136 (Continued)

For $x_{11} = 1$ *and* $x_{32} = 0$ *or for* $x_{11} = 0$ *and* $x_{32} = 1$ *we have*

1	*configuration*	*with*	*one* 1
7	*configurations*	*with*	*two* 1s
21	*configurations*	*with*	*three* 1s
35	*configurations*	*with*	*four* 1s
35	*configurations*	*with*	*five* 1s
21	*configurations*	*with*	*six* 1s
7	*configurations*	*with*	*seven* 1s
1	*configuration*	*with*	*eight* 1s.

For $x_{11} = 1$ *and* $x_{32} = 1$ *we have*

1	*configuration*	*with*	*two* 1s
7	*configurations*	*with*	*three* 1s
21	*configurations*	*with*	*four* 1s
35	*configurations*	*with*	*five* 1s
35	*configurations*	*with*	*six* 1s
21	*configurations*	*with*	*seven* 1s
7	*configurations*	*with*	*eight* 1s
1	*configuration*	*with*	*nine* 1s.

So, over all, we have

1	*configuration*	*with*	*zero* 1s
9	*configurations*	*with*	*one* 1
36	*configurations*	*with*	*two* 1s
84	*configurations*	*with*	*three* 1s
126	*configurations*	*with*	*four* 1s
126	*configurations*	*with*	*five* 1s
84	*configurations*	*with*	*six* 1s
36	*configurations*	*with*	*seven* 1s
9	*configurations*	*with*	*eight* 1s
1	*configuration*	*with*	*nine* 1s.

So,

$$Z = 1 \times e^0 + 9 \times e^1 + 36 \times e^2 + 84 \times e^3 + 126 \times e^4 + 126 \times e^5 + 84 \times e^6 + 36 \times e^7 + 9 \times e^8 + 1 \times e^9$$
$$\Rightarrow Z = 135,856.59878. \tag{3.401}$$

The joint probability density function $p(x_{11} = 1, x_{32} = 0)$ may be computed by integrating the Gibbs distribution over all possible values of all other variables:

$$p(x_{11} = 1, x_{32} = 0) = \sum_{x_{12}}\sum_{x_{13}}\sum_{x_{21}}\sum_{x_{22}}\sum_{x_{23}}\sum_{x_{31}}\sum_{x_{33}} p(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = \frac{1}{Z}\sum_{all\_configurations\_with\ clamped\_values\_of\ x_{11}\_and\_x_{32}} e^{\sum_{ij} x_{ij}}. \tag{3.402}$$

This sum may be computed by using the table of values we extracted earlier. We obtain:

$$p(x_{11} = 1, x_{32} = 0) = \frac{1}{135856.59878}[1 \times e^1 + 7 \times e^2 + 21 \times e^3 + 35 \times e^4 + 35 \times e^5 + 21 \times e^6 + 7 \times e^7 + 1 \times e^8] = 0.1966. \tag{3.403}$$

Note that this value is different from that obtained in Example 3.135, which was 0.25. This is because in this case, there is an external bias that influences the value a single pixel takes, even though in both cases neighbouring pixels do not influence the values of each other directly.

Example B3.137

For a 3×3 binary image compute the joint probability density function $p(x_{11} = 1, x_{32} = 0)$ if the image structure is described by the Gibbs distribution of Equation (3.397) with $U_0 = 0$ and $U_1 = 1$ and $U_2 = -1$. Assume that the image is repeated ad infinitum in all directions.

The joint probability density function $p(x_{11} = 1, x_{32} = 0)$ can be found by integrating the Gibbs distribution over all possible values of all other variables:

$$p(x_{11} = 1, x_{32} = 0) = \sum_{x_{12}}\sum_{x_{13}}\sum_{x_{21}}\sum_{x_{22}}\sum_{x_{23}}\sum_{x_{31}}\sum_{x_{33}} p(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = \frac{1}{Z}\sum_{all\_configurations\_with\ clamped\_values\_of\ x_{11}\_and\_x_{32}} e^{\sum_{ij}(x_{ij}x_{i+1,j}U_1 + x_{ij}x_{i,j+1}U_2)}. \tag{3.404}$$

We note that unless both pixels in a clique have value 1, the corresponding term in the exponent of the above equation will be 0.

We count, therefore, first the number of cliques we have consisting of two 1s next to each other along the horizontal direction. To ensure that all pixels have horizontal neighbours to the right, we must imagine that the first column of the image is repeated once more to the right of the image.

(Continued)

Example B3.137 (Continued)

Table 3.33 shows the number of horizontal cliques $(1,1)$ *found in the configurations of Figure 3.148. The entries of the table correspond to the spatial arrangement of the different configurations in the figure.*

Table 3.33 The number of **horizontal** pairs $(1,1)$ in the configurations of Figure 3.148. The spatial arrangement of these numbers follows the spatial arrangement of the images in the figure.

$x_{11}=1, x_{32}=0$							
0	1	1	0	0	0	0	0
3	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0
0	0	1	1	1	3	3	3
3	3	2	1	1	1	1	1
1	2	2	2	2	1	1	1
1	1	1	2	2	2	1	1
1	1	1	1	1	1	1	3
4	3	3	3	3	3	3	4
4	4	2	2	2	2	2	2
2	2	2	4	2	2	2	2
2	2	2	2	2	4	2	2
2	3	3	4	4	4	4	4
4	4	4	4	6	3	3	3
4	4	3	3	3	4	4	4
5	5	5	6	6	5	5	7

Table 3.34 The number of **vertical** pairs $(1,1)$ in the configurations of Figure 3.148. The spatial arrangement of these numbers follows the spatial arrangement of the images in the figure.

$x_{11}=1, x_{32}=0$							
0	0	0	1	0	1	0	0
0	1	1	1	1	0	1	0
1	0	1	1	3	1	1	1
1	0	1	1	0	1	1	1
1	1	2	3	2	1	2	3
1	2	1	1	1	3	1	2
1	1	1	1	2	1	3	2
1	3	3	1	2	1	1	1
2	3	2	2	2	3	2	2
2	2	4	4	2	4	3	2
4	2	3	2	3	2	2	3
4	2	2	2	2	2	4	3
2	3	2	4	4	3	4	4
3	4	3	4	3	6	4	4
4	4	4	4	3	4	3	4
6	5	5	5	5	6	5	7

Then we count the number of cliques that consist of two 1s along the vertical direction. To ensure that all pixels have vertical neighbours below them, we repeat the first row of each image at its bottom, so that pixels at the bottom row have neighbours below them as well.

The number of vertical $(1,1)$ *pairs for each configuration of Figure 3.148 are shown in the corresponding position of Table 3.34.*

Tables 3.35–3.40 show the numbers of vertical and horizontal cliques consisting of pixels both of which have value 1 for all remaining possible images.

To compute the joint probability density function using Equation (3.404) we compute first the numerator. This is the sum of as many exponentials as we have possible configurations with values $x_{11}=1$ *and* $x_{32}=0$*, i.e. all the entries of Figure 3.148. Since* $U_1=1$ *and* $U_2=-1$*, each*

Table 3.35 The number of **horizontal** pairs $(1,1)$ in the configurations of Figure 3.148 if the two grey pixels have both value 0.

$x_{11}=0, x_{32}=0$							
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	3
2	1	1	1	1	1	1	2
2	2	1	1	1	1	1	1
1	1	1	3	1	1	1	1
1	1	1	1	1	3	2	2
2	3	3	2	2	2	2	2
2	2	2	2	4	2	2	2
3	3	2	2	2	3	3	4
3	3	3	4	4	4	4	5

Table 3.36 The number of **vertical** pairs $(1,1)$ in the configurations of Figure 3.148 if the two grey pixels have both value 0.

$x_{11}=0, x_{32}=0$							
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	0	1	0	1	0	0	0
1	0	0	0	0	0	1	0
1	1	1	1	1	0	1	3
1	1	0	1	0	1	0	1
0	1	1	0	1	1	1	1
0	1	1	0	1	0	1	0
1	1	1	1	1	3	2	1
1	2	2	3	1	2	1	1
3	1	3	1	1	1	1	1
2	1	1	1	2	1	2	1
1	1	1	2	3	2	2	2
2	3	2	4	2	4	2	3
2	3	2	2	2	2	2	2
4	3	4	3	4	4	3	5

exponential has the form e^{H-V}, where H and V are the counts of the horizontal and vertical pairs in the configuration, respectively, in which both pixels have value 1. We use the entries of Tables 3.33 and 3.34 together, reading the value of H from the first table and the value of V from the second, and compute this numerator to be 254.45889.

To compute the denominator of (3.404), we use Equation (3.398). We observe that this is going to be a sum of as many exponentials as we have possible configurations (i.e. 512). Each exponential has the same form as before, only now we have to make use of the remaining tables in pairs. Tables 3.35 and 3.36 will be used together, to produce all those terms that come from configurations in which both pixels of interest have value 0. Tables 3.37 and 3.38 will be used together, to produce all those terms that come from configurations in which both pixels of interest have value 1. Finally, Tables 3.39 and 3.40 will be used together, to produce all those terms that come from configurations in which the two pixels of interest have exchanged values.

If we do that, we find that $Z = 917.44455$.

Therefore, the joint probability density function we had to compute is: $p(x_{11}=1, x_{32}=0) = 254.45889/917.44455 = 0.2774$.

(Continued)

Example B3.137 (Continued)

Table 3.37 The number of **horizontal** pairs (1, 1) in the configurations of Figure 3.148 if the two grey pixels have both value 1.

$x_{11} = 1, x_{32} = 1$							
0	1	1	1	1	0	0	0
3	2	2	1	1	1	2	2
1	1	1	3	1	1	1	1
1	1	1	1	1	4	4	3
3	3	4	2	2	2	2	2
2	2	2	2	4	2	2	2
2	2	2	2	2	2	3	3
3	2	2	2	2	2	2	3
6	4	4	4	4	4	4	4
4	4	4	4	4	3	3	3
3	3	3	4	4	4	4	3
3	3	3	3	3	4	4	4
4	4	4	6	6	6	5	5
5	5	5	5	6	5	5	5
5	5	5	5	5	5	5	6
7	7	7	7	7	7	7	9

Table 3.38 The number of **vertical** pairs (1, 1) in the configurations of Figure 3.148 if the two grey pixels have both value 1.

$x_{11} = 1, x_{32} = 1$							
0	0	1	1	0	1	0	1
1	1	1	1	1	1	2	1
2	1	3	1	3	1	2	1
1	1	1	2	1	2	2	2
2	3	2	3	2	2	2	3
2	2	2	2	2	4	2	4
2	2	3	2	4	3	3	2
2	3	4	2	2	2	2	2
3	4	3	4	3	4	4	3
4	4	4	4	3	4	4	3
4	3	4	3	4	3	4	4
6	4	3	4	4	4	4	4
3	4	3	5	5	5	5	6
5	5	5	6	5	6	5	5
5	5	5	6	5	6	5	5
7	7	7	7	7	7	7	9

Table 3.39 The number of **horizontal** pairs (1, 1) in the configurations of Figure 3.148 if the grey pixels exchange their values.

$x_{11} = 0, x_{32} = 1$							
0	0	0	1	1	0	0	0
1	1	1	0	0	0	1	1
0	0	0	3	1	1	1	1
1	1	1	1	1	2	2	1
1	1	3	1	1	1	1	1
1	1	1	1	3	1	1	1
1	1	1	1	1	1	3	3
3	2	2	2	2	2	2	3
4	2	2	2	2	2	2	2
2	2	3	3	3	2	2	2
2	2	2	3	3	3	3	2
2	2	2	2	2	3	4	4
4	4	4	4	4	4	3	3
3	3	3	3	4	4	4	4
4	4	4	4	4	4	4	6
5	5	5	5	5	6	6	7

Table 3.40 The number of **vertical** pairs (1, 1) in the configurations of Figure 3.148 if the grey pixels exchange their values.

$x_{11} = 0, x_{32} = 1$							
0	0	1	0	0	0	0	1
1	0	1	0	1	1	1	1
1	1	3	0	1	0	1	0
1	1	0	1	1	1	2	1
2	3	1	1	1	1	1	3
2	1	1	2	1	2	1	3
1	2	3	1	3	3	1	1
1	1	2	1	1	1	2	1
2	2	2	3	2	4	4	2
3	4	2	3	2	2	2	2
3	2	4	2	2	2	3	2
4	3	2	3	4	3	2	2
2	2	2	3	4	4	3	4
4	4	4	6	4	4	3	4
3	4	3	4	4	4	4	3
5	5	6	5	6	5	5	7

Example B3.138

For a 3×3 binary image compute the joint probability density function $p(x_{11} = 1, x_{32} = 0)$ if the image structure is described by the Gibbs distribution of Equation (3.397) with $U_0 = -1$, $U_1 = 1$ and $U_2 = -1$. Assume that the image is repeated ad infinitum in all directions.

The joint probability density function $p(x_{11} = 1, x_{32} = 0)$ now has the form:

$$p(x_{11} = 1, x_{32} = 0) = \sum_{x_{12}} \sum_{x_{13}} \sum_{x_{21}} \sum_{x_{22}} \sum_{x_{23}} \sum_{x_{31}} \sum_{x_{33}} p(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = \frac{1}{Z} \sum_{all\_configurations\_with\ clamped\_values\_of\ x_{11}\_and\_x_{32}} e^{\sum_{ij} x_{ij}(U_0 + x_{i+1,j} U_1 + x_{i,j+1} U_2)}. \quad (3.405)$$

The only difference with Example 3.137 is that now we have to take into consideration the single pixels that have non-zero value as well. Table 3.41 gives for each configuration of Figure 3.148 the number of pixels with value 1. Tables 3.42–3.44 give the number of pixels with value 1 for all other possible configurations. Note that Table 3.44 is the same as Table 3.41 since here the relative position of the pixels does not matter; we only count the number of 1s in each configuration.

Table 3.41 The number of single pixels in each configuration of Figure 3.148 with value 1.

$x_{11} = 1, x_{32} = 0$							
1	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	6	6	6	6	6
6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	8

Table 3.42 The number of single pixels with value 1 in each configuration produced from Figure 3.148 by setting both highlighted pixels to 0.

$x_{11} = 0, x_{32} = 0$							
0	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
4	4	4	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	7

(Continued)

Example B3.138 (Continued)

Table 3.43 The number of single pixels with value 1 in each configuration produced from Figure 3.148 by setting both highlighted pixels to 1.

$x_{11}=1, x_{32}=1$							
2	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6
6	6	6	7	7	7	7	7
7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	9

Table 3.44 The number of single pixels with value 1 in each configuration produced from Figure 3.148 by exchanging the values of the highlighted pixels.

$x_{11}=0, x_{32}=1$							
1	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	6	6	6	6	6
6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	8

To compute the joint probability density function now we have to add terms of the form e^{-S+H-V} *where S counts the number of 1s in each configuration and its value must be read from Tables 3.41–3.44, and H and V count the number of pairs of 1s in the horizontal and vertical directions respectively, and their values must be read from Tables 3.33–3.40. For the numerator we must use only Tables 3.33 and 3.34, while for the denominator all possible configurations must be taken into consideration. We find that in this case* $p(x_{11}=1, x_{32}=0) = 5.73381/24.94446 = 0.229863$.

Example B3.139

For the Gibbs distribution of Equation (3.397) identify the most probable configuration of a binary 3×3 **image that repeats periodically in all directions, for the following sets of clique potentials:**

$U_0 = 1$ $U_1 = 0$ $U_2 = 0$

$U_0 = 0$ $U_1 = 1$ $U_2 = 2$

$U_0 = -1$ $U_1 = 1$ $U_2 = -1$

$U_0 = 0$ $U_1 = 1$ $U_2 = -1$

$U_0 = 0$ $U_1 = -1$ $U_2 = 1$.

We use the entries of Tables 3.33–3.44 to compute the numerator of Equation (3.397) for each set of clique potentials. We do not need to compute the denominator if we are simply interested in identifying the most probable configuration. However, if we want to know how probable a configuration is, we do need the value of Z too. Table 3.45 presents the results of these calculations, and the most probable configurations identified in each case.

In the case $U_0 = 0, U_1 = 1, U_2 = 2$ *both vertical and horizontal cliques of pixels with identical values are encouraged, producing a flat image consisting of 1s only. In the case* $U_0 = 0, U_1 = 1, U_2 = -1$ *horizontal cliques of pixels with identical values are encouraged, but the vertical cliques are encouraged to consist of pixels with different values. Six equally probable configurations were identified in this case. Note the difference with the case* $U_0 = -1, U_1 = 1, U_2 = -1$ *which is only different in the value of* U_0*: the non-zero value of this parameter creates a bias that makes the black flat configuration the most probable one. The case* $U_0 = 0, U_1 = -1, U_2 = 1$ *has also six equiprobable configurations with vertical stripes instead of horizontal. The reason of the existence of six equiprobable configurations is the toroidal boundary conditions we use, which make all these configurations equivalent. These probabilities should be compared with* $1/512 = 0.00195$ *which is the probability of picking a particular image if the configuration space were totally unstructured, i.e. if all images were equally probable.*

Table 3.45 Most probable configurations for different sets of clique potentials. The values of the numerator and the probability p correspond to the most probable configuration which is shown in the last column. In the last two cases, six configurations are equally probable, due to the toroidal boundary conditions we use.

Clique potentials	Z	Numerator	p	Most probable configuration
$U_0 = 1, U_1 = U_2 = 0$	135 856.6	8100	0.060	111
				111
				111
$U_0 = 0, U_1 = 1, U_2 = 2$	544×10^9	532×10^9	0.978	111
				111
				111
$U_0 = -1, U_1 = 1, U_2 = -1$	24.94	1	0.040	000
				000
				000
$U_0 = 0, U_1 = 1, U_2 = -1$	917.44	20.09	0.022	111 000 000
				000 111 000
				000 000 111
				111 111 000
				111 000 111
				000 111 111
$U_0 = 0, U_1 = -1, U_2 = 1$	917.44	20.09	0.022	100 010 001
				100 010 001
				100 010 001
				110 101 011
				110 101 011
				110 101 011

Example B3.140

Consider the most probable configurations identified in Example 3.139 shown in Table 3.45. For each one of them compute the joint probability density function $p(x_{ij} = 1, x_{i+2,j+1} = 0)$ assuming periodic boundary conditions. Compare the values you compute with those computed in Examples 3.136, 3.137 and 3.138. Comment on the answers you get.

The joint probability density function we compute here is spatial, i.e. computed from a single image, i.e. computed over all pairs of pixels in the image with values 1 and 0 in the same relative position from each other. The joint probability density function we computed in Examples 3.136, 3.137 and 3.138 was in the ensemble sense. The two values that correspond to the same set of clique potentials would have been the same if the images were ergodic. (For the concept of ergodicity, see Book I [75].)

Table 3.46 Values of probability $p(x_{ij} = 1, x_{i+2,j+1} = 0)$ for the most probable configurations of example 3.139. In the last column the values computed using the method of examples 3.136, 3.137 and 3.138.

Clique potentials			$p(x_{ij} = 1, x_{i+2,j+1} = 0)$	Ensemble calculation
$U_0 = 1,$	$U_1 = 0$	$U_2 = 0$	0	0.1966
$U_0 = 0,$	$U_1 = 1,$	$U_2 = 2$	0	0.0026
$U_0 = -1,$	$U_1 = 1,$	$U_2 = -1$	0	0.2299
$U_0 = 0,$	$U_1 = 1,$	$U_2 = -1$	1/3	0.2774
$U_0 = 0,$	$U_1 = -1,$	$U_2 = 1$	1/3	0.2774

What is the relationship between Gibbs distributions and co-occurrence matrices?

A co-occurrence matrix may be thought of as a *marginal* of the joint probability of the whole configuration of grey values to arise, like for example the marginal computed in Example 3.140. That marginal may be computed in two ways: in Examples 3.136–3.138 it was computed over the ensemble of all versions of the image. In Example 3.140, it was computed over the same image, allowing the pair of pixels to scan the whole image. It is this version that corresponds to the co-occurrence matrix, as it captures the properties of the specific texture pattern, which is assumed stationary.

What is the relationship between the clique potentials and the Markov parameters?

This relationship emerges naturally when one works out the equivalence of Gibbs distributions and MRFs. This equivalence is known as the **Hammersley–Clifford theorem**. For its proof see Example 3.141 and Box 3.26. Then Example 3.145 shows that the Markov parameters are the same as the clique potentials, when the latter are considered to be constants.

Example B3.141

Prove that if we have a Gibbs distribution defined over a configuration of pixel values we may describe the same configuration by a Markov random field model.

Let us first imagine that all pixels are labelled sequentially, so that we avoid having double indices. This is only for convenience of notation and it has no other implication for the proof. Let us call the total number of pixels M and let us denote the joint configuration of their values $x_1, x_2, \dots, x_M$, by X.

We assume that we know the joint probability density function of all pixels $p(X) = p(x_1, x_2, \dots, x_M)$ and that this is a Gibbs distribution as defined by Equation (3.396). Let us consider one pixel, x_i. Its conditional probability density function is $p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_M)$. To prove that we are dealing with an Markov random field this conditional probability density function must depend only on the neighbours of pixel i and not on all other pixels in the image. From Bayesťheorem we know that $p(A, B) = p(A|B)p(B)$, so

$$p(x_1, x_2, \dots, x_M) = p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_M) \times p(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_M)$$

$$\Rightarrow \qquad p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_M) = \frac{p(x_1, x_2, \dots, x_M)}{p(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_M)}. \tag{3.406}$$

We also know that $p(A) = \sum_{all\_values\_of\_B} p(A, B)$. So, we may re-introduce x_i in the denominator of the above expression, provided we sum over all possible values of it:

$$p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_M) = \frac{p(x_1, x_2, \dots, x_M)}{\sum_{x_i} p(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_M)}. \tag{3.407}$$

Now let us substitute the joint probability density function that appears in the numerator and the denominator of the above equation from the Gibbs formula (3.396):

$$p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_M) =$$

$$\frac{\frac{1}{Z} e^{\sum_{\text{all\_types\_of\_clique}} \sum_{\text{all\_cliques\_of\_this\_type}} x_{i_1} x_{i_2} \dots x_{i_{c_k}} U_k}}{\frac{1}{Z} \sum_{x_i} e^{\sum_{\text{all\_types\_of\_clique}} \sum_{\text{all\_cliques\_of\_this\_type}} x_{i_1} x_{i_2} \dots x_{i_{c_k}} U_k}}. \tag{3.408}$$

Note that the quantities that appear in the numerator and the denominator are products of exponentials. Each such factor contains the values of some pixels that belong to the same clique. All those factors that do not involve pixel x_i will cancel, and we shall have left in the numerator and the denominator only those factors that refer to cliques that contain pixel x_i, i.e. the neighbourhood of x_i. This shows that the conditional probability on the left-hand side of (3.408) depends only on the neighbours of pixel x_i, i.e. that the lattice possesses the Markovian property and thus it is a Markov random field.

Example B3.142

Consider a joint probability density function $p(x_1, x_2, \dots, x_M)$. Show that you may write

$$\frac{p(X)}{p(Y)} = \prod_{i=1}^{M} \frac{p(x_i|x_1, \dots, x_{i-1}, y_{i+1}, \dots, y_M)}{p(y_i|x_1, \dots, x_{i-1}, y_{i+1}, \dots, y_M)} \tag{3.409}$$

where X stands for $x_1, \dots, x_M$ and Y stands for $y_1, \dots, y_M$.
From Bayesítheorem we may write

$$p(X) = p(x_1, \dots, x_M) = p(x_M|x_1, \dots, x_{M-1}) \times p(x_1, \dots, x_{M-1}). \tag{3.410}$$

Let us apply this formula for $x_M = y_M$:

$$p(x_1, \dots, x_{M-1}, y_M) = p(y_M|x_1, \dots, x_{M-1}) \times p(x_1, \dots, x_{M-1}). \tag{3.411}$$

Solving for $p(x_1, \dots, x_{M-1})$ yields:

$$p(x_1, \dots, x_{M-1}) = \frac{p(x_1, \dots, x_{M-1}, y_M)}{p(y_M|x_1, \dots, x_{M-1})}. \tag{3.412}$$

Substitute from (3.412) into (3.410) to obtain:

$$p(X) = \frac{p(x_M|x_1, \dots, x_{M-1})}{p(y_M|x_1, \dots, x_{M-1})} \times p(x_1, \dots, x_{M-1}, y_M). \tag{3.413}$$

Next, apply Bayesítheorem again, to write:

$$p(x_1, \dots, x_{M-1}, y_M) = p(x_{M-1}|x_1, \dots, x_{M-2}, y_M) \times p(x_1, \dots, x_{M-2}, y_M). \tag{3.414}$$

Apply this formula for $x_{M-1} = y_{M-1}$

$$p(x_1, \dots, y_{M-1}, y_M) = p(y_{M-1}|x_1, \dots, x_{M-2}, y_M) \times p(x_1, \dots, x_{M-2}, y_M) \tag{3.415}$$

which yields

$$p(x_1, \dots, x_{M-2}, y_M) = \frac{p(x_1, \dots, y_{M-1}, y_M)}{p(y_{M-1}|x_1, \dots, x_{M-2}, y_M)}. \tag{3.416}$$

Substitute from (3.416) into (3.414) to obtain:

$$p(x_1, \dots, x_{M-1}, y_M) = \frac{p(x_{M-1}|x_1, \dots, x_{M-2}, y_M)}{p(y_{M-1}|x_1, \dots, x_{M-2}, y_M)} \times p(x_1, \dots, y_{M-1}, y_M). \tag{3.417}$$

Then substitute from (3.417) into (3.413):

$$p(X) = \frac{p(x_M|x_1, \dots, x_{M-1})}{p(y_M|x_1, \dots, x_{M-1})} \times \frac{p(x_{M-1}|x_1, \dots, x_{M-2}, y_M)}{p(y_{M-1}|x_1, \dots, x_{M-2}, y_M)} \times p(x_1, \dots, y_{M-1}, y_M). \tag{3.418}$$

This process may be repeated until the last factor on the right-hand side of (3.418) contains only y_i variables, i.e. until the last factor is $p(Y)$. Then, by dividing both sides of the equation with $p(Y)$, we obtain (3.409).

Example B3.143

We define a function $Q(X)$ as

$$Q(X) \equiv \ln \frac{p(X)}{p(O)} \tag{3.419}$$

where X is configuration $x_1, \dots, x_M$, as defined in Example 3.141, O is a particular realisation of this configuration where all pixels take value 0, and it is assumed that such a configuration has non-zero probability of existing, i.e. $p(O) \neq 0$. Prove that

$$e^{Q(X)-Q(X_i)} = \frac{p(x_i|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_M)}{p(0|x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_M)} \tag{3.420}$$

where X_i is the configuration where all pixels have the same value as in configuration X, except pixel i which now has value 0.

We start by re-arranging Equation (3.419) and applying it for configuration X_i:

$$e^{Q(X)} = \frac{p(X)}{p(O)} \qquad e^{Q(X_i)} = \frac{p(X_i)}{p(O)}. \tag{3.421}$$

We divide the above two equations by parts to obtain:

$$e^{Q(X)-Q(X_i)} = \frac{p(X)}{p(X_i)}. \tag{3.422}$$

We then substitute $p(X)$ and $p(X_i)$ from the result proven in Example 3.142, i.e. from Equation (3.409). All factors, except the factors that have as conditioned variable x_i in the numerator and 0 in the denominator, cancel. Equation (3.420) then follows.

Example B3.144

Show that function $Q(X)$ of Example 3.143 may be expanded in a unique way as follows:

$$Q(X) = \sum_{1\le i\le M} x_i G_i(x_i) + \sum_{1\le i<j\le M}\sum x_i x_j G_{ij}(x_i, x_j) + \sum_{1\le i<j<k\le M}\sum\sum x_i x_j x_k G_{ijk}(x_i, x_j, x_k)$$
$$+ \dots + x_1 x_2 \dots x_M G_{12\dots M}(x_1, x_2, \dots, x_M). \tag{3.423}$$

Consider first the state $(x_1, 0, \dots, 0)$, where $x_1 \neq 0$, and apply Equation (3.423). We obtain:

$$Q(x_1, 0, \dots, 0) = x_1 G_1(x_1). \tag{3.424}$$

This expression uniquely determines the unknown function $G_1(x_1)$. Consider then all configurations where all pixels have value 0, except one pixel. This way you can define uniquely all functions $G_i(x_i)$. Then consider all configurations for which only two sites have values different from 0. You will be able to define uniquely all functions $G_{ij}(x_i, x_j)$, and so on.

Box 3.26 Prove the equivalence of Markov random fields and Gibbs distributions (Hammersley–Clifford theorem).

Step 1: Proof that a Gibbs distribution implies a Markov random field

To prove this, we must show that the conditional probability that appears on the right-hand side of (3.420) is conditioned only on the neighbours of x_i and it is independent of the values of the pixels that are not in the cliques that involve x_i. This was shown in Example 3.141. However, we would like also to identify the correspondence between clique potentials and Markov parameters in the most general form. So, we prove it again here in a different way.

Without loss of generality, let us concentrate on pixel x_1. Let us consider the expansion of $Q(X)$ given by Equation (3.423) and write it for configuration X_1, i.e. the configuration where all pixels have identical values with those in X, except x_1 which has value 0: $0, x_2, \ldots, x_M$. Then all terms in the expansion that contain x_1 will be zero, and the terms that remain will be those that do not depend on x_1. So, if we subtract the expansion of $Q(X_1)$ from that of $Q(X)$, we shall be left with only those terms that include x_1:

$$Q(X) - Q(X_1) = x_1 \left[G_1(x_1) + \sum_{2 \le j \le M} x_j G_{1j}(x_1, x_j) + \right.$$
$$\left. \sum_{2 \le j < k \le M} \sum \sum x_j x_k G_{1jk}(x_1, x_j, x_k) + \ldots + x_2 x_3 \ldots x_M G_{12\ldots M}(x_1, x_2, \ldots, x_M) \right]. \quad (3.425)$$

Let us consider now a pixel l that is not a neighbour of pixel 1. In Example 3.141 it was shown that the conditional probabilities that appear on the right-hand side of Equation (3.420) do not depend on non-neighbours of x_1, so $Q(X) - Q(X_1)$ should not depend on them either. Therefore, the left-hand side of (3.425) does not depend on pixel l, so all terms on the right-hand side that include l must also vanish. We can show that each one of the corresponding G functions that depends on x_l must be zero. For example, by writing (3.425) for the configuration where all pixels have values 0 except x_1 and x_l, we shall be left on the right-hand side with the term $x_1 x_l G_{1l}(x_1, x_l)$ only, which will have to be 0.

So, we have just shown that all G functions on the right-hand side of (3.425) will be zero unless the pixels that are involved in them form cliques with pixel 1. On the other hand, if a G function depends on a pixel l, so will $Q(X) - Q(X_1)$, and from (3.420) we infer that the conditional probability for the value of x_1 should also depend on it. In other words, the pixels on which the G functions depend define the neighbourhood structure of $p(x_1|x_2, \ldots, x_M)$. This realisation helps us prove the second part of the theorem, namely:

Step 2: Proof that a Markov random field implies a Gibbs distribution.

If in the expansion of Equation (3.423) we consider a finite set of non-zero G functions, the expansion is legitimate. From this expansion and with the help of the first of Equations (3.421) we may infer the joint probability density function of configuration X, $p(X)$, i.e. the Gibbs distribution. Note that the denominator $p(O)$ should not bother us as it is a constant.

Example B3.145

For the Gibbs distribution of Equation (3.397) work out the Markov parameters it implies.

First we repeat here Equation (3.397):

$$p(X) = \frac{1}{Z} e^{\sum_i \sum_j x_{ij} U_0 + \sum_i \sum_j x_{ij} x_{i+1,j} U_1 + \sum_i \sum_j x_{ij} x_{i,j+1} U_2}. \tag{3.426}$$

We start by computing function $Q(X)$ of Equation (3.419). If we set all pixels equal to 0, we have $p(O) = 1/Z$, so:

$$Q(X) = \sum_i \sum_j x_{ij} U_0 + \sum_i \sum_j x_{ij} x_{i+1,j} U_1 + \sum_i \sum_j x_{ij} x_{i,j+1} U_2. \tag{3.427}$$

We pick up a particular pixel (k, l) and separate it from the rest:

$$Q(X) = \sum_{ij,(i,j)\neq(k,l)} \sum \left\{ x_{ij} \left(U_0 + x_{i+1,j} U_1 + x_{i,j+1} U_2 \right) \right\} + x_{kl} \left(U_0 + x_{k+1,l} U_1 + x_{k,l+1} U_2 \right). \tag{3.428}$$

If we set the value of (k, l) to 0, we shall have a configuration we call X_{kl}, for which:

$$Q(X_{kl}) = \sum_{ij,(i,j)\neq(k,l)} \sum \left\{ x_{ij} U_0 + x_{ij} x_{i+1,j} U_1 + x_{ij} x_{i,j+1} U_2 \right\}. \tag{3.429}$$

By subtracting (3.429) from (3.428) we obtain:

$$Q(X) - Q(X_{kl}) = x_{kl} U_0 + x_{kl} x_{k+1,l} U_1 + x_{kl} x_{k,l+1} U_2. \tag{3.430}$$

By comparison with expansions (3.423) and (3.425) we deduce that:

$$\begin{array}{lcll} G_i(x_i) & \leftrightarrow & G_{kl}(x_{kl}) = U_0 & \text{for every } (k,l) \\ G_{ij}(x_i, x_j) & \leftrightarrow & G_{kl,mn}(x_{kl}, x_{mn}) = U_1 & \text{if } (m,n) = (k+1, l) \\ & & G_{kl,mn}(x_{kl}, x_{mn}) = U_2 & \text{if } (m,n) = (k, l+1) \\ & & G_{kl,mn}(x_{kl}, x_{mn}) = 0 & \text{otherwise} \end{array}$$

while all other G functions are 0.

We may then use Equation (3.420) to work out the conditional probability density function for pixel (k, l):

$$\begin{aligned} p(x_{kl}|all\_other\_pixels) &= p(x_{kl}|all\_nbring\_pixels\_in\_1st\_order\_Markov\_nbrhd) \\ &= \underbrace{p(0|all\_other\_pixels) e^{Q(X_{kl})}}_{\text{normalising constant}} e^{x_{kl}[U_0 + x_{k+1,l} U_1 + x_{k,l+1} U_2]} \end{aligned} \tag{3.431}$$

This expansion shows that the Markov parameters are U_0, U_1 and U_2.

Example B3.146

For the auto-binomial Markov random field model given by Equation (3.360) with function s given by (3.361), work out the corresponding Gibbs potentials.

We start by working out $Q(X) - Q(X_i)$ as it appears in Equation (3.420). The right-hand side of this equation expresses the probability of a pixel i to take a certain value x_i, over the probability

(Continued)

Example B3.146 (Continued)

to take value 0. First we revert back to pixels being identified by two indices, as it is customary for 2D images, i.e. we identify a particular pixel by its indices (i,j) in the grid:

$$e^{Q(X)-Q(X_{ij})} = \frac{p(x_{ij}|\text{values\_of\_all\_other\_pixels})}{p(0|\text{values\_of\_all\_other\_pixels})} \tag{3.432}$$

Because of the Markovian property, the probabilities on the right-hand side may be replaced by those conditioned on the neighbouring pixels only:

$$e^{Q(X)-Q(X_{ij})} = \frac{p(x_{ij}|\text{values\_of\_nbrs})}{p(0|\text{values\_of\_nbrs})} \tag{3.433}$$

We then re-write Equation (3.360) with function s given by (3.361) using the usual pixel notation

$$p(x_{ij}|x_{i-1,j},x_{i+1,j},x_{i,j-1},x_{i,j+1}) = \frac{n!}{x_{ij}!(n-x_{ij})!}\frac{e^{ax_{ij}(x_{i-1,j}+x_{i+1,j})+bx_{ij}(x_{i,j-1}+x_{i,j+1})}}{D} \tag{3.434}$$

where here $D \equiv \left[1+e^{ax_{ij}(x_{i-1,j}+x_{i+1,j})+bx_{ij}(x_{i,j-1}+x_{i,j+1})}\right]^n$. *We note that* $p(0|x_{i-1,j},x_{i+1,j},x_{i,j-1},x_{i,j+1}) = \frac{n!}{0!n!}\frac{e^0}{D}$. *If we substitute then into the right-hand side of (3.433), we obtain:*

$$e^{Q(X)-Q(X_{ij})} = \frac{n!}{x_{ij}!(n-x_{ij})!}e^{ax_{ij}(x_{i-1,j}+x_{i+1,j})+bx_{ij}(x_{i,j-1}+x_{i,j+1})}. \tag{3.435}$$

We deduce, therefore, that

$$\begin{aligned} Q(X)-Q(X_{ij}) &= \ln\frac{n!}{x_{ij}!(n-x_{ij})!} + ax_{ij}(x_{i-1,j}+x_{i+1,j}) + bx_{ij}(x_{i,j-1}+x_{i,j+1}) \\ &= x_{ij}\left\{\frac{1}{x_{ij}}\ln\frac{n!}{x_{ij}!(n-x_{ij})!} + a(x_{i-1,j}+x_{i+1,j})\right. \\ &\qquad \left. +b(x_{i,j-1}+x_{i,j+1})\,.\right\} \end{aligned} \tag{3.436}$$

By comparison with expansions (3.423) and (3.425) we deduce that the clique potentials of the corresponding Gibbs distribution are

$$\begin{array}{lcllll} G_i(x_i) & \leftrightarrow & G_{ij}(x_{ij}) & = & U_0 & = \frac{1}{x_{ij}}\ln\frac{n!}{x_{ij}!(n-x_{ij})!} \\ G_{ij}(x_i,x_j) & \leftrightarrow & G_{ij,i+1,j}(x_{ij},x_{i+1,j}) & = & U_1 & = a \\ & & G_{ij,i,j+1}(x_{ij},x_{i,j+1}) & = & U_2 & = b \end{array}$$

while all other G functions are 0.

Example B3.147

Imagine that a particular 3×3 binary image from those of Example 3.136 was created by flipping a coin to decide the value of each pixel (1 for heads, 0 for tails). What is the probability θ for the coin we used of obtaining heads in a single throw?

We assume that for this image the Gibbs model of Equation (3.397) applies, with $U_0 = 1$ and $U_1 = U_2 = 0$. So, in this case function s of Equation (3.359) is $s = 1 \times x_{ij}$. For $x_{ij} = 1$, $s = 1$. Therefore, $\theta = e^1/(1 + e^1) = 0.731$. This result shows that the coin is strongly biased towards the value 1 (heads). This is in agreement with the result of Example 3.139 where we found that the most probable configuration consists of 1s only.

Example B3.148

For the Gaussian (auto-normal) Markov random field model given by Equation (3.359) with the first order neighbourhood structure shown in Figure 3.116a work out the corresponding Gibbs potentials.

We proceed as in Example 3.146, only now we use (3.365) to work out the probabilities that appear on the right-hand side of Equation (3.433):

$$p(x_{ij}|x_{i-1,j}, x_{i+1,j}, x_{i,j-1}, x_{i,j+1}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{\left(x_{ij} - \left[a_1(x_{i-1,j} + x_{i+1,j}) + a_2(x_{i,j-1} + x_{i,j+1})\right]\right)^2}{2\sigma^2}\right\}. \tag{3.437}$$

We note that:

$$p(0|x_{i-1,j}, x_{i+1,j}, x_{i,j-1}, x_{i,j+1}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{\left[a_1(x_{i-1,j} + x_{i+1,j}) + a_2(x_{i,j-1} + x_{i,j+1})\right]^2}{2\sigma^2}\right\}. \tag{3.438}$$

If we substitute then into the right-hand side of (3.433), we obtain:

$$e^{Q(X)-Q(X_{ij})} = \exp\left\{-\frac{1}{2\sigma^2}\left(x_{ij}^2 - 2x_{ij}\left[a_1(x_{i-1,j} + x_{i+1,j}) + a_2(x_{i,j-1} + x_{i,j+1}\right]\right)\right\}. \tag{3.439}$$

We deduce, therefore, that

$$Q(X) - Q(X_{ij}) = -\frac{1}{2\sigma^2}\left(x_{ij}^2 - 2x_{ij}\left[a_1(x_{i-1,j} + x_{i+1,j}) + a_2(x_{i,j-1} + x_{i,j+1}\right]\right) \tag{3.440}$$

which may be written as:

$$Q(X) - Q(X_{ij}) = x_{ij}\left\{-\frac{x_{ij}}{2\sigma^2} + \frac{a_1}{\sigma^2}(x_{i-1,j} + x_{i+1,j}) + \frac{a_2}{\sigma^2}(x_{i,j-1} + x_{i,j+1})\right\}. \tag{3.441}$$

By comparison with expansions (3.423) and (3.425) we deduce that the clique potentials are

$$\begin{array}{lcllll} G_i(x_i) & \leftrightarrow & G_{ij}(x_{ij}) & = & U_0 & = & -\frac{x_{ij}}{2\sigma^2} \\ G_{ij}(x_i, x_j) & \leftrightarrow & G_{ij,i+1,j}(x_{ij}, x_{i+1,j}) & = & U_1 & = & \frac{a_1}{\sigma^2} \\ & & G_{ij,ij+1}(x_{ij}, x_{ij+1}) & = & U_2 & = & \frac{a_2}{\sigma^2} \end{array}$$

while all other G functions are 0.

How can we use the Gibbs distribution to create textures?

We may do that by iteratively relaxing a random field with the help of a **greedy** or **single-minded** algorithm. Such an algorithm always moves forward towards a solution, without worrying whether this solution is the best solution.

Step 0: Choose the functional form and the clique potential values of the Gibbs distribution from which you want to create a sample image.
Choose the histogram of the image you wish to create.
Step 1: Set counters $m = 0$ and $c = 0$.
Set N equal to the total number of pixels in the image you wish to create.
Set M equal to the maximum number of iterations you wish to allow.
Step 2: Start from a configuration of pixel values X, constructed so that it has the desirable histogram.
Step 3: Set $m = m + 1$.
If $m \geq M$ exit the algorithm, producing as output X.
If $m < M$ proceed to step 4.
Step 4: Set $c = c + 1$.
If $c \geq N$, set $c = 0$ and go to step 3.
If $c < N$, pick at random two pixels and proceed to step 5.
Step 5: If the two pixels have the same grey value, go to step 4.
If the two pixels do not have the same value, proceed to step 6.
Step 6: Exchange the values of the two pixels, thus creating a new configuration Y.
Step 7: Use Equation (3.409) to compute the relative probability $q \equiv p(Y)/p(X)$ of the two configurations.
Step 8: If $q \geq 1$ accept the exchange. Set $X = Y$ and go to step 4.
If $q < 1$ keep the old configuration X intact and go to step 4.

This algorithm is constructed in such a way that an iteration consists of picking up as many pairs of pixels as we have pixels in the image, irrespective of whether they have identical or different values. So, counter m counts the iteration steps, while counter c counts the pairs of pixels we choose and it is set back to 0 every time it reaches N, the total number of pixels. The algorithm terminates when the total number of iterations is M. Alternative termination criteria might be used. For example, at the end of each iteration step, one may estimate the parameters of the created image and check how close they are to the desired parameters. If they are close within a certain tolerance, the algorithm may be terminated. In the examples we present later the algorithm was terminated when the probability of existence of the created configuration was not changing much from one update to the next, i.e. when $q \simeq 1$.

In step 7, q is the ratio of the two Gibbs distributions. Of course, it may be computed by computing separately $p(X)$ and $p(Y)$ using the Gibbs distribution formula. This, however, is wasteful, as the two distributions differ only in the positions of the two pixels, the values of which were exchanged. So, the use of formula (3.409) allows one to consider only the cliques that were affected by the exchange.

The above algorithm will create an image which may be described by the model used. However, if we are interested in constructing **the most probable** image according to the chosen probability density function, we must use a variation of the **simulated annealing algorithm**. This algorithm consists of the following steps.

Step 0: Choose the functional form and the clique potential values of the Gibbs distribution from which you want to create a sample image.
Choose the histogram of the image you wish to create.
Step 1: Set counters $m = 0$, $c = 0$ and $l = 0$.
Set $T_0 = 1$ or another value of similar order of magnitude.
Set $\alpha = 0.99$ or a similar number smaller than 1, but close to 1.
Set N equal to the total number of pixels in the image you wish to create.
Set M equal to the maximum number of iterations you wish to allow.
Step 2: Start from a configuration of pixel values X, constructed so that it has the desirable histogram.
Set $X_{\text{old}} = X$.
Step 3: Set $m = m + 1$.
If $m \geq M$ exit the algorithm, producing as output X_{old}.
If $m < M$, set $T_m = \alpha T_{m-1}$ and proceed to step 4.
Step 4: Set $c = c + 1$.
If $c \geq N$, set $c = 0$ and go to step 3.
If $c < N$, pick at random two pixels and proceed to step 5.
Step 5: If the two pixels have the same grey value, go to step 4.
If the two pixels do not have the same value, proceed to step 6.
Step 6: Exchange the values of the two pixels, thus creating a new configuration Y.
Step 7: Use Equation (3.409) to compute the relative probability $q \equiv p(Y)/p(X)$ of the two configurations.
Step 8: If $q \geq 1$ accept the exchange. Set $X = Y$, $l = 0$, $X_{\text{old}} = Y$ and go to step 4.
If $q < 1$ proceed to step 9.
Step 9: Set $l = l + 1$.
If $l \geq L$ terminate and exit the algorithm producing as output X_{old}.
If $l < L$, draw a random number ξ in the range $[0, 1]$ and proceed to step 10.
Step 10: If $\xi < q^{1/T_m}$, accept the exchange. Set $X = Y$ and go to step 4.
Step 11: If $\xi \geq q^{1/T_m}$ keep the old configuration X intact and go to step 4.

The core of this algorithm is step 10: this step allows one to accept unfavourable changes with decreasing probability. The unfavourable changes allow one to escape from local optima of the configuration space, i.e. allow one to take some "bad" steps in order to find a better route to the target configuration: moving always forward is not usually the best way to reach our destination. Note that as $m \to \infty$, $T_m \to 0$, $1/T_m \to \infty$ and $q^{1/T_m} \to 0$, since $q < 1$. So, as iterations proceed, it becomes less and less likely to accept changes that decrease the probability of drawing a random number ξ smaller than q^{1/T_m}, i.e. it becomes less and less likely to accept a configuration that is less probable than the previous one. The algorithm terminates either when the total number of

iterations is M, or when in L successive updates the only accepted updates were those that were reducing the probability of the current configuration to exist. The algorithm exits by going back to the last update that produced a more probable configuration than the previous one.

This algorithm is a variation of the non-linear optimisation method of simulated annealing. The reason for this terminology is parameter T_m, which is called the **temperature** parameter. As its value drops, we say that the system **cools**. The whole process corresponds to the **annealing** process used in physics to produce, among other things, crystal configurations on a large scale. Parameter α is the **cooling parameter** and equation $T_m = \alpha T_{m-1}$ is called the **cooling schedule**. The algorithm is designed to produce the most probable configuration according to the adopted probability density function, and so it is useful when we are trying to solve inverse problems, like, for example, trying to answer the question: "what is the most probable configuration of pixel values that are consistent with the data, the model of degradation suffered by the data and the constraints that we know apply to the data?" Then the algorithm has to be applied with a carefully chosen cooling schedule. Note that the particular version of the simulated annealing algorithm we presented here, as well as the greedy algorithm presented earlier, are designed to **preserve the histogram** of grey values of the image, because the total number of pixels with a certain grey value does not change.

Example B3.149

Use expression (3.409) proven in Example 3.142 to construct a 64×64 Markov random field with a given Gibbs distribution and with the same number of pixels at all grey levels. Assume that the number of grey levels is $G = 2$, 4 and 64. The Gibbs distribution of configuration X is given by Equation (3.397) with $U_0 = 1$, $U_1 = 1$ and $U_2 = -1$.

We can easily compute the number of pixels for each grey level as given by the first two columns of Table 3.47.

We start by creating arrays of size 64×64 with the same number of pixels for each grey value. The starting configurations are shown in the first row of Figure 3.149. For visualisation purposes, the grey values of each image have been stretched to be integers in the range $[0, 256)$. For example, the image with $G = 2$, instead of having values 0 and 1, has values 0 and 255, and the image with $G = 4$, instead of having values $0, 1, 2, 3$, has values computed by

$$Scaled\_value = \left\lfloor \frac{g}{G-1} \times 255 \right\rfloor \tag{3.442}$$

where g is the original pixel value in the range $[0, G-1]$, and $\lfloor x \rfloor$ means integer part of x. However, in all the algorithmic computations, pixel values x are chosen to be in the range $[0, 1]$. Variable g as appears in Equation (3.442) is computed for each pixel from its x value, by using $g = Gx$. Then in order to allow the pattern to emerge, we apply the greedy algorithm. The third column of Table 3.47 gives the total number of iterations performed.

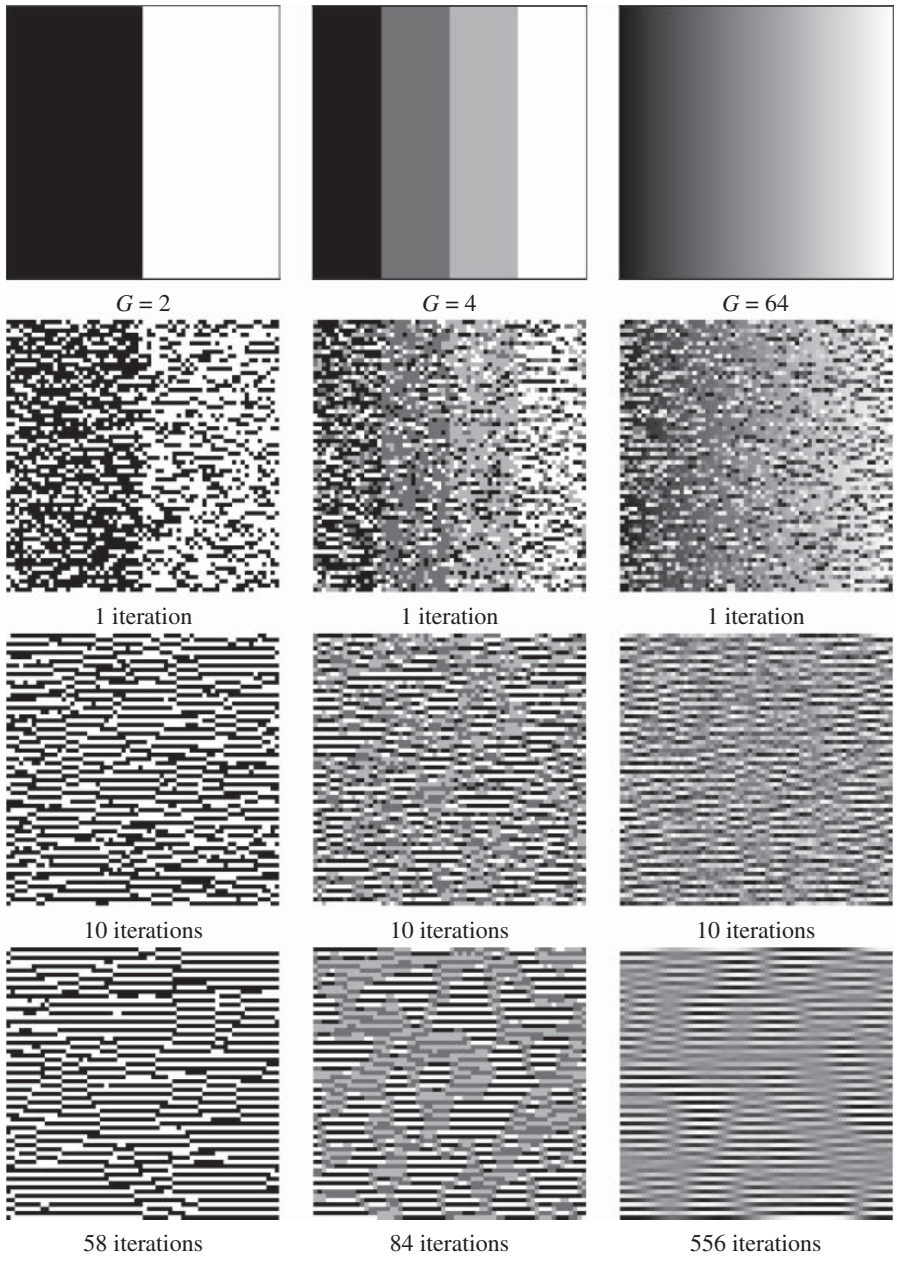


Figure 3.149 Results of the greedy algorithm applied to a starting configurations consisting of vertical stripes with the same number of pixels, for $G = 2$, $G = 4$, and $G = 64$, when the Gibbs potentials were fixed to be $U_0 = 1$, $U_1 = 1$ and $U_2 = -1$. These parameters favour the creation of horizontal stripes. Source: Maria Petrou.

(Continued)

Example B3.149 (Continued)

Table 3.47 Number of iterations required for the greedy algorithm to converge using different numbers of grey levels. The number of pixels per grey level and the figure that shows the results are also shown.

G	Pixels per grey level	Iterations to converge	Figure that shows the result
2	2048	58	3.149 first column
4	1024	84	3.149 second column
64	64	556	3.149 third column

Example B3.150

Repeat Example 3.149 where the starting configurations are not striped images but the pixels of the different grey values are randomly mixed, keeping still the same number of pixels per grey level. Comment on the results in comparison with those obtained in Example 3.149.

The results are shown in Figure 3.150. It is clear that although the starting images are totally different from those in example 3.149, the final images are very similar.

In general, fewer updates are needed now in order to reach convergence than in Example 3.149. This is because in Example 3.149, the starting configuration had some order. "Energy" had to be pumped into the system to destroy the old order before new order was imposed.

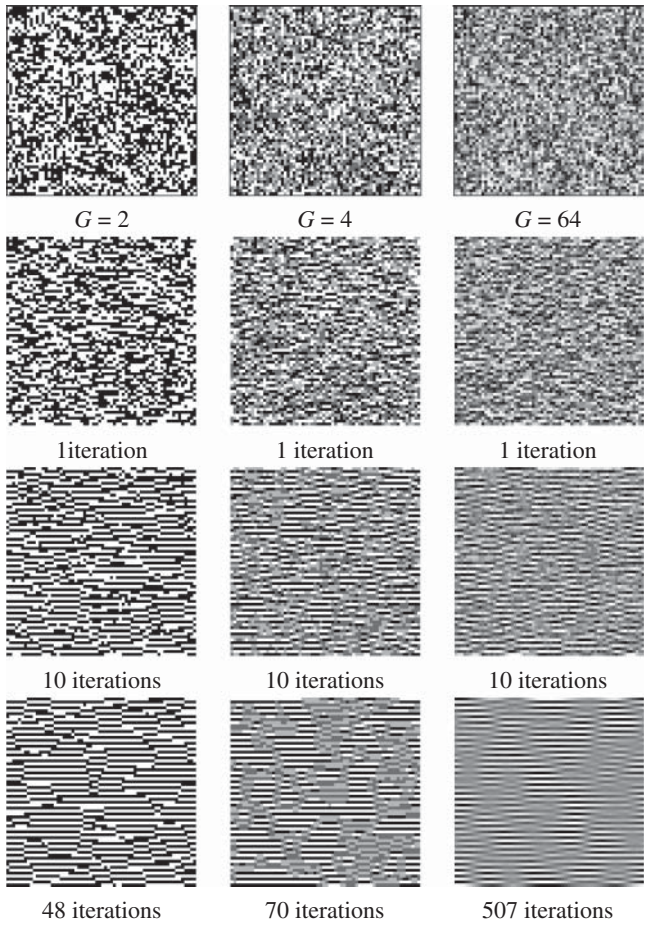


Figure 3.150 Results of the greedy algorithm applied to random images with the same number of pixels per grey level, for $G = 2$, $G = 4$ and $G = 64$, when the Gibbs potentials were fixed to be $U_0 = 1$, $U_1 = 1$ and $U_2 = -1$. Source: Maria Petrou.

Example B3.151

Apply the greedy algorithm to produce texture images of size 64×64 with the number of pixels in each grey value given by the equation $N(g) = \alpha g$ where a is some constant you must first specify, and $g = 0, 1, \ldots, G-1$.

The total number of pixels must be $64^2 = 4096$. So $\sum_{g=0}^{G-1} N(g) = 4096$ or $\alpha \sum_{g=0}^{G-1} g = 4096$. The second column of Table 3.48 gives the value of α for each value of G. To create the starting configurations we take $N(g) = \lfloor \alpha g \rfloor$ for various values of g, up to $G-2$. We give to all remaining pixels value $G-1$.

The results are shown in Figure 3.151.

(Continued)

Example B3.151 (Continued)

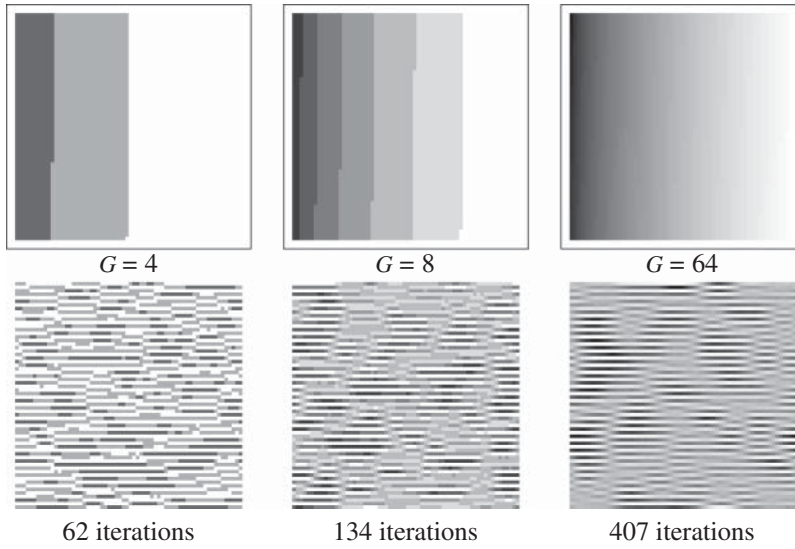


Figure 3.151 Results of the greedy algorithm applied to starting configurations consisting of vertical stripes with the number of pixels per grey level given by $N(g) = \alpha g$, for $G = 4$, $G = 8$ and $G = 64$, when the Gibbs potentials were fixed to be $U_0 = 1$, $U_1 = 1$ and $U_2 = -1$. Source: Maria Petrou.

Table 3.48 Value of α for number of grey levels $G = 4, 8, 16, 32, 64$.

G	α
4	682.67
8	146.29
16	34.13
32	8.26
64	2.03

How can we create an image compatible with a Gibbs model if we are not interested in fixing the histogram of the image?

If we are not interested in preserving the number of pixels per grey value, we have to modify the algorithms described earlier, so instead of exchanging the values of two pixels, we allow each pixel to choose freely its new value. The modified greedy algorithm is as follows.

Step 0: Choose the functional form and the clique potential values of the Gibbs distribution from which you want to create a sample image.
Step 1: Set counters $m = 0$ and $c = 0$.
Set M equal to the maximum number of iterations you wish to allow.
Set C equal to the number of different sets of pixels identified by the coding scheme applicable to the MRF neighbourhood structure of your model.
Denote by S_c, where $c = 1, 2, \ldots, C$, the cth set of pixels identified by the coding you use.
Step 2: Start from configuration of pixel values X, chosen at random, or in any other way. Create an output array Y and set $Y = X$.
Step 3: Set $X = Y$
Set $m = m + 1$.
If $m \geq M$, exit the algorithm, producing as output Y.
If $m < M$, set $c = c + 1$.
If $c > C$, set $c = 1$ and proceed to step 4. If $c \leq C$, proceed to step 4.
Step 4: If you have scanned all pixels in set S_c go to step 3.
If you have not scanned all pixels in set S_c, pick the next pixel from set S_c and proceed to step 5.
Step 5: Choose a new value for the current pixel to create a new configuration Y_{temp}.
Step 6: Use Equation (3.409) to compute the relative probability $q \equiv p(Y_{temp})/p(X)$ of the two configurations.
Step 7: If $q \geq 1$, update the value of the pixel in Y and go to step 4.

If $q < 1$, keep the old configuration Y intact and go to step 4.

The above algorithm will give us a configuration drawn from the defined probability density function of configurations. If we want the **most probable** configuration, we must use the simulated annealing algorithm that follows.

Step 0: Choose the functional form and the clique potential values of the Gibbs distribution from which you want to create a sample image.
Step 1: Set counters $m = 0$, $c = 0$ and $l = 0$.
Set $T_0 = 1$ or another value of similar order of magnitude.
Set $\alpha = 0.99$ or a similar number smaller than 1, but close to 1.
Set M equal to the maximum number of iterations you wish to allow.
Set C equal to the number of different sets of pixels identified by the coding scheme applicable to the Markov random field neighbourhood structure of your model.
Denote by S_c, where $c = 1, 2, \ldots, C$, the cth set of pixels identified by the coding you use.
Step 2: Start from configuration of pixel values X, chosen at random, or in any other way.
Create an output array Y and set $Y = X$.
Set $Y_{\text{old}} = Y$.

Step 3: Set $X = Y$. Set $m = m + 1$.
If $m \geq M$, exit the algorithm, producing as output Y_{old}.
If $m < M$, set $c = c + 1$.
If $c > C$, set $c = 1$ and proceed to step 4.
If $c \leq C$, proceed to Step 4.
Step 4: Set $T_m = \alpha T_{m-1}$ and proceed to step 5.
Step 5: If you have scanned all pixels in set S_c go to step 3.
If you have not scanned all pixels in set S_c, pick the next pixel from set S_c and proceed to step 6.
Step 6: Choose a new value for the current pixel to create a new configuration Y_{temp}.
Step 7: Use Equation (3.409) to compute the relative probability $q \equiv p(Y_{\text{temp}})/p(X)$ of the two configurations.
Step 8: If $q \geq 1$, update the value of the pixel in Y, set $l = 0$, $Y_{\text{old}} = Y$ and go to step 5.
If $q < 1$ proceed to step 9.
Step 9: Set $l = l + 1$.
If $l \geq L$ terminate and exit the algorithm producing as output Y_{old}.
If $l < L$, draw a random number ξ in the range $[0, 1]$ and proceed to step 10.
Step 10: If $\xi < q^{1/T_m}$, update the value of the pixel in Y and go to step 5.
Step 11: If $\xi \geq q^{1/T_m}$, keep the old configuration Y intact and go to step 5.

In both the above algorithms, for simplicity, the grey values of all pixels should be chosen to be real numbers in the range $[0, 1]$. At the end the output values y should be quantised to the desired number of grey levels G by using $g = yG$ and Equation (3.442).

In step 6 the new value of a pixel is chosen from a uniform distribution in the range $[0, 1]$. So this algorithm uses a **Metropolis sampler**. (For alternative samplers, see Book I .) Note that the difference from the previous algorithms is that an iteration here is equivalent to the updating of all pixels in the same coding set, as opposed to picking up as many pairs of pixels as number of pixels in the image. The pixels now are visited in a systematic way, and because their values are updated by taking into consideration the values of the neighbours in the configuration of the previous step, all pixels in the set may be updated in parallel in each iteration step.

Example B3.152

Use the modified greedy algorithm that works with totally free number of pixels per grey value and repeat the experiments of Example 3.150, for $G = 2,\ 32$ and 256.
The results are shown in Figure 3.152. Note that due to the external field, and because we are not trying to preserve the histogram of the original image, as the iterations progress, the constructed image tends to a flat white image.

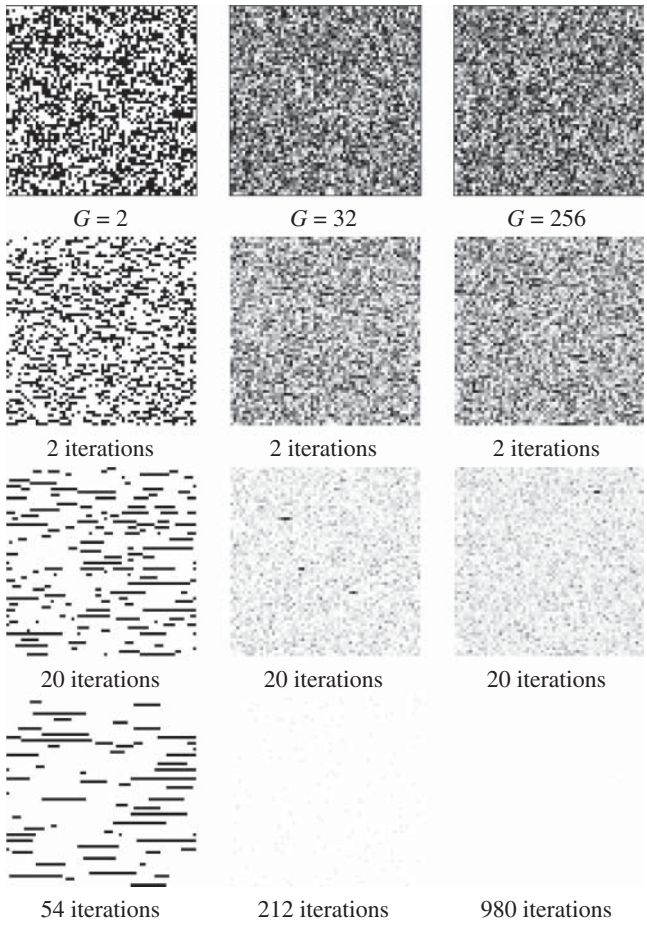


Figure 3.152 Results of the modified greedy algorithm applied to a random image, for $G = 2$, $G = 32$ and $G = 256$, when the Gibbs potentials were fixed to be $U_0 = 1$, $U_1 = 1$ and $U_2 = -1$. Source: Maria Petrou.

Example B3.153

Repeat the experiment of Example 3.152 using the following clique potentials: $U_0 = 0$**,** $U_1 = 1$**, and** $U_2 = -1$**.**

The results are shown in Figure 3.153. Now as the external field has been switched off (by setting $U_0 = 0$*), the pattern is allowed to emerge.*

(Continued)

Example B3.153 (Continued)

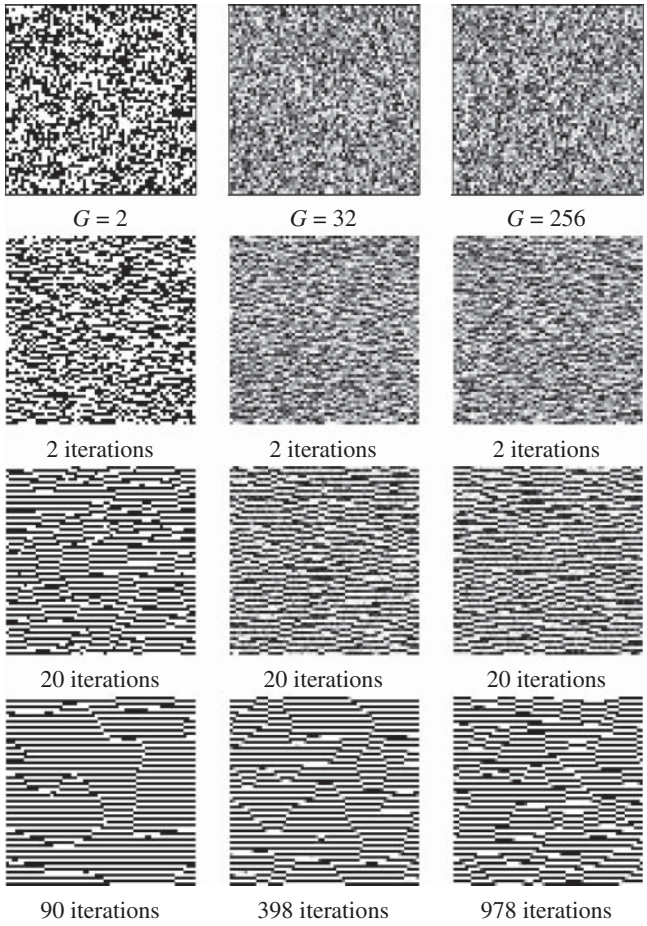


Figure 3.153 Results of the modified greedy algorithm applied to a random image, for $G = 2$, $G = 32$ and $G = 256$, when the Gibbs potentials were fixed to be $U_0 = 0$, $U_1 = 1$ and $U_2 = -1$. Note the difference from Figure 3.152 where there is an external bias ($U_0 \neq 0$) that dominates the outcome. Source: Maria Petrou.

What is the temperature of a Gibbs distribution?

The term temperature comes from physics where Gibbs distributions were first used. It is a parameter that multiplies the exponent of Equation (3.396). Its effect is twofold:

- It determines how strong the correlations between distant pixels are (see Example 3.163).
- It determines how sharp the joint probability density function is in discriminating between different configurations.

How does the temperature parameter of the Gibbs distribution determine how distinguishable one configuration is from another?

Let us redefine the exponent of Equation (3.396) as

$$p(X) = \frac{1}{Z} e^{-\frac{1}{T}H(X)} \tag{3.443}$$

where $H(X)$ is called the **energy** or **cost function** of configuration X and it contains the sum over all cliques in the configuration. In (3.443), $H(X)$ would be

$$H(X) = -\sum_{ij} x_{ij} \left(1 + x_{i+1,j}\frac{U_1}{U_0} + x_{i,j+1}\frac{U_2}{U_0}\right) \tag{3.444}$$

where we have identified $1/T$ with U_0 and without loss of generality we have assumed U_0 to be positive. (If U_0 were negative, we could have defined $1/T \equiv |U_0|$ and changed the definition of $H(X)$ accordingly.)

Consider two different possible pixel configurations X_1 and X_2, and compute their relative probability of arising:

$$\frac{p(X_1)}{p(X_2)} = e^{-\frac{1}{T}[H(X_1)-H(X_2)]}. \tag{3.445}$$

Let us assume that $H(X_1) > H(X_2)$, without loss of generality. (If this is not true, we can always rename the two configurations, so that X_1 is always the one with the highest energy.) Then, if $T \to \infty$, i.e. if $U_0 = 1/T \to 0$, the exponent on the right-hand side of (3.445) tends to zero, i.e. the exponential tends to 1, i.e. $p(X_1) \sim p(X_2)$, i.e. the two configurations are almost equally probable, no matter how different they are. If, on the contrary, $T \to 0$, i.e. if $U_0 = 1/T \to \infty$, the exponent on the right-hand side of (3.445) tends to negative infinity, i.e. the exponential tends to 0, i.e. $p(X_1) << p(X_2)$, i.e. the two configurations have very different probabilities of arising, with the lowest energy one being the most probable.

Example B3.154

For each of the possible configurations of a 3×3 binary image compute the average grey value.

The results are shown in Tables 3.49–3.52.

(Continued)

Example B3.154 (Continued)

Table 3.49 Average grey value of each configuration of Figure 3.148.

$x_{11} = 1, x_{32} = 0$							
1/9	2/9	2/9	2/9	2/9	2/9	2/9	2/9
3/9	3/9	3/9	3/9	3/9	3/9	3/9	3/9
3/9	3/9	3/9	3/9	3/9	3/9	3/9	3/9
3/9	3/9	3/9	3/9	3/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	6/9	6/9	6/9	6/9	6/9
6/9	6/9	6/9	6/9	6/9	6/9	6/9	6/9
6/9	6/9	6/9	6/9	6/9	6/9	6/9	6/9
7/9	7/9	7/9	7/9	7/9	7/9	7/9	8/9

Table 3.50 Average grey value of each configuration of Figure 3.148 if the two highlighted pixels have value 0.

$x_{11} = 0, x_{32} = 0$							
0/9	1/9	1/9	1/9	1/9	1/9	1/9	1/9
2/9	2/9	2/9	2/9	2/9	2/9	2/9	2/9
2/9	2/9	2/9	2/9	2/9	2/9	2/9	2/9
2/9	2/9	2/9	2/9	2/9	3/9	3/9	3/9
3/9	3/9	3/9	3/9	3/9	3/9	3/9	3/9
3/9	3/9	3/9	3/9	3/9	3/9	3/9	3/9
3/9	3/9	3/9	3/9	3/9	3/9	3/9	3/9
3/9	3/9	3/9	3/9	3/9	3/9	3/9	3/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
6/9	6/9	6/9	6/9	6/9	6/9	6/9	7/9

Table 3.51 Average grey value of each configuration of Figure 3.148 if the two highlighted pixels have value 1.

$x_{11} = 1, x_{32} = 1$							
2/9	3/9	3/9	3/9	3/9	3/9	3/9	3/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
6/9	6/9	6/9	6/9	6/9	6/9	6/9	6/9
6/9	6/9	6/9	6/9	6/9	6/9	6/9	6/9
6/9	6/9	6/9	6/9	6/9	6/9	6/9	6/9
6/9	6/9	6/9	6/9	6/9	6/9	6/9	6/9
6/9	6/9	6/9	7/9	7/9	7/9	7/9	7/9
7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9
7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9
8/9	8/9	8/9	8/9	8/9	8/9	8/9	9/9

Table 3.52 Average grey value of each configuration of Figure 3.148 if the two highlighted pixels exchange their values.

$x_{11} = 0, x_{32} = 1$							
1/9	2/9	2/9	2/9	2/9	2/9	2/9	2/9
3/9	3/9	3/9	3/9	3/9	3/9	3/9	3/9
3/9	3/9	3/9	3/9	3/9	3/9	3/9	3/9
3/9	3/9	3/9	3/9	3/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
4/9	4/9	4/9	4/9	4/9	4/9	4/9	4/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	5/9	5/9	5/9	5/9	5/9
5/9	5/9	5/9	6/9	6/9	6/9	6/9	6/9
6/9	6/9	6/9	6/9	6/9	6/9	6/9	6/9
6/9	6/9	6/9	6/9	6/9	6/9	6/9	6/9
7/9	7/9	7/9	7/9	7/9	7/9	7/9	8/9

Example B3.155

For each of the possible configurations of a 3×3 binary image compute its prior probability of arising when this is given by a Gibbs distribution with sets of clique potentials:

$U_0 = 1, \quad U_1 = 0, \quad U_2 = 0$
$U_0 = 0, \quad U_1 = 1, \quad U_2 = 1$
$U_0 = 0, \quad U_1 = 1, \quad U_2 = -1$
$U_0 = -1, \quad U_1 = 1, \quad U_2 = -1$
$U_0 = 1, \quad U_1 = 1, \quad U_2 = -1.$

These probabilities may be computed using Equation (3.397), repeated here:

$$p(X) = \frac{1}{Z} e^{\sum_i \sum_j x_{ij} U_0 + \sum_i \sum_j x_{ij} x_{i+1\,j} U_1 + \sum_i \sum_j x_{ij} x_{i,j+1} U_2}. \tag{3.446}$$

As an indicative example, we present in Tables 3.53–3.56 the results that correspond to the first case only, i.e. the case with $U_0 = 1, U_1 = U_2 = 0$. This case represents a situation where "order" in the configuration space is imposed by an external field, since the pixels are not allowed to influence each other ($U_1 = U_2 = 0$).

Table 3.53 Probability of each configuration in Figure 3.148 arising, when expressed by a Gibbs distribution with clique potentials $U_0 = 1$, $U_1 = U_2 = 0$. All values must be multiplied with 10^{-6}.

$x_{11} = 1, x_{32} = 0$							
20.0	54.4	54.4	54.4	54.4	54.4	54.4	54.4
147.8	147.8	147.8	147.8	147.8	147.8	147.8	147.8
147.8	147.8	147.8	147.8	147.8	147.8	147.8	147.8
147.8	147.8	147.8	147.8	147.8	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	2969.5	2969.5	2969.5	2969.5	2969.5
2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5
2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5
8072.0	8072.0	8072.0	8072.0	8072.0	8072.0	8072.0	21941.9

(Continued)

Example B3.155 (Continued)

Table 3.54 Probability of each configuration in Figure 3.148 arising if the two highlighted pixels have value 0. This probability is modelled by a Gibbs distribution with clique potentials $U_0 = 1$, $U_1 = U_2 = 0$. All values must be multiplied with 10^{-6}.

$x_{11} = 0, x_{32} = 0$							
7.4	20.0	20.0	20.0	20.0	20.0	20.0	20.0
54.4	54.4	54.4	54.4	54.4	54.4	54.4	54.4
54.4	54.4	54.4	54.4	54.4	54.4	54.4	54.4
54.4	54.4	54.4	54.4	54.4	147.8	147.8	147.8
147.8	147.8	147.8	147.8	147.8	147.8	147.8	147.8
147.8	147.8	147.8	147.8	147.8	147.8	147.8	147.8
147.8	147.8	147.8	147.8	147.8	147.8	147.8	147.8
147.8	147.8	147.8	147.8	147.8	147.8	147.8	147.8
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	8072.0

Table 3.55 Probability of each configuration in Figure 3.148 arising if the two highlighted pixels have value 1. This probability is modelled by a Gibbs distribution with clique potentials $U_0 = 1$, $U_1 = U_2 = 0$. All values must be multiplied with 10^{-6}.

$x_{11} = 1, x_{32} = 1$							
54.4	147.8	147.8	147.8	147.8	147.8	147.8	147.8
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5
2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5
2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5
2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5
2969.5	2969.5	2969.5	8072.0	8072.0	8072.0	8072.0	8072.0
8072.0	8072.0	8072.0	8072.0	8072.0	8072.0	8072.0	8072.0
8072.0	8072.0	8072.0	8072.0	8072.0	8072.0	8072.0	8072.0
21941.9	21941.9	21941.9	21941.9	21941.9	21941.9	21941.9	59644.4

Table 3.56 Probability of each configuration in Figure 3.148 arising if the two highlighted pixels exchange their values (i.e. x_{11} takes value 0 and x_{32} takes value 1). This probability is modelled by a Gibbs distribution with clique potentials $U_0 = 1, U_1 = U_2 = 0$. All values must be multiplied with 10^{-6}.

$x_{11} = 0, x_{32} = 1$							
20.0	54.4	54.4	54.4	54.4	54.4	54.4	54.4
147.8	147.8	147.8	147.8	147.8	147.8	147.8	147.8
147.8	147.8	147.8	147.8	147.8	147.8	147.8	147.8
147.8	147.8	147.8	147.8	147.8	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
401.9	401.9	401.9	401.9	401.9	401.9	401.9	401.9
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4	1092.4
1092.4	1092.4	1092.4	2969.5	2969.5	2969.5	2969.5	2969.5
2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5
2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5	2969.5
8072.0	8072.0	8072.0	8072.0	8072.0	8072.0	8072.0	21941.9

Example B3.156

Plot the normalised histogram of the mean grey values you produced in Example 3.154 for the case when all configurations are equally likely to arise and the cases when the prior probability of each configuration is as computed in Example 3.155.

This is the same as computing the probability of having an image with a given average grey value.

When the image space does not have a Gibbs structure, to produce this histogram we simply count how many images have a particular mean value and then divide each number by the total number of possible images, i.e. 512 in this case.

When, however, each image has a different prior probability of arising, given by the Gibbs distribution, in order to produce this histogram we add up the probabilities of all configurations that have a particular mean value. In other words, the probability of getting an image with average

(Continued)

Example B3.156 (Continued)

grey value M must be given by the sum of the Gibbs distribution over all configurations that have average grey value M:

$$\begin{aligned} p(M) &= \sum_{all\_configurations\_with\_average\_grey\_value\_M} p(X) \\ &= \frac{1}{Z} \sum_{all\_configurations\_with\_average\_grey\_value\_M} e^{\sum_{ij} x_{ij}(U_0 + U_1 x_{i+1,j} + U_2 x_{i,j+1})} \end{aligned} \tag{3.447}$$

In our example, the average grey value is determined by the number of 1s in the image. As we can have, $0, 1, \ldots, 9$ *1s, and we have nine pixels in all, the only possible values of M are* $0, 1/9, 2/9, \ldots, 1$.

Table 3.57 gives the value of $p(M)$ *for each possible value of M.*

Figure 3.154 shows the plots of these functions.

Table 3.57 Probability of having an image with average grey level M for different sets of clique potentials, and when there is no Gibbs structure. The numbers in each column have to sum up to 1. If they do not, it is because of the truncation to three decimal points only. Note that no 0 entry is exactly 0. The numbers in the last column were computed by simply counting the number of configurations with a certain number of 1s and dividing the result by the total number of possible 3×3 binary configurations, i.e. by 512.

		$p(M)$					
M	Number of images	$U_0 = 1$ $U_1 = 0$ $U_2 = 0$	$U_0 = 0$ $U_1 = 1$ $U_2 = 1$	$U_0 = 0$ $U_1 = 1$ $U_2 = -1$	$U_0 = -1$ $U_1 = 1$ $U_2 = -1$	$U_0 = 1$ $U_1 = 1$ $U_2 = -1$	No Gibbs structure
0	1	0.000	0.000	0.001	0.040	0.000	0.002
1/9	9	0.000	0.000	0.010	0.133	0.000	0.018
2/9	36	0.002	0.000	0.050	0.248	0.002	0.070
3/9	84	0.012	0.000	0.172	0.315	0.016	0.164
4/9	126	0.051	0.000	0.267	0.180	0.066	0.246
5/9	126	0.138	0.000	0.267	0.066	0.180	0.246
6/9	84	0.249	0.003	0.172	0.016	0.315	0.164
7/9	36	0.291	0.019	0.050	0.002	0.248	0.070
8/9	9	0.197	0.138	0.010	0.000	0.133	0.018
1	1	0.060	0.840	0.001	0.000	0.040	0.002

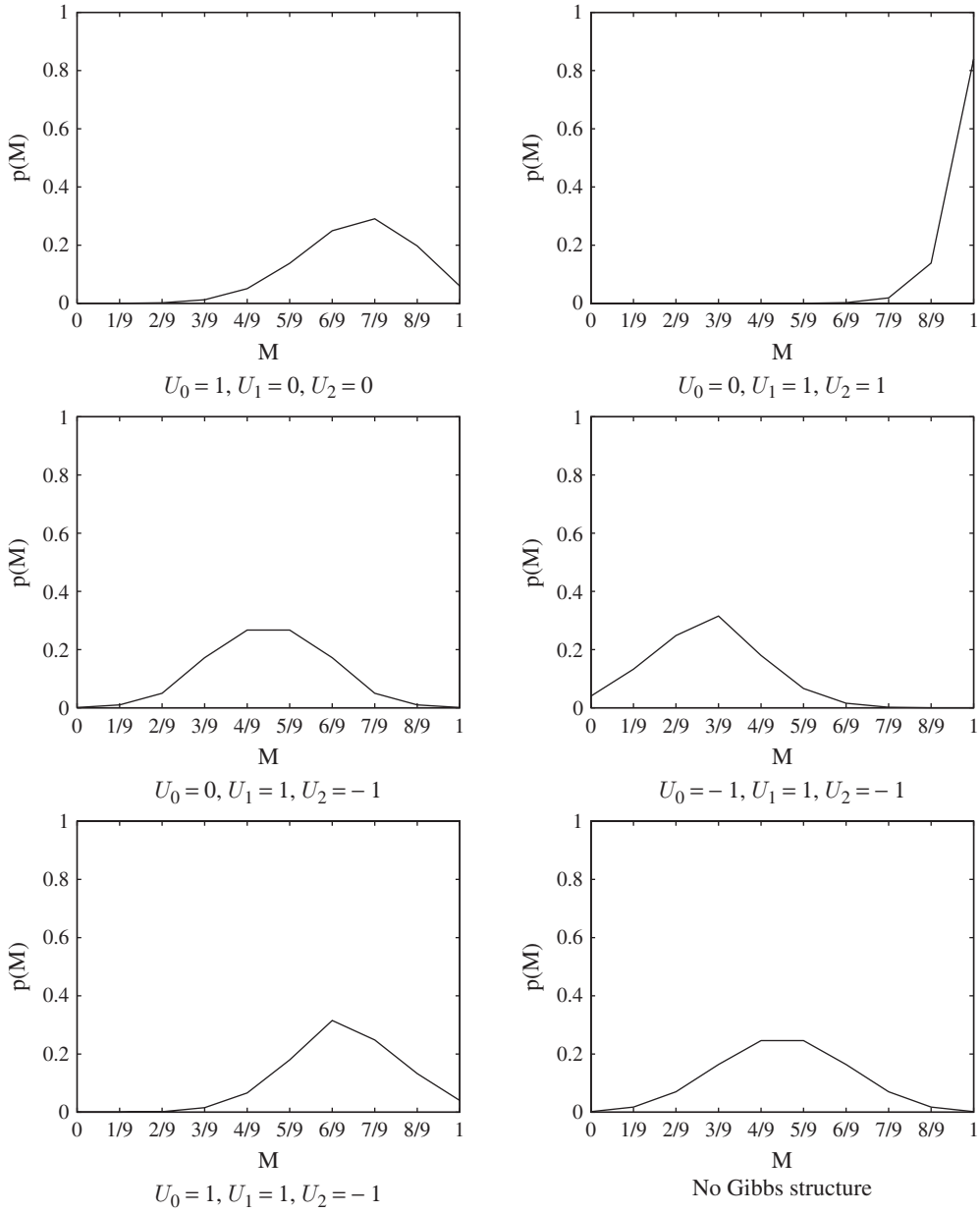


Figure 3.154 Probability density function for having an image with average grey level M for different sets of clique potentials and when there is no Gibbs structure.

Example B3.157

Physicists define a quantity called *mean free energy*, $F(M)$, as follows:

$$e^{-F(M)} \equiv \sum_{\text{all\_configurations\_with\_average\_grey\_value\_}M} Zp(X) \tag{3.448}$$

where M is the average grey value of a configuration. What is the meaning of $F(M)$ for an image? Compute its value for the images of Example 3.156.

For the particular Gibbs distribution we have been using, this equation takes the form:

$$e^{-F(M)} = \sum_{all\_configurations\_with\_average\_grey\_value\_M} e^{\sum_{ij} x_{ij}\left(U_0 + U_{1xi+1,j} + U_2 x_{i,j+1}\right)} \tag{3.449}$$

We notice that the right-hand side of this equation is the numerator of Equation (3.447) which determines how likely it is to find a configuration with mean grey value M. Given that the denominator is constant, this quantity then determines how probable a particular average grey value is, how "stable" our system is in the following sense: if something changes in the image, how likely it is that its overall appearance, as expressed by its average grey value, will remain the same. As search engines often retrieve images according to their mean colour, the concept of mean free energy is related to such tasks. When there is no Gibbs structure, $F(M) = -\ln N_M$, where N_M is the number of images that have mean value M.

Table 3.58 shows the values of $F(M)$ for the different sets of clique potentials we have been using. Figure 3.155 shows the plots of these functions versus M.

Table 3.58 Free energy $F(M)$ for different structures of the configuration space.

		$F(M)$					
	Number of	$U_0 = 1$	$U_0 = 0$	$U_0 = 0$	$U_0 = -1$	$U_0 = 1$	
M	images	$U_1 = 0$ $U_2 = 0$	$U_1 = 1$ $U_2 = 1$	$U_1 = 1$ $U_2 = -1$	$U_1 = 1$ $U_2 = -1$	$U_1 = 1$ $U_2 = -1$	No Gibbs structure
0	1	0.000	0.000	0.000	0.000	0.000	0.000
1/9	9	−3.197	−2.197	−2.197	−1.197	−3.197	−2.197
2/9	36	−5.584	−4.204	−3.824	−1.824	−5.824	−3.584
3/9	84	−7.431	−6.195	−5.062	−2.062	−8.062	−4.431
4/9	126	−8.836	−8.164	−5.501	−1.501	−9.501	−4.836
5/9	126	−9.836	−10.164	−5.501	−0.501	−10.501	−4.836
6/9	84	−10.431	−12.195	−5.062	0.938	−11.062	−4.431
7/9	36	−10.584	−14.204	−3.824	3.176	−10.824	−3.584
8/9	9	−10.197	−16.197	−2.197	5.803	−10.197	−2.197
1	1	−9.000	−18.000	0.000	9.000	−9.000	0.000

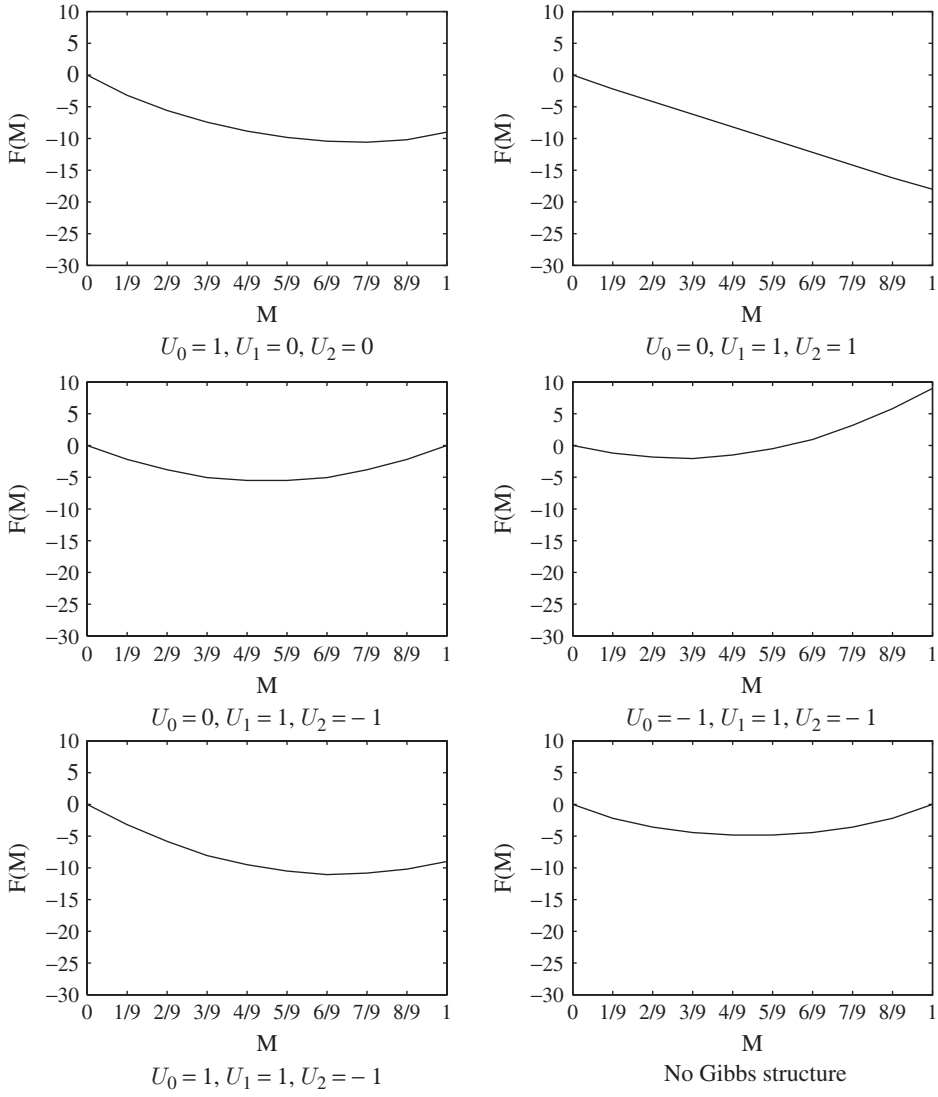


Figure 3.155 Free energy $F(M)$ as a function of the mean grey value M.

What is the critical temperature of a Markov random field?

The critical temperature of an MRF is the value of parameter T for which the MRF behaves like a fractal. At that parameter value, the autocorrelation function of the MRF implies correlation among the pixels at all scales, i.e. at all distances from each other (see Example 3.163).

Example B3.158

Show that for the Gibbs model of Equation 3.397 applied to a binary image with clique potentials $U_2 = U_1 = -U_0/2$ and $U_0 > 0$, the probability of a configuration X existing depends on the number of cliques of the form $(0,1)$ and $(1,0)$ present in the configuration.

Let us first make a transformation so that the pixels take values -1 and 1 instead of 0 and 1

$$x_{ij} = \frac{n_{ij}+1}{2} \Rightarrow n_{ij} = 2x_{ij} - 1 \tag{3.450}$$

where x_{ij} is the old value of a pixel (being either 0 or 1) and n_{ij} is the new value, which may be either 1 or -1.

Function $H(X)$ then, which appears in the exponent of the numerator of $p(X)$, defined here so that $p(X) \equiv \frac{1}{Z}e^{-H(X)}$, is:

$$\begin{aligned} H &= \sum_{ij} U_0 \frac{n_{ij}+1}{2} + U_1 \frac{n_{ij}+1}{2} \times \frac{n_{i+1,j}+1}{2} + U_2 \frac{n_{ij}+1}{2} \times \frac{n_{i,j+1}+1}{2} \\ &= \left\{ \frac{U_0}{2}\sum_{ij} n_{ij} + \frac{U_0}{2}\sum_{ij} 1 + \frac{U_1}{4}\sum_{ij} n_{ij}n_{i+1,j} + \frac{U_1}{4}\sum_{ij} n_{ij} + \frac{U_1}{4}\sum_{ij} n_{i+1,j} \right. \\ &\quad \left. + \frac{U_1}{4}\sum_{ij} 1 + \frac{U_2}{4}\sum_{ij} n_{ij}n_{i,j+1} + \frac{U_2}{4}\sum_{ij} n_{ij} + \frac{U_2}{4}\sum_{ij} n_{i,j+1} + \frac{U_2}{4}\sum_{ij} 1 \right\}. \end{aligned} \tag{3.451}$$

First we note that since the sums run over all possible values of the pixels, we have

$$\sum_{ij} n_{ij} = \sum_{ij} n_{i+1,j} = \sum_{ij} n_{i,j+1} = N_1 - N_{-1} \tag{3.452}$$

where N_1 is the number of pixels with value 1 in the image and N_{-1} is the number of pixels with value -1 (or 0 originally) in the image. In addition, if the total number of pixels in the image is N, $\sum_{ij} 1 = N$. Then Equation (3.451) is simplified as follows:

$$\begin{aligned} H &= -\left(\frac{U_0}{2} + 2\frac{U_1}{4} + 2\frac{U_2}{4}\right)\sum_{ij} n_{ij} - \frac{U_1}{4}\sum_{ij} n_{ij}n_{i+1,j} - \frac{U_2}{4}\sum_{ij} n_{ij}n_{i,j+1} \\ &\quad -\left(\frac{U_0}{2} + \frac{U_1}{4} + \frac{U_2}{4}\right)\sum_{ij} 1 \\ &= -\left(\frac{U_0}{2} + 2\frac{U_1}{4} + 2\frac{U_2}{4}\right)(N_1 - N_{-1}) - \frac{U_1}{4}\sum_{ij} n_{ij}n_{i+1,j} - \frac{U_2}{4}\sum_{ij} n_{ij}n_{i,j+1} \\ &\quad -\left(\frac{U_0}{2} + \frac{U_1}{4} + \frac{U_2}{4}\right)N. \end{aligned} \tag{3.453}$$

Upon substitution of $U_1 = U_2 = -U_0/2$, the coefficient of the first term becomes 0 and we obtain:

$$H = \frac{U_0}{4}\sum_{ij}\left\{n_{ij}n_{i+1,j} + n_{ij}n_{i,j+1}\right\} - \frac{U_0 N}{4}. \tag{3.454}$$

Let us call C the total number of cliques in the image, vertical or horizontal, and let us call C_o the number of odd cliques, i.e. cliques of the form $(1,-1)$ or $(-1,1)$ and C_e the number of even

cliques, i.e. cliques of the form $(1,1)$ *or* $(-1,-1)$*. The odd cliques will produce* $n_{ij}n_{i+1,j}$ *or* $n_{ij}n_{i,j+1}$ *equal to* -1*, while the even cliques will make the same products equal to 1. So, the sum in (3.454) will be equal to the number of even cliques minus the number of odd cliques*

$$H = \frac{U_0}{4}(C_e - C_o) - \frac{U_0 N}{4} = \frac{U_0}{4}(C - 2C_o) - \frac{U_0 N}{4} = -\frac{U_0}{2}C_o + \frac{U_0(C-N)}{4} \tag{3.455}$$

where we used $C = C_o + C_e$*.*

Note that the last term on the right-hand side of (3.455) is a constant and, when we use this expression in the calculation of $p(X)$*, the exponential factor it will produce in the numerator of the expression will cancel with an identical factor of the normalising constant in the denominator. In other words, a constant in* H *does not affect the probability of a configuration to exist. So, the probability of a configuration* X *with the particular model parameters is given by*

$$p(X) = \frac{1}{Z} e^{\frac{U_0}{2}C_o} \tag{3.456}$$

which is a function only of the number of transitions in the image, from a black pixel to a white one or vice versa.

Example B3.159

Show that the result of Example 3.158 is also valid for clique potentials that do not necessarily obey the condition $U_1 = U_2 = -U_0/2$**, as long as the average grey value of the image is fixed.**

In this case the coefficient of the first term on the right-hand side of Equation (3.453) will not vanish. This coefficient multiplies factor $N_1 - N_{-1}$ *which is the total number of 1s and* -1*s in the image. This number divided with the total number* N *of pixels in the image is the average grey value of the image, which is assumed fixed. Then the first term on the right-hand side of (3.453) is another constant term that does not affect the probability of the configuration existing, which in this case will be given by*

$$p(X) = \frac{1}{Z} e^{\frac{U_1}{2}CH_o + \frac{U_2}{2}C_{V_o}} \tag{3.457}$$

where C_{H_o} *is the number of odd cliques in the horizontal direction and* C_{V_o} *is the number of odd cliques in the vertical direction.*

Example B3.160

Use a 10×10 **binary texture image to demonstrate that the number of odd cliques in both the vertical and horizontal directions is equal to the length of the loops which isolate "islands" of pixels with grey value 0 from pixels with grey value 1, if each pixel is treated like a tile of size** $l \times l$ **and lengths are measured with unit** l**.**

(Continued)

Example B3.160 (Continued)

With reference to Figure 3.156a we see that the total length of the perimeter of the three islands of 0s is $16 + 14 + 6 = 36$*. The horizontal odd cliques, highlighted in 3.156b, are counted to be 16, while the vertical such cliques, highlighted in 3.156c, are 20. Their sum is 36.*

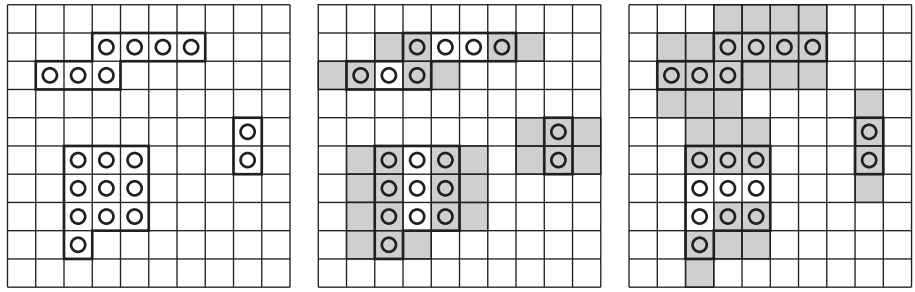


Figure 3.156 (a) A texture binary image. The circles indicate pixels with grey value 0, while the blank pixels have grey value 1. Grey is used to highlight the horizontal cliques of $(1, 0)$ and $(0, 1)$ in (b), and the vertical ones in (c).

Example B3.161

Show that there are at most $r(L) = 4L^2 3^L$ **different islands of 0s with perimeter** L**.**
Consider an internal point O of such an island. The furthest point of an island with perimeter L from O may be at most $L/2$ *pixels away. So, all islands of perimeter L that contain O must be restricted in an* $L \times L$ *square around O. Any point inside that square may be used as a starting point for drawing an island. Once we choose the starting point, we have the freedom to move in one of four possible directions. Once we reach the next point, we cannot go back, so we have the freedom to choose one of three possible directions. The same for each succeeding point. So, the total number of islands we can draw is:*

$$Number\_of\_starting\_points \times Number\_of\_options\_for\_the\_first\_step \times$$
$$Number\_of\_options\_for\_the\_second\_step \times$$
$$Number\_of\_options\_for\_the\_third\_step \times \ldots = L^2 \times 4 \times 3^L$$

.

The fact that this formula is an over-estimate is demonstrated in Figure 3.157 where we show all possible islands of perimeter 8 that may be constructed around a pixel. They are only 26, much fewer than the upper limit predicted by the formula, which is $4 \times 8^2 \times 3^8 = 1,679,616$*! The upper limit predicted by the formula becomes much tighter for higher values of L.*

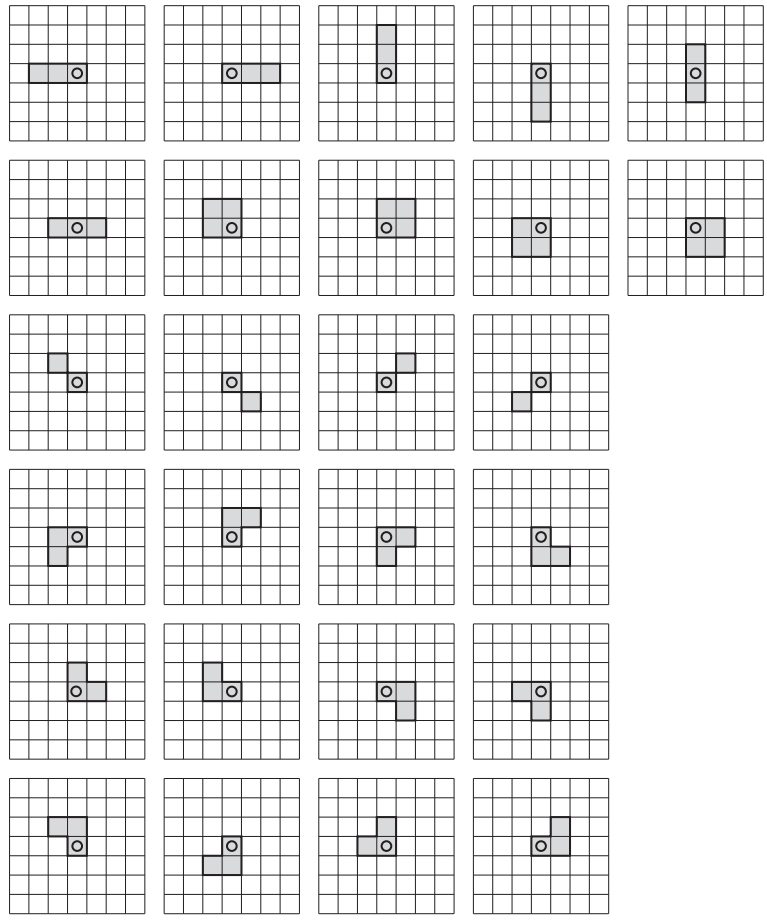


Figure 3.157 All islands of perimeter 8 one can draw around a pixel.

Example B3.162

Show that series $\sum_n n^2 x^n$ converges if $x < 1$.

For a series to converge the ratio of two successive terms must be less than 1 as $n \to \infty$. Let us call $a_n \equiv n^2 x^n$. We must show that $a_{n+1}/a_n < 1$:

$$\frac{a_{n+1}}{a_n} = \frac{(n+1)^2 x^{n+1}}{n^2 x^n} = \left(1 + \frac{1}{n}\right)^2 x. \tag{3.458}$$

For $n \to \infty$ the factor in the bracket tends to 1, and the ratio will be less than 1 if x is less than 1. Therefore, $\sum_n n^2 x^n$ converges to a finite number if $x < 1$.

Example B3.163

Show that for a configuration with probability density function given by Equation (3.456) it is possible to choose parameter U_0 so that the image behaves like a fractal in the sense that the value of a pixel is influenced by the values of all other pixels, no matter how far away they are.

Consider a finite square image and assume that the boundary around it consists of pixels with grey value 1. We shall show that for certain values of U_0, the probability of the pixel in the centre of the image taking a certain value will be affected by this boundary, irrespective of the size of the image, i.e. even when the boundary is placed very far away.

Let us consider the probability of the central pixel x_c having value 0. Since all boundary pixels around the image have value 1, for this pixel to have value 0 it must be part of, or form by itself, an island of 0s. Let us call the length of the perimeter of this island L. Let us denote by $\overline{X}$ any configuration that has the same island in the middle. For every such configuration $\overline{X}$ there exists a configuration X' which is identical to $\overline{X}$, but it does not contain the central island. Figure 3.158 shows some examples of configurations $\overline{X}$, and their corresponding configurations X'.

Let us call an island of this particular shape, island_A. The probability of having a configuration with such an island in the middle is given by

$$p(island\_A) = \frac{\sum_{\overline{X}} e^{\frac{U_0}{2} C_o(\overline{X})}}{\sum_X e^{\frac{U_0}{2} C_o(X)}} \tag{3.459}$$

where we make explicit that the number of odd cliques C_o that appears in Equation (3.456) is different for different configurations.

If we reduce the number of configurations over which we sum in the denominator, we may obtain an upper limit for this probability. We achieve this by summing over configurations X' only, instead of summing over all possible configurations:

$$p(island\_A) \leq \frac{\sum_{\overline{X}} e^{\frac{U_0}{2} C_o(\overline{X})}}{\sum_{X'} e^{\frac{U_0}{2} C_o(X')}}. \tag{3.460}$$

However, for each configuration $\overline{X}$ we have a corresponding configuration X' that has L fewer odd cliques since their only difference is the removal of island_A of perimeter L (see Figure 3.158). So,

$$C_o(\overline{X}) = C_o(X') + L. \tag{3.461}$$

Upon substitution in (3.460) we obtain:

$$p(island\_A) \leq \frac{\sum_{\overline{X}} e^{\frac{U_0}{2} C_o(\overline{X})}}{\sum_{\overline{X}} e^{\frac{U_0}{2} (C_o(\overline{X}) - L)}} = e^{\frac{U_0}{2} L}. \tag{3.462}$$

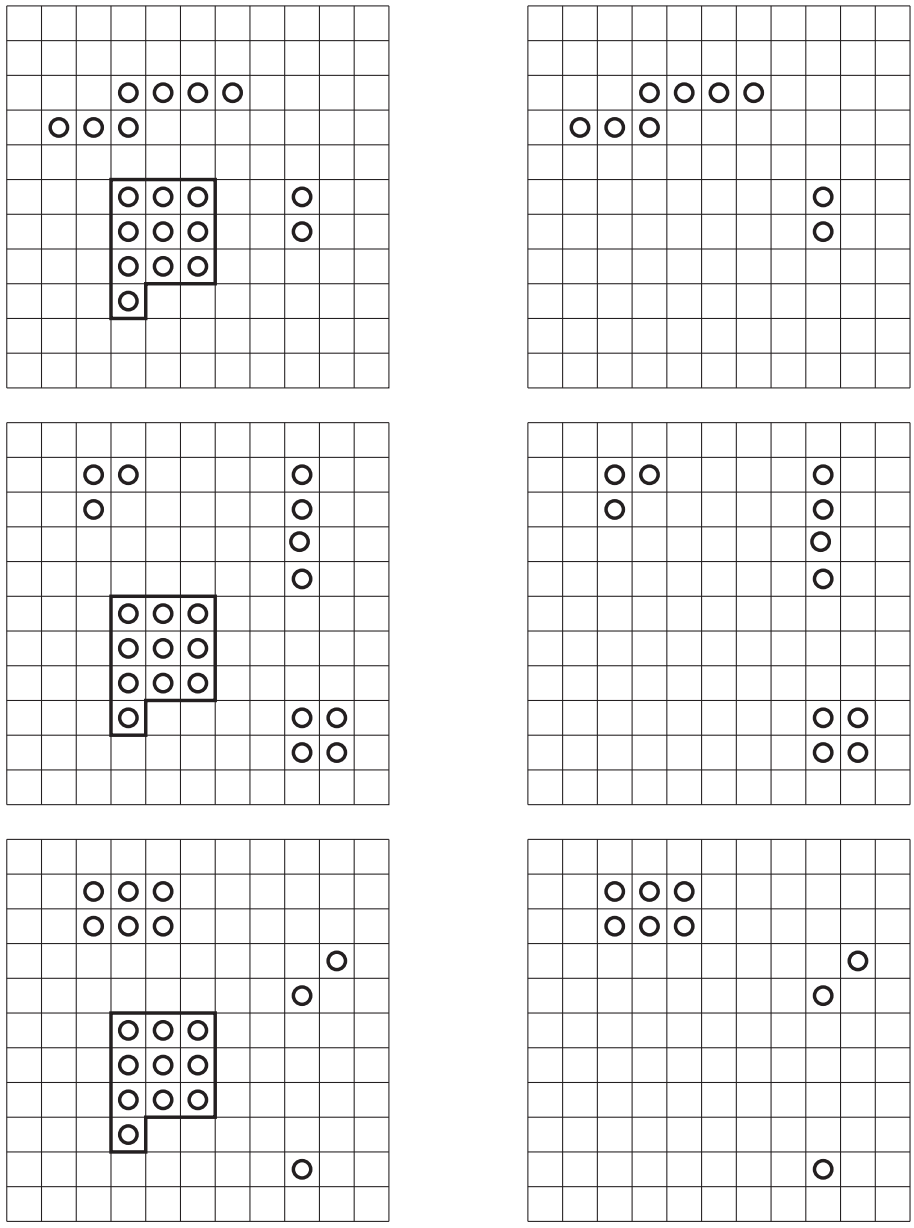


Figure 3.158 On the left some configurations $\overline{X}$ with the same island of 0s in the middle and on the right their corresponding configurations X' without the island. Note that in all cases all border pixels have value 1 (indicated by blank here).

The probability of the central pixel having value 0 is the same as the probability of having around it an island of 0s of whatever perimeter. If there are $r(L)$ different ways to draw an island of perimeter L, then the probability of the central pixel having value 0 is:

$$p(x_c = 0) = \sum_L r(L)p(island\_with\_perimeter\_L) \leq \sum_L r(L)\mathrm{e}^{\frac{U_0}{2}L}. \tag{3.463}$$

(Continued)

Example B3.163 (Continued)

If we make use of the result of Example 3.160, we have

$$p(x_c = 0) \leq \sum_L 4L^2 3^L \mathrm{e}^{\frac{U_0}{2}L} = 4\sum_L L^2 \left(3\mathrm{e}^{\frac{U_0}{2}}\right)^L. \tag{3.464}$$

According to the result of Example 3.161 the sum on the right-hand side of the above expression is a finite number if we choose U_0 small enough so that the expression in the bracket is less than 1 (remember that U_0 is assumed to be positive). Let us say that we choose $U_0/2$ so that the right-hand side of (3.464) has value $\alpha < 0.5$. Then $p(x_c = 0) < \alpha$ while $p(x_c = 1) > \alpha$.

*Now assume that the border pixels all around the image had 0 value instead of 1. Following exactly the same procedure, we would have found that in this case $p(x_c = 1) < \alpha$ while $p(x_c = 0) > \alpha$. In other words, when the border pixels had value 1, the probability of the central pixel to be 1 was **more** than α, while when the border pixels had value 0, the probability of the central pixel to be 1 was **less** than α. So, the central pixel "feels" the effect of the boundary no matter how far that boundary is. This shows that for the particular choice of U_0, the Markov random field no longer possesses the Markovian property. Instead, it behaves more like a fractal with long-range correlations present. This behaviour can be observed when U_0 is sufficiently small, i.e. its inverse, which corresponds to the temperature parameter in the terminology borrowed from physics, is sufficiently high. The value of $1/U_0$ for which this happens is the **critical temperature** of the image model. An MRF has to be below its critical temperature in order to correspond to a Gibbs distribution.*

3.8 Texture Repair

What is texture repair?

Texture repair is the process of restoring part of a texture image. It involves filling up gaps in the image so that the gap, or the place where the gap was, is no longer discernible by eye.

How can we perform texture repair?

There are three broad approaches:

i) Statistical methods, where the missing texture is filled-in while preserving some statistical property of the texture image; an example of such an approach is Kriging.
ii) Filtering methods, where the values of neighbouring pixels are propagated to the gaps through some filtering process; examples of such methods are **inpainting**, **normalised convolution** and **error correction** methods.
iii) Non-parametric MRF methods; examples of such approaches are the **Efros and Leung** algorithm and all its variations.

In general, filtering based methods are only appropriate for filling in small parts of the texture. They are excellent for non-textured images with relatively flat regions, as they are actually interpolation methods relying on the assumption of smoothness, rather than on the idea of replicating a pattern. Statistical methods are appropriate mostly for random textures. The non-parametric Markov random field methods are the most powerful ones for generic textures.

What is Kriging?

Kriging assigns to a pixel in the gap a value that is a weighted linear combination of the values of the available pixels, with the weights chosen so that the **covariance** of the image is preserved. Kriging belongs to the type of estimator known as the **best linear unbiased estimator** or **BLUE**.

How does Kriging work?

Let us assume that we have values for the following pixels, indexed here with a single index for simplicity: $P_1, P_2, \dots, P_N$. Let us assume that their mean grey value is μ and their variance is σ^2. Let $V(P_i)$ be the grey value of pixel P_i. Finally, let us denote by P_0 the pixel for which we do not have a value. Our problem is to assign a grey value to pixel P_0, so that the overall appearance of the texture is not affected.

According to Kriging, the value of P_0 should be

$$\hat{V}(P_0) = \sum_{i=1}^{N} w_i(P_0) V(P_i) \tag{3.465}$$

with the weights w_i chosen so that the variance of error between the estimated value $\hat{V}(P_0)$ and the true (unknown) value $V(P_0)$ is minimised. The residual error is defined as

$$R(P_0) \equiv \hat{V}(P_0) - V(P_0). \tag{3.466}$$

If we treat the image as a random field, $V(P_0)$ is a random variable and so is $R(P_0)$. We may then define the variance of the error and choose weights w_i so that the variance of error $R(P_0)$ is minimised.

It can be shown (see Example 3.164) that the variance σ_R^2 of error $R(P_0)$ is given by

$$\sigma_R^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j C_{ij} - 2\sum_{i=1}^{N} w_i C_{i0} + \sigma^2 + \mu^2 \tag{3.467}$$

where C_{ij} is the correlation between random variables $V(P_i)$ and $V(P_j)$ and C_{i0} is the correlation between random variables $V(P_i)$ and $V(P_0)$.

It turns out (see Example 3.165) that the weights may be estimated from:

$$W = C^{-1} D \tag{3.468}$$

where C is an $(N+1) \times (N+1)$ matrix, of the form

$$C = \begin{pmatrix} C_{11} & \dots & C_{1N} & 1 \\ \dots & \dots & \dots & \dots \\ C_{N1} & \dots & C_{NN} & 1 \\ 1 & \dots & 1 & 0 \end{pmatrix} \tag{3.469}$$

D is an $(N+1)\times 1$ vector:

$$D = \begin{pmatrix} 2C_{10} \\ \cdots \\ 2C_{N0} \\ 1 \end{pmatrix} \tag{3.470}$$

and W is the $(N+1)\times 1$ vector of the unknowns,

$$W = \begin{pmatrix} w_1(P_0) \\ \cdots \\ w_N(P_0) \\ \lambda \end{pmatrix}. \tag{3.471}$$

The value of λ, which is an extra unknown introduced in the process, is disregarded after the solution has been obtained.

Example B3.164

Starting from Equations (3.465) and (3.466) prove Equation (3.467).
Let us start from Equation (3.466) and substitute on the right-hand side the estimate $\hat{V}(P_0)$ from (3.465):

$$R(P_0) \equiv \hat{V}(P_0) - V(P_0) = \sum_{i=1}^{N} w_i(P_0)V(P_i) - V(P_0). \tag{3.472}$$

Treating $R(P_0)$ as a random variable allows us to write a formula for its variance with the help of the expectation operator $E\{\ \}$:

$$\sigma_R^2 \equiv E\left\{ \left(R(P_0) - \overline{R(P_0)} \right)^2 \right\}. \tag{3.473}$$

Here $\overline{R(P_0)}$ is the mean value of $R(P_0)$. First, we shall show that this is 0, as long as the weights we choose sum up to 1, i.e. $\sum_{i=1}^{N} w_i = 1$:

$$\begin{aligned} \overline{R(P_0)} &\equiv E\left\{ \sum_{i=1}^{N} w_i(P_0)V(P_i) - V(P_0) \right\} \\ &= \sum_{i=1}^{N} w_i E\left\{V(P_i)\right\} - E\left\{V(P_0)\right\} \\ &= \sum_{i=1}^{N} w_i\mu - \mu = \mu\sum_{i=1}^{N} w_i - \mu = \mu - \mu = 0. \end{aligned} \tag{3.474}$$

Here we made use of the fact that the expectation operator is applied only to the random variables, and that the expectation value at each pixel position is the same, equal to μ. This assumption is based on the fact that we are dealing with stationary textures and we may invoke the assumption of ergodicity. For more on ergodicity the reader is referred to Book I [75].

Then:

$$\begin{aligned}
\sigma_R^2 &= E\left\{\left(R(P_0)\right)^2\right\} \\
&= E\left\{\left(\sum_{i=1}^{N} w_i(P_0)V(P_i) - V(P_0)\right)\left(\sum_{j=1}^{N} w_j(P_0)V(P_j) - V(P_0)\right)\right\} \\
&= E\left\{\sum_{i=1}^{N} w_i(P_0)V(P_i)\sum_{j=1}^{N} w_j(P_0)V(P_j) - V(P_0)\sum_{i=1}^{N} w_i(P_0)V(P_i) - \right. \\
&\quad \left. V(P_0)\sum_{j=1}^{N} w_j(P_0)V(P_j) + V(P_0)^2\right\} \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} w_i(P_0)w_j(P_0)E\left\{V(P_i)V(P_j)\right\} - \sum_{i=1}^{N} w_i(P_0)E\left\{V(P_i)V(P_0)\right\} - \\
&\quad \sum_{j=1}^{N} w_j(P_0)E\left\{V(P_j)V(P_0)\right\} + E\left\{V(P_0)^2\right\} \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} w_i(P_0)w_j(P_0)C_{ij} - \sum_{i=1}^{N} w_i(P_0)C_{i0} - \sum_{j=1}^{N} w_j(P_0)C_{j0} + E\left\{V(P_0)^2\right\} \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} w_i(P_0)w_j(P_0)C_{ij} - 2\sum_{i=1}^{N} w_i(P_0)C_{i0} + E\left\{V(P_0)^2\right\}. \qquad (3.475)
\end{aligned}$$

The last equation was derived by observing that j was a dummy index and so it could be renamed i. We also have for the variance of the grey values:

$$\begin{aligned}
\sigma^2 &\equiv E\left\{\left(V(P_0) - \mu\right)^2\right\} \\
&= E\left\{V(P_0)^2 - 2V(P_0)\mu + \mu^2\right\} \\
&= E\left\{V(P_0)^2\right\} - 2\mu E\left\{V(P_0)\right\} + \mu^2 \\
&= E\left\{V(P_0)^2\right\} - 2\mu^2 + \mu^2 \Rightarrow \\
E\left\{V(P_0)^2\right\} &= \sigma^2 + \mu^2. \qquad (3.476)
\end{aligned}$$

After substitution in (3.475), we obtain (3.467).

In (3.475) C_{ij} is the correlation matrix between positions P_i and P_j, while C_{i0} is the correlation matrix between position P_i and P_0. Assuming isotropy and stationarity, the value of C_{ij} depends only on the relative distance between pixels P_i and P_j, while the value of C_{i0} also depends only on the relative distance between positions P_i and P_0. So, both values may be computed from the available data. The same goes for σ^2 and μ: they are both estimated from the available pixel values. Note that a good practice would be to remove the mean of the image before we perform Kriging, so that $\mu = 0$ and the correlation matrix is the same as the covariance matrix. The mean may always be added at the end.

Example B3.165

Work out the weights that will minimise the variance of the estimation error, as expressed by Equation (3.467), and at the same time they will sum up to 1.

This is a constrained optimisation problem. We shall use the method of ***Lagrange multipliers****. According to this method, we create a linear combination of the original function that has to be minimised and the constraint, and then we choose the weights to minimise this linear combination. In this particular problem, the linear combination we create is*

$$\tilde{\sigma}_R^2 \equiv \underbrace{\sum_{i=1}^{N}\sum_{j=1}^{N} w_i(P_0)w_j(P_0)C_{ij} - 2\sum_{i=1}^{N} w_i(P_0)C_{i0} + \sigma^2 + \mu^2}_{\text{original\_function\_to\_be\_minimised}} + \underbrace{\lambda\left(\sum_{i=1}^{N} w_i(P_0) - 1\right)}_{=0(constraint)} \tag{3.477}$$

where λ is the so called Lagrange multiplier. We have $(N+1)$ unknowns now, the N weights plus λ. We must differentiate $\tilde{\sigma}_R^2$ with respect to the $N+1$ unknowns and set these first partial derivatives to zero, to obtain a system of linear equations for the unknowns.

First we must work out the derivative of $\tilde{\sigma}_R^2$ with respect to one of the weights. Differentiating both sides of (3.477) with respect to $w_i(P_0)$, we obtain:

$$\frac{\partial\tilde{\sigma}_R^2}{\partial w_i(P_0)} = \sum_{j=1}^{N} w_j(P_0)C_{ij} - 2C_{i0} + \lambda. \tag{3.478}$$

Let us set the right-hand side equal to 0:

$$w_1(P_0)C_{i1} + w_2(P_0)C_{i2} + \cdots + w_N(P_0)C_{iN} + \lambda = 2C_{i0}. \tag{3.479}$$

We can produce one such equation for each value of i, that is we shall have N such equations for the N unknown weights.

Let us next differentiate $\tilde{\sigma}_R^2$ with respect to λ:

$$\frac{\partial\tilde{\sigma}_R^2}{\partial\lambda} = \sum_{i=1}^{N} w_i(P_0) - 1. \tag{3.480}$$

Setting the right-hand side of this expression equal to 0 yields:

$$w_1(P_0) + w_2(P_0) + \cdots + w_N(P_0) = 1. \tag{3.481}$$

Equations (3.479), for $i = 1, \ldots, N$, and Equation (3.481) constitute a system of $N+1$ linear equations for the $N+1$ unknowns, which may be written as

$$CW = D \tag{3.482}$$

with C given by (3.469), W given by (3.471) and D given by (3.470).

How can we estimate the correlation matrix of the data?

This is done via the variogram (see Example 3.75. To reduce the effect of noise, we use first a parametric model to fit the variogram (see Examples 3.75 and 3.76) and then work out the correlation matrix from the variogram model as

$$C(d) = \sigma^2 - \gamma(d) \tag{3.483}$$

where σ^2 is the variance of the grey values of the pixels of the image.

How do we perform Kriging interpolation in practice?

We apply the following algorithm.

Step 0: Decide the number of neighbours you will use to fill in a missing pixel value. Let us call this number N.
Step 1: Calculate the mean μ of the pixels with correct grey values and remove it from all pixel values of the image.
Step 2: From the part of the image that is not damaged, compute the variogram of the image using the algorithm in the section How do we compute the variogram of an image in practice?
Step 3: Fit the variogram with a chosen model, using the algorithm in the section How can we fit automatically a model to a variogram?
Step 4: From the variogram work out the covariance $C(d)$ of the image using Equation (3.483). (Note that since we have removed the mean, correlation and covariance is the same.)
Step 5: For each pixel with no value, consider the N nearest neighbours it has. For neighbour i, work out its distance d_{i0} in order to set $C_{i0} = C(d_{i0})$. For neighbours i and j work out their distance d_{ij}, so you may set $C_{ij} = C(d_{ij})$.
Step 6: Use Equation (3.468) to work out the weights for this particular pixel.
Step 7: Use Equation (3.465) to work out the grey value of the pixel as the linear combination of the values of the N nearest neighbours you selected.
Step 8: Repeat the process for each pixel without grey value, using as pixels with known values also the pixels to which you have already assigned values.
Step 9: At the end add the mean μ to all pixel values.

Can we use another statistic to guide us in selecting the missing pixel values?

Yes, we may select any statistic we believe expresses best the appearance of the texture, and then select the values of the damaged pixels so that the particular statistic is preserved. Kriging uses the covariance of the texture as it is believed that the human visual system is sensitive up to second order statistics of the image. However, second order statics may also be expressed by the use of co-occurrence matrices. So, one may devise a texture interpolation method where the co-occurrence matrix is preserved (see Example 3.166).

How can we repair a texture while preserving its co-occurrence matrices?

We have first to capture the texture characteristics by constructing the appropriate co-occurrence matrices and then use them to repair the texture. So, first we do the following.

Step 1: Decide upon the co-occurrence matrices you wish to preserve.
Step 2: Construct the co-occurrence matrices you wish to preserve by using the undamaged part of the image.
Step 3: Normalise each co-occurrence matrix by dividing all its elements with the number of pairs you used to construct it.

These normalised co-occurrence matrices will be used as lookup tables. Each such co-occurrence matrix gives the probability of occurrence of a pair of grey level values at the particular relative position.

Let us say that we decided to construct all directional co-occurrence matrices that correspond to all pairs of relative positions one may create by pairing the central pixel of an $M \times M$ window with all other pixels inside the window. This way one will have $M^2 - 1$ different co-occurrence matrices. When we want then to interpolate the texture image, we scan the image again. Around each pixel with no value, we identify all pixels with valid values inside an $M \times M$ window. For each one of these pixels we consult the corresponding lookup table, according to its relative position with respect to the pixel under repair, in order to infer the possible values of the pixel in the centre and the probabilities with which these values are possible. As we may have as many as $M^2 - 1$ co-occurrence matrices to consult, in principle, for each possible grey value, we may have up to $M^2 - 1$ probabilities read from these co-occurrence matrices. We then sum up the probabilities that correspond to each possible grey value and so we end up with a list of possible grey values for each pixel and a corresponding sum of probabilities for each value. Because we are interested in *relative* probabilities for each grey value for the pixel under consideration, we normalise these sums of probabilities by dividing them with their sum. Thus, every pixel now has associated with it a list of possible grey values and a relative probability for each grey value to arise. We may create a first reconstruction of the image by assigning to each pixel the value with the highest probability. This reconstruction is expected to be quick.

We apply this algorithm starting from the points around the border of the area to be repaired, so the damaged pixels have enough neighbours with valid values.

The reconstruction may be performed according to the following algorithm, which we call **co-occurrence mode** algorithm.

Step 1: Visit every pixel that does not have a value, but it has neighbours with valid values inside its $M \times M$ neighbourhood.

Step 2: Go through the corresponding lookup table for each neighbour with a valid value, according to its relative location with respect to the focal pixel, and compile a list of possible values for the focal pixel, with their corresponding probabilities. You will have at most $M^2 - 1$ probabilities for each possible grey value, from the $M^2 - 1$ lookup tables.

Step 3: Compute the sum of probabilities for each possible grey value.

Step 4: Assign to the focal pixel the grey value with the maximum probability.

Step 5: Pixels with assigned values may be treated as neighbours with valid values when the $M \times M$ neighbourhoods of other pixels are considered. This way, the pattern will grow inwards in the damaged area.

One may alternatively use a more sophisticated algorithm, where the probabilities of all possible grey values of an image are taken into consideration. We call this algorithm **co-occurrence probabilistic** algorithm. The following steps have to be used after step 3 of the co-occurrence mode algorithm.

Step 4: Divide the probability of each possible grey value by the sum of probabilities of all grey values, i.e. normalise the probabilities of all possible grey values of the focal pixel, so they sum up to 1.

Step 5: Create for each pixel a list of possible grey values with their corresponding normalised probabilities and the cumulative distribution from them, like the one shown in Table 3.59. This

Table 3.59 All possible grey values g_i of a pixel, their corresponding probabilities P_i and their cumulative distribution needed for drawing a random number according to the probabilities listed in the second column.

g_1	P_1	$[0, P_1)$
g_2	P_2	$[P_1, P_1 + P_2)$
g_3	P_3	$[P_1 + P_2, P_1 + P_2 + P_3)$
...	...	...
g_N	P_N	$[\sum_{i=1}^{N-1} P_i, 1]$
Sum	1	

list assigns to each possible grey value a range of real numbers, all in the interval $[0, 1]$. The more probable a grey value is, the larger the range of values it is assigned.

Step 6: Draw a number uniformly distributed in the range $[0, 1]$.

Step 7: Assign to the focal pixel the grey value that corresponds to the interval inside which this random number happens to fall. In other words, we assign to the focal pixel a grey value according to the probabilities dictated by the co-occurrence matrices. The more probable a grey value is, the larger the range of values it is assigned, the more likely it is for a random number from a uniform distribution between 0 and 1 to fall in its interval, without excluding the option to fall in the interval of one of the other possible grey values.

Step 8: Pixels with assigned values may be treated as neighbours with valid values when the $M \times M$ neighbourhoods of other pixels are considered. This way, the pattern will grow inwards in the damaged area.

Finally, we may use an iterative scheme, according to which the values of the damaged pixels are constantly updated, so that the co-occurrence matrices of the reconstructed image become more and more similar to the co-occurrence matrices of the undamaged pixels we had to begin with. In order to measure the similarity between an original co-occurrence matrix and the corresponding one of the reconstructed image, we may use the χ^2 statistic defined in the section What is the χ^2 test? So, we define the cost function U, for a particular type of co-occurrence matrix corresponding to a fixed relative location of pixels, as:

$$U \equiv \sum_{i=0}^{255} \sum_{j=0}^{255} \frac{\left[m_1 \times C_{\text{after}}(i,j) - m_2 \times C_{\text{data}}(i,j)\right]^2}{C_{\text{after}}(i,j) + C_{\text{data}}(i,j)} \tag{3.484}$$

where

$$m_1 = \sqrt{\frac{\sum_{i=0}^{255} \sum_{j=0}^{255} C_{\text{data}}(i,j)}{\sum_{i=0}^{255} \sum_{j=0}^{255} C_{\text{after}}(i,j)}} \tag{3.485}$$

and

$$m_2 = \sqrt{\frac{\sum_{i=0}^{255}\sum_{j=0}^{255} C_{\text{after}}(i,j)}{\sum_{i=0}^{255}\sum_{j=0}^{255} C_{\text{data}}(i,j)}} \tag{3.486}$$

with C_{data} being the co-occurrence matrix constructed from the undamaged pairs of pixels and C_{after} the corresponding co-occurrence matrix constructed by using all pairs of pixels in the repaired image.

The iterative algorithm one may then use starts with the output of one of the previous algorithms and tries to improve it. We call this algorithm **co-occurrence iterative**. One single pass of $M \times M$ non-overlapping windows of the image constitutes one **phase**. At each phase, we consider only the pixels that do not have $M \times M$ overlapping neighbourhoods. This is so that if we update the value of a pixel to agree with the value its neighbours tell it to have, we must avoid then changing immediately the values of the neighbours according to their own neighbours and destroy any benefit we had achieved. Several phases constitute one **iteration**. An iteration consists of the number of necessary passes so that all pixels we have to reconstruct are visited once each, and a proposed value is considered by using the $M \times M$ neighbourhood of each one of them, provided of course this neighbourhood contains at least one pixel with known value. This algorithm has the same steps as the co-occurrence probabilistic algorithm up and including step 6. Then one should use the following steps.

Step 7: The new value of the pixel is checked first whether it reduces the cost function or not. We denote the value of the cost function before updating the value of the pixel by U_{before}. U_{after} is computed by using the proposed new value of the pixel under consideration. The new value of the reconstructed pixel is accepted only if $U_{\text{after}} < U_{\text{before}}$, for all co-occurrence matrices.

Step 8: Perform all passes and do all iterations needed to visit once all pixels under repair.

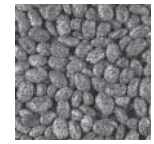
Step 9: If the cost function converges to zero or reaches a steady state, i.e. if there are no points updated in, say, 5 successive iterations, then exit.

All these algorithms rely on adequate statistical representation of the co-occurrence properties of the image. It is possible that an image with 256 grey levels does not supply sufficient statistics for the construction of the co-occurrence matrices. One way to deal with the problem is to reduce the number of grey levels. For example, we may use grey levels in the range $[0, 85]$, by appropriately re-quantising the grey values. After the reconstruction of the texture, we may restore the values of the original pixels and stretch the values of the repaired pixels to the range $[0, 255]$ by appropriate scaling. To avoid having missing grey levels in the repaired region of the image, we may choose at random one of the following three numbers to add to the grey value of a repaired pixel: $\{-1, 0, +1\}$. A pixel with grey value 1 when the range $[0, 85]$ was used, would have acquired value $1 \times 255/85 = 3$ when scaled to the range $[0, 255]$, while a pixel with grey value 2 when the range $[0, 85]$ was used, would have acquired value $2 \times 255/85 = 6$ when scaled to the range $[0, 255]$. This would leave gaps in the grey levels, as no pixels will have grey value 1, 2, 4 or 5. When we randomly add one of the three numbers $\{-1, 0, +1\}$ to a repaired pixel value, we repopulate the empty grey levels. We can do this without noticeable change in the appearance of the texture, as the human eye can hardly discern differences in grey values smaller than 3 levels.

Example 3.166

Use the three co-occurrence algorithms to reconstruct the damaged image of Figure 3.159, by setting $M = 5$.

Figure 3.159 The damaged image to be reconstructed. Source: Maria Petrou.



To facilitate our work, let us encode the 24 neighbours of a pixel with the letters shown in Figure 3.160.

Let us denote the co-occurrence matrices we construct according to the paired letters in Figure 3.160, as C_{AB}, C_{AC}, C_{AD}, etc. To create these matrices we follow this algorithm.

Step 1: *Create 24, 256×256 arrays and set all their elements to 0.*
Step 2: *Initialise counters: $N_{AB} = 0$, $N_{AD} = 0$, etc.*
Step 3: *Scan the image. At every non-empty pixel A you find, check if its neighbour B is non-empty.*
If non-empty, set
$N_{AB} = N_{AB} + 1$
$C_{AB}(I(\mathrm{A}), I(\mathrm{B})) = C_{AB}(I(\mathrm{A}), I(\mathrm{B})) + 1$
where C_{AB} is the co-occurrence matrix for the pairs of pixels that are at relative position A and B, and $I(\mathrm{A})$ and $I(\mathrm{B})$ represent the grey values of A and B, respectively.
Step 4: *Check if neighbour C is non-empty. If non-empty set*
$N_{AC} = N_{AC} + 1$
$C_{AC}(I(\mathrm{A}), I(\mathrm{C})) = C_{AC}(I(\mathrm{A}), I(\mathrm{C})) + 1.$
Repeat this for all 24 relative positions around pixel A.
Step 5: *After all pixels have been processed, replace*
$C_{\zeta\xi}(i,j)$ *with* $\frac{C_{\zeta\xi}(i,j)}{N_{\zeta\xi}}$ *for all* (i,j) *and for all* $(\zeta,\xi) \in \{(A, B), (A, D), \ldots, (A, Y)\}$

This way we have constructed the look up tables we need for the next stage. We calculate the C_{AB} as shown in Figure 3.161

Figure 3.160 Each position, encoded here with a letter, in this 5×5 neighbourhood, will be paired with position A to define a co-occurrence matrix.

K	J	I	H	G
L	N	D	E	F
P	O	(A)	B	C
Q	R	S	T	M
Y	X	W	V	U

(Continued)

Example 3.166 (Continued)

Figure 3.161 The feature map of co-occurrence matrix C_{AB}. Source: Maria Petrou.

What is in-painting?

It is a method that fills-in the damaged region of an image by propagating inwards the lines that end up at the border of that region, as well as the grey values of the surrounding region. It may be used to fill in small damaged patches of a texture, like, for example, the gaps created if we mask out some scratches or some text written over the texture. There are several variations of the approach. We give here the steps of the original basic algorithm.

Step 0: Use a binary mask to identify the damaged region.

Step 1: Consider the mask that identifies the damaged region and dilate it with a 3×3 structuring element.

Step 2: Fill in the damaged region with the mode grey value of the part of the image that is in the augmented minus the initial mask that marks the damaged region.

Step 3: Inside the augmented region compute the Laplacian of each pixel using

$$L(i,j) = I(i+1,j) + I(i-1,j) + I(i,j+1) + I(i,j-1) - 4I(i,j) \tag{3.487}$$

where $I(i,j)$ is the grey value of pixel (i,j).

Step 4: For each pixel inside the region that is to be repaired, compute the gradient of the Laplacian as $\boldsymbol{\delta}\mathbf{L}(\mathbf{i},\mathbf{j}) = (\delta L_x(i,j), \delta L_y(i,j))$, with:

$$\delta L_x(i,j) = L(i+1,j) - L(i-1,j) \qquad \delta L_y(i,j) = L(i,j+1) - L(i,j-1). \tag{3.488}$$

Step 5: For each pixel inside the region that is to be repaired, compute the normal $\mathbf{n(i,j)} = (n_x(i,j), n_y(i,j))$ to the gradient vector $(g_x(i,j), g_y(i,j)) = (I(i+1,j) - I(i-1,j), I(i,j+1) - I(i,j-1))$, as follows:

$$n_x(i,j) = -g_y(i,j) = I(i,j-1) - I(i,j+1) \qquad n_y(i,j) = g_x(i,j) = I(i+1,j) - I(i-1,j). \tag{3.489}$$

Step 6: For each pixel inside the region that is to be repaired, compute:

$$\gamma(i,j) \equiv \delta\mathbf{L}(i,j) \cdot \mathbf{n(i,j)}. \tag{3.490}$$

Step 7: If $\gamma(i,j) \neq 0$, compute:

$$\beta(i,j) \equiv \frac{\gamma(i,j)}{\sqrt{n_x(i,j)^2 + n_y(i,j)^2}}. \tag{3.491}$$

Step 8: If $\gamma(i,j) > 0$, set:

$$|\nabla I(i,j)| = \Big[\big[\min\{I(i,j) - I(i-1,j), 0\}\big]^2 + \big[\max\{I(i+1,j) - I(i,j), 0\}\big]^2$$

$$+\left[\min\{I(i,j)-I(i,j-1),0\}\right]^2+\left[\max\{I(i,j+1)-I(i,j),0\}\right]^2\Big]^{\frac{1}{2}}. \quad (3.492)$$

If $\gamma(i,j)<0$, set:

$$|\nabla I(i,j)|=\Big[\left[\max\{I(i,j)-I(i-1,j),0\}\right]^2+\left[\min\{I(i+1,j)-I(i,j),0\}\right]^2$$
$$+\left[\max\{I(i,j)-I(i,j-1),0\}\right]^2+\left[\min\{I(i,j+1)-I(i,j),0\}\right]^2\Big]^{\frac{1}{2}}. \quad (3.493)$$

Step 9: Update the value of each pixel in the damaged region according to

$$I^{n+1}(i,j)=I^n(i,j)+\Delta t\beta^n(i,j)\,|\nabla I^n(i,j)| \quad (3.494)$$

where superscript n indicates the values in the previous iteration step and Δt is an updating coefficient that might be set to 1.

The philosophy behind each step of the algorithm is as follows. The algorithm wants to propagate the smoothness of the image along the lines of constant intensity. As the gradient vector indicates the direction of maximum change of the image, the direction of constant intensity (or least change) is expected to be locally orthogonal to that. This is the direction of the local isophote (curve of constant brightness). So, the isophote vector we compute in step 5 **n(i,j)** when denoted with the gradient vector **g(i,j)** yields 0, i.e. it is orthogonal to it. Now, we have the direction of the local isophote and we want to propagate along this direction some measure of smoothness of the image. The smoothness of the image may be expressed by the local value of the Laplacian, which is the sum of the second differences of the image. This is given by Equation (3.487) used in step 3. However, we need to know how much this local smoothness varies locally, so we compute its gradient in step 4. Note that as we need to take differences of the Laplacian to compute its gradient inside a local window of size 3×3, we must have the value of the Laplacian in pixels just outside the region we have to repair. Thus, in step 2 we enlarge the mask of the damaged region by a pixel all around, in order to compute the Laplacian inside that region. In addition, as the calculation of the Laplacian itself requires the use of a 3×3 neighbourhood, we need to have a 2-pixel thick layer all around the region we wish to repair. In steps 6 and 7, we project the change of the measure of smoothness, i.e. the Laplacian, along the direction of the local isophote. Then we scale it according to the strength of the local gradient. Now this could have been done by simply omitting the denominator in Equation (3.491), since that denominator is actually the magnitude of the local gradient vector. However, the proponents of this algorithm observed that this leads to numerical instabilities, so they proposed to use instead, as a measure of the local gradient magnitude, the quantity calculated in step 9, which is called the **slope-limited norm of the gradient vector**. See Example 3.167 for an intuitive understanding of definitions (3.492) and (3.493).

Finally, in step 9, we update the value of the pixel, increasing it if the Laplacian increases in the direction of the isophote, and decreasing it if the Laplacian decreases in the direction of the isophote (see Example 3.168).

Example 3.167

Consider formulae ((3.492)) and ((3.493)). Explain their logic with the help of the image patch shown in Figure 3.162.

(Continued)

Example 3.167 (Continued)

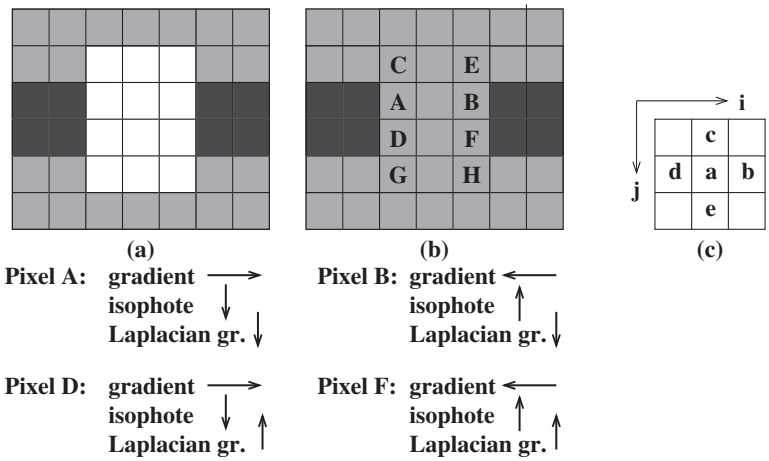


Figure 3.162 (a) An image patch that is to be repaired, by giving values to the white pixels. (b) The initial assignment of values according to the mode of the surrounding pixels. (c) Neighbourhood of a pixel that determines how much its value will change according to the value of $|\nabla I(i,j)|$ computed with the help of (3.492) and (3.493).

We have to initialise this algorithm by giving to the masked pixels in 3.162a the mode value of the surrounding pixels. This creates the first approximation of the repaired image shown in Figure 3.162b. Now we have to update the values of the pixels that were masked in 3.162a. Let us consider the updating of pixels A, B, D and F. At the bottom of Figure 3.162 we show the directions of the key vectors associated with these pixels: the gradient vector (that points always towards the brightest neighbour in the direction of maximum change), the isophote vector (that is defined so its horizontal component is minus the vertical component of the gradient vector and its vertical component is the same as the horizontal component of the gradient vector) and the gradient of the Laplacian (that points in the direction of maximum dissimilarity of a pixel and its four nearest neighbours).

For pixels A and F, for which the gradient of the Laplacian points in the same direction as the isophote vector ($\gamma > 0$), we must use formula (3.492) to compute $|\nabla I(i,j)|$. We may express this formula in terms of the values of the neighbouring pixels of the pixel under consideration, if we encode the values of the relevant neighbours as shown in Figure 3.162c:

$$|\nabla I(i,j)| = \Big[(\min\{a-d,0\})^2 + (\max\{b-a,0\})^2 + (\min\{a-c,0\})^2 + (\max\{e-a,0\})^2\Big]^{\frac{1}{2}}. \tag{3.495}$$

Then for pixel A:

$$a = c = b = e > d \Rightarrow |\nabla I(i,j)| = |\min\{a-d,0\}| = 0. \tag{3.496}$$

For pixel F:

$$a = c = d = e > b \Rightarrow |\nabla I(i,j)| = |\max\{b-a,0\}| = 0. \tag{3.497}$$

For pixels B and D, the direction of the gradient of the Laplacian is opposite the direction of the isophote ($\gamma < 0$) and so we must use formula (3.493), which takes the form:

$$|\nabla I(i,j)| = \Big[(\max\{a-d,0\})^2 + (\min\{b-a,0\})^2$$

$$+(\max\{a-c,0\})^2+(\min\{e-a,0\})^2\Big]^{\frac{1}{2}}. \tag{3.498}$$

Then for pixel B:

$$a=c=e=d>b \Rightarrow |\nabla I(i,j)| = |\min\{b-a,0\}| = |b-a|. \tag{3.499}$$

For pixel D:

$$a=c=e=b>d \Rightarrow |\nabla I(i,j)| = |\max\{a-d,0\}| = |a-d|. \tag{3.500}$$

In other words, pixels B and D will have their values reduced, while pixels A and F will have their values unchanged. If we were computing $|\nabla I(i,j)|$ from the central difference, for all four pixels it would have the same value, equal to the contrast between the horizontal dark stripe and the side regions. This would serve as a factor by which the values of pixels A and F would have been increased, while the values of B and D would have been decreased, something clearly undesirable.

Example 3.168

By looking at Figure 3.162, justify why the value of a pixel should increase if the Laplacian changes along the same direction as the isophote, and it should be decreased if the Laplacian changes along the opposite direction of the isophote.

To understand what we have to do, let us fix our attention on a pixel, the value of which we want to update, and its right neighbour as we move along the direction of the isophote vector. The question is: in which circumstances do we want to make this pixel more similar or less similar to its right neighbour?

We understand that an increasing Laplacian implies an increasing dissimilarity. Also, from the way the isophote vector is defined, we understand that when we move along the isophote, the neighbour on the right is always darker than the pixel under consideration. So, if the direction of increasing Laplacian coincides with the direction of the isophote, we want to increase the dissimilarity of the pixel with its right neighbour, so we want to make it even brighter by increasing its value. If the direction of increasing Laplacian is opposite the direction of the isophote, we want to increase the similarity of the pixel with its right neighbour, so we want to make it darker.

Example 3.169

Apply the in-painting algorithm to assign values to the empty pixels of image 3.163.

Figure 3.163 A 2-bit image that needs repair.

3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	2
3	3					2	2
3	3					2	2
3	3					2	3
3	3	3	2	2	2	3	3
3	3	2	2	2	3	3	3
3	2	2	2	3	3	3	3

(Continued)

Example 3.169 (Continued)

Figure 3.164 shows the detailed steps of the first iteration of the algorithm. This way we construct the image after the first iteration shown at the top left of Figure 3.165.

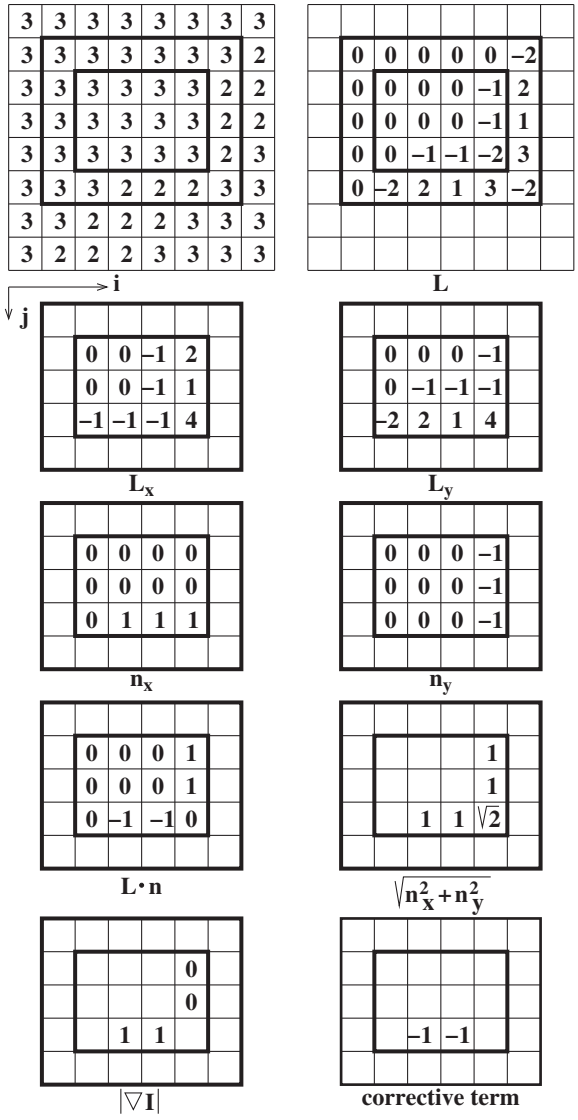


Figure 3.164 The first iteration of the in-painting algorithm.

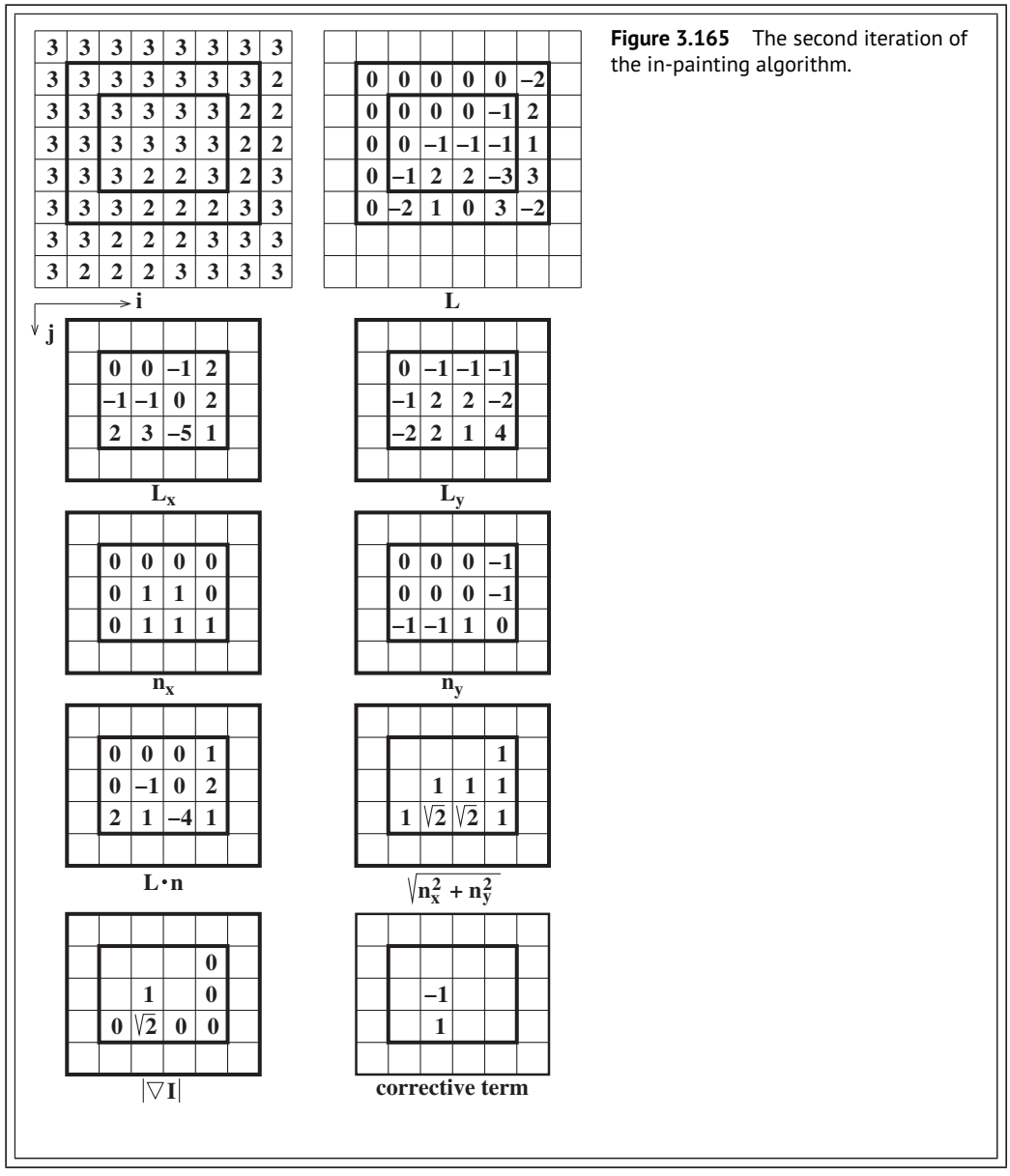


Figure 3.165 The second iteration of the in-painting algorithm.

What is normalised convolution?

It is a very simple interpolating algorithm with the following steps.

Step 0: Assign to pixels with damaged or missing values the value 0.
Step 1: Convolve the image with a low pass filter.
Step 2: Give to the pixels with known values the value 1, while the pixels with the unknown values value 0, thus creating the so-called **sampling mask** of the image.
Step 3: Convolve the sampling mask of the image with the low pass filter you used in step 1.

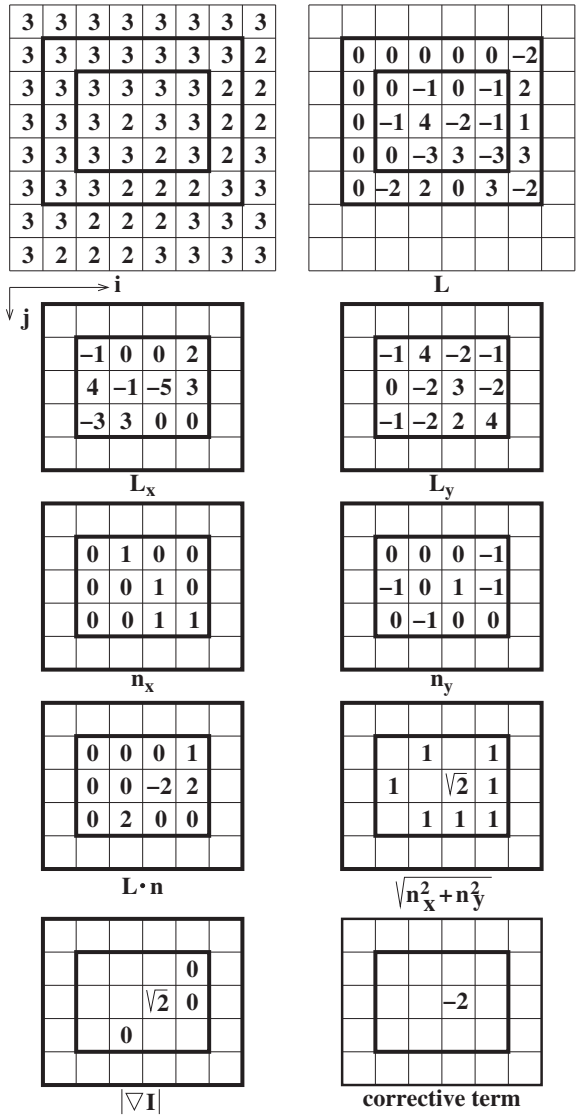


Figure 3.166 The third iteration of the in-painting algorithm.

Step 4: Divide the output of step 1 by the output of step 2, point by point, to obtain the interpolated image.

Example 3.170

Repair the image of Figure 3.163 using normalised convolution. For a low pass filter use one of the masks shown in Figure 3.167, the one which, according to your opinion, is the most appropriate.

1	2	1
2	4	2
1	2	1

0	1	1	1	0
1	2	4	2	1
1	4	8	4	1
1	2	4	2	1
0	1	1	1	0

Figure 3.167 Two low pass filters that may be used in normalised convolution.

Clearly, the size of the filter we use has to be larger than the size of the gaps we wish to fill, otherwise we shall have to use more than one pass, and it is not certain even then that the information will propagate correctly from the rim of the damaged region inwards. So, we select to use the 5×5 *filter.*

Since we decided to use a 5×5 *convolution filter, we must consider the minimum enclosing rectangle of the area we wish to repair and augment it by a band two-pixels wide all around. Then we must convolve the augmented sub-image with the filter to produce the first convolution output. Subsequently, we must use the same sub-image but replace all known pixel values with 1 to create the sampling mask. Next, we must convolve this sampling mask with the low pass filter to produce the second convolution output. Finally, we must divide the first output with the second one, pixel by pixel, and round the result to the nearest integer. All stages of this process are shown in Figure 3.165. Frame 3.165f should be inserted in place of the 0s in frame 3.165a to produce the repaired image.*

What is the error correction method?

It is an interpolation method that relies on the assumption of image stationarity. The algorithm consists of the following steps.

Step 0: Consider the minimum enclosing rectangle of the damaged region, with a strip of at least one pixel wide all around it, consisting of pixels with valid values.

Step 1: Create the sampling mask S such that it is the same size as the sub-image you constructed in step 1 and it has value 1 at the positions where valid pixel values exist and value 0 at the positions where pixel values have to be assigned.

Step 2: Interpolate the sub-image you identified in step 1 by using any simple interpolation method, e.g. nearest neighbour interpolation. You can do that by considering each point marked with 0 in mask S and computing its distance from all points in the mask marked with 1, keeping track of the smallest distance so far. At the end, assign to the selected pixel the value of the pixel at the site with the smallest distance. Call the interpolated sub-image you create this way I_0.

Step 3: Take the DFT of I_0. Call it $\hat{I}_0$.

Step 4: Set the amplitudes of all frequencies above a certain pair of cut-off frequency indices (m_0, n_0) equal to 0. Call the result $\tilde{I}_0$.

Step 5: Take the inverse DFT of $\tilde{I}_0$. Call it I_1.

Step 6: Compare I_1 with I_0 point by point, only at the places where the sampling mask has value 1. Thus, create array E_0, with elements $E_0(i,j) = S(i,j)[I_0(i,j) - I_1(i,j)]$.

3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	2
3	3	0	0	0	0	2	2
3	3	0	0	0	0	2	2
3	3	0	0	0	0	2	3
3	3	3	2	2	2	3	3
3	3	2	2	2	3	3	3
3	2	2	2	3	3	3	3

(a)

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

(b)

66	45	43	56
50	25	21	39
61	35	33	50

(c)

22	15	15	22
17	9	9	17
22	15	15	22

(d)

3	3	2.9	2.5
2.9	2.8	2.3	2.3
2.8	2.3	2.2	2.3

(e)

3	3	3	3
3	3	2	2
3	2	2	2

(f)

Figure 3.168 (a) The image that is to be repaired with 0 values assigned to the damaged pixels. The black frame identifies the sub-image that is to be convolved with the low pass filter. (b) The sampling mask of the image, with 1 indicating the places with valid pixel values. (c) The result of convolving the subimage indicated by the thick black frame in (a) with mask 3.167b. (d) The result of convolving the sub-image indicated by the thick black frame in (b) with mask 3.167b. (e) The result of dividing (c) with (d) point by point. (f) The result of rounding (e) to the nearest integer.

Step 7: Interpolate the error array E_0, using the same interpolation method you used in step 2, to assign values to the elements of E_0 marked with 0 in sampling mask S, and thus create array E_1.
Step 8: Use array E_1 to correct array I_1 and form array $I_2 = I_1 + E_1$.
Step 9: Take the DFT of I_2. Call it $\hat{I}_2$.
Step 10: Set the amplitudes of all frequencies above a certain pair of cut-off frequency indices (m_1, n_1) equal to 0. Call the result $\tilde{I}_2$.
Step 11: Take the inverse DFT of $\tilde{I}_2$. Call it I_3.
Step 12: Compare I_3 with I_2 point by point, only at the places where the sampling mask has value 1. Thus, create array E_2, with elements $E_2(i,j) = S(i,j)[I_0(i,j) - I_2(i,j)]$.
Step 13: Use array E_2 to correct array I_3 and form array $I_4 = I_2 + E_2$.

You can carry on this way until the error array you compute has all its elements below a certain threshold, in which case you may consider that the algorithm has converged. After the algorithm has converged, you should round the pixel values to the nearest integer.

The philosophy of this algorithm is the following. In step 2, where we perform the nearest neighbour interpolation, we effectively know that we mess up the high frequencies of the image, as we actually create our 0th order estimate from flat patches. So, when we take its DFT, we know that the high frequency content of this estimate is wrong, and we set it to 0 in step 4. This way, however, we damage also the pixels that had correct values to begin with. We work out how much the error

is in those locations (step 6), and we interpolate the error (step 7) in order to correct the estimates in the other locations (step 8). This loop is repeated until convergence.

The key parameters of this algorithm is the cut-off frequency pairs (m_0, n_0), (m_1, n_1), etc. This might have to be determined by trial and error, or by considering the spectrum of an undamaged window of the image from which the highest image frequency is inferred.

What is the non-parametric Markov random field method for texture repair?

It is a family of methods inspired by the **Efros and Leung algorithm**, and its variations. We give here the original algorithm, on which many subsequent algorithms are based.

Step 1: Select the size of the Markov neighbourhood you will use for a pixel. Let us call it $M \times M$.
Step 2: Consider a pixel without value, which, however, has a valid Markov neighbourhood, even if it is incomplete.
Step 3: Scan the part of the image with valid values, using an $M \times M$ window, and compute the similarity of the Markov neighbourhood of the pixel with every such window. As a measure of similarity use a weighted sum of square differences, with the weights computed from a Gaussian function (see Example 3.171).
Step 4: Assign to the pixel the grey value of the focal pixel of the most similar $M \times M$ image patch you found.
Step 5: When a pixel has been assigned a value, treat it as one of the original pixels, so that the pattern grows inwards from the borders of the region under repair.

In subsequent versions, the most important improvements of this algorithm have been the following:

i) Once the best partial neighbourhood of the pixel being repaired has been identified, the values of a whole patch (possibly the full missing neighbourhood) are copied and not just the value of the focal pixel.
ii) The order of treating the damaged pixels is selected so that damaged pixels that border valid pixels forming the endings of image lines or edges are treated first, for the continuity of those lines or edges to be preserved when valid values are propagated towards the centre of the damaged region.

Example 3.171

Figure 3.169a shows the incomplete Markov neighbourhood of a pixel that is to be restored. Figure 3.169b shows the mask that identifies the pixels with and without grey values. Figure 3.169c shows an image patch of the same size, where there are no missing values. Use a Gaussian window to work out the distance between neighbourhoods (a) and (c).

(Continued)

Example 3.171 (Continued)

5	3			
4	4	3	2	
5	5	4	1	3

(a)

0	0	0	0	0
0	0	0	0	0
1	1	0	0	0
1	1	1	1	0
1	1	1	1	1

(b)

7	6	6	5	5
7	7	6	5	5
6	6	4	4	5
7	5	4	4	3
6	6	5	5	4

(c)

Figure 3.169 The incomplete neighbourhood in (a) has to be compared with the image patch in (c). In (b), the mask identifying the overlapping part of the two neighbourhoods.

$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$
$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
2	1	0	1	2
$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$

(a)

.157	.315	.397	.315	.157
.315	.630	.794	.630	.315
.397	.794	1	.794	.397
.315	.630	.794	.630	.315
.157	.315	.397	.315	.157

(b)

16	31	40	31	16
31	63	79	63	31
40	79	100	79	40
31	63	79	63	31
16	31	40	31	16

(c)

Figure 3.170 (a) The distances from the centre of all positions of the neighbourhood. (b) The value of the Gaussian function at each position. (c) The integer valued weights constructed from the values in (b).

First we must create the Gaussian weights we shall use to work out the weighted difference between the two neighbourhoods. We are dealing with a 5×5 *neighbourhood, so the Gaussian mask should also be of the same size. It will be created by sampling function* $e^{-\frac{x^2}{2\sigma^2}}$. *The standard deviation* σ *of this Gaussian function determines the relative importance we give to the matching of the various neighbours. Let us say that we would like neighbours that are at distance 1 from the focal pixel to weigh twice as much as neighbours that are at distance 2. So, we want:*

$$\frac{e^{-\frac{1^2}{2\sigma^2}}}{e^{-\frac{2^2}{2\sigma^2}}} = 2 \Rightarrow e^{\frac{3}{2\sigma^2}} = 2 \Rightarrow \frac{3}{2\sigma^2} = \ln 2 \Rightarrow \sigma^2 = \frac{3}{2\ln 2} = 2.164 \tag{3.501}$$

Figure 3.170a shows the value of x *to be used in the exponent of* $e^{-\frac{x^2}{2\sigma^2}}$ *for each position of the* 5×5 *grid. The corresponding value of the Gaussian mask is shown in 3.170b. To avoid having to deal with real numbers, we multiply the weights with 100 and round them to the nearest integer, as shown in 3.170c.*

Let us denote by (i_0, j_0) *the index of the pixel under repair. Let us also denote by* (k, l) *the centre of the patch with which we compare its neighbourhood. Let us call* $g(i, j)$ *the weight of the Gaussian mask at the* (i, j) *position, assuming that these indices take values in the range* $[-2, 2]$. *Finally, let us call* $m(i, j)$ *the value of the mask, as given in 3.169b, which tells us whether position* (i, j) *has a value assigned (*$m(i, j) = 1$*), or not (*$m(i, j) = 0$*). We may then compute the distance between the two neighbourhoods as follows:*

$$D((i_0, j_0), (k, l)) \equiv \frac{\sum_{i=-2}^{2} \sum_{j=-2}^{2} m(i_0 + i, j_0 + j) g(i, j) \left[I(i_0 + i, j_0 + j) - I(k + i, l + j) \right]^2}{\sum_{i=-2}^{2} \sum_{j=-2}^{2} m(i_0 + i, j_0 + j) g(i, j)}. \tag{3.502}$$

Here $I(i,j)$ is the grey value of the image in position (i,j). Note that the use of the mask, which indicates which pixels have values and which not, avoids the contribution of pixels with damaged values. The use of the mask and of partial neighbourhoods allows us also to deal with pixels near the border of the image. For example, we may augment the image with a strip of 2 pixels all around, so even border pixels have geometrically complete neighbourhoods, and give to the pixels we add any value we like, as long as we also augment the corresponding mask that marks the pixels with valid values, and give to the added pixels value 0 in the mask.

In this example, the distance of the two neighbourhoods is:

$$\begin{aligned}&\left(1^2\times 40+3^2\times 79+3^2\times 31+1^1\times 63+1^2\times 79+2^2\times 63\right.\\&\left.+1^2\times 16+1^2\times 31+1^2\times 40+4^2\times 31+1^2\times 16\right)/\\&(40+79+31+63+79+63+16+31+40+31+16)=\frac{2023}{489}=4.13.\end{aligned}\tag{3.503}$$

What is the "take home" message of this chapter?

This chapter has described some methods used for characterising textures. It was concerned with texture samples that were isolated, i.e. it assumed that the whole image (or texture sample) was filled with the same texture pattern.

Some of the methods discussed are also appropriate for creating texture patters. These are, for example, the Markov random field method and the Gibbs distribution method. However, it was shown that the parameters that control these methods are not robust enough to allow one-to-one mapping between patterns and parameter sets. There are many reasons for that.

1. The parameters computed rely on the model assumed, e.g. the size of the Markovian neighbourhood. This is not always easy to determine a priori.
2. The neighbourhood used must not be self-contradictory. We saw examples where it was not possible to get repeatable patterns because the neighbourhoods we used were not expressible in terms of clique potentials, i.e. the "Markov random fields" they supposingly defined did not correspond to any Gibbs distribution. A neighbourhood is self-consistent when pixels which are neighbours of each other in a particular relative position always interact in the same way. This was not the case in some of the examples we discussed in the section on Markov random fields.
3. The methods used to estimate the parameters rely on the number of samples available for each combination of neighbourhoods. Given that the samples of textures we use are limited in size, we are forced to re-quantise the grey values in order to increase the number of distinct neighbourhood structures and thus compute reliable statistics from them.

In short, no model fits the observed textures perfectly, and so no model parameters are perfect in capturing all characteristics of a texture pattern. However, one has to remember that the estimated parameters do not have to be adequate for reproducing the pattern in order for these parameters to allow the discrimination of different textures. All one needs is to compute some numbers that represent a texture without necessarily capturing all aspects of the pattern for its reproduction. Fractal features are far from unique descriptors of textures, but in conjunction with other descriptors they may help solve problems of texture discrimination.

On the contrary, non-parametric methods, such as co-occurrence matrices, tend to be more successful. In particular, several studies have shown that the use of the elements of the co-occurrence

matrix itself and in particular ratios of element values are excellent texture discriminators. Features defined from the co-occurrence matrix, such as contrast, homogeneity, etc. are popular because of the perceptual meaning they have. However, they are not adequate for texture and object discrimination as they throw away most of the information conveyed by the co-occurrence matrices. Often people identify the use of co-occurrence matrices with the use of such features and thus draw the wrong conclusions about the usefulness of co-occurrence matrices to texture discrimination.

Finally, we discussed methods of repairing damaged textures. We saw that by far the most powerful such methods are those based on the use of non-parametric random fields, while statistical methods, like Kriging, are suitable for random textures.

4

Non-stationary Grey Texture Images

What is a non-stationary texture image?

A non-stationary texture image is a texture image that contains more than one type of texture in it. The problem we are interested in then is that of **texture segmentation**, i.e. the division of the image into regions each of which is occupied by a single texture type.

What is this chapter about?

This chapter is about the development of methodology that allows the extraction of **localised** texture information that may be used to characterise an image locally and so allow the identification of different texture patches in it.

Why can't we use the methods developed in the previous chapter here?

The methods developed in the previous chapter are **global** methods assuming that the whole image is filled up by a single texture. Some of those methods may be applied locally. In such a case, we simply choose local image windows and treat the parts of the image inside the windows as separate images, with supposedly uniform texture, to which the methodology developed for stationary textures is applicable.

How can we be sure that the texture inside an image window is stationary?

We cannot be sure. To minimise the risk of having windows that contain two or more different textures, we have to choose windows as small as possible. However, texture is a spatial property, so we need to have windows large enough to capture the texture characteristics by the performed analysis. This contradiction in requirements is known as the **uncertainty principle in image processing** and it is related to the well known **uncertainty principle in signal processing**.

4.1 The Uncertainty Principle and its Implications in Signal and Image Processing

What is the uncertainty principle in signal processing?

In plain language, the uncertainty principle in signal processing tells us that we cannot simultaneously know "what happens when". Something "happens" when something changes: for example, we hear something if something vibrates. "What happens", therefore, is associated with

Image Processing: Dealing with Texture, Second Edition. Maria Petrou and Sei-ichiro Kamata.

Companion Website: www.wiley.com/go/kamataText2

frequencies that indicate motion, change, variation, vibration, etc. Of course, in order to have a complete picture, we need to know the exact time this event is happening. However, if we know the exact time, we cannot measure frequency, i.e. change, i.e. "what happens". If we allow for a time duration, we **can** measure frequency, i.e. we **can** identify "what happened", but we do not know exactly **when** it happened since all we know is that it happened during the particular time interval we considered. Formally this is expressed as

$$(\Delta t)^2(\Delta\omega)^2 \geq \frac{1}{4} \tag{4.1}$$

where Δt is the duration of the signal we consider and $\Delta\omega$ is its bandwidth in the frequency domain. The uncertainty principle then says: the product of the spectral bandwidth multiplied with the time duration of a signal cannot be less than a certain minimum value.

Note that here the definition of bandwidth we use is not the classical definition used in signal processing, where the bandwidth of the signal is defined as $\Delta\nu$ where $\nu = \omega/(2\pi)$. This is no problem, as long as we are consistent in our notation and definitions. Obviously, in terms of ν, the uncertainty principle is stated as

$$(\Delta t)^2(\Delta\nu)^2 \geq \frac{1}{16\pi^2}. \tag{4.2}$$

Example B4.1

Show that if $F(\omega)$ is the Fourier transform of a function $f(t)$, the Fourier transform of the derivative of the function, $f'(t)$, is $\mathrm{j}\omega F(\omega)$.

By definition:

$$F(\omega) \equiv \int_{-\infty}^{\infty} f(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t. \tag{4.3}$$

The inverse Fourier transform allows us to write:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega. \tag{4.4}$$

To differentiate an integral with respect to a parameter, we use Leibniz's rule. Assume that we have the integral

$$I(\lambda) \equiv \int_{\alpha(\lambda)}^{\beta(\lambda)} f(x;\lambda)\mathrm{d}x \tag{4.5}$$

and we need its derivative with respect to λ. Leibniz's rule says that this is given by:

$$\frac{\mathrm{d}I(\lambda)}{\mathrm{d}\lambda} = \frac{\mathrm{d}\beta(\lambda)}{\mathrm{d}\lambda}f(\beta(\lambda);\lambda) - \frac{\mathrm{d}\alpha(\lambda)}{\mathrm{d}\lambda}f(\alpha(\lambda);\lambda) + \int_{\alpha(\lambda)}^{\beta(\lambda)}\frac{\partial f(x;\lambda)}{\partial\lambda}\mathrm{d}x. \tag{4.6}$$

If we differentiate both sides of (4.4) with respect to t, by applying Leibniz's rule, we obtain

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\mathrm{j}\omega\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega \tag{4.7}$$

or after some rearrangement:

$$f'(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}[F(\omega)\mathrm{j}\omega]\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega. \tag{4.8}$$

For this equation to be right, the quantity inside the square brackets must be the Fourier transform of $f'(t)$. This proves the theorem.

Example B4.2

Show that if $F(\omega)$ is the Fourier transform of $f(t)$, then the Fourier transform of the complex conjugate of $f(t)$, $f^*(t)$, is $F^*(-\omega)$.

Take the complex conjugate of both sides of Equation (4.4):

$$f^*(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F^*(\omega)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}\omega. \tag{4.9}$$

We may define a new variable of integration $\tilde{\omega}$ as $\tilde{\omega} \equiv -\omega$, so we may write

$$f^*(t) = \frac{1}{2\pi}\int_{\infty}^{-\infty} F^*(-\tilde{\omega})\mathrm{e}^{\mathrm{j}\tilde{\omega}t}(-\mathrm{d}\tilde{\omega}) \tag{4.10}$$

which upon changing the order of the limits of integration and dropping the tilde from the dummy variable $\tilde{\omega}$, becomes:

$$f^*(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F^*(-\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega. \tag{4.11}$$

This equation is only valid if $F^*(-\omega)$ is the Fourier transform of $f^*(t)$. This proves the theorem.

Example B4.3

If $F(\omega)$ is the Fourier transform of $f(t)$, show that the Fourier transform of $f(t)f^*(t)$ is $\frac{1}{2\pi}\int_{-\infty}^{\infty} F(u)F^*(u-\omega)\mathrm{d}u$.

From the result of Example 4.2 we have:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega \qquad f^*(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F^*(-\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega. \tag{4.12}$$

To prove the conjecture we must show that the inverse Fourier transform of

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} F(u)F^*(u-\omega)\mathrm{d}u \tag{4.13}$$

is $f(t)f^*(t)$.

Let us consider the inverse Fourier transform of (4.13):

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} F(u)F^*(u-\omega)du\right]\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega. \tag{4.14}$$

We define a new variable of integration $v \equiv \omega - u$. Then $\omega = u + v$, $\mathrm{d}\omega = \mathrm{d}v$ and the above expression takes the form:

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} F(u)F^*(-v)\mathrm{d}u\right]\mathrm{e}^{\mathrm{j}(u+v)t}\mathrm{d}v.$$

Since $\mathrm{e}^{\mathrm{j}(u+v)t} = \mathrm{e}^{\mathrm{j}ut}\mathrm{e}^{\mathrm{j}vt}$, we may easily separate the double integral into a product of two single integrals:

$$\left\{\frac{1}{2\pi}\int_{-\infty}^{\infty} F(u)\mathrm{e}^{\mathrm{j}ut}\mathrm{d}u\right\}\left\{\frac{1}{2\pi}\int_{-\infty}^{\infty} F^*(-v)\mathrm{e}^{\mathrm{j}vt}\mathrm{d}v\right\}. \tag{4.15}$$

By comparison with Equations (4.12) we recognise that the above expression is the product $f(t)f^*(t)$, which proves the conjecture.

Example B4.4

If$F(\omega)$ is the Fourier transform of a function $f(t)$, show that

$$\int_{-\infty}^{\infty} |f(t)|^2 \mathrm{d}t = \frac{1}{2\pi} \int_{-\infty}^{\infty} |F(\omega)|^2 \mathrm{d}\omega. \tag{4.16}$$

(Parseval's theorem).

Example 4.3 showed that if $F(\omega)$ is the Fourier transform of $f(t)$, the Fourier transform of $f(t)f^*(t)$ is $\frac{1}{2\pi}\int_{-\infty}^{\infty} F(u)F^*(u-\omega)\mathrm{d}u$. Let us write this explicitly:

$$\int_{-\infty}^{\infty} f(t)f^*(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(u)F^*(u-\omega)\mathrm{d}u. \tag{4.17}$$

This equation is valid for any value of ω. If we set $\omega = 0$ we obtain the result we wish to prove.

Example B4.5

Prove that

$$\int_{-\infty}^{\infty} f(t)^2 \mathrm{d}t \int_{-\infty}^{\infty} g(t)^2 \mathrm{d}t \geq \left| \int_{-\infty}^{\infty} f(t)g(t)\mathrm{d}t \right|^2 \tag{4.18}$$

where $f(t)$ and $g(t)$ are some real functions (Schwartz's inequality).

Let us consider the expression $m(x)$ defined as

$$m(x) \equiv \int_{-\infty}^{\infty} [f(t) + xg(t)]^2 \mathrm{d}t \tag{4.19}$$

where x is a real variable. If we expand the square in the above definition we may write:

$$m(x) = \int_{-\infty}^{\infty} f(t)^2 \mathrm{d}t + 2x \int_{-\infty}^{\infty} f(t)g(t)\mathrm{d}t + x^2 \int_{-\infty}^{\infty} g(t)^2 \mathrm{d}t. \tag{4.20}$$

Let:

$$a \equiv \int_{-\infty}^{\infty} g(t)^2 \mathrm{d}t \qquad b \equiv 2\int_{-\infty}^{\infty} f(t)g(t)\mathrm{d}t \qquad c \equiv \int_{-\infty}^{\infty} f(t)^2 \mathrm{d}t. \tag{4.21}$$

Since $m(x)$ is the integral of a square function, $m(x)$ must always be real and positive for all real values of x:

$$m(x) = ax^2 + bx + c > 0. \tag{4.22}$$

This is only true if the discriminant of the quadratic expression is non-positive:

$$b^2 - 4ac \leq 0 \Rightarrow ac \geq \frac{1}{4}b^2. \tag{4.23}$$

Substituting the expressions for a, b and c from (4.21) in the last expression proves Schwartz's inequality.

Box 4.1 Prove the uncertainty principle in signal processing

Let $f(t)$ be a function and $F(\omega)$ its Fourier transform, defined by Equation (4.3).
The **energy content** of the signal is defined as:

$$E_\infty \equiv \int_{-\infty}^{\infty} f(t)^2 \mathrm{d}t. \tag{4.24}$$

From Parseval's theorem (see Example 4.4), this may be written as:

$$E_\infty = \frac{1}{2\pi} \int_{-\infty}^{\infty} |F(\omega)|^2 \mathrm{d}\omega. \tag{4.25}$$

The **time dispersion** of the signal is given by

$$(\Delta t)^2 \equiv \frac{1}{E_\infty} \int_{-\infty}^{\infty} (t - \bar{t})^2 f(t)^2 \mathrm{d}t \tag{4.26}$$

where $\bar{t}$ is the **centre of gravity** of the signal defined as:

$$\bar{t} \equiv \frac{1}{E_\infty} \int_{-\infty}^{\infty} t f(t)^2 \mathrm{d}t. \tag{4.27}$$

We may shift the origin of t so that $\bar{t} = 0$, in which case:

$$(\Delta t)^2 = \frac{1}{E_\infty} \int_{-\infty}^{\infty} t^2 f(t)^2 \mathrm{d}t. \tag{4.28}$$

In an analogous way, the **spectral bandwidth** of the signal is given by

$$(\Delta \omega)^2 \equiv \frac{1}{2\pi E_\infty} \int_{-\infty}^{\infty} (\omega - \overline{\omega})^2 |F(\omega)|^2 \mathrm{d}\omega \tag{4.29}$$

where $\overline{\omega}$ is the **spectral centre of gravity** of the signal defined as:

$$\overline{\omega} \equiv \frac{1}{E_\infty} \int_{-\infty}^{\infty} \omega |F(\omega)|^2 \mathrm{d}\omega. \tag{4.30}$$

For $\overline{\omega} = 0$:

$$(\Delta \omega)^2 \equiv \frac{1}{2\pi E_\infty} \int_{-\infty}^{\infty} \omega^2 |F(\omega)|^2 \mathrm{d}\omega \tag{4.31}$$

If $f'(t)$ is the derivative of the function, its Fourier transform is $\mathrm{j}\omega F(\omega)$ (see Example 4.1). By applying Parseval's theorem to the Fourier pair $f'(t) \leftrightarrow \mathrm{j}\omega F(\omega)$ we obtain:

$$\int_{-\infty}^{\infty} \omega^2 |F(\omega)|^2 \mathrm{d}\omega = 2\pi \int_{-\infty}^{\infty} f'(t)^2 \mathrm{d}t. \tag{4.32}$$

By substituting in Equation (4.31), we have:

$$(\Delta \omega)^2 = \frac{1}{E_\infty} \int_{-\infty}^{\infty} f'(t)^2 \mathrm{d}t \tag{4.33}$$

(Continued)

Box 4.1 (Continued)

We use Equations (4.28) and (4.33) to calculate:

$$(\Delta t)^2(\Delta\omega)^2 = \frac{1}{E_\infty^2}\int_{-\infty}^{\infty} t^2 f(t)^2 \mathrm{d}t \int_{-\infty}^{\infty} f'(t)^2 \mathrm{d}t. \tag{4.34}$$

If we apply Schwartz's inequality (see Example 4.5), for the integrals on the right-hand side of (4.34), we obtain:

$$\int_{-\infty}^{\infty} t^2 f(t)^2 \mathrm{d}t \int_{-\infty}^{\infty} f'(t)^2 \mathrm{d}t \geq \left| \int_{-\infty}^{\infty} t f(t) f'(t) \mathrm{d}t \right|^2. \tag{4.35}$$

We may integrate by parts the integral on the right-hand side of (4.35):

$$\int_{-\infty}^{\infty} t f(t) f'(t) \mathrm{d}t = \frac{1}{2} t f(t)^2 \Big|_{-\infty}^{\infty} - \frac{1}{2}\int_{-\infty}^{\infty} f(t)^2 \mathrm{d}t. \tag{4.36}$$

If $\lim_{t\to\infty} t f(t)^2 = 0$, the first term on the right-hand side of (4.36) vanishes and from Equation (4.24) we have

$$\int_{-\infty}^{\infty} t f(t) f'(t) \mathrm{d}t = -\frac{1}{2} E_\infty. \tag{4.37}$$

If we use this in (4.35) and then in (4.34), we obtain:

$$(\Delta t)^2(\Delta\omega)^2 \geq \frac{1}{4}. \tag{4.38}$$

This is the mathematical statement of the uncertainty principle in signal processing.

Example B4.6

Compute the time dispersion of the Gaussian signal

$$g(t) = \mathrm{e}^{-\frac{t^2}{2\sigma^2}}. \tag{4.39}$$

This signal is symmetric, so its centre of gravity, $\bar{t}$, defined by Equation (4.27) is zero. Therefore, we may compute its time dispersion by using Equation (4.28). However, first we must compute the total energy of the signal, using Equation (4.24)

$$E_\infty = \int_{-\infty}^{\infty} g(t)^2 \mathrm{d}t = \int_{-\infty}^{\infty} \mathrm{e}^{-\left(\frac{t}{\sigma}\right)^2} dt = \sigma \int_{-\infty}^{\infty} \mathrm{e}^{-x^2} \mathrm{d}x = \sqrt{\pi}\sigma \tag{4.40}$$

where we changed the variable of integration from t to $x \equiv t/\sigma$ and made use of the result of Example 3.36.

Next we compute the time dispersion of the signal using Equation (4.28)

$$(\Delta t)^2 = \frac{1}{\sqrt{\pi}\sigma}\int_{-\infty}^{\infty} t^2 \mathrm{e}^{-\left(\frac{t}{\sigma}\right)^2} \mathrm{d}t = \frac{1}{\sqrt{\pi}\sigma}\sigma^3 \int_{-\infty}^{\infty} x^2 \mathrm{e}^{-x^2} \mathrm{d}x = \frac{\sigma^2}{\sqrt{\pi}}\frac{\sqrt{\pi}}{2} = \frac{\sigma^2}{2} \tag{4.41}$$

where we made the substitution $x \equiv t/\sigma$ and used the result of Example 3.37 with $a = 1$.

Example B4.7

Compute the spectral bandwidth of the Gaussian signal given by Equation (4.39).
We shall use formula (4.29) with $\overline{\omega} = 0$ since the signal is real, which implies a symmetric Fourier spectrum. We shall also make use of the results stated by Equations (4.40) and (3.311):

$$(\Delta\omega)^2 = \frac{1}{2\pi\sqrt{\pi}\sigma}\int_{-\infty}^{\infty}\omega^2 2\pi\sigma^2 e^{-\sigma^2\omega^2}d\omega = \frac{\sigma}{\sqrt{\pi}}\frac{1}{\sigma^3}\int_{-\infty}^{\infty}x^2 e^{-x^2}dx = \frac{1}{2\sigma^2}. \quad (4.42)$$

Here we made the substitution $x \equiv \omega\sigma$ and used the result of Example 3.37.

Example B4.8

Use the results of Examples 4.6 and 4.7 to show that the Gaussian signal has the minimal joint uncertainty in terms of being localised in time and having its frequency content estimated.
By making use of Equations (4.41) and (4.42), we obtain:

$$(\Delta t)^2(\Delta\omega)^2 = \frac{1}{4}. \quad (4.43)$$

Therefore, the Gaussian achieves the minimum in the uncertainty inequality (4.38) of signal processing.

Does the window we choose in order to extract local information influence the result?

Yes. We have already seen that the **size** of the window affects the accuracy of what we compute. The **shape** as well as the **shift** of the window do as well. For example, let us consider a signal $g(t)$ with Fourier transform $G(\omega)$. Let us assume that we observe only part of the signal through a window $w(t)$, with Fourier transform $W(\omega)$, centred at t_0:

$$h(t) = g(t)w(t - t_0). \quad (4.44)$$

Due to the shifting property of the Fourier transform, the Fourier transform of the window is $e^{-j\omega t_0}W(\omega)$. Since the window multiplies the signal, the Fourier transform of the window is **convolved** with the Fourier transform of the signal. Therefore, the Fourier transform of what we observe is given by:

$$H(\omega) = \int_{-\infty}^{\infty}G(\omega - u)e^{-jut_0}W(u)du. \quad (4.45)$$

In general, $H(\omega)$ is expected to be different from $G(\omega)$ (which we actually wish to observe), and depends on the locality of the window t_0.

Example B4.9

Compute the Fourier transform of the delta function.
We insert the delta function into the definition formula of the Fourier transform

$$\Delta(\omega) = \int_{-\infty}^{\infty}\delta(t)e^{-j\omega t}dt = e^{-j\omega t}|_{t=0} = 1 \quad (4.46)$$

(Continued)

Example B4.9 (Continued)

where we made use of the following property of the delta function: when the delta function is integrated with another function from $-\infty$ to ∞, it picks up the value of the other function at the point where the argument of the delta function is zero. In this particular case, the argument of the delta function is t and it is zero at $t = 0$.

Example B4.10

Prove that:

$$\delta(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{j}xy}\mathrm{d}y. \tag{4.47}$$

If we take the inverse Fourier transform of (4.46), we have $\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} 1\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega$, which proves the conjecture if we replace t by x and ω by y.

Example 4.11

Consider a signal $g(t) = A\sin\omega_0 t$, where A is a positive constant, and a window $w(t)$ defined as:

$$w(t) = \begin{cases} 1 & \text{if } t_1 \le t \le t_2 \\ 0 & \text{elsewhere} \end{cases}. \tag{4.48}$$

Compute the estimate of the Fourier transform of the signal obtained when seen through this window and compare it with the true Fourier transform of the signal.
We start by computing first the true Fourier transform of the signal. The signal may be written as:

$$g(t) = A\sin\omega_0 t = \frac{A}{2\mathrm{j}}(\mathrm{e}^{\mathrm{j}\omega_0 t} - \mathrm{e}^{-\mathrm{j}\omega_0 t}). \tag{4.49}$$

Then its Fourier transform is given by

$$\begin{aligned} G(\omega) &= \int_{-\infty}^{\infty} g(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \\ &= \frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}(\mathrm{e}^{\mathrm{j}\omega_0 t} - \mathrm{e}^{-\mathrm{j}\omega_0 t})\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \\ &= \frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}\mathrm{e}^{\mathrm{j}(\omega_0-\omega)t}\mathrm{d}t - \frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}\mathrm{e}^{-\mathrm{j}(\omega_0+\omega)t}\mathrm{d}t \\ &= \frac{A}{2\mathrm{j}}2\pi\delta(\omega_0-\omega) - \frac{A}{2\mathrm{j}}2\pi\delta(\omega_0+\omega) \\ &= -\mathrm{j}A\pi\delta(\omega_0-\omega) + \mathrm{j}A\pi\delta(\omega_0+\omega) \end{aligned} \tag{4.50}$$

where we made use of the result of Example 4.10, the property of delta function $\delta(-t) = \delta(t)$, and the fact that $1/\mathrm{j} = -\mathrm{j}$.

Next we compute the Fourier transform of the windowed signal

$$H(\omega) = \int_{t_1}^{t_2} A\sin\omega_0 t\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t$$

$$
\begin{aligned}
&= A\int_{t_1}^{t_2} \sin\omega_0 t\cos\omega t\mathrm{d}t - \mathrm{j}A\int_{t_1}^{t_2} \sin\omega_0 t\sin\omega t\mathrm{d}t \\
&= \frac{A}{2}\int_{t_1}^{t_2} \sin(\omega_0+\omega)t\mathrm{d}t + \frac{A}{2}\int_{t_1}^{t_2} \sin(\omega_0-\omega)t\mathrm{d}t \\
&\quad -\mathrm{j}\frac{A}{2}\int_{t_1}^{t_2} \cos(\omega_0-\omega)t\mathrm{d}t + \mathrm{j}\frac{A}{2}\int_{t_1}^{t_2} \cos(\omega_0+\omega)t\mathrm{d}t
\end{aligned} \tag{4.51}
$$

where we have used the trigonometric identities:

$$
\begin{aligned}
\sin A\cos B &= \frac{1}{2}\sin(A+B) + \frac{1}{2}\sin(A-B) \\
\sin A\sin B &= \frac{1}{2}\cos(A-B) - \frac{1}{2}\cos(A+B).
\end{aligned} \tag{4.52}
$$

Upon integration we obtain:

$$
\begin{aligned}
H(\omega) &= \left[-\frac{A}{2}\frac{\cos(\omega_0+\omega)t}{\omega_0+\omega} - \frac{A}{2}\frac{\cos(\omega_0-\omega)t}{\omega_0-\omega}\right|_{t_1}^{t_2} \\
&\quad + \left[-\mathrm{j}\frac{A}{2}\frac{\sin(\omega_0-\omega)t}{\omega_0-\omega} + \mathrm{j}\frac{A}{2}\frac{\sin(\omega_0+\omega)t}{\omega_0+\omega}\right|_{t_1}^{t_2} \\
&= -\frac{A}{2}\left[\frac{\cos(\omega_0+\omega)t_2}{\omega_0+\omega} - \frac{\cos(\omega_0+\omega)t_1}{\omega_0+\omega} + \frac{\cos(\omega_0-\omega)t_2}{\omega_0-\omega} - \frac{\cos(\omega_0-\omega)t_1}{\omega_0-\omega} + \right. \\
&\quad \left. \mathrm{j}\frac{\sin(\omega_0-\omega)t_2}{\omega_0-\omega} - \mathrm{j}\frac{\sin(\omega_0-\omega)t_1}{\omega_0-\omega} - \mathrm{j}\frac{\sin(\omega_0+\omega)t_2}{\omega_0+\omega} + \mathrm{j}\frac{\sin(\omega_0+\omega)t_1}{\omega_0+\omega}\right].
\end{aligned} \tag{4.53}
$$

If we use the trigonometric identities

$$
\begin{aligned}
\cos A - \cos B &= 2\sin\frac{A+B}{2}\sin\frac{B-A}{2} \\
\sin A - \sin B &= 2\cos\frac{A+B}{2}\sin\frac{A-B}{2}
\end{aligned} \tag{4.54}
$$

we may re-write this result as

$$
\begin{aligned}
H(\omega) &= -\frac{A}{2}\left[\frac{2}{\omega_0+\omega}\sin\frac{(\omega_0+\omega)t_2+(\omega_0+\omega)t_1}{2}\sin\frac{(\omega_0+\omega)t_1-(\omega_0+\omega)t_2}{2}\right. \\
&\quad + \frac{2}{\omega_0-\omega}\sin\frac{(\omega_0-\omega)t_2+(\omega_0-\omega)t_1}{2}\sin\frac{(\omega_0-\omega)t_1-(\omega_0-\omega)t_2}{2} \\
&\quad + \mathrm{j}\frac{2}{\omega_0-\omega}\cos\frac{(\omega_0-\omega)t_2+(\omega_0-\omega)t_1}{2}\sin\frac{(\omega_0-\omega)t_2-(\omega_0-\omega)t_1}{2} \\
&\quad \left. - \mathrm{j}\frac{2}{\omega_0+\omega}\cos\frac{(\omega_0+\omega)t_2+(\omega_0+\omega)t_1}{2}\sin\frac{(\omega_0+\omega)t_2-(\omega_0+\omega)t_1}{2}\right] \\
&= \frac{A}{\omega_0+\omega}\sin(\omega_0+\omega)\frac{t_1+t_2}{2}\sin(\omega_0+\omega)\frac{t_2-t_1}{2} \\
&\quad + \frac{A}{\omega_0-\omega}\sin(\omega_0-\omega)\frac{t_1+t_2}{2}\sin(\omega_0-\omega)\frac{t_2-t_1}{2} \\
&\quad - \mathrm{j}\frac{A}{\omega_0-\omega}\cos(\omega_0-\omega)\frac{t_2+t_1}{2}\sin(\omega_0-\omega)\frac{t_2-t_1}{2} \\
&\quad + \mathrm{j}\frac{A}{\omega_0+\omega}\cos(\omega_0+\omega)\frac{t_2+t_1}{2}\sin(\omega_0+\omega)\frac{t_2-t_1}{2}.
\end{aligned} \tag{4.55}
$$

(Continued)

Example 4.11 (Continued)

This expression makes explicit the dependence of the result on the length of the window, $t_2 - t_1 \equiv 2\tau$, and on the locality of the window $(t_2 + t_1)/2 \equiv t_0$:

$$\begin{aligned} H(\omega) = & \frac{A}{\omega_0 + \omega} \sin(\omega_0 + \omega)t_0 \sin(\omega_0 + \omega)\tau \\ & + \frac{A}{\omega_0 - \omega} \sin(\omega_0 - \omega)t_0 \sin(\omega_0 - \omega)\tau \\ & - \mathrm{j}\frac{A}{\omega_0 - \omega} \cos(\omega_0 - \omega)t_0 \sin(\omega_0 - \omega)\tau \\ & + \mathrm{j}\frac{A}{\omega_0 + \omega} \cos(\omega_0 + \omega)t_0 \sin(\omega_0 + \omega)\tau. \end{aligned} \tag{4.56}$$

Figures 4.1–4.3 demonstrate this result for a signal with $\omega_0 = 5$ and $A = 2$. Figure 4.1 shows the continuous signal and its Fourier transform in the form of real and imaginary parts, and magnitude and phase. Figure 4.2 shows various windowed parts of the signal and their corresponding Fourier transforms in the form of real and imaginary parts. Figure 4.3 is the same as Figure 4.2, but it shows the magnitude and phase of each Fourier transform. These Fourier transforms should be compared with their counterparts in Figure 4.1 in order to appreciate the effect of both the size of the window and the locality of the window. In all cases the main peaks, which correspond to delta function impulses at $\omega = \pm 5$ in the continuous case, are preserved, but they are blunt, with secondary peaks appearing even when a reasonably long part of the signal is windowed. In a real situation, when the signal is more complicated than a simple sinusoid, and when we do not know its true nature, we may have difficulty in separating the true peaks from the false secondary ones.

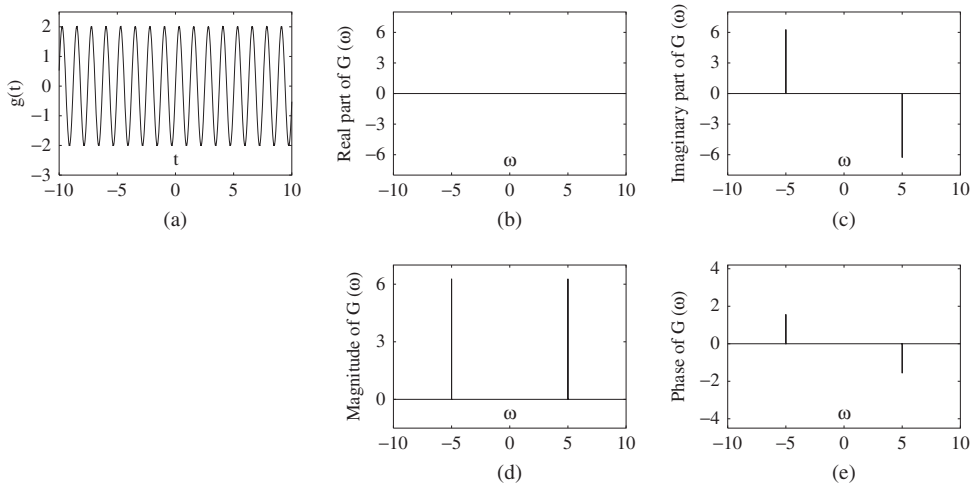


Figure 4.1 A continuous function (a), and the real part (b), imaginary part (c), magnitude (d) and phase (e) of its Fourier transform.

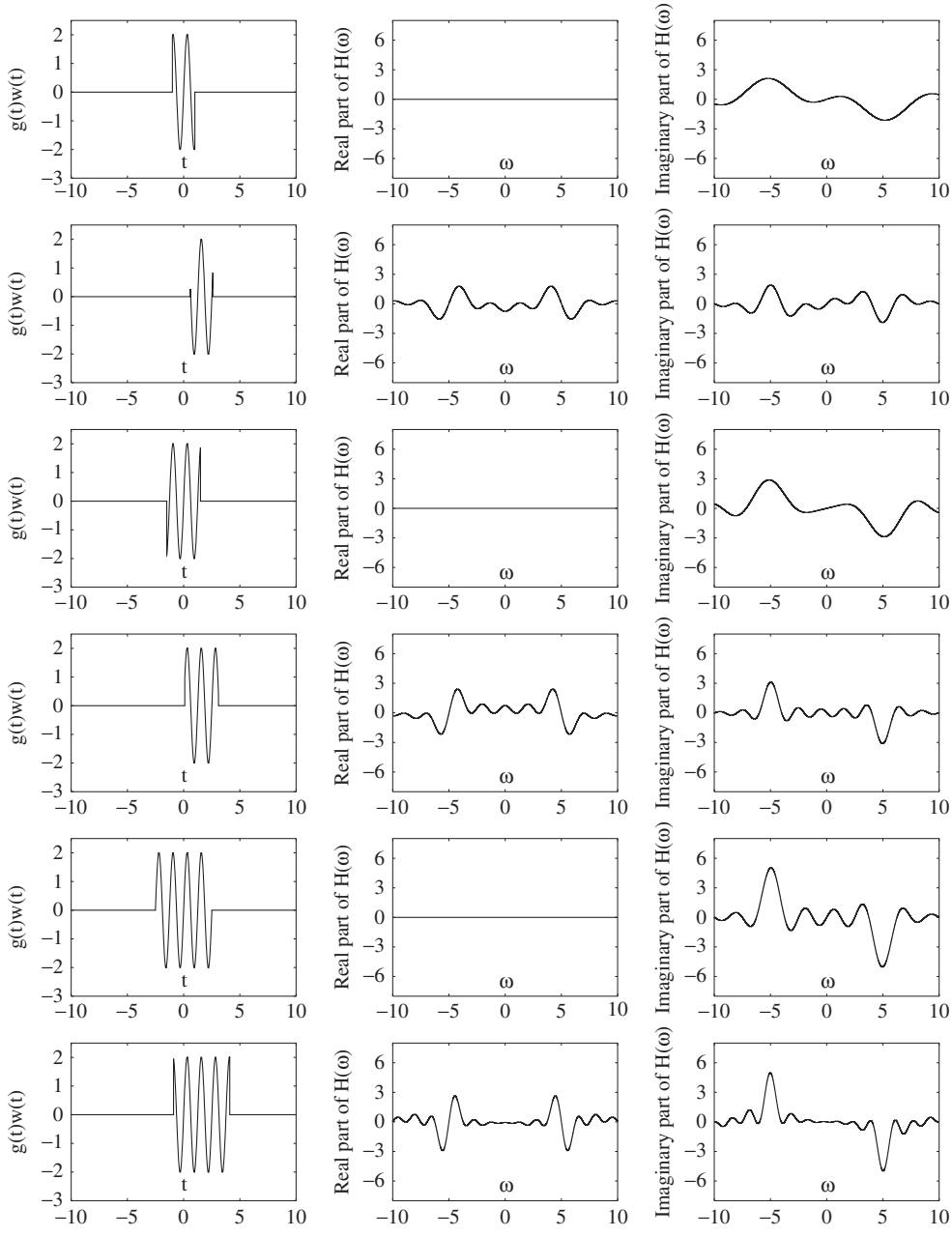


Figure 4.2 The effects of a rectangular window: windowed parts of the signal (first column) and the corresponding real and imaginary parts of their Fourier transforms (second and third columns). From top to bottom: ($\tau = 1, t_0 = 0$), ($\tau = 1, t_0 = 1.6$), ($\tau = 1.5, t_0 = 0$), ($\tau = 1.5, t_0 = 1.6$), ($\tau = 2.5, t_0 = 0$), ($\tau = 2.5, t_0 = 1.6$), respectively.

(Continued)

Example 4.11 (Continued)

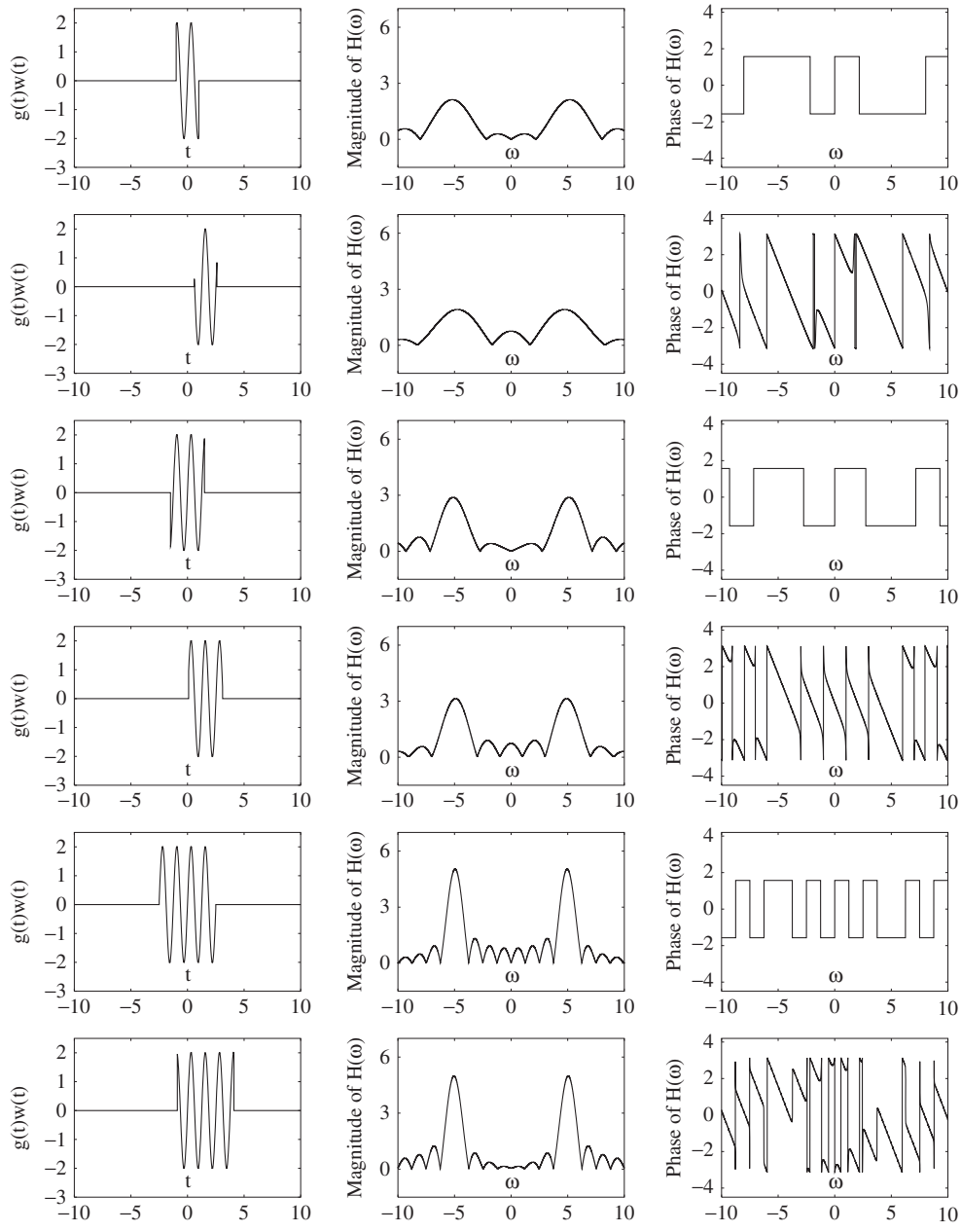


Figure 4.3 The effects of a rectangular window: windowed parts of the signal (first column) and the corresponding magnitude and phase of their Fourier transforms (second and third columns). From top to bottom: ($\tau = 1, t_0 = 0$), ($\tau = 1, t_0 = 1.6$), ($\tau = 1.5, t_0 = 0$), ($\tau = 1.5, t_0 = 1.6$), ($\tau = 2.5, t_0 = 0$), ($\tau = 2.5, t_0 = 1.6$), respectively.

Example 4.12

Repeat Example 4.11 using a Gaussian instead of a rectangular window.

A Gaussian window is infinite in extent, so it is characterised by its locality t_0 and its standard deviation, which in this context is called "spread" and is denoted by τ:

$$w(t) = \mathrm{e}^{\frac{(t-t_0)^2}{2\tau^2}}. \tag{4.57}$$

Equation (4.51) now takes the form:

$$\begin{aligned}
H(\omega) &= \int_{-\infty}^{\infty} A \sin\omega_0 t \mathrm{e}^{-\frac{(t-t_0)^2}{2\tau^2}} \mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \\
&= \frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}(\mathrm{e}^{\mathrm{j}\omega_0 t} - \mathrm{e}^{-\mathrm{j}\omega_0 t})\mathrm{e}^{-\frac{(t-t_0)^2}{2\tau^2}}\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \\
&= \frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{(t-t_0)^2}{2\tau^2}+\mathrm{j}(\omega_0-\omega)t}\mathrm{d}t - \frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{(t-t_0)^2}{2\tau^2}-\mathrm{j}(\omega_0+\omega)t}\mathrm{d}t \\
&= \frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{t^2+t_0^2-2tt_0}{2\tau^2}+\mathrm{j}(\omega_0-\omega)t}\mathrm{d}t \\
&\quad -\frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{t^2+t_0^2-2tt_0}{2\tau^2}-\mathrm{j}(\omega_0+\omega)t}\mathrm{d}t \\
&= \frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{t}{\sqrt{2}\tau}\right)^2+t\left(\frac{t_0}{\tau^2}+\mathrm{j}(\omega_0-\omega)\right)-\frac{t_0^2}{2\tau^2}}\mathrm{d}t \\
&\quad -\frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{t}{\sqrt{2}\tau}\right)^2+t\left(\frac{t_0}{\tau^2}-\mathrm{j}(\omega_0+\omega)\right)-\frac{t_0^2}{2\tau^2}}\mathrm{d}t.
\end{aligned} \tag{4.58}$$

We may compute this integral by completing the square in the exponent:

$$\begin{aligned}
&H(\omega) = \\
&\frac{A}{2\mathrm{j}}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{t}{\sqrt{2}\tau}\right)^2+2\left(\frac{t}{\sqrt{2}\tau}\right)\frac{1}{2}\sqrt{2}\tau\left(\frac{t_0}{\tau^2}+\mathrm{j}(\omega_0-\omega)\right)-\left(\frac{t_0}{\sqrt{2}\tau}+\frac{\tau\mathrm{j}(\omega_0-\omega)}{\sqrt{2}}\right)^2+\left(\frac{t_0}{\sqrt{2}\tau}+\frac{\tau\mathrm{j}(\omega_0-\omega)}{\sqrt{2}}\right)^2-\frac{t_0^2}{2\tau^2}}\mathrm{d}t \\
-\frac{A}{2\mathrm{j}}&\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{t}{\sqrt{2}\tau}\right)^2+2\left(\frac{t}{\sqrt{2}\tau}\right)\frac{1}{2}\sqrt{2}\tau\left(\frac{t_0}{\tau^2}-\mathrm{j}(\omega_0+\omega)\right)-\left(\frac{t_0}{\sqrt{2}\tau}-\frac{\tau\mathrm{j}(\omega_0+\omega)}{\sqrt{2}}\right)^2+\left(\frac{t_0}{\sqrt{2}\tau}-\frac{\tau\mathrm{j}(\omega_0+\omega)}{\sqrt{2}}\right)^2-\frac{t_0^2}{2\tau^2}}\mathrm{d}t = \\
&\qquad \frac{A}{2\mathrm{j}}\mathrm{e}^{\left(\frac{t_0}{\sqrt{2}\tau}+\frac{\tau\mathrm{j}(\omega_0-\omega)}{\sqrt{2}}\right)^2-\frac{t_0^2}{2\tau^2}}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{t}{\sqrt{2}\tau}-\left(\frac{t_0}{\sqrt{2}\tau}+\frac{\tau\mathrm{j}(\omega_0-\omega)}{\sqrt{2}}\right)\right)^2}\mathrm{d}t \\
&\qquad -\frac{A}{2\mathrm{j}}\mathrm{e}^{\left(\frac{t_0}{\sqrt{2}\tau}-\frac{\tau\mathrm{j}(\omega_0+\omega)}{\sqrt{2}}\right)^2-\frac{t_0^2}{2\tau^2}}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{t}{\sqrt{2}\tau}-\left(\frac{t_0}{\sqrt{2}\tau}-\frac{\tau\mathrm{j}(\omega_0+\omega)}{\sqrt{2}}\right)\right)^2}\mathrm{d}t.
\end{aligned} \tag{4.59}$$

(Continued)

Example 4.12 (Continued)

Then, by changing variables of integration and making use of the result of Example 3.36, we obtain:

$$\begin{aligned}
H(\omega) &= \frac{A}{2\mathrm{j}}\mathrm{e}^{\left(\frac{t_0}{\sqrt{2}\tau}+\frac{\tau\mathrm{j}(\omega_0-\omega)}{\sqrt{2}}\right)^2-\frac{t_0^2}{2\tau^2}}\sqrt{2}\tau\int_{-\infty}^{\infty}\mathrm{e}^{-x^2}\mathrm{d}x \\
&\quad -\frac{A}{2\mathrm{j}}\mathrm{e}^{\left(\frac{t_0}{\sqrt{2}\tau}-\frac{\tau\mathrm{j}(\omega_0+\omega)}{\sqrt{2}}\right)^2-\frac{t_0^2}{2\tau^2}}\sqrt{2}\tau\int_{-\infty}^{\infty}\mathrm{e}^{-x^2}\mathrm{d}x \\
&= \frac{A}{2\mathrm{j}}\mathrm{e}^{\left(\frac{t_0}{\sqrt{2}\tau}+\frac{\tau\mathrm{j}(\omega_0-\omega)}{\sqrt{2}}\right)^2-\frac{t_0^2}{2\tau^2}}\sqrt{2}\tau\sqrt{\pi} \\
&\quad -\frac{A}{2\mathrm{j}}\mathrm{e}^{\left(\frac{t_0}{\sqrt{2}\tau}-\frac{\tau\mathrm{j}(\omega_0+\omega)}{\sqrt{2}}\right)^2-\frac{t_0^2}{2\tau^2}}\sqrt{2}\tau\sqrt{\pi}.
\end{aligned} \tag{4.60}$$

Remembering that $\mathrm{j}^2=-1$ *and that* $1/\mathrm{j}=-\mathrm{j}$*, we may re-arrange this result:*

$$\begin{aligned}
H(\omega) &= -\mathrm{j}\frac{A\tau\sqrt{\pi}}{\sqrt{2}}\mathrm{e}^{-\frac{\tau^2(\omega_0-\omega)^2}{2}+\mathrm{j}t_0(\omega_0-\omega)} \\
&\quad +\mathrm{j}\frac{A\tau\sqrt{\pi}}{\sqrt{2}}\mathrm{e}^{-\frac{\tau^2(\omega_0+\omega)^2}{2}-\mathrm{j}t_0(\omega_0+\omega)}.
\end{aligned} \tag{4.61}$$

We may separate the real and the imaginary parts by expanding the complex exponentials:

$$\begin{aligned}
H(\omega) &= -\mathrm{j}\frac{A\tau\sqrt{\pi}}{\sqrt{2}}\mathrm{e}^{-\frac{\tau^2(\omega_0-\omega)^2}{2}}(\cos t_0(\omega_0-\omega)+\mathrm{j}\sin t_0(\omega_0-\omega)) \\
&\quad +\mathrm{j}\frac{A\tau\sqrt{\pi}}{\sqrt{2}}\mathrm{e}^{-\frac{\tau^2(\omega_0+\omega)^2}{2}}(\cos t_0(\omega_0+\omega)-\mathrm{j}\sin t_0(\omega_0+\omega)) \\
&= \frac{A\tau\sqrt{\pi}}{\sqrt{2}}\left(\mathrm{e}^{-\frac{\tau^2(\omega_0-\omega)^2}{2}}\sin t_0(\omega_0-\omega)+\mathrm{e}^{-\frac{\tau^2(\omega_0+\omega)^2}{2}}\sin t_0(\omega_0+\omega)\right) \\
&\quad -\mathrm{j}\frac{A\tau\sqrt{\pi}}{\sqrt{2}}\left(\mathrm{e}^{-\frac{\tau^2(\omega_0-\omega)^2}{2}}\cos t_0(\omega_0-\omega)-\mathrm{e}^{-\frac{\tau^2(\omega_0+\omega)^2}{2}}\cos t_0(\omega_0+\omega)\right).
\end{aligned} \tag{4.62}$$

Figures 4.4 and 4.5 show the corresponding results to those shown in Figures 4.2 and 4.3 for the rectangular window, respectively. Comparing these two sets of results we see that the shape of the window we use to estimate the Fourier transform of the signal **does** *affect our estimate. The use of the Gaussian window eliminated all secondary peaks in the Fourier transform of the windowed signal.*

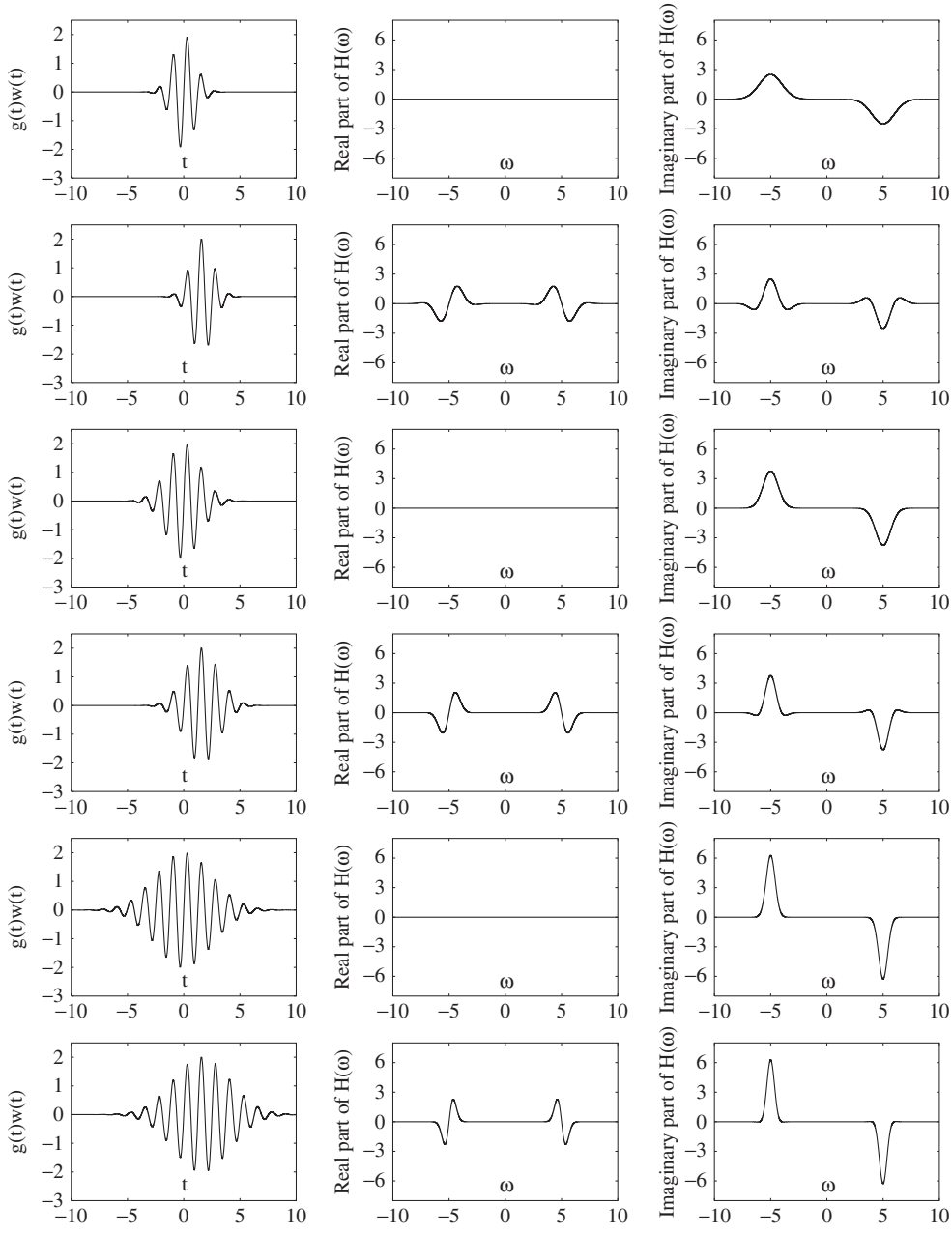


Figure 4.4 The effects of a Gaussian window: windowed parts of the signal (first column) and the corresponding real and imaginary parts of their Fourier transforms (second and third columns). From top to bottom: ($\tau = 1, t_0 = 0$), ($\tau = 1, t_0 = 1.6$), ($\tau = 1.5, t_0 = 0$), ($\tau = 1.5, t_0 = 1.6$), ($\tau = 2.5, t_0 = 0$), ($\tau = 2.5, t_0 = 1.6$), respectively.

(Continued)

Example 4.12 (Continued)

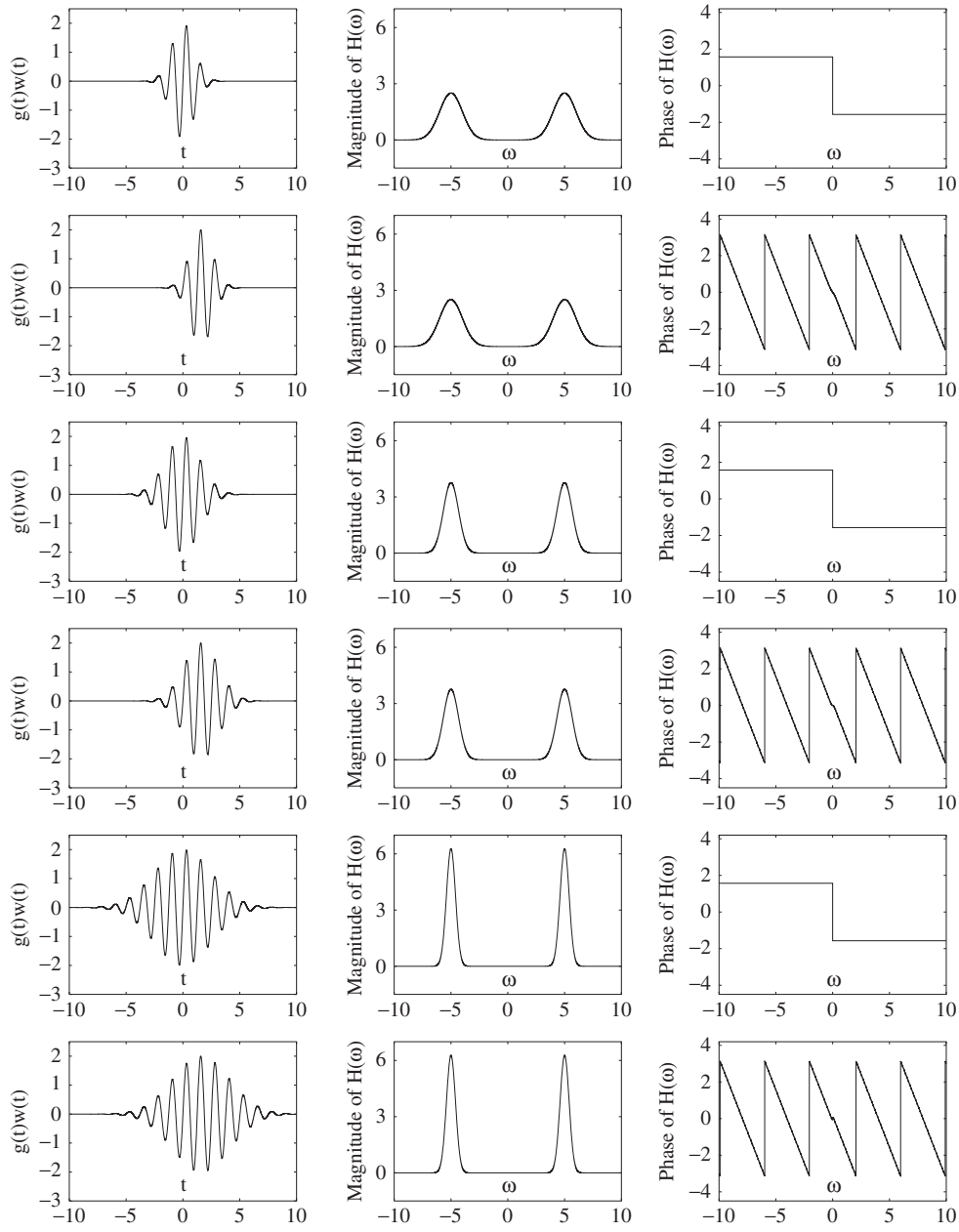


Figure 4.5 The effects of a Gaussian window: windowed parts of the signal (first column) and the corresponding magnitude and phase of their Fourier transforms (second and third columns). From top to bottom: $(\tau = 1, t_0 = 0)$, $(\tau = 1, t_0 = 1.6)$, $(\tau = 1.5, t_0 = 0)$, $(\tau = 1.5, t_0 = 1.6)$, $(\tau = 2.5, t_0 = 0)$, $(\tau = 2.5, t_0 = 1.6)$, respectively.

How can we estimate "what is happening where" in a digital signal?

The simplest way is to consider segments of the signal and compute the discrete Fourier transform (DFT) of each segment. This is called **short time Fourier transform**. If we consider a window of odd size and associate the DFT we compute inside it with the sample at the centre of the window, we shall be associating with each sample of the signal a "small" Fourier transform ("small" in this case means equal in length to the length of the window used).

The problem with this method is that by taking a piece of the signal at a time is equivalent to using a rectangular window. Such a window has sharp edges. When we compute the Fourier transform of a discrete signal, we make the implicit assumption that the signal is repeated periodically outside the window. This immediately introduces this repetition frequency. If the starting part of the windowed signal is different from the ending part, when the two parts are placed next to each other, a sudden jump is created. This induces high frequencies in the Fourier domain. In the continuous case this manifests itself in the form of side ripples appearing in the computed Fourier transform (see for example Figures 4.2 and 4.3).

This is shown schematically in Figure 4.6 where we show an infinitely long windowed signal and the signals for which DFT is computed for six different positions of the same window.

Example 4.13

Use the short time Fourier transform to compute local features of the signal defined by

$$g(n) = A \sin \frac{2\pi n}{N} \qquad \text{for } n = 0, 1, 2, ..., T \tag{4.63}$$

where $N = 10$, $T = 127$ **and** $A = 5$.

Let us consider a window of length $2M + 1$. *The first sample on which we can place the centre of the window is sample number* M, *and the last is sample number* $T - M$, *as otherwise part of the*

(Continued)

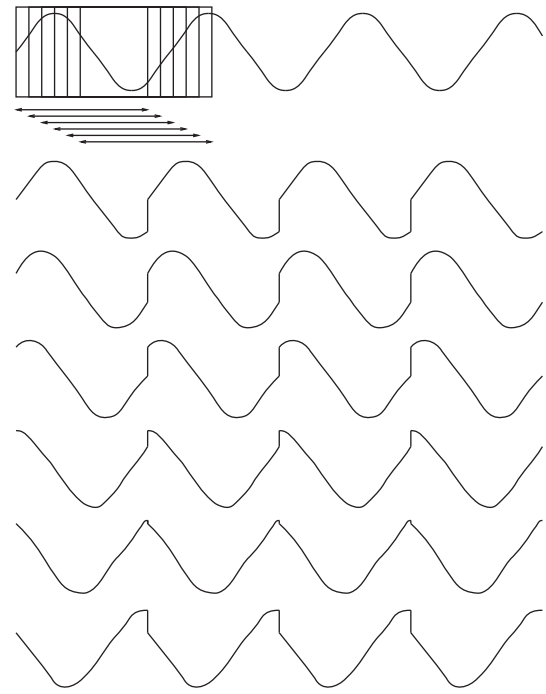


Figure 4.6 At the top an infinitely long signal with parts of it seen through six different positions of the same rectangular window explicitly shown. Below that we can see the signals which DFT "sees" for each different location of the window. The variety of the signals will be reflected in the variety of DFTs that will be computed, and ultimately in the variability of the features that will be extracted to characterise the signal locally.

Example 4.13 (Continued)

window will "stick out" of the signal. At each position n of the window we compute the short time Fourier transform

$$\begin{aligned} F(k;n) &= \frac{1}{\sqrt{2M+1}} \sum_{m=-M}^{M} A \sin\frac{2\pi(n+m)}{N} e^{-j\frac{2\pi km}{2M+1}} \\ &= \frac{A}{\sqrt{2M+1}} \sum_{m=-M}^{M} \sin\frac{2\pi(n+m)}{N} \cos\frac{2\pi km}{2M+1} \\ &\quad -j\frac{A}{\sqrt{2M+1}} \sum_{m=-M}^{M} \sin\frac{2\pi(n+m)}{N} \sin\frac{2\pi km}{2M+1} \end{aligned} \tag{4.64}$$

for $n = M, \ldots, T-M$.

This way, each sample n is associated with $2 \times (2M+1)$ *numbers, the values of the real and imaginary parts of the short time Fourier transform computed when the centre of the window is placed at n. This is schematically shown in Figure 4.7 for some values of n and for* $M = 8$. *Each one of these numbers may be thought of as a feature of the signal computed locally. We may plot the value of each feature as a function of n, remembering that due to border effects the first and the last M samples do not have feature values associated with them. Figure 4.8 shows all* $34(= 2 \times (2 \times 8 + 1))$ *feature sequences computed from the original signal, which is shown at the top of each panel. Figure 4.8a shows the real parts of the Fourier transforms, while Figure 4.8b shows the corresponding imaginary parts.*

Often, instead of the raw values of the real and imaginary parts of the Fourier transform at different frequencies, we use the magnitude and phase of the computed Fourier transform. In Chapter 3 we saw that one cannot extract useful features from the wrapped phase. So, we use only the magnitude information here. The corresponding 34 sequences of features produced this way are shown in Figure 4.9. In all these figures the first and last eight samples, which do not have a feature value computed for them, are given value 0.

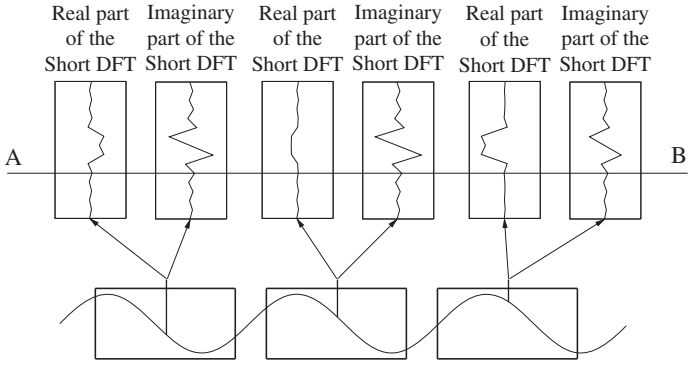


Figure 4.7 At the bottom an original signal with three different positions of the same rectangular window shown explicitly. At the central sample of each window we assign the real and imaginary parts of the short time Fourier transform we compute inside the window. If we read the numbers inside the corresponding panels along line AB and plot them as functions of the position of the window, we create feature sequences of the original signal, two for each frequency, coming from the real and the imaginary parts of the Fourier transform. By changing the position of line AB we can create different feature sequences.

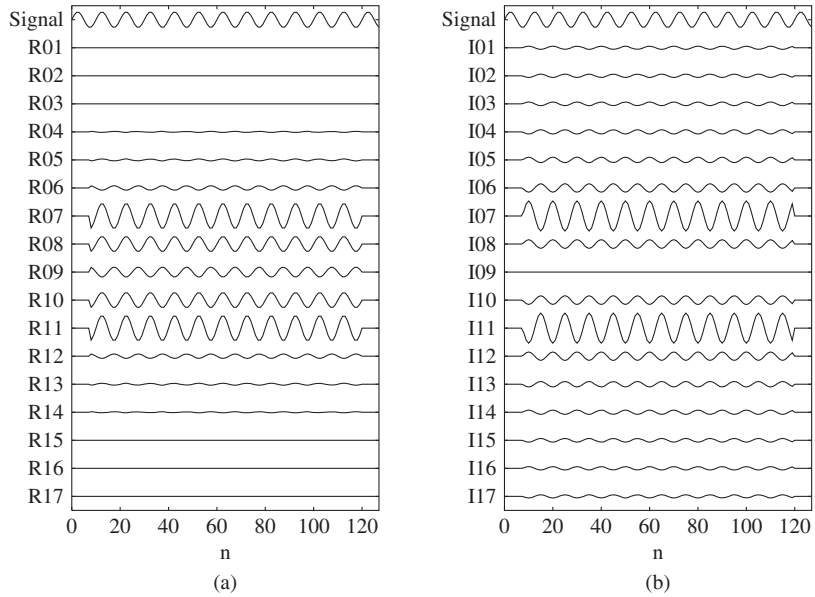
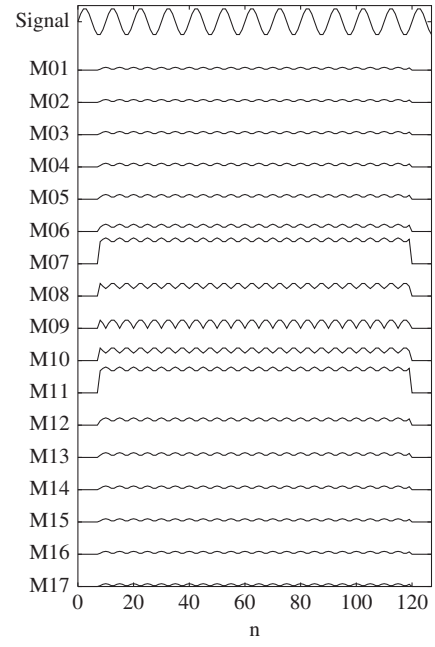


Figure 4.8 Feature sequences obtained from the real (a) and the imaginary (b) part of the short time Fourier transform. The original signal is shown at the top.

Figure 4.9 Feature sequences obtained from the magnitude of the short time Fourier transform. The original signal is shown at the top of the figure.



How can we deal with the variability of the values of a feature?

From the results of Example 4.13 we notice that most of the 34 features show great variability in the values they take from one location to the other, in spite of the fact that they are supposed to characterise the same signal. Such variability is undesirable because we wish to have features that change value only when the signal changes, so that we can detect this change. If we use a window of length $2M+1$, there are $2M+1$ different positions inside the window at which a particular sample may find itself, as the window is shifted across the signal. Thus, there will be $2M+1$ different signals the DFT algorithm "sees" when it is applied. Figure 4.6 demonstrates some of these different signals for a particular example. Due to this effect, if we use the components of the power spectrum associated with each sample to characterise the signal locally, these components will exhibit a lot of variation from one location to the next, in spite of the fact that they refer to the same signal. To have features that are invariant to the relative position of the sample inside the window, we may average over all relative positions of the particular sample. This is equivalent to taking the average value of the particular feature over the neighbouring $2M+1$ samples around the sample of interest. This is schematically shown in Figure 4.10.

Example 4.14

Plot signal

$$g(n)=\begin{cases} A\sin\frac{2\pi n}{N_1} & \text{for } n=0,1,2,...,T_1 \\ A\sin\frac{2\pi n}{N_2} & \text{for } n=T_1+1,\dots,T \end{cases} \tag{4.65}$$

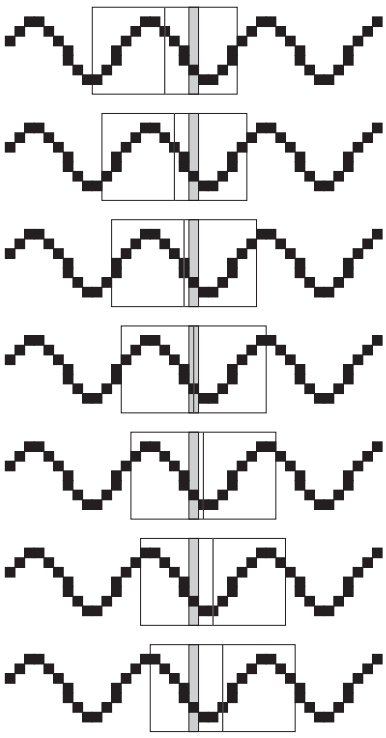
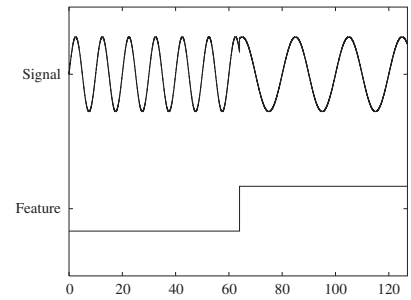


Figure 4.10 This figure shows a digital signal scanned by a window 15 samples long. The black vertical line inside the window marks the centre of the window. If we fix our attention to a particular sample of the signal, highlighted with the grey vertical strip, as the window is shifted along the signal, this sample finds itself in all possible relative positions with respect to the centre of the window. Therefore, to have features invariant to this relative position, we may average the values of a particular feature as computed over ± 7 neighbouring samples around the highlighted sample.

where $N_1 = 10$, $N_2 = 20$, $T_1 = 63$, $T = 127$ and $A = 5$, and explain how an ideal feature, which can help us segment the signal, should behave.
The signal is plotted at the top of Figure 4.11. The ideal feature should be such that it has the same value for samples $0, \ldots, T_1$, and changes to a different value for samples $T_1 + 1, \ldots, T$. Such a behaviour is shown schematically at the bottom of Figure 4.11. Such a feature can help us segment this composite signal into its constituent parts by using simple thresholding.

Figure 4.11 Top: the signal of Equation (4.65). Bottom: an ideal feature that characterises the signal locally and is useful in segmenting it into its constituent parts by simple thresholding.



Example 4.15

Use the short time Fourier transform with a window of size $2M + 1 = 25$ to segment the signal of Example 4.14.
For $M = 12$ we compute for $n = 12, \ldots, 115$ the following function:

$$\begin{aligned} F(k;n) &= \frac{1}{\sqrt{2M+1}} \sum_{m=-M}^{M} g(n+m) e^{-j\frac{2\pi km}{2M+1}} \\ &= \frac{1}{\sqrt{2M+1}} \sum_{m=-M}^{M} g(n+m) \cos\frac{2\pi km}{2M+1} \\ &\quad -j\frac{1}{\sqrt{2M+1}} \sum_{m=-M}^{M} g(n+m) \sin\frac{2\pi km}{2M+1} \\ &\equiv F_R(k;n) + jF_I(k;n). \end{aligned} \tag{4.66}$$

In Figure 4.12a we plot the real part $F_R(k;n)$ of this function against n for all 25 values of k, while the imaginary part $F_I(k;n)$ is plotted in Figure 4.12b. In Figure 4.13a we plot the magnitude, $S(k;n)$ against n for all values of k, one under the other, computed as:

$$S(k;n) = \sqrt{F_R(k;n)^2 + F_I(k;n)^2}. \tag{4.67}$$

Note that we gave value 0 to the first and the last 12 samples, for which no feature values were computed.

We observe that none of these 75 features of Figures 4.12 and 4.13a behaves as the ideal feature at the bottom of Figure 4.11. However, features M10, M11, M12, M14, M15 and M16 could be used

(Continued)

Example 4.15 (Continued)

for segmentation, as the amplitude of the fluctuations they show is less than that of the sudden change they show when the signal changes.

To reduce the fluctuation within the same part of the composite signal, we average the value of every feature over 25 successive samples and assign it to the central sample. Note that this introduces yet another boundary leaving the first and last 12 samples at either end with non-averaged values. We can either ignore those samples, or assign to them values that are not the average over 25 samples, but of as many samples as happen to be inside the averaging window and have computed feature values. This is the strategy we follow here to produce Figure 4.13b from the magnitude of the Fourier transform.

Now features AvM10, AvM11, AvM12, AvM14, AvM15 and AvM16 appear to have behaviour that approaches that of an ideal feature. By thresholding any one of them, one may obtain a reasonable segmentation of the signal. Due to the averaging we performed, the step edges we wished to observe in the plots of these features, when the signal changes, are not sharp. However, even blurred, these ramp edges allow us to segment the signal by the appropriate choice of a threshold.

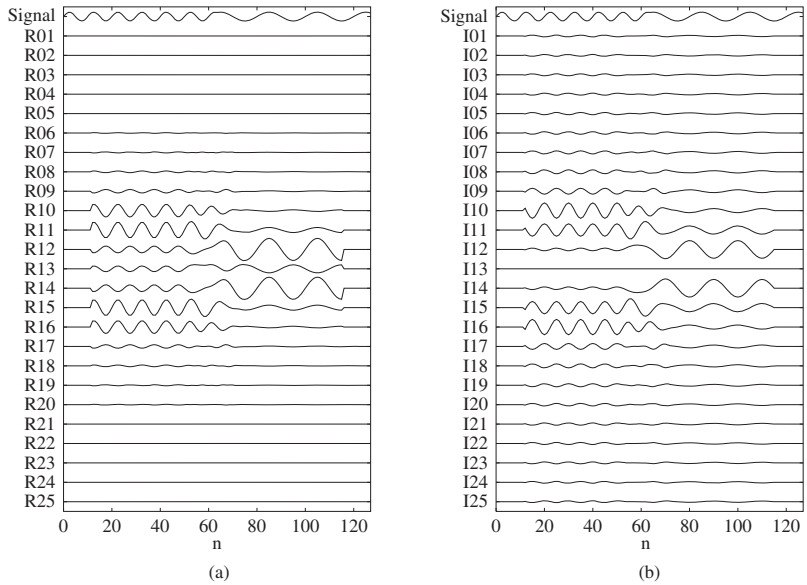


Figure 4.12 Feature sequences obtained from the real (a) and imaginary (b) part of the short time Fourier transform. The original signal is shown at the top of each panel.

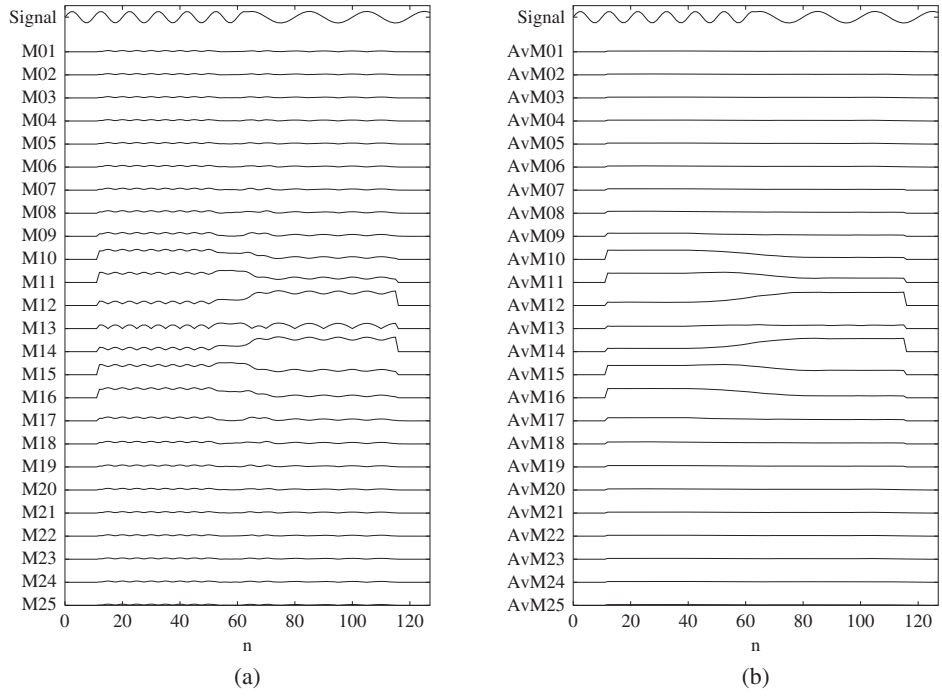


Figure 4.13 (a) Feature sequences obtained from the magnitude of the short time Fourier transform. (b) Averaged feature sequences obtained from the magnitude of the short Fourier transform. The original signal is shown at the top of each panel.

Example 4.16

Repeat Example 4.15 with a window of size 5.

We now use formula (4.66) with $M = 2$. The features computed from the power spectrum are shown in Figure 4.14, while Figure 4.15 shows the same features after smoothing. We observe that the best features now are only marginally appropriate for segmenting the signal (e.g. features AvM02 and AvMo4). This is because this time the window we used was too small and it did not capture effectively the different characteristics of the different parts of the signal.

(Continued)

Example 4.16 (Continued)

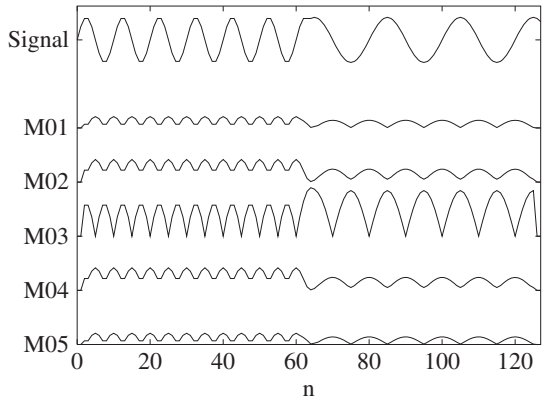


Figure 4.14 Feature sequences obtained from the magnitude of the short Fourier transform using $M = 2$. The original signal is shown at the top of the figure.

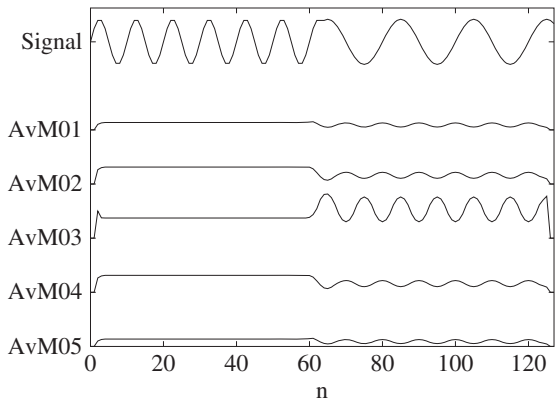


Figure 4.15 Averaged feature sequences obtained from the magnitude of the short Fourier transform using $M = 2$. The original signal is shown at the top of the figure.

How do we know which size window we should use?

We do not know! There are two issues we should deal with. (1) First of all, we must make sure that we search all frequencies in the frequency domain of the signal, because we do not know a priori which is the characteristic frequency of the signal that will allow us to pick it up. For example, in Example 4.16, the use of a very small window implied that we were looking for very high frequencies. This is not immediately obvious, but it can be understood in terms of Figure 4.6. We note from Figure 4.6 that, in spite of the distorted version of the original signal seen by the DFT algorithm, DFT basically sees a signal that has the same fundamental period as the original signal. This is because the window used was large enough to capture that basic variation of the original signal. If, however, the window used was much smaller than the typical wavelength of the original signal, then the fundamental frequency of the original signal would have been totally washed out by the periodicity of the window itself. This is demonstrated in Figure 4.16 where the window used is about one third the size of the window used in Figure 4.6.

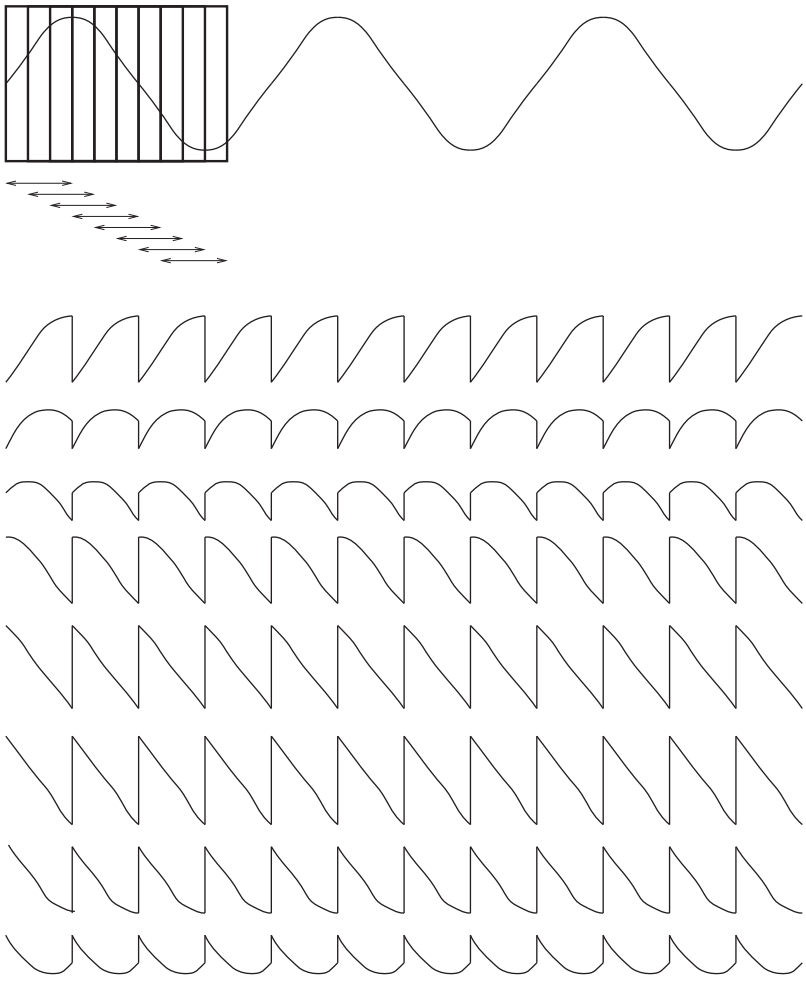


Figure 4.16 The original signal of Figure 4.6 scanned by a narrow window. If the discrete Fourier transform is applied at the signal inside the window, depending on the locality of the window, the algorithm "sees" different signals that have no relevance to the true signal. Some example such signals are shown here for eight different locations of the small window.

(2) Even when the window is large enough to capture the fundamental periodicity of a signal, it may not be large enough to correspond to a small enough window in the frequency domain that will allow one to resolve two parts of the signal that have similar frequencies. If two nearby frequencies are present in the same signal, the convolution of the Fourier transform of the window with the Fourier transform of the signal may result in the two frequencies being totally swamped by the spectrum of the window. This situation is exacerbated by the use of a rectangular window that has strong side lobes. To reduce this effect, the use of a Gaussian window is preferable over the use of a rectangular window. This is demonstrated schematically in Figure 4.17.

Further, a systematic search of all possible frequencies that might be present in a signal is necessary, to make sure that no frequency is overlooked. This can be achieved by expanding the spectrum of the signal in terms of **Gabor functions**. This is nothing other than the use of a local Gaussian window before computing the Fourier transform.

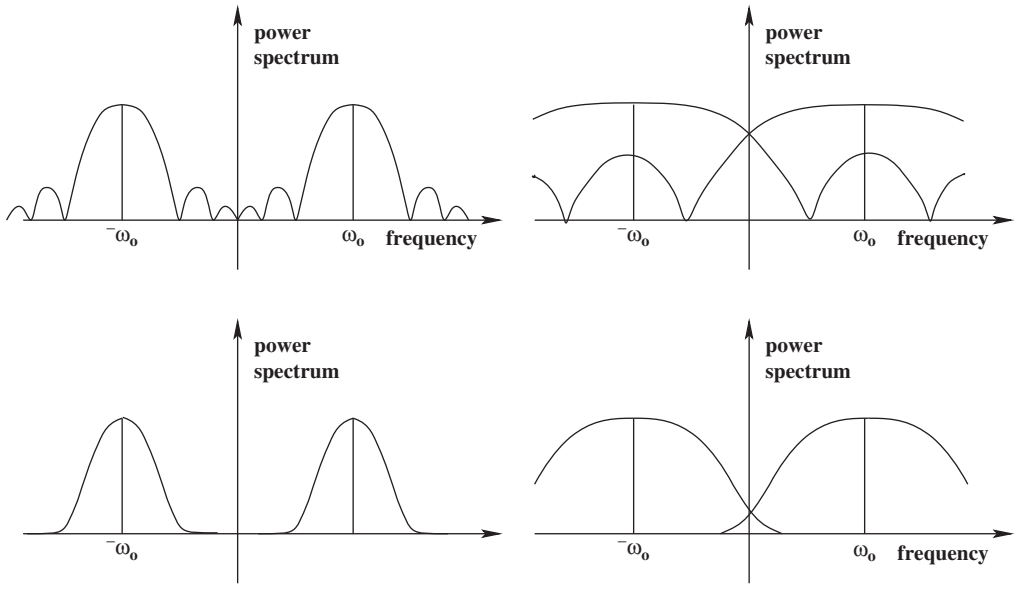


Figure 4.17 Suppose that we are seeing a signal with fundamental frequency ω_0 using a sufficiently large window. The spectrum of what we see will look like the graph in the top left if the window is rectangular: the fundamental frequency of the original signal is preserved but the details of the original signal vary due to interference of the high frequencies of the two components of the spectrum. This is the situation in Figure 4.6. If we use a too narrow window, its spectrum is very broad and the interference between the two branches of the spectrum is such that the fundamental frequency of the original signal is totally lost (top-right panel). This is the situation in Figure 4.16. If we use a Gaussian window, there is improvement in both cases because the spectrum of such a window does not have side lobes (bottom panels).

How is the uncertainty principle generalised to 2D?

For each spatial dimension we introduce, we have a new frequency dimension. So, for variables x and y we have the corresponding frequencies ω_x and ω_y. If we have a function $f(x,y)$, we may compute three different second-order dispersions for it, namely $(\Delta x)^2$, $(\Delta y)^2$ and $\Delta x\Delta y$, defined as

$$\begin{aligned}(\Delta x)^2 &\equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\bar{x})^2 f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y\\ (\Delta y)^2 &\equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(y-\bar{y})^2 f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y\\ \Delta x\Delta y &\equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\bar{x})(y-\bar{y})f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y\end{aligned} \tag{4.68}$$

where

$$E_\infty \equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y \tag{4.69}$$

and

$$\begin{aligned}\bar{x} &\equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xf(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y\\ \bar{y} &\equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} yf(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y.\end{aligned} \tag{4.70}$$

We may assume that axes x and y are chosen in such a way that dispersion $\Delta x\Delta y$ is zero. Then axes x and y are the **principal axes** of the signal function $f(x,y)$. If this is not the case, we can always define a new set of axes in which the cross-dispersion $\Delta x\Delta y$ is zero (see Example 4.17).

The spectral bandwidths $\Delta\omega_x$ and $\Delta\omega_y$ of the signal are defined as

$$(\Delta\omega_x)^2 \equiv \frac{1}{4\pi^2 E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\omega_x-\overline{\omega}_x)^2 F(\omega_x,\omega_y)F^*(\omega_x,\omega_y)\mathrm{d}\omega_x\mathrm{d}\omega_y$$
$$(\Delta\omega_y)^2 \equiv \frac{1}{4\pi^2 E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\omega_y-\overline{\omega}_y)^2 F(\omega_x,\omega_y)F^*(\omega_x,\omega_y)\mathrm{d}\omega_x\mathrm{d}\omega_y \tag{4.71}$$

where $F(\omega_x,\omega_y)$ is the Fourier transform of signal $f(x,y)$ and

$$\overline{\omega}_x \equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\omega_x F(\omega_x,\omega_y)F^*(\omega_x,\omega_y)\mathrm{d}\omega_x\mathrm{d}\omega_y$$
$$\overline{\omega}_y \equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\omega_y F(\omega_x,\omega_y)F^*(\omega_x,\omega_y)\mathrm{d}\omega_x\mathrm{d}\omega_y. \tag{4.72}$$

Then the signal satisfies a separate uncertainty principle for each coordinate/frequency pair:

$$(\Delta x)^2(\Delta\omega_x)^2 \geq \frac{1}{4} \qquad (\Delta y)^2(\Delta\omega_y)^2 \geq \frac{1}{4}. \tag{4.73}$$

The joint uncertainty of the 2D signal is the product of these two uncertainties:

$$(\Delta x)^2(\Delta\omega_x)^2(\Delta y)^2(\Delta\omega_y)^2 \geq \frac{1}{16}. \tag{4.74}$$

Example B4.17

Assume that the cross-dispersion of a function $f(x,y)$ as defined by the third of Equations (4.68) is not zero. Define a new coordinate system $(\tilde{x},\tilde{y})$ in which the cross-dispersion is zero.

Let us assume that the new axes $(\tilde{x},\tilde{y})$ can be obtained from the old ones by rotation by an angle θ:

$$\tilde{x} = x\cos\theta + y\sin\theta$$
$$\tilde{y} = -x\sin\theta + y\cos\theta. \tag{4.75}$$

Our purpose is to define angle θ. Let us start by computing the cross-dispersion of the function in the new coordinate system and setting it to 0:

$$\Delta\tilde{x}\Delta\tilde{y} \equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\tilde{x}-\overline{\tilde{x}})(\tilde{y}-\overline{\tilde{y}})f(\tilde{x},\tilde{y})f^*(\tilde{x},\tilde{y})\mathrm{d}\tilde{x}\mathrm{d}\tilde{y} = 0. \tag{4.76}$$

We use Equations (4.75) to express $\Delta\tilde{x}\Delta\tilde{y}$ in terms of the old coordinates x and y. We observe that

$$\mathrm{d}\tilde{x}\mathrm{d}\tilde{y} = \begin{vmatrix} \frac{\partial\tilde{x}}{\partial x} & \frac{\partial\tilde{x}}{\partial y} \\ \frac{\partial\tilde{y}}{\partial x} & \frac{\partial\tilde{y}}{\partial y} \end{vmatrix}\mathrm{d}x\mathrm{d}y = \begin{vmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{vmatrix}\mathrm{d}x\mathrm{d}y = (\cos^2\theta+\sin^2\theta)\mathrm{d}x\mathrm{d}y = \mathrm{d}x\mathrm{d}y. \tag{4.77}$$

Then

$$\frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x\cos\theta + y\sin\theta - \overline{x}\cos\theta - \overline{y}\sin\theta)\times$$
$$(-x\sin\theta + y\cos\theta + \overline{x}\sin\theta - \overline{y}\cos\theta)f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y = 0$$
$$\Rightarrow \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}[(x-\overline{x})\cos\theta + (y-\overline{y})\sin\theta]\times$$
$$[-(x-\overline{x})\sin\theta + (y-\overline{y})\cos\theta)]f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y = 0$$

(Continued)

Example B4.17 (Continued)

$$\Rightarrow \frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[-(x-\bar{x})^2\cos\theta\sin\theta+(x-\bar{x})(y-\bar{y})\cos^2\theta\right.$$
$$\left.-(y-\bar{y})(x-\bar{x})\sin^2\theta+(y-\bar{y})^2\sin\theta\cos\theta\right]f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y=0$$
$$\Rightarrow -\cos\theta\sin\theta\frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\bar{x})^2f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y$$
$$+(\cos^2\theta-\sin^2\theta)\frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\bar{x})(y-\bar{y})f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y$$
$$+\sin\theta\cos\theta\frac{1}{E_\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(y-\bar{y})^2f(x,y)f^*(x,y)\mathrm{d}x\mathrm{d}y=0$$
$$\Rightarrow -\cos\theta\sin\theta(\Delta x)^2+(\cos^2\theta-\sin^2\theta)\Delta x\Delta y+\sin\theta\cos\theta(\Delta y)^2=0 \tag{4.78}$$

where we made use of the definitions of $(\Delta x)^2$, $(\Delta y)^2$ *and* $\Delta x\Delta y$ *as given by Equations (4.68).*

We also observe that $\sin\theta\cos\theta=\frac{1}{2}\sin(2\theta)$ *and that* $\cos^2\theta-\sin^2\theta=\cos(2\theta)$. *Then we have an equation for angle* 2θ*:*

$$\frac{1}{2}\sin(2\theta)[-(\Delta x)^2+(\Delta y)^2]+\cos(2\theta)\Delta x\Delta y=0\Rightarrow\tan(2\theta)=\frac{2\Delta x\Delta y}{(\Delta x)^2-(\Delta y)^2}. \tag{4.79}$$

This equation allows us to define the angle by which we have to rotate the coordinate system, so that we define axes which coincide with the principal axes of the signal, i.e. axes in which the signal has zero cross-dispersion.

Example B4.18

Find the principal axes of function

$$f(x,y)\equiv\begin{cases}\sqrt{a_1x^2+a_2y^2+a_3xy+a_4} & \text{for } |x|\le 1 \text{ and } |y|\le 1\\ 0 & \text{elsewhere}\end{cases}. \tag{4.80}$$

We have to use Equation (4.79) in conjunction with Equations (4.68)–(4.70). By applying Equations (4.70) we may easily show that $\bar{x}=\bar{y}=0$. *We also note from Equation (4.79) that* E_∞*, which appears in equations (4.68), cancels out. So we need not compute it. Then we must compute the double integrals that appear in equations (4.68):*

$$(\Delta x)^2=\frac{1}{E_\infty}\int_{-1}^{1}\int_{-1}^{1}(a_1x^4+a_2x^2y^2+a_3x^3y+a_4x^2)\mathrm{d}x\mathrm{d}y=\frac{1}{E_\infty}\left(\frac{4a_1}{5}+\frac{4a_2}{9}+\frac{4a_4}{3}\right).$$

The result for $(\Delta y)^2$ *can be computed by exchanging the roles of* a_1 *and* a_2*. So, it is expected to be* $(\Delta y)^2=\frac{1}{E_\infty}\left(\frac{4a_2}{5}+\frac{4a_1}{9}+\frac{4a_4}{3}\right)$. *Also:*

$$\Delta x\Delta y=\frac{1}{E_\infty}\int_{-1}^{1}\int_{-1}^{1}(a_1x^3y+a_2xy^3+a_3x^2y^2+a_4xy)\mathrm{d}x\mathrm{d}y=\frac{1}{E_\infty}\frac{4a_3}{9}. \tag{4.81}$$

Using all these results in Equation (4.79), yields:

$$\tan(2\theta)=\frac{5a_3}{2(a_1-a_2)}\Rightarrow\theta=\frac{1}{2}\tan^{-1}\left(\frac{5a_3}{2(a_1-a_2)}\right). \tag{4.82}$$

This means that the first principal axis of the function is at angle θ *with respect to the x axis, and the second principal axis is at angle* $\theta+90^\circ$ *with respect to the same axis.*

4.2 Gabor Functions

What is a Gabor function?

A Gabor function in 1D is defined as:

$$G(t; t_0, \omega_0, \sigma) \equiv \mathrm{e}^{-\frac{(t-t_0)^2}{2\sigma^2}+\mathrm{j}\omega_0 t}. \tag{4.83}$$

The Fourier transform of the Gabor function is:

$$\hat{G}(\omega; t_0, \omega_0, \sigma) = \sqrt{2\pi}\sigma \mathrm{e}^{-\mathrm{j}(\omega-\omega_0)t_0}\mathrm{e}^{-\frac{\sigma^2(\omega-\omega_0)^2}{2}}. \tag{4.84}$$

Gabor showed that this function is the most general function that obtains the minimum in the uncertainty principle for signal processing.

Example B4.19

Prove that the Fourier transform of the Gabor function is given by Equation (4.84).
The Fourier transform of the Gabor function may be computed as follows:

$$\hat{G}(\omega; t_0, \omega_0, \sigma) = \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(t-t_0)^2}{2\sigma^2}+\mathrm{j}\omega_0 t}\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t. \tag{4.85}$$

If we multiply the integrand by $\mathrm{e}^{\mathrm{j}(\omega-\omega_0)t_0}\mathrm{e}^{-\mathrm{j}(\omega-\omega_0)t_0}$*, we may write:*

$$\hat{G}(\omega; t_0, \omega_0, \sigma) = \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(t-t_0)^2}{2\sigma^2}-\mathrm{j}(\omega-\omega_0)(t-t_0)}\mathrm{d}t\ \mathrm{e}^{-\mathrm{j}(\omega-\omega_0)t_0}. \tag{4.86}$$

We change variable of integration from t *to* $x \equiv t - t_0$ *and for brevity we define* $\tilde{\omega} \equiv \omega - \omega_0$*:*

$$\hat{G}(\omega; t_0, \omega_0, \sigma) = \mathrm{e}^{-\mathrm{j}\tilde{\omega}t_0}\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{x^2}{2\sigma^2}-\mathrm{j}\tilde{\omega}x}\mathrm{d}x. \tag{4.87}$$

This integral is similar to that of Equation (3.303) and we may make use of Equation (3.311) to obtain

$$\hat{G}(\omega; t_0, \omega_0, \sigma) = \mathrm{e}^{-\mathrm{j}\tilde{\omega}t_0}\sqrt{2\pi}\sigma\mathrm{e}^{-\frac{\sigma^2\tilde{\omega}^2}{2}} \tag{4.88}$$

which upon re-arrangement yields (4.84).

Example B4.20

Prove that the Gabor function achieves the minimum of the uncertainty inequality (4.38).
We note that in order to compute the time dispersion and the spectral bandwidth of a signal we need expressions $G(t; t_0, \omega_0, \sigma)^2$ *and* $\hat{G}(\omega; t_0, \omega_0, \sigma)^2$ *which appear in equations (4.24), (4.26), (4.27) and (4.29). These expressions are given by*

$$\begin{aligned} G(t; t_0, \omega_0, \sigma)^2 &= G(t; t_0, \omega_0, \sigma)G^*(t; t_0, \omega_0, \sigma) \\ &= \mathrm{e}^{-\frac{(t-t_0)^2}{2\sigma^2}+\mathrm{j}\omega_0(t-t_0)}\mathrm{e}^{-\frac{(t-t_0)^2}{2\sigma^2}-\mathrm{j}\omega_0(t-t_0)} \\ &= \mathrm{e}^{-\frac{(t-t_0)^2}{\sigma^2}} \end{aligned} \tag{4.89}$$

(Continued)

Example B4.20 (Continued)

and

$$\begin{aligned}\hat{G}(\omega;t_0,\omega_0,\sigma)^2 &= \hat{G}(\omega;t_0,\omega_0,\sigma)\hat{G}^*(\omega;t_0,\omega_0,\sigma)\\ &= \sqrt{2\pi}\sigma e^{-\frac{\sigma^2(\omega-\omega_0)^2}{2}} e^{-j(\omega-\omega_0)t_0}\sqrt{2\pi}\sigma e^{-\frac{\sigma^2(\omega-\omega_0)^2}{2}} e^{j(\omega-\omega_0)t_0}\\ &= 2\pi\sigma^2 e^{-\sigma^2(\omega-\omega_0)^2}. \end{aligned} \tag{4.90}$$

Using (4.89) we may compute the time dispersion of this signal following the steps of Example 4.6. The only adaptation we have to make is to change the variable of integration from t to $\tilde{t} \equiv t - t_0$, with everything else remaining the same.

Using (4.90) we may compute the spectral bandwidth of the signal following the steps of Example 4.7. Again, the only adaptation we have to make is to change the variable of integration from ω to $\tilde{\omega} \equiv \omega - \omega_0$, with everything else remaining the same. So, the Gabor function has the same time dispersion and spectral bandwidth as the Gaussian function, and therefore it minimises their product just as the Gaussian function does, as it was shown in Example 4.8.

Why are Gabor functions useful in analysing a signal?

Gabor functions allow us to represent a signal in time and frequency domain simultaneously. Notice that $G(t;t_0,\omega_0,\sigma)$ may be thought of as a family of functions of t for different pairs of values of t_0 and ω_0. A schematic representation of this family of functions is shown in Figure 4.18. This family of functions covers the (t,ω) space completely, so we must be able to represent any signal function $g(t)$ in terms of this family of functions.

In a similar way, $\hat{G}(\omega;t_0,\omega_0,\sigma)$ may be thought of as a family of functions of ω for different pairs of values of t_0 and ω_0. A schematic representation of this family of functions is shown in Figure 4.19. This family of functions also covers the (t,ω) space completely, so we must be able to represent any function $\hat{g}(\omega)$ in terms of this family of functions.

So, we may think of the family of $G(t;t_0,\omega_0,\sigma)$ functions as forming a set of basis functions in terms of which any function $g(t)$ may be expanded. Each basis function expresses a different answer to the question "what is where?". Each coefficient of the expansion of a signal in terms of these elementary functions tells us how much of the particular type of "what" is at each locality of the original function. To compute this coefficient for a particular basis function, all we have to do is to project function $g(t)$ onto the basis function. If we are dealing with digital functions, we can think of them as vectors and projecting one digital function onto another is nothing more than projecting one vector onto another, which is nothing more than taking the dot product of the two vectors, i.e. multiplying them component by component and adding the results. For continuous functions the process is the same: we multiply them point by point and integrate over all points. If the functions are complex, we must take the complex conjugate of one of them. So, to find how much of the particular "what", expressed by $\omega = \omega_1$, is present at the particular "where", expressed by $t = t_1$, in function $g(t)$, we must project $g(t)$ onto the particular basis function $G(t;t_0 = t_1,\omega_0 = \omega_1,\sigma)$, i.e. we must simply compute the integral:

$$\begin{aligned} a(t_1,\omega_1) &\equiv \int_{-\infty}^{\infty} g(t)G^*(t;t_0 = t_1,\omega_0 = \omega_1,\sigma)\mathrm{d}t\\ &= \int_{-\infty}^{\infty} g(t)e^{-\frac{(t-t_1)^2}{2\sigma^2} - j\omega_1 t}\mathrm{d}t. \end{aligned} \tag{4.91}$$

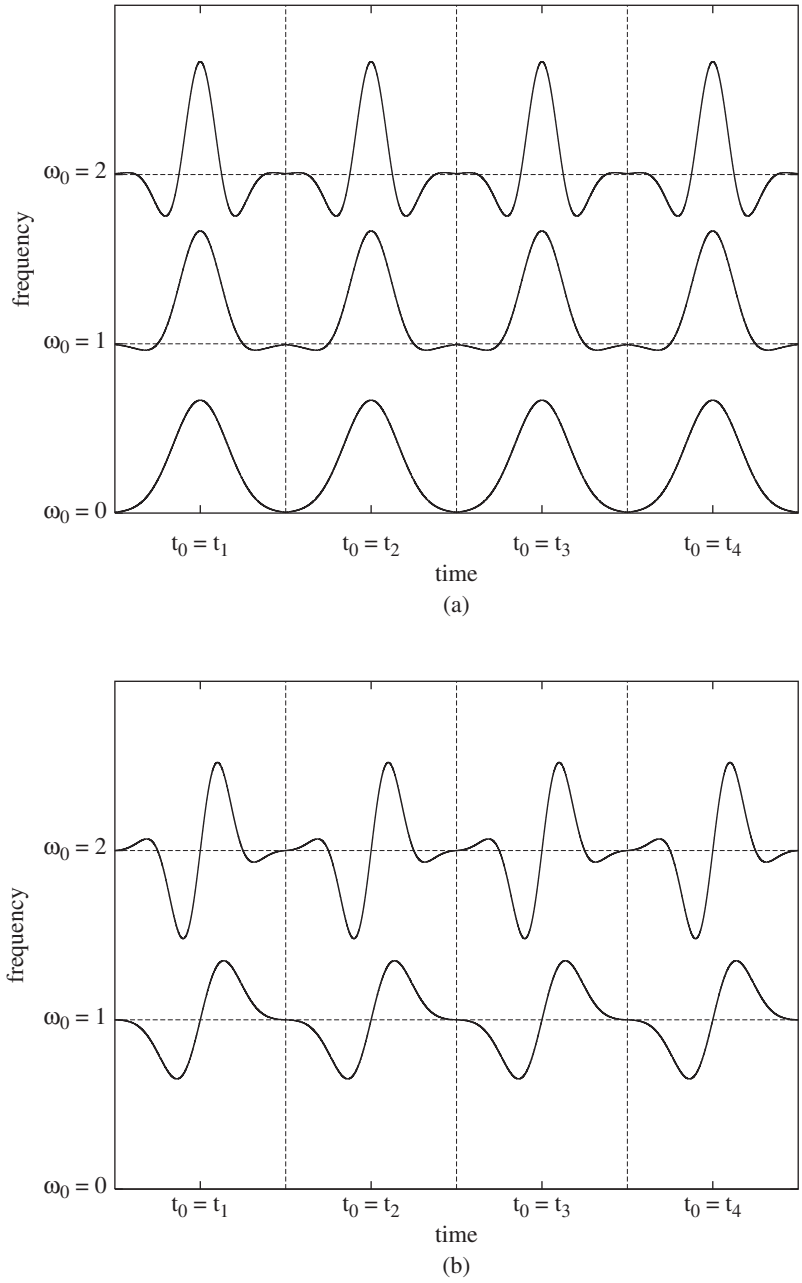


Figure 4.18 Gabor function $G(t; t_0, \omega_0, \sigma)$ may be used to create elementary signals in terms of which any arbitrary signal $g(t)$ may be represented in a joint time–frequency representation. This figure shows schematically some of these elementary signals. The real part of each elementary signal is shown in (a) and its imaginary part in (b).

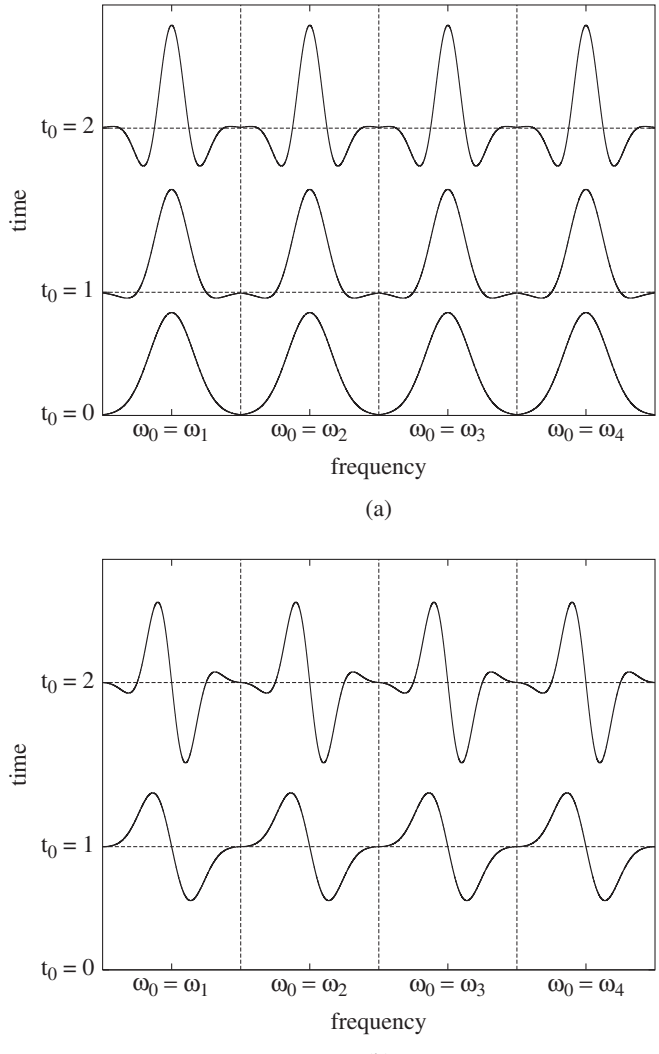


Figure 4.19 Gabor function $\hat{G}(\omega; t_0, \omega_0, \sigma)$ may be used to create elementary signals in terms of which an arbitrary signal $\hat{g}(\omega)$ may be represented in a joint time–frequency representation. This figure shows schematically some of these elementary signals. The real part of each elementary signal is shown in (a) and its imaginary part in (b).

If we substitute $G(t; t_0, \omega_0, \sigma)$ from its definition, it becomes obvious that the above integral is nothing other than the original function $g(t)$, multiplied with a Gaussian window to give preference to some part of it only (i.e. to answer the "where" part of the question), and then taking the Fourier transform, concentrating at a particular frequency ω_1, to answer the "what" part of the question as well.

In a similar way, we may think of the family of $\hat{G}(\omega; t_0, \omega_0, \sigma)$ functions as forming a set of basis functions in terms of which any function $\hat{g}(\omega)$ may be expanded. Again, each basis function expresses a different answer to the question "what is where?". So, we may analyse function $\hat{g}(\omega)$ in terms of this basis and compute the coefficients of this expansion. Each coefficient will tell us how

much of the particular basis function is present in the original function. These coefficients may again be computed by projecting function $\hat{g}(\omega)$ onto each basis function in turn:

$$\begin{aligned} b(t_1,\omega_1) &\equiv \int_{-\infty}^{\infty} \hat{g}(\omega)\hat{G}^*(\omega;t_0=t_1,\omega_0=\omega_1,\sigma)\mathrm{d}f \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{g}(\omega)\hat{G}^*(\omega;t_0=t_1,\omega_0=\omega_1,\sigma)\mathrm{d}\omega \\ &= \frac{\sigma}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \hat{g}(\omega)\mathrm{e}^{\mathrm{j}(\omega-\omega_1)t_1}\mathrm{e}^{-\frac{\sigma^2(\omega-\omega_1)^2}{2}}\mathrm{d}\omega \\ &= \frac{\sigma}{\sqrt{2\pi}}\mathrm{e}^{-\mathrm{j}\omega_1 t_1}\int_{-\infty}^{\infty} \hat{g}(\omega)\mathrm{e}^{\mathrm{j}\omega t_1}\mathrm{e}^{-\frac{\sigma^2(\omega-\omega_1)^2}{2}}\mathrm{d}\omega. \end{aligned} \tag{4.92}$$

This integral is nothing more than multiplying function $\hat{g}(\omega)$ with a Gaussian window centred at ω_1, and then taking the inverse Fourier transform, apart from some constant factor and a phase coefficient.

As both these expansions answer the question "what is where?" in a signal, if $\hat{g}(\omega)$ is the Fourier transform of $g(t)$, either expansion should give the same result. This is indeed the case as it is shown in Example 4.21.

It is important to note that neither of the two sets of basis functions is orthogonal. Each Gaussian window is infinite in extent and, therefore, each segment of either function we isolate, inevitably contains information from all other parts of the function due to the infinite tails of the Gaussian window. The expansion of a function, therefore, in terms of Gabor functions contains redundant information, which, in the particular case of frequency band segmentation, plays the role of interference from all bands to all other bands. In practice this is not a problem because the tails of the Gaussian die out quite fast.

Example B4.21

Prove that $a(t_1,\omega_1)$ defined by Equation (4.91) and $b(t_1,\omega_1)$ defined by Equation (4.92) are identical, if functions $g(t)$ and $\hat{g}(\omega)$ form a Fourier transform pair.

Since functions $g(t)$ and $\hat{g}(\omega)$ form a Fourier transform pair, we may write:

$$\hat{g}(\omega)=\int_{-\infty}^{\infty} g(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \qquad g(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{g}(\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega. \tag{4.93}$$

We start from Equation (4.92) and substitute $\hat{g}(\omega)$ from (4.93):

$$\begin{aligned} b(t_1,\omega_1) &= \frac{\sigma}{\sqrt{2\pi}}\mathrm{e}^{-\mathrm{j}\omega_1 t_1}\int_{-\infty}^{\infty}\left\{\int_{-\infty}^{\infty} g(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t\right\}\mathrm{e}^{\mathrm{j}\omega t_1}\mathrm{e}^{-\frac{\sigma^2(\omega-\omega_1)^2}{2}}\mathrm{d}\omega \\ &= \frac{\sigma}{\sqrt{2\pi}}\mathrm{e}^{-\mathrm{j}\omega_1 t_1}\int_{-\infty}^{\infty} g(t)\left\{\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{\sigma^2(\omega-\omega_1)^2}{2}-\mathrm{j}\omega(t-t_1)}\mathrm{d}\omega\right\}\mathrm{d}t. \end{aligned} \tag{4.94}$$

We shall compute separately the integral inside the curly brackets

$$\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{\sigma^2(\omega-\omega_1)^2}{2}-\mathrm{j}\omega(t-t_1)}\mathrm{d}\omega =$$

$$\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{\sigma\omega}{\sqrt{2}}\right)^2-\left(\frac{\sigma\omega_1}{\sqrt{2}}\right)^2+\sigma^2\omega\omega_1-\mathrm{j}\omega(t-t_1)}\mathrm{d}\omega =$$

$$e^{-\left(\frac{\sigma\omega_1}{\sqrt{2}}\right)^2}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{\sigma\omega}{\sqrt{2}}\right)^2+\omega(\sigma^2\omega_1-\mathrm{j}(t-t_1))}\mathrm{d}\omega =$$

(Continued)

Example B4.21 (Continued)

$$
\begin{aligned}
& e^{-\left(\frac{\sigma\omega_1}{\sqrt{2}}\right)^2} \int_{-\infty}^{\infty} e^{-\left(\frac{\sigma\omega}{\sqrt{2}}\right)^2 + 2\left(\frac{\sigma\omega}{\sqrt{2}}\right)\frac{1}{2}\frac{\sqrt{2}}{\sigma}(\sigma^2\omega_1 - j(t-t_1))} d\omega = \\
& e^{-\left(\frac{\sigma\omega_1}{\sqrt{2}}\right)^2} \int_{-\infty}^{\infty} e^{-\left(\frac{\sigma\omega}{\sqrt{2}}\right)^2 + 2\left(\frac{\sigma\omega}{\sqrt{2}}\right)\left(\frac{\sigma\omega_1}{\sqrt{2}} - j\frac{(t-t_1)}{\sqrt{2}\sigma}\right) - \left(\frac{\sigma\omega_1}{\sqrt{2}} - j\frac{(t-t_1)}{\sqrt{2}\sigma}\right)^2 + \left(\frac{\sigma\omega_1}{\sqrt{2}} - j\frac{(t-t_1)}{\sqrt{2}\sigma}\right)^2} = \\
& e^{-\left(\frac{\sigma\omega_1}{\sqrt{2}}\right)^2} e^{\left(\frac{\sigma\omega_1}{\sqrt{2}} - j\frac{(t-t_1)}{\sqrt{2}\sigma}\right)^2} \int_{-\infty}^{\infty} e^{-\left(\frac{\sigma\omega}{\sqrt{2}} - \frac{\sigma\omega_1}{\sqrt{2}} + j\frac{(t-t_1)}{\sqrt{2}\sigma}\right)^2} = \\
& e^{-\left(\frac{\sigma\omega_1}{\sqrt{2}}\right)^2} e^{\left(\frac{\sigma\omega_1}{\sqrt{2}} - j\frac{(t-t_1)}{\sqrt{2}\sigma}\right)^2} \frac{\sqrt{2}}{\sigma} \int_{-\infty}^{\infty} e^{-x^2} dx = \\
& e^{-\left(\frac{\sigma\omega_1}{\sqrt{2}}\right)^2} e^{\left(\frac{\sigma\omega_1}{\sqrt{2}}\right)^2} e^{-\left(\frac{(t-t_1)}{\sqrt{2}\sigma}\right)^2} e^{-2j\frac{\sigma\omega_1}{\sqrt{2}}\frac{(t-t_1)}{\sqrt{2}\sigma}} \frac{\sqrt{2}}{\sigma}\sqrt{\pi} = \\
& e^{-\frac{(t-t_1)^2}{2\sigma^2}} e^{-j\omega_1(t-t_1)} \frac{\sqrt{2\pi}}{\sigma}
\end{aligned}
\tag{4.95}
$$

where we made the substitution $x \equiv \frac{\sigma\omega}{\sqrt{2}} - \frac{\sigma\omega_1}{\sqrt{2}} + j\frac{(t-t_1)}{\sqrt{2}\sigma}$ *and used the result of Example (3.36). Upon substitution in (4.94) we obtain:*

$$
\begin{aligned}
b(t_1, \omega_1) &= \frac{\sigma}{\sqrt{2\pi}} e^{-j\omega_1 t_1} \int_{-\infty}^{\infty} g(t) e^{-\frac{(t-t_1)^2}{2\sigma^2}} e^{-j\omega_1(t-t_1)} \frac{\sqrt{2\pi}}{\sigma} dt \\
&= \int_{-\infty}^{\infty} g(t) e^{-\frac{(t-t_1)^2}{2\sigma^2}} e^{-j\omega_1 t} dt \\
&= a(t_1, \omega_1).
\end{aligned}
\tag{4.96}
$$

The last result was produced by direct comparison with (4.91).

Example B4.22

Show that the set of $G(t; t_0 = t_1, \omega_0 = \omega_1, \sigma)$ **functions is not an orthogonal set.**

A set of functions is orthogonal if the projection of one of them onto any other is zero. In the discrete domain, projection means dot product between the two signals (vectors), while in the continuous domain projection means the integral of the product of the two functions. When the functions are complex, the product must be computed between one function and the complex conjugate of the other function. This is so because the projection of the imaginary unit function on itself must be 1 ($jj^* = j(-j) = 1$)*. To show that the set of* $G(t; t_0 = t_1, \omega_0 = \omega_1, \sigma)$ *functions is not an orthogonal set, we must show that for* $t_1 \neq t_2$ *and* $\omega_1 \neq \omega_2$*:*

$$
\int_{-\infty}^{\infty} G(t; t_0 = t_1, \omega_0 = \omega_1, \sigma) G^*(t; t_0 = t_2, \omega_0 = \omega_2, \sigma) dt \neq 0. \tag{4.97}
$$

We start by trying to compute this integral, which for convenience we call $I(t_1, t_2, \omega_1, \omega_2, \sigma)$*:*

$$
\begin{aligned}
& I(t_1, t_2, \omega_1, \omega_2, \sigma) = \\
& \int_{-\infty}^{\infty} e^{-\frac{(t-t_1)^2}{2\sigma^2} + j\omega_1 t} e^{-\frac{(t-t_2)^2}{2\sigma^2} - j\omega_2 t} dt = \\
& \int_{-\infty}^{\infty} e^{-\frac{(t-t_1)^2 + (t-t_2)^2}{2\sigma^2} + j(\omega_1 - \omega_2)t} dt = \\
& \int_{-\infty}^{\infty} e^{-\frac{2t^2 - 2(t_1+t_2)t + t_1^2 + t_2^2}{2\sigma^2} + j(\omega_1 - \omega_2)t} dt.
\end{aligned}
\tag{4.98}
$$

After we re-arrange the terms in the exponent:

$$I(t_1,t_2,\omega_1,\omega_2,\sigma)=\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{t^2}{\sigma^2}+2t\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma^2}-\frac{t_1^2+t_2^2}{2\sigma^2}}\mathrm{d}t=$$
$$\mathrm{e}^{-\frac{t_1^2+t_2^2}{2\sigma^2}}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{t}{\sigma}\right)^2+2\left(\frac{t}{\sigma}\right)\left(\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma}\right)}\mathrm{d}t \tag{4.99}$$

we add and subtract from the exponent $(t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2)^2/(2\sigma)^2$*:*

$$I(t_1,t_2,\omega_1,\omega_2,\sigma)=$$
$$\mathrm{e}^{-\frac{t_1^2+t_2^2}{2\sigma^2}}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{t}{\sigma}\right)^2+2\left(\frac{t}{\sigma}\right)\left(\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma}\right)}\times$$
$$\mathrm{e}^{-\left(\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma}\right)^2+\left(\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma}\right)^2}\mathrm{d}t=$$

$$\mathrm{e}^{-\frac{t_1^2+t_2^2}{2\sigma^2}}\mathrm{e}^{\left(\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma}\right)^2}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\frac{t}{\sigma}-\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma}\right)^2}\mathrm{d}t. \tag{4.100}$$

We make the substitution $x\equiv\frac{t}{\sigma}-\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma}$*:*

$$I(t_1,t_2,\omega_1,\omega_2,\sigma)=\mathrm{e}^{-\frac{t_1^2+t_2^2}{2\sigma^2}}\mathrm{e}^{\left(\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma}\right)^2}\sigma\int_{-\infty}^{\infty}\mathrm{e}^{-x^2}\mathrm{d}x. \tag{4.101}$$

We use the result of Example 3.36 to obtain:

$$I(t_1,t_2,\omega_1,\omega_2,\sigma)=\mathrm{e}^{-\frac{t_1^2+t_2^2}{2\sigma^2}}\mathrm{e}^{\left(\frac{t_1+t_2+\mathrm{j}\sigma^2\omega_1-\mathrm{j}\sigma^2\omega_2}{2\sigma}\right)^2}\sigma\sqrt{\pi}. \tag{4.102}$$

The last expression is not equal to zero, so the basis consisting of the elementary functions $G(t;t_0=t_1,\omega_0=\omega_1,\sigma)$ *is not orthogonal.*

How can we use the Gabor functions in practice?

In practice, we analyse a signal in terms of Gabor functions by taking the Fourier transform of the signal, multiplying it with a Gaussian window centred at various frequencies, and taking the inverse Fourier transform of the result. We make the choice of the central frequency of each Gaussian in a systematic way, so that all frequency bands of the signal are covered. This way we obtain several versions of the original signal. Each version expresses the frequency content of the original signal at a particular band. Thus, a single sample of the original signal is now characterised by as many numbers as bands we considered, i.e. as many numbers as the number of different Gaussians we used.

Example 4.23

Expand in terms of Gabor functions the signal of Example 4.14.

First we compute the Fourier transform of the signal

$$G(k)=\sum_{n=0}^{T}g(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi nk}{T+1}} \tag{4.103}$$

where k takes values from $-(T+1)/2$ *to* $(T-1)/2$*.*

(Continued)

Example 4.23 (Continued)

For $T = 127$, k takes values from -64 to 63 for this signal. At the top of Figure 4.20a we plot the real and imaginary parts of $G(k)$ as functions of k.

Let us say that the Gaussian window we shall use in the frequency domain is given by

$$W_{\rm I}(k) = {\rm e}^{-\frac{(k-k_{\rm I})^2}{2\Sigma_k^2}} \tag{4.104}$$

where $k_{\rm I}$ is the central index of each window. Let us say that we shall place the windows $2\Sigma_k$ apart. This way, at the places of transition between different bands, i.e. when $k = k_{\rm I} \pm \Sigma_k$, the value of each window is $\exp\left(-\frac{\Sigma_k^2}{2\Sigma_k^2}\right) = {\rm e}^{-0.5}$, i.e. it is equal to 0.607 $(= {\rm e}^{-0.5})$ times the value of the window at the central point $k_{\rm I}$.

Note that the width of the bands we choose is $2\Sigma_k$ which is different from the bandwidth of the Gaussian we use. The latter, according to Equation (4.42), is $\Delta\omega = 1/(2\sigma^2)$, with $\sigma = 1/\Sigma$ and $\Sigma = \Sigma_k(T+1)/(4\pi^2)$, since Σ is the standard deviation when the independent variable is ω and Σ_k is the standard deviation when the independent variable is just k. (Remember that $\omega = 2\pi k/(T+1)$.) If the bands we choose had widths equal to the bandwidth of the corresponding Gaussians, there would have been significant overlap between neighbouring bands, preventing the discrimination of the different frequencies that may be present in the signal.

When we multiply the Fourier transform of the original signal with any one of these windows and take the inverse Fourier transform, we wish to obtain a real signal. This may only be achieved if the inverse transform we compute is the inverse transform of a symmetric function. So, each Gaussian window is placed at symmetric frequencies about $k = 0$, i.e. centred at positions $k_{\rm I}$ and $-k_{\rm I}$.

Figure 4.20a shows all these windows for $\Sigma_k = 9$. The centres of the windows are at $k_0 = 0$, $k_1 = 18$, $k_2 = 36$ and $k_3 = 54$.

Next we multiply the Fourier transform of the original signal with each one of the four windows. The functions produced that way are shown in Figure 4.20b. Then we take the inverse Fourier transform of each one of these functions using

$$g_{\rm I}(n) = \frac{1}{128}\sum_{k=-64}^{63} w_{\rm I}(k){\rm e}^{{\rm j}\frac{2\pi nk}{128}} \quad \text{for } n = 0, \ldots, 127 \tag{4.105}$$

where $w_{\rm I}(k) = G(k)W_{\rm I}(k)$.

These inverse functions are plotted in Figure 4.21. Each one of these functions tells us how much of each frequency band is present at each position.

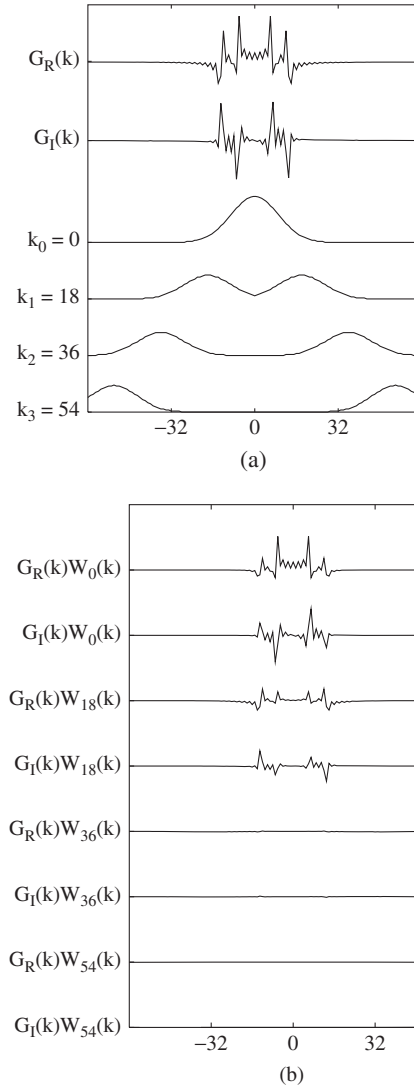


Figure 4.20 (a) At the top the real and imaginary parts of $G(k)$, the Fourier transform of the original signal. Below them, the four symmetric Gaussian windows to be used for windowing the Fourier transform of the signal. (b) The real and imaginary parts of $G(k)$ multiplied with the four Gaussian windows, plotted pairwise one under the other.

Example 4.24

Segment with the help of Gabor functions the signal of Example 4.14.

The original signal is shown at the top of Figure 4.11. To produce features out of its Gabor representation, shown in Figure 4.21, we compute the local average energy of each sequence of Figure 4.21, either by using a Gaussian window, or by using a rectangular window. Figure 4.22 shows the features produced when the values of every 11 successive samples are squared and averaged and assigned to the central sample. Figure 4.23 shows the features produced when the squared signals of Figure 4.21 are convolved with a truncated Gaussian window of standard deviation $\sigma = 5.5$. This value of σ was chosen so that the Gaussian window was roughly equal in width to the uniform window ($2\sigma = 11$). The size of the window was chosen so that when the Gaussian was truncated, the discontinuity created was about 0.1 of its peak value:

$$e^{-\frac{x^2}{2\sigma^2}} = 0.1 \Rightarrow -\frac{x^2}{2\sigma^2} = \ln 0.1 \Rightarrow x = \sqrt{-2\sigma^2 \ln 0.1} \simeq 11.8. \tag{4.106}$$

So, this window was chosen to be 25 samples wide, defined for indices in the range $[-12, 12]$.

We can see that if we threshold the features plotted either in Figure 4.22 or in Figure 4.23 we shall segment the signal.

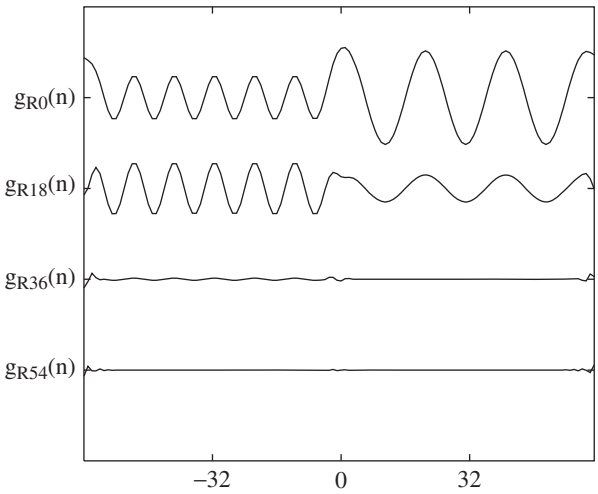


Figure 4.21 Inverse Fourier transform of all windowed Fourier transforms of the signal (functions $w_I(k)$ of Equation (4.105)). In all cases only the real part is shown as the imaginary part is zero.

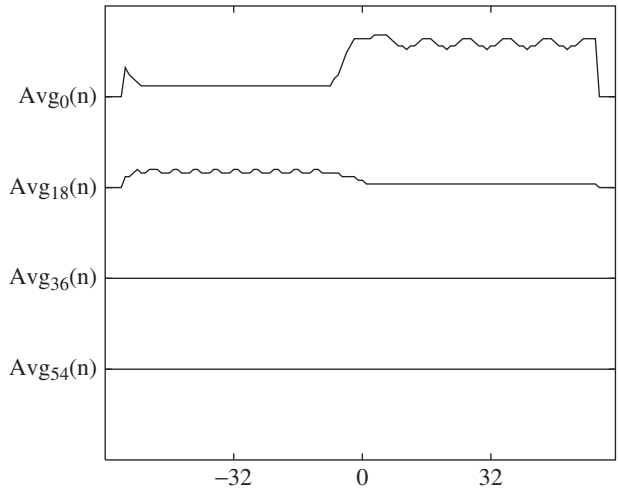


Figure 4.22 Gabor features produced by squaring and convolving the functions plotted in Figure 4.21 with a rectangular window of size 11.

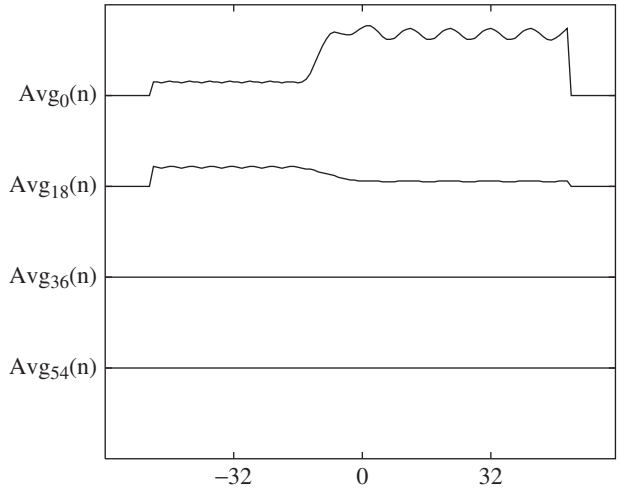


Figure 4.23 Gabor features produced by squaring and convolving the functions plotted in Figure 4.21 with a Gaussian window of size 25 and $\sigma = 5.5$.

How is a Gabor function generalised in 2D?

A 2D Gabor function is defined as:

$$G(x,y;x_0,\omega_{x_0},\sigma_x,y_0,\omega_{y_0},\sigma_y) \equiv e^{-\frac{(x-x_0)^2}{2\sigma_x^2}-\frac{(y-y_0)^2}{2\sigma_y^2}} e^{j\omega_{x_0}x+j\omega_{y_0}y}. \tag{4.107}$$

The Fourier transform of this function is:

$$\hat{G}(\omega_x,\omega_y;x_0,\omega_{x_0},\sigma_x,y_0,\omega_{y_0},\sigma_y) = 2\pi\sigma_x\sigma_y e^{-\frac{\sigma_x^2(\omega_x-\omega_{x_0})^2}{2}-\frac{\sigma_y^2(\omega_y-\omega_{y_0})^2}{2}} e^{-j(\omega_y-\omega_{y_0})y_0-j(\omega_x-\omega_{x_0})x_0}. \tag{4.108}$$

Example B4.25

Prove that (4.108) is the Fourier transform of function (4.107).

TheFourier transform of function (4.107) is given by:

$$\begin{aligned}
&\hat{G}(\omega_x,\omega_y;x_0,\omega_{x_0},\sigma_x,y_0,\omega_{y_0},\sigma_y) \equiv \\
&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(x,y;x_0,\omega_{x_0},\sigma_x,y_0,\omega_{y_0},\sigma_y)e^{-j\omega_x x-j\omega_y y}dxdy = \\
&\int_{-\infty}^{\infty} e^{-\frac{(x-x_0)^2}{2\sigma_x^2}} e^{j\omega_{x_0}x}e^{-j\omega_x x}dx \int_{-\infty}^{\infty} e^{-\frac{(y-y_0)^2}{2\sigma_y^2}} e^{j\omega_{y_0}y}e^{-j\omega_y y}dy = \\
&\int_{-\infty}^{\infty} e^{-\frac{(x-x_0)^2}{2\sigma_x^2}-j(\omega_x-\omega_{x_0})(x-x_0)}dx e^{-j(\omega_x-\omega_{x_0})x_0}\times \\
&\int_{-\infty}^{\infty} e^{-\frac{(y-y_0)^2}{2\sigma_y^2}-j(\omega_y-\omega_{y_0})(y-y_0)}dy e^{-j(\omega_y-\omega_{y_0})y_0}.
\end{aligned} \tag{4.109}$$

Let us use the auxiliary variables $\tilde{\omega}_x \equiv \omega_x - \omega_{x_0}$, $\tilde{\omega}_y \equiv \omega_y - \omega_{y_0}$, $\tilde{x} \equiv x - x_0$ and $\tilde{y} \equiv y - y_0$. Then:

$$\begin{aligned}
&\hat{G}(\omega_x,\omega_y;x_0,\omega_{x_0},\sigma_x,y_0,\omega_{y_0},\sigma_y) = \\
&e^{-j\tilde{\omega}_x x_0}e^{-j\tilde{\omega}_y y_0}\int_{-\infty}^{\infty} e^{-\frac{\tilde{x}^2}{2\sigma_x^2}-j\tilde{\omega}_x\tilde{x}}d\tilde{x}\int_{-\infty}^{\infty} e^{-\frac{\tilde{y}^2}{2\sigma_y^2}-j\tilde{\omega}_y\tilde{y}}d\tilde{y} = \\
&e^{-j\tilde{\omega}_x x_0}e^{-j\tilde{\omega}_y y_0}\sqrt{2\pi}\sigma_x e^{-\frac{\tilde{\omega}_x^2\sigma_x^2}{2}}\sqrt{2\pi}\sigma_y e^{-\frac{\tilde{\omega}_y^2\sigma_y^2}{2}} = \\
&2\pi\sigma_x\sigma_y e^{-j(\omega_x-\omega_{x_0})x_0}e^{-j(\omega_y-\omega_{y_0})y_0}e^{-\frac{(\omega_x-\omega_{x_0})^2\sigma_x^2}{2}}e^{-\frac{(\omega_y-\omega_{y_0})^2\sigma_y^2}{2}}.
\end{aligned} \tag{4.110}$$

This expression after re-arrangement is exactly the right-hand side of Equation (4.108). In this calculation we made use of the result of computing the integral of Equation (3.303), i.e. Equation (3.311), since each one of the two integrals we have to compute is identical to function $\hat{G}(\omega)$ of Example 3.99.

Example B4.26

Show that the cross-dispersion of the Gabor function defined by Equation (4.107) is zero.

First we compute the energy of the function

$$\begin{aligned}
E_\infty &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(x,y;x_0,\omega_{x_0},\sigma_x,y_0,\omega_{y_0},\sigma_y)G^*(x,y;x_0,\omega_{x_0},\sigma_x,y_0,\omega_{y_0},\sigma_y)\mathrm{d}x\mathrm{d}y \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}-\frac{(y-y_0)^2}{\sigma_y^2}}\mathrm{d}x\mathrm{d}y \\
&= \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}}\mathrm{d}x \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(y-y_0)^2}{\sigma_y^2}}\mathrm{d}y \\
&= \sigma_x\sigma_y \int_{-\infty}^{\infty} \mathrm{e}^{-\tilde{x}^2}\mathrm{d}\tilde{x} \int_{-\infty}^{\infty} \mathrm{e}^{-\tilde{y}^2}\mathrm{d}\tilde{y} \\
&= \sigma_x\sigma_y\pi
\end{aligned} \tag{4.111}$$

where we used new variables of integration $\tilde{x}$ and $\tilde{y}$ defined as $\tilde{x} \equiv (x-x_0)/\sigma_x$ and $\tilde{y} \equiv (y-y_0)/\sigma_y$, respectively, and made use of the result of Example 3.36. The coordinates $(\bar{x},\bar{y})$ of the centre of gravity of the 2D Gabor function are computed as

$$\begin{aligned}
\bar{x} &= \frac{1}{\sigma_x\sigma_y\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}-\frac{(y-y_0)^2}{\sigma_y^2}}\mathrm{d}x\mathrm{d}y \\
&= \frac{1}{\sigma_x\sigma_y\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-x_0)\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}-\frac{(y-y_0)^2}{\sigma_y^2}}\mathrm{d}x\mathrm{d}y + \\
&\quad \frac{x_0}{\sigma_x\sigma_y\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}-\frac{(y-y_0)^2}{\sigma_y^2}}\mathrm{d}x\mathrm{d}y \\
&= \frac{1}{\sigma_x\sigma_y\pi}\sigma_x^2\int_{-\infty}^{\infty}\frac{x-x_0}{\sigma_x}\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}}\mathrm{d}\left(\frac{x-x_0}{\sigma_x}\right)\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{(y-y_0)^2}{\sigma_y^2}}\mathrm{d}y + \\
&\quad \frac{1}{\sigma_x\sigma_y\pi}\sigma_x\sigma_y x_0\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}}\mathrm{d}\left(\frac{x-x_0}{\sigma_x}\right)\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{(y-y_0)^2}{\sigma_y^2}}\mathrm{d}\left(\frac{y-y_0}{\sigma_y}\right) \\
&= \frac{\sigma_x}{\sigma_y\pi}\int_{-\infty}^{\infty}\tilde{x}\mathrm{e}^{-\tilde{x}^2}\mathrm{d}\tilde{x}\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{(y-y_0)^2}{\sigma_y^2}}\mathrm{d}y + \\
&\quad \frac{1}{\pi}x_0\int_{-\infty}^{\infty}\mathrm{e}^{-\tilde{x}^2}\mathrm{d}\tilde{x}\int_{-\infty}^{\infty}\mathrm{e}^{-\tilde{y}^2}\mathrm{d}\tilde{y} \\
&= 0 + \frac{1}{\pi}x_0\sqrt{\pi}\sqrt{\pi} = x_0
\end{aligned} \tag{4.112}$$

where we again used the same substitution for the variables of integration, and made use of the results of Examples 3.36 and 3.37, and the fact that $\int_{-\infty}^{\infty}\tilde{x}\mathrm{e}^{-\tilde{x}^2}\mathrm{d}\tilde{x} = 0$ because this is the integral of an antisymmetric function over a symmetric interval of integration.

(Continued)

Example B4.26 (Continued)

Similarly, we can show that $\bar{y} = y_0$. *Then the cross-dispersion of the Gabor function is given by*

$$\begin{aligned}\Delta x\Delta y &= \frac{1}{\sigma_x\sigma_y\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-x_0)(y-y_0)\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}-\frac{(y-y_0)^2}{\sigma_y^2}}\,\mathrm{d}x\mathrm{d}y\\ &= \frac{1}{\sigma_x\sigma_y\pi}\sigma_x^2\sigma_y^2\int_{-\infty}^{\infty}\frac{x-x_0}{\sigma_x}\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}}\,\mathrm{d}\left(\frac{x-x_0}{\sigma_x}\right)\times\\ &\quad\int_{-\infty}^{\infty}\frac{y-y_0}{\sigma_y}\mathrm{e}^{-\frac{(y-y_0)^2}{\sigma_y^2}}\,\mathrm{d}\left(\frac{y-y_0}{\sigma_y}\right)\\ &= \frac{\sigma_x\sigma_y}{\pi}\int_{-\infty}^{\infty}\tilde{x}\mathrm{e}^{-\tilde{x}^2}\mathrm{d}\tilde{x}\int_{-\infty}^{\infty}\tilde{y}\mathrm{e}^{-\tilde{y}^2}\mathrm{d}\tilde{y}=0\end{aligned}\tag{4.113}$$

since $\int_{-\infty}^{\infty}\tilde{x}\mathrm{e}^{-\tilde{x}^2}d\tilde{x} = \int_{-\infty}^{\infty}\tilde{y}\mathrm{e}^{-\tilde{y}^2}d\tilde{y} = 0$.

So, the way Gabor function is defined by Equation (4.107) implies that axes x and y are the principal axes of the function.

Example B4.27

Derive the dispersion of the Gabor function defined by Equation (4.107) along the x **and** y **axes.**

Making use of the calculation of E_∞ *and* $\bar{x}$ *and* $\bar{y}$ *in Example 4.26, we proceed to compute the first two dispersions defined by Equation (4.68). We shall make use of the auxiliary variables* $\tilde{x}$ *and* $\tilde{y}$, *defined here as* $\tilde{x} \equiv (x-x_0)/\sigma_x$ *and* $\tilde{y} \equiv (y-y_0)/\sigma_y$:

$$\begin{aligned}(\Delta x)^2 &= \frac{1}{\sigma_x\sigma_y\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-x_0)^2\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}-\frac{(y-y_0)^2}{\sigma_y^2}}\,\mathrm{d}x\mathrm{d}y\\ &= \frac{1}{\sigma_x\sigma_y\pi}\sigma_x^3\sigma_y\int_{-\infty}^{\infty}\frac{(x-x_0)^2}{\sigma_x^2}\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}}\,\mathrm{d}\left(\frac{x-x_0}{\sigma_x}\right)\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{(y-y_0)^2}{\sigma_y^2}}\,\mathrm{d}\left(\frac{y-y_0}{\sigma_y}\right)\\ &= \frac{\sigma_x^2}{\pi}\int_{-\infty}^{\infty}\tilde{x}^2\mathrm{e}^{-\tilde{x}^2}\mathrm{d}\tilde{x}\int_{-\infty}^{\infty}\mathrm{e}^{-\tilde{y}^2}\mathrm{d}\tilde{y}\\ &= \frac{\sigma_x^2}{\pi}\frac{\sqrt{\pi}}{2}\sqrt{\pi} = \frac{\sigma_x^2}{2}.\end{aligned}\tag{4.114}$$

Here we used $\int_{-\infty}^{\infty}\tilde{x}^2\mathrm{e}^{-\tilde{x}^2}\mathrm{d}\tilde{x} = \sqrt{\pi}/2$ *from Example 3.37 and* $\int_{-\infty}^{\infty}\mathrm{e}^{-\tilde{y}^2}\mathrm{d}\tilde{y} = \sqrt{\pi}$ *from Example 3.36.*

Similarly:

$$\begin{aligned}(\Delta y)^2 &= \frac{1}{\sigma_x\sigma_y\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(y-y_0)^2\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}-\frac{(y-y_0)^2}{\sigma_y^2}}\,\mathrm{d}x\mathrm{d}y\\ &= \frac{1}{\sigma_x\sigma_y\pi}\sigma_x\sigma_y^3\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{(x-x_0)^2}{\sigma_x^2}}\,\mathrm{d}\left(\frac{x-x_0}{\sigma_x}\right)\int_{-\infty}^{\infty}\frac{(y-y_0)^2}{\sigma_y^2}\mathrm{e}^{-\frac{(y-y_0)^2}{\sigma_y^2}}\,\mathrm{d}\left(\frac{y-y_0}{\sigma_y}\right)\\ &= \frac{\sigma_y^2}{\pi}\int_{-\infty}^{\infty}\mathrm{e}^{-\tilde{x}^2}\mathrm{d}\tilde{x}\int_{-\infty}^{\infty}\tilde{y}^2\mathrm{e}^{-\tilde{y}^2}\mathrm{d}\tilde{y}\\ &= \frac{\sigma_y^2}{\pi}\frac{\sqrt{\pi}}{2}\sqrt{\pi} = \frac{\sigma_y^2}{2}.\end{aligned}\tag{4.115}$$

Therefore:

$$\Delta x = \frac{\sigma_x}{\sqrt{2}} \qquad \Delta y = \frac{\sigma_y}{\sqrt{2}}. \tag{4.116}$$

Example B4.28

Derive the spectral bandwidth of the Gabor function defined by Equation (4.108) along the axes ω_x and ω_y.

We use definitions (4.71) and (4.108) to compute $(\Delta\omega_x)^2$ and $(\Delta\omega_y)^2$, making also use of the result $E_\infty = \pi\sigma_x\sigma_y$:

$$\begin{aligned}
(\Delta\omega_x)^2 &= \frac{4\pi^2\sigma_x^2\sigma_y^2}{4\pi^3\sigma_x\sigma_y}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\omega_x-\omega_{x_0})^2 e^{-\sigma_x^2(\omega_x-\omega_{x_0})^2-\sigma_y^2(\omega_y-\omega_{y_0})^2}d\omega_x d\omega_y \\
&= \frac{\sigma_x\sigma_y}{\pi}\int_{-\infty}^{\infty}(\omega_x-\omega_{x_0})^2 e^{-\sigma_x^2(\omega_x-\omega_{x_0})^2}d\omega_x \int_{-\infty}^{\infty} e^{-\sigma_y^2(\omega_y-\omega_{y_0})^2}d\omega_y \\
&= \frac{\sigma_x\sigma_y}{\pi}\frac{1}{\sigma_x^3\sigma_y}\int_{-\infty}^{\infty}\sigma_x^2(\omega_x-\omega_{x_0})^2 e^{-\sigma_x^2(\omega_x-\omega_{x_0})^2}d(\sigma_x(\omega_x-\omega_{x_0}))\times \\
&\quad \int_{-\infty}^{-\infty} e^{-\sigma_y^2(\omega_y-\omega_{y_0})^2}d(\sigma_y(\omega_y-\omega_{y_0})) \\
&= \frac{1}{\pi\sigma_x^2}\int_{-\infty}^{-\infty}\tilde{\omega}_x^2 e^{-\tilde{\omega}_x^2}d\tilde{\omega}_x \int_{-\infty}^{-\infty} e^{-\tilde{\omega}_y^2}d\tilde{\omega}_y \\
&= \frac{1}{\pi\sigma_x^2}\frac{\sqrt{\pi}}{2}\sqrt{\pi} = \frac{1}{2\sigma_x^2}.
\end{aligned} \tag{4.117}$$

Here we used auxiliary variables $\tilde{\omega}_x \equiv \sigma_x(\omega_x - \omega_{x_0})$ and $\tilde{\omega}_y \equiv \sigma_y(\omega_y - \omega_{y_0})$, and the results of Examples 3.36 and 3.37.

Similarly, we may compute:

$$(\Delta\omega_y)^2 = \frac{1}{2\sigma_y^2}. \tag{4.118}$$

Therefore:

$$\Delta\omega_x = \frac{1}{\sqrt{2}\sigma_x} \qquad \Delta\omega_y = \frac{1}{\sqrt{2}\sigma_y}. \tag{4.119}$$

Example B4.29

Prove that the 2D Gabor function as defined by Equations (4.107) and (4.108) satisfies the uncertainty principle with an equality.

If we substitute from Equations (4.116) and (4.119) into equation (4.74), we obtain the equality. Therefore, the 2D Gabor function has the minimum uncertainty in specifying what is where in an image.

How may we use the 2D Gabor functions to analyse an image?

To analyse an image into Gabor functions, we proceed as in the 1D case. We take the DFT of the image, we multiply it with the appropriate Gaussian window, and take the inverse Fourier transform of the result. To make sure we consider all possible frequencies that might be present in the image, we tessellate the frequency space into rectangular bands and place Gaussian windows in all of them in turn.

A possible tessellation of the 2D frequency domain is shown in Figure 4.24. At the places marked by crosses (the centres of the frequency bands/tiles) we place the centres of the windows. Then we create Gaussian envelopes centred at each band using the magnitude of expression (4.108). We use each one of them to multiply the Fourier transform of the whole image (which occupies the whole frequency plane) point by point. This process effectively kills all frequencies outside the chosen frequency band, and keeps the frequencies inside the chosen band. We then Fourier transform back this product. The result is an "image" that contains only frequencies in the chosen band. This "image", therefore, tells us which parts of the original image contain frequencies in this band. (The parts which do not contain such frequencies will be very dark and structureless.) We repeat this process for all frequency bands we have created. So, at the end we have a stack of "images". For each pixel now we have as many values as frequency bands we used, one value per output "image". The set of these values is the "frequency signature" of the pixel. We can obtain texture segmentation by using clustering in an N-dimensional space, where N is the number of bands we used. However, these values are often too oscillatory to constitute good local texture features. That is why we often consider around each pixel a small local window and inside that window

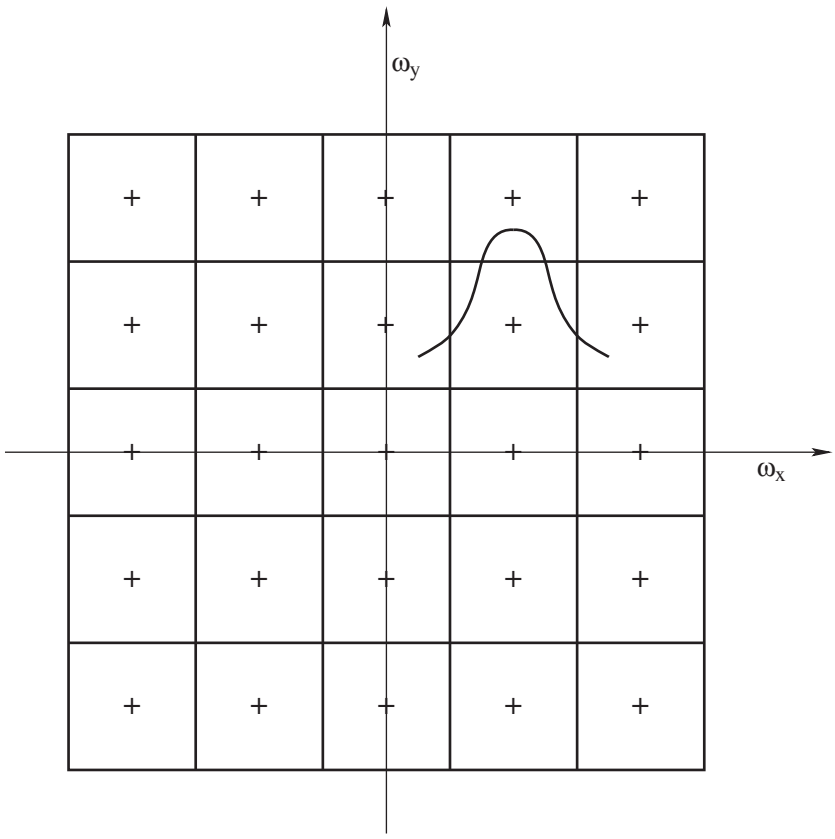


Figure 4.24 A schematic tessellation of the 2D frequency domain. The crosses mark the centres of the frequency bands where we may place the 2D Gaussian windows.

we compute the "energy" of the "image". This is nothing other than the sum of the squares of the "image" values, but according to Parseval's theorem, it represents the local image energy inside the relevant frequency band. It is these local energies that are considered as "frequency signatures" of the pixels and are used for performing pixel by pixel classification and thus image segmentation.

Note that in order for the "image" obtained by the inverse Fourier transform to be real, its Fourier transform must be symmetric about the origin of the axes. This means that when we choose a band specified by spatial frequencies ω_x and ω_y such that $\omega_1 < \omega_x < \omega_2$ and $\omega_3 < \omega_y < \omega_4$, we must also include the band where $-\omega_1 > \omega_x > -\omega_2$ and $-\omega_3 > \omega_y > -\omega_4$, on which a window function should also be placed. (Here we have assumed $\omega_1, \omega_2, \omega_3, \omega_4 > 0$.)

The whole process is schematically shown in Figure 4.25.

In practice, however, we often use tessellations of the frequency space different from that of Figure 4.24.

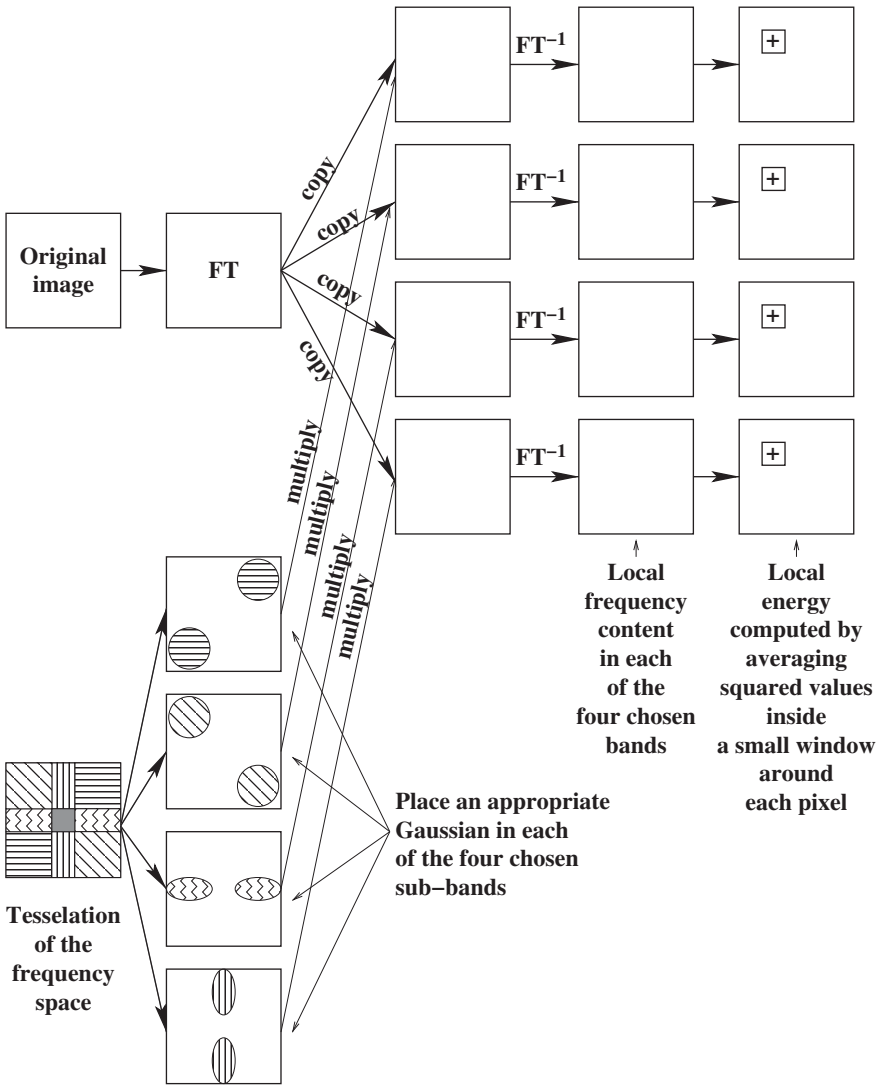


Figure 4.25 A schematic representation of the analysis of an image into contents of its frequency bands. Here we assume that we have chosen four pairs of frequency bands. In each band we have placed a window. We multiply the Fourier transform of the image with each one of the windows in turn, and Fourier transform the result back to obtain an "image" that contains only frequencies in the chosen band. Note: we use pairs of bands symmetric about the origin in order to have real outputs.

Example B4.30

Prove that the DFT $F(k,l)$ of an image $f(m,n)$ of size $M\times N$, where both M and N are odd, and $m\in[0,M-1]$, $n\in[0,N-1]$, does not change if we shift the image so that its indices take values in the range $\left[-\frac{M-1}{2},\frac{M-1}{2}\right]$ and $\left[-\frac{N-1}{2},\frac{N-1}{2}\right]$ respectively.

We start from the conventional definition of DFT

$$F(k,l)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)\mathrm{e}^{-2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)} \tag{4.120}$$

where $k\in[0,M-1]$ and $l\in[0,N-1]$. Let us split the double sum as follows:

$$F(k,l)=\sum_{m=0}^{\frac{M-1}{2}}\sum_{n=0}^{N-1}f(m,n)\mathrm{e}^{-2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)}+\sum_{\frac{M+1}{2}}^{M-1}\sum_{n=0}^{N-1}f(m,n)\mathrm{e}^{-2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)}. \tag{4.121}$$

In the second sum we define some new variable of summation $\tilde{m}$, such that:

$$\tilde{m}=m-M\Rightarrow m=\tilde{m}+M\quad\text{and}\quad\left[\frac{M+1}{2},M-1\right]\to\left[-\frac{M-1}{2},-1\right]. \tag{4.122}$$

Then:

$$F(k,l)=\sum_{m=0}^{\frac{M-1}{2}}\sum_{n=0}^{N-1}f(m,n)\mathrm{e}^{-2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)}+\sum_{\tilde{m}=-\frac{M-1}{2}}^{-1}\sum_{n=0}^{N-1}f(\tilde{m}+M,n)\mathrm{e}^{-2\pi\mathrm{j}\left(\frac{k\tilde{m}}{M}+k+\frac{ln}{N}\right)}. \tag{4.123}$$

We remember that DFT assumes that an image is repeated periodically in all directions. So, we may say that $f(\tilde{m}+M,n)=f(\tilde{m},n)$. Also, $\mathrm{e}^{\pm\mathrm{j}(2\pi k+\phi)}=\mathrm{e}^{\pm\mathrm{j}\phi}$. So, the above expression may be written as

$$F(k,l)=\sum_{m=0}^{\frac{M-1}{2}}\sum_{n=0}^{N-1}f(m,n)\mathrm{e}^{-2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)}+\sum_{\tilde{m}=-\frac{M-1}{2}}^{-1}\sum_{n=0}^{N-1}f(\tilde{m},n)\mathrm{e}^{-2\pi\mathrm{j}\left(\frac{k\tilde{m}}{M}+\frac{ln}{N}\right)} \tag{4.124}$$

which upon replacing the dummy variable $\tilde{m}$ by m and combining the two sums may be written as:

$$F(k,l)=\sum_{m=-\frac{M-1}{2}}^{\frac{M-1}{2}}\sum_{n=0}^{N-1}f(m,n)\mathrm{e}^{-2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)}. \tag{4.125}$$

Next we define for the inner sum a new variable of summation $\tilde{n}$, such that

$$\tilde{n}=n-N\Rightarrow n=\tilde{n}+N\quad\text{and}\quad\left[\frac{N+1}{2},N-1\right]\to\left[-\frac{N-1}{2},-1\right] \tag{4.126}$$

and working as for the sum over m, we prove the conjecture:

$$F(k,l)=\sum_{m=-\frac{M-1}{2}}^{\frac{M-1}{2}}\sum_{n=-\frac{N-1}{2}}^{\frac{N-1}{2}}f(m,n)\mathrm{e}^{-2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)}\quad\text{for } k\in[0,M-1],l\in[0,N-1]. \tag{4.127}$$

Example B4.31

Prove that an image $f(m,n)$ of size $M \times N$, where both M and N are odd does not change, if its DFT $F(k,l)$ is written so that the dc component is in the middle of the axes.

Due to the duality property of the Fourier transform, $f(m,n)$ is the Fourier transform of $F(k,l)$. We saw in Example 4.30 that the Fourier transform of an image does not change if we shift the origin of the axes to the image centre. Therefore, $f(m,n)$ will not change if we shift the origin of the axes of $F(k,l)$ to its centre, i.e. we may write

$$\begin{aligned} f(m,n) &= \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} F(k,l)\mathrm{e}^{2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)} \\ &= \frac{1}{MN}\sum_{k=-\frac{M-1}{2}}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} F(k,l)\mathrm{e}^{2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)} \end{aligned} \tag{4.128}$$

where $m \in [0, M-1]$ and $n \in [0, N-1]$, or $m \in \left[-\frac{M-1}{2}, \frac{M-1}{2}\right]$ and $n \in \left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$.

Example B4.32

Prove that if $F(k,l)$ is the DFT of a real image $f(m,n)$ of size $M \times N$, where M and N are odd, then $|F(k,l)| = |F(-k,-l)|$.

From Equation (4.127), we have:

$$F(-k,-l) = \sum_{m=-\frac{M-1}{2}}^{\frac{M-1}{2}}\sum_{n=-\frac{N-1}{2}}^{\frac{N-1}{2}} f(m,n)\mathrm{e}^{2\pi\mathrm{j}\left(\frac{km}{M}+\frac{ln}{N}\right)} = F^*(k,l) \Rightarrow |F(k,l)| = |F(-k,-l)|. \tag{4.129}$$

Example B4.33

Show that if the DFT $F(k,l)$ of an image $f(m,n)$ of size $M \times N$, where M and N are odd, has the property $F(-k,-l) = F^*(k,l)$, then the image is real.

The inverse DFT of $f(m,n)$ according to Equation (4.128) may be written as:

$$\begin{aligned} f(m,n) = &\frac{1}{MN}\sum_{k=-\frac{M-1}{2}}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} F(k,l)\cos\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right] \\ &+\mathrm{j}\frac{1}{MN}\sum_{k=-\frac{M-1}{2}}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} F(k,l)\sin\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right]. \end{aligned} \tag{4.130}$$

(Continued)

Example B4.33 (Continued)

We must show that the imaginary part of this expression is zero. If $F(k,l) = F_R(k,l) + \mathrm{j}F_\mathrm{I}(k,l)$*, the imaginary part of* $f(m,n)$*, which we may call* $F_\mathrm{I}(m,n)$*, is:*

$$F_\mathrm{I}(m,n) = \frac{1}{MN}\sum_{k=-\frac{M-1}{2}}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} F_\mathrm{I}(k,l)\cos\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right] + \frac{1}{MN}\sum_{k=-\frac{M-1}{2}}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} F_R(k,l)\sin\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right]. \tag{4.131}$$

Note that in the above expression there is no term $k = l = 0$*, because* $\sin 0 = 0$ *and because relation* $F(-k,-l) = F^*(k,l)$ *implies that* $F(0,0)$ *must be real, i.e.* $F_\mathrm{I}(0,0) = 0$*. We split each double sum into four sums plus the terms with either* $k = 0$ *or* $l = 0$*, as follows:*

$$\begin{aligned}
F_\mathrm{I}(m,n) = {} & \left\{\sum_{k=-\frac{M-1}{2}}^{-1}\sum_{l=-\frac{N-1}{2}}^{-1} + \sum_{k=-\frac{M-1}{2}}^{-1}\sum_{l=1}^{\frac{N-1}{2}} + \sum_{k=1}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{-1} + \sum_{k=1}^{\frac{M-1}{2}}\sum_{l=1}^{\frac{N-1}{2}}\right\} \\
& F_\mathrm{I}(k,l)\cos\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right] \\
& + \sum_{k=-\frac{M-1}{2}}^{-1} F_\mathrm{I}(k,0)\cos\left[2\pi\frac{km}{M}\right] + \sum_{k=1}^{\frac{M-1}{2}} F_\mathrm{I}(k,0)\cos\left[2\pi\frac{km}{M}\right] \\
& + \sum_{l=-\frac{N-1}{2}}^{-1} F_\mathrm{I}(0,l)\cos\left[2\pi\frac{ln}{N}\right] + \sum_{l=1}^{\frac{N-1}{2}} F_\mathrm{I}(0,l)\cos\left[2\pi\frac{ln}{N}\right] + \\
& \left\{\sum_{k=-\frac{M-1}{2}}^{-1}\sum_{l=-\frac{N-1}{2}}^{-1} + \sum_{k=-\frac{M-1}{2}}^{-1}\sum_{l=1}^{\frac{N-1}{2}} + \sum_{k=1}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{-1} + \sum_{k=1}^{\frac{M-1}{2}}\sum_{l=1}^{\frac{N-1}{2}}\right\} \\
& F_R(k,l)\sin\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right] \\
& + \sum_{k=-\frac{M-1}{2}}^{-1} F_R(k,0)\sin\left[2\pi\frac{km}{M}\right] + \sum_{k=1}^{\frac{M-1}{2}} F_R(k,0)\sin\left[2\pi\frac{km}{M}\right] \\
& + \sum_{l=-\frac{N-1}{2}}^{-1} F_R(0,l)\sin\left[2\pi\frac{ln}{N}\right] + \sum_{l=1}^{\frac{N-1}{2}} F_R(0,l)\sin\left[2\pi\frac{ln}{N}\right].
\end{aligned} \tag{4.132}$$

There are 16 sums in this expression. Let us enumerate them sequentially, from left to right and top to bottom. In sums 1, 2, 5, 7, 9, 10, 13 and 15 we introduce new summation variables: $\tilde{k} = -k \Rightarrow k = -\tilde{k}$ *and* $\tilde{l} = -l \Rightarrow l = -\tilde{l}$*. Then:*

$$F_\mathrm{I}(m,n) = \left\{\sum_{\tilde{k}=\frac{M-1}{2}}^{1}\sum_{\tilde{l}=\frac{N-1}{2}}^{1} + \sum_{\tilde{k}=\frac{M-1}{2}}^{1}\sum_{\tilde{l}=-1}^{-\frac{N-1}{2}}\right\} F_\mathrm{I}(-\tilde{k},-\tilde{l})\cos\left[2\pi\left(-\frac{\tilde{k}m}{M}-\frac{\tilde{l}n}{N}\right)\right]$$

$$+\left\{\sum_{k=1}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{-1}+\sum_{k=0}^{\frac{M-1}{2}}\sum_{l=1}^{\frac{N-1}{2}}\right\}F_{\mathrm{I}}(k,l)\cos\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right]$$

$$+\sum_{\tilde{k}=\frac{M-1}{2}}^{1}F_{\mathrm{I}}(-\tilde{k},0)\cos\left[-2\pi\frac{\tilde{k}m}{M}\right]+\sum_{k=1}^{\frac{M-1}{2}}F_{\mathrm{I}}(k,0)\cos\left[2\pi\frac{km}{M}\right]$$

$$+\sum_{\tilde{l}=\frac{N-1}{2}}^{1}F_{\mathrm{I}}(0,-\tilde{l})\cos\left[-2\pi\frac{\tilde{l}n}{N}\right]+\sum_{l=1}^{\frac{N-1}{2}}F_{\mathrm{I}}(0,l)\cos\left[2\pi\frac{ln}{N}\right]$$

$$+\left\{\sum_{\tilde{k}=\frac{M-1}{2}}^{1}\sum_{\tilde{l}=\frac{N-1}{2}}^{1}+\sum_{\tilde{k}=\frac{M-1}{2}}^{1}\sum_{\tilde{l}=-1}^{-\frac{N-1}{2}}\right\}F_{R}(-\tilde{k},-\tilde{l})\sin\left[2\pi\left(-\frac{\tilde{k}m}{M}-\frac{\tilde{l}n}{N}\right)\right]$$

$$+\left\{\sum_{k=1}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{-1}+\sum_{k=1}^{\frac{M-1}{2}}\sum_{l=1}^{\frac{N-1}{2}}\right\}F_{R}(k,l)\sin\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right]$$

$$+\sum_{\tilde{k}=\frac{M-1}{2}}^{1}F_{R}(-\tilde{k},0)\sin\left[-2\pi\frac{\tilde{k}m}{M}\right]+\sum_{k=1}^{\frac{M-1}{2}}F_{R}(k,0)\sin\left[2\pi\frac{km}{M}\right]$$

$$+\sum_{\tilde{l}=\frac{N-1}{2}}^{1}F_{R}(0,-\tilde{l})\sin\left[-2\pi\frac{\tilde{l}n}{N}\right]+\sum_{l=1}^{\frac{N-1}{2}}F_{R}(0,l)\sin\left[2\pi\frac{ln}{N}\right]. \tag{4.133}$$

We observe that $\cos(-\phi)=\cos\phi$, $\sin(-\phi)=-\sin\phi$, *and that if* $F(-k,-l)=F^{*}(k,l)$, *then* $F_{\mathrm{R}}(-k,-l)+\mathrm{j}F_{\mathrm{I}}(-k,-l)=F_{\mathrm{R}}(k,l)-\mathrm{j}F_{\mathrm{I}}(k,l)$, *which implies that* $F_{\mathrm{R}}(-k,-l)=F_{\mathrm{R}}(k,l)$ *and* $F_{\mathrm{I}}(-k,-l)=-F_{\mathrm{I}}(k,l)$. *Obviously, we have the special cases* $F_{\mathrm{R}}(-k,0)=F_{\mathrm{R}}(k,0)$, $F_{\mathrm{R}}(0,-l)=F_{\mathrm{R}}(0,l)$, $F_{\mathrm{I}}(-k,0)=-F_{\mathrm{I}}(k,0)$ *and* $F_{\mathrm{I}}(0,-l)=-F_{\mathrm{I}}(0,l)$. *We also note that in a sum the order in which we add the various terms does not matter, so we may exchange the upper and lower limits of each summation with no further consequences. We just choose to start summing from the lower index and stop at the highest index value. Then:*

$$F_{\mathrm{I}}(m,n)=-\left\{\sum_{\tilde{k}=1}^{\frac{M-1}{2}}\sum_{\tilde{l}=1}^{\frac{N-1}{2}}+\sum_{\tilde{k}=1}^{\frac{M-1}{2}}\sum_{\tilde{l}=-\frac{N-1}{2}}^{-1}\right\}F_{\mathrm{I}}(\tilde{k},\tilde{l})\cos\left[2\pi\left(\frac{\tilde{k}m}{M}+\frac{\tilde{l}n}{N}\right)\right]$$

$$+\left\{\sum_{k=1}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{-1}+\sum_{k=1}^{\frac{M-1}{2}}\sum_{l=1}^{\frac{N-1}{2}}\right\}F_{\mathrm{I}}(k,l)\cos\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right]$$

$$-\sum_{\tilde{k}=1}^{\frac{M-1}{2}}F_{\mathrm{I}}(\tilde{k},0)\cos\left[2\pi\frac{\tilde{k}m}{M}\right]+\sum_{k=1}^{\frac{M-1}{2}}F_{\mathrm{I}}(k,0)\cos\left[2\pi\frac{km}{M}\right]$$

$$-\sum_{\tilde{l}=1}^{\frac{N-1}{2}}F_{\mathrm{I}}(0,\tilde{l})\cos\left[2\pi\frac{\tilde{l}n}{N}\right]+\sum_{l=1}^{\frac{N-1}{2}}F_{\mathrm{I}}(0,l)\cos\left[2\pi\frac{ln}{N}\right]$$

(Continued)

Example B4.33 (Continued)

$$
\begin{aligned}
&-\left\{\sum_{\tilde{k}=1}^{\frac{M-1}{2}}\sum_{\tilde{l}=1}^{\frac{N-1}{2}}+\sum_{\tilde{k}=1}^{\frac{M-1}{2}}\sum_{\tilde{l}=-\frac{N-1}{2}}^{-1}\right\}F_{\mathrm{R}}(\tilde{k},\tilde{l})\sin\left[2\pi\left(\frac{\tilde{k}m}{M}+\frac{\tilde{l}n}{N}\right)\right]\\
&+\left\{\sum_{k=1}^{\frac{M-1}{2}}\sum_{l=-\frac{N-1}{2}}^{-1}+\sum_{k=1}^{\frac{M-1}{2}}\sum_{l=1}^{\frac{N-1}{2}}\right\}F_{\mathrm{R}}(k,l)\sin\left[2\pi\left(\frac{km}{M}+\frac{ln}{N}\right)\right]\\
&-\sum_{\tilde{k}=1}^{\frac{M-1}{2}}F_{\mathrm{R}}(\tilde{k},0)\sin\left[2\pi\frac{\tilde{k}m}{M}\right]+\sum_{k=1}^{\frac{M-1}{2}}F_{\mathrm{R}}(k,0)\sin\left[2\pi\frac{km}{M}\right]\\
&-\sum_{\tilde{l}=1}^{\frac{N-1}{2}}F_{\mathrm{R}}(0,\tilde{l})\sin\left[2\pi\frac{\tilde{l}n}{N}\right]+\sum_{l=1}^{\frac{N-1}{2}}F_{\mathrm{R}}(0,l)\sin\left[2\pi\frac{ln}{N}\right]. \qquad (4.134)
\end{aligned}
$$

We may drop the tildes from the dummy summation indices without causing any confusion. Then we can see that the sums cancel each other to yield 0. This proves the theorem.

Can we have alternative tessellations of the frequency domain?

Yes, in fact when using Gaussian windows, most commonly the frequency plane is tessellated using polar coordinates. This is shown schematically in Figure 4.26. This choice is biologically inspired. It has been observed that the 2D receptive field profiles of simple cells in the mammalian visual cortex may be modelled by 2D Gabor functions defined at various orientations and various radial frequencies.

Alternatively, we often use bands that are narrower for higher frequencies and broader for low frequencies because texture is associated with high frequencies rather than low frequencies, and so, if we wish to discriminate texture better, we must tessellate more finely the high frequencies. Such a tessellation is shown schematically in Figure 4.27. Alternative schemes may also be devised.

How can we define a Gaussian window in polar coordinates in the frequency domain?

Let us denote by ρ the polar radius, and and ϕ the polar angle in frequency space, so that:

$$\rho \equiv \sqrt{\omega_x^2+\omega_y^2} \qquad \cos\phi \equiv \frac{\omega_x}{\rho} \qquad \sin\phi \equiv \frac{\omega_y}{\rho}. \qquad (4.135)$$

Let us assume that we choose the Gaussian windows to be centred at points $(\rho_{\mathrm{I}}, \phi_j)$, with standard deviation along the polar radius $\Sigma_{\rho_{\mathrm{I}}}$ and along the orthogonal direction $\Sigma_{\phi_{\mathrm{I}}}$.

If we define a local coordinate system $(\tilde{\omega}_{x;ij}, \tilde{\omega}_{y;ij})$, centred at $(\rho_{\mathrm{I}}, \phi_j)$ and rotated with respect to the original one by angle ϕ_j, the coordinate positions of a point $(\omega_{x;ij}, \omega_{y;ij})$ in terms of its coordinates in this new coordinate system are (see Figure 4.28):

$$
\begin{aligned}
\omega_{x;ij} &= (\rho_{\mathrm{I}}+\tilde{\omega}_{x;ij})\cos\phi_j - \tilde{\omega}_{y;ij}\sin\phi_j\\
\omega_{y;ij} &= (\rho_{\mathrm{I}}+\tilde{\omega}_{x;ij})\sin\phi_j + \tilde{\omega}_{y;ij}\cos\phi_j. \qquad (4.136)
\end{aligned}
$$

If we multiply the first equation with $\cos\phi_j$ and the second with $\sin\phi_j$ and add them, we can solve this system for $\rho_{\mathrm{I}}+\tilde{\omega}_{x;ij}$ and if we multiply the first equation with $-\sin\phi_j$ and the second with

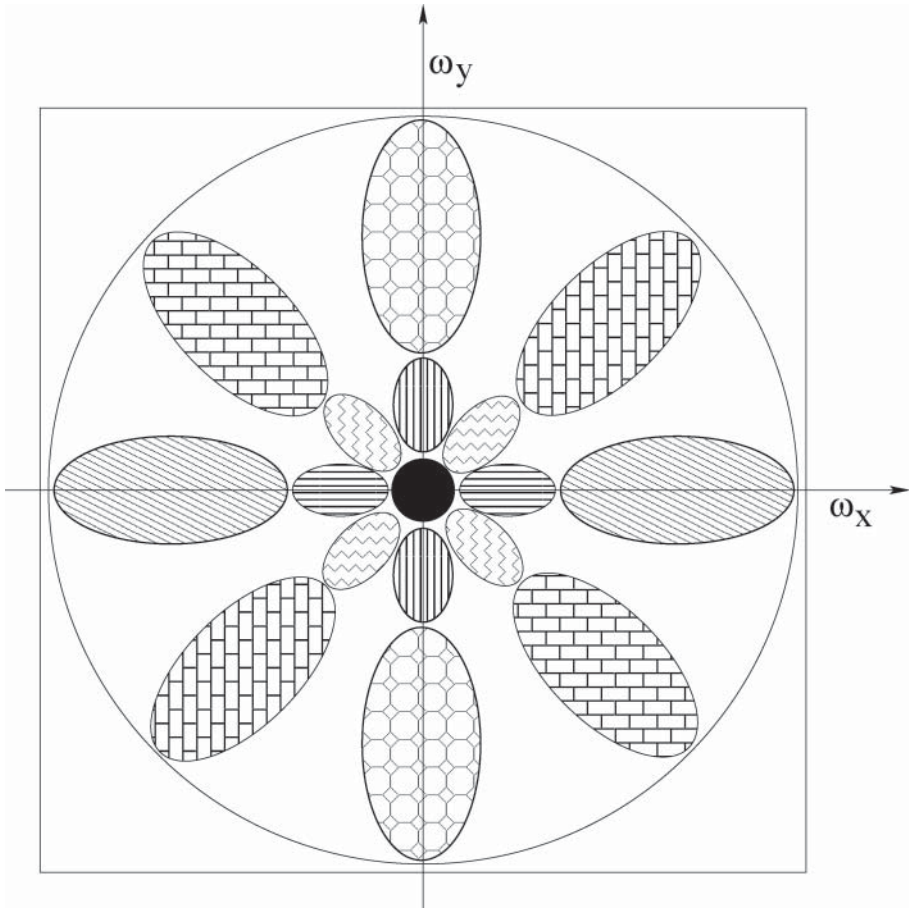


Figure 4.26 A polar coordinates-based tessellation of the frequency domain. Each ellipse represents a Gaussian window placed at that location. For the creation of real texture features, windows symmetrically placed about the centre of the axes (the dc component) should be simultaneously activated. Such pairs of windows are marked with the same shading.

$\cos\phi_j$ and add them, we can solve this system for $\tilde{\omega}_{y;ij}$:

$$\begin{aligned}\tilde{\omega}_{x;ij} &= \omega_{x;ij}\cos\phi_j + \omega_{y;ij}\sin\phi_j - \rho_{\mathrm{I}} \\ \tilde{\omega}_{y;ij} &= -\omega_{x;ij}\sin\phi_j + \omega_{y;ij}\cos\phi_j.\end{aligned} \tag{4.137}$$

In the $(\tilde{\omega}_{x;ij}, \tilde{\omega}_{y;ij})$ coordinate system the equation of a Gaussian is:

$$\tilde{G}(\tilde{\omega}_{x;ij}, \tilde{\omega}_{y;ij}) = \exp\left\{-\frac{\tilde{\omega}^2_{x;ij}}{2\Sigma^2_{\rho_{\mathrm{I}}}} - \frac{\tilde{\omega}^2_{y;ij}}{2\Sigma^2_{\phi_{\mathrm{I}}}}\right\}. \tag{4.138}$$

If we express $(\tilde{\omega}_{x;ij}, \tilde{\omega}_{y;ij})$ in terms of $(\omega_{x;ij}, \omega_{y;ij})$, using (4.137), we obtain:

$$\tilde{G}(\omega_{x;ij}, \omega_{y;ij}) = \exp\left\{-\frac{(\omega_{x;ij}\cos\phi_j + \omega_{y;ij}\sin\phi_j - \rho_{\mathrm{I}})^2}{2\Sigma^2_{\rho_{\mathrm{I}}}} - \frac{(-\omega_{x;ij}\sin\phi_j + \omega_{y;ij}\cos\phi_j)^2}{2\Sigma^2_{\phi_{\mathrm{I}}}}\right\}. \tag{4.139}$$

This definition depends on some parameters that have to be specified, otherwise the definition is not complete.

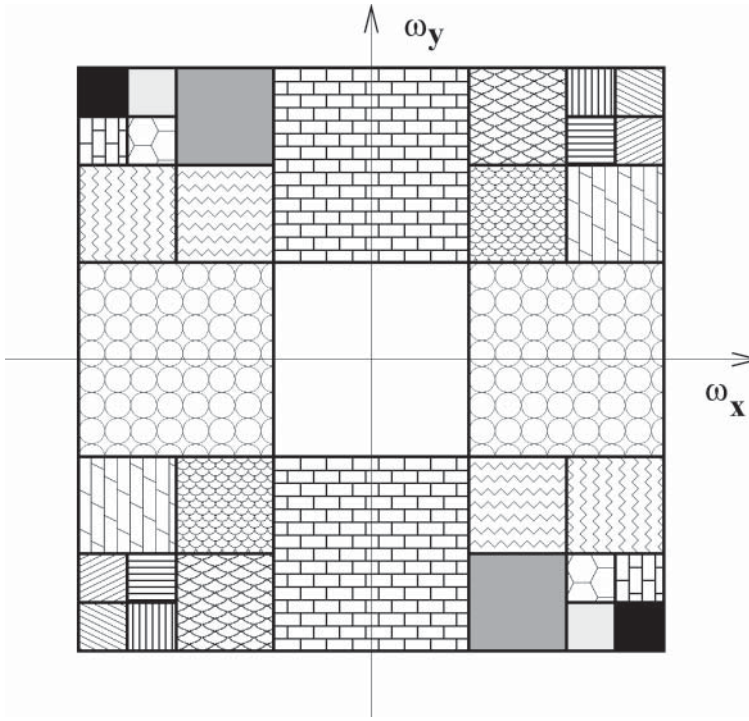


Figure 4.27 A tessellation of the frequency domain where higher frequencies are more finely tessellated. For the creation of real texture features, windows symmetrically placed about the centre of the axes should be simultaneously activated. Such windows are marked with the same shading.

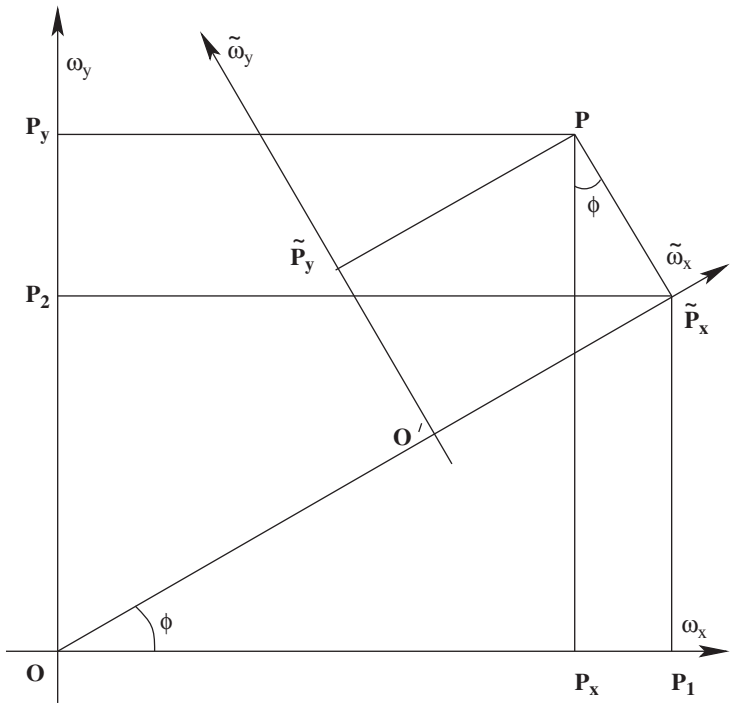


Figure 4.28 The coordinates of point P in the large coordinate system are (OP_x, OP_y), and in the rotated coordinate system are $(O'\tilde{P}_x, O'\tilde{P}_y)$. It is obvious that $OP_x = OP_1 - P_xP_1 = O\tilde{P}_x \cos\phi - P\tilde{P}_x \sin\phi = (OO' + O'\tilde{P}_x)\cos\phi - O'\tilde{P}_y \sin\phi$. Also, $OP_y = OP_2 + P_2P_y = O\tilde{P}_x \sin\phi + P\tilde{P}_x \cos\phi = (OO' + O'\tilde{P}_x)\sin\phi + O'\tilde{P}_y \cos\phi$.

As Gabor functions are biologically inspired, we often use the convention of choosing the parameters in accordance with the parameters used by the human visual system. It has been shown that the human visual system has filters similar to Gabor filters which have central frequency responses one **octave** apart. Thus, frequencies ρ_I are measured in octaves.

What is an octave?

An octave is the bandwidth between two frequencies, one of which is double the other.

How do we express a frequency in octaves?

This is better explained in conjunction with the diagram of Figure 4.29. Let us say that two frequencies f_1 and f_2 are one octave apart. This means that $f_2 = 2f_1$. If a third frequency f_3 is one octave away from f_2, it must be $f_3 = 2f_2 = 2 \times 2f_1 = 2^2 f_1$. Frequencies f_1 and f_3 are two octaves apart. A fourth frequency, f_4, which is one octave away from f_3, is $f_4 = 2f_3 = 2 \times 2f_2 = 2 \times 2 \times 2f_1 = 2^3 f_1$. We say that f_2 is one octave away, f_3 is two octaves away and f_4 is three octaves away from f_1, or that there are three octaves between frequencies f_4 and f_1. It is evident from this that the number of octaves between two frequencies is given by:

$$\log_2 \frac{f_{\max}}{f_{\min}}. \tag{4.140}$$

For an unknown frequency f_x, we may say that it is α octaves away from frequency f_1 and compute α in the same way:

$$\alpha \equiv \log_2 \frac{f_x}{f_{\min}}. \tag{4.141}$$

Example B4.34

If a frequency band is $\Delta\alpha$ octaves wide, what is its width $\Delta\omega$ in cycles per unit length?
If the central frequency of the band is ω, the upper limit of the band is $\omega + \frac{\Delta\omega}{2}$ and the lower limit is $\omega - \frac{\Delta\omega}{2}$. This width in octaves, $\Delta\alpha$, is given by:

$$\Delta\alpha = \log_2 \frac{\omega + \frac{\Delta\omega}{2}}{\omega - \frac{\Delta\omega}{2}}. \tag{4.142}$$

(Continued)

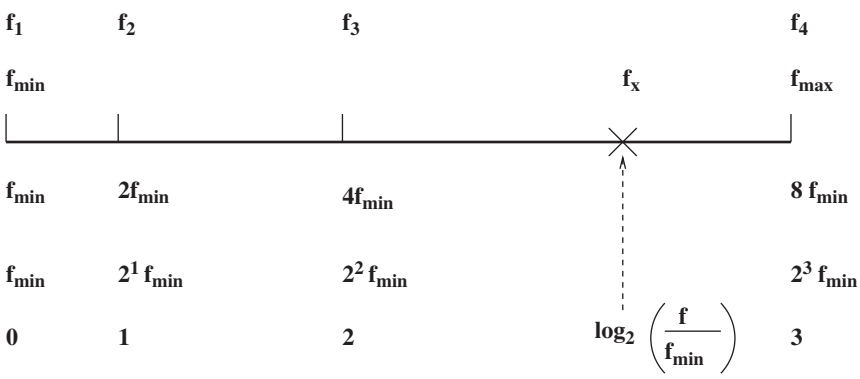


Figure 4.29 An octave is the interval between a frequency and its double. The bottom row of numbers counts the number of octaves starting from frequency $f_1 = f_{\min}$.

Example B4.34 (Continued)

This equation may be re-arranged as follows:

$$\begin{aligned}
&\frac{\omega+\frac{\Delta\omega}{2}}{\omega-\frac{\Delta\omega}{2}} = 2^{\Delta\alpha} \\
\Rightarrow\ & \omega 2^{\Delta\alpha} - \frac{\Delta\omega}{2}2^{\Delta\alpha} = \omega + \frac{\Delta\omega}{2} \\
\Rightarrow\ & \omega(2^{\Delta\alpha}-1) = \frac{\Delta\omega}{2}(2^{\Delta\alpha}+1) \\
\Rightarrow\ & \Delta\omega = 2\omega\frac{2^{\Delta\alpha}-1}{2^{\Delta\alpha}+1}.
\end{aligned} \tag{4.143}$$

From this expression we may compute the width of a frequency band in cycles per unit length if we know its width in octaves and its central frequency.

How may we choose the parameters of the Gaussian window in the frequency space?

We have the following parameters to choose:

- The positions of the centres of the functions in the frequency space, so that we cover the whole of space with the minimum number of functions.
- The standard deviations of each Gaussian along its principal axes.

Let us consider Figure 4.26. The circular band placed in the middle represents the low-frequency component, as it is centred at zero frequency. Texture attributes are really measured by the Gabor functions outside that circle. Let us call ω_0 the limit of the low-frequency band.

The maximum possible frequency of an image along the x or y axes is defined by the size of the image. If the image is $N \times M$ in size, the frequency domain is also $N \times M$ in size, with the fundamental frequency along the x axis being $2\pi/N$ and along the y axis $2\pi/M$. The maximum frequency along each axis is $\pi(N-1)/N$ and $\pi(M-1)/M$, respectively when N and M are both odd (in which case the indices along the frequency axes vary in the ranges $\left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$ and $\left[-\frac{M-1}{2}, \frac{M-1}{2}\right]$, respectively). If N and M are both even, the ranges of the frequency indices are $\left[-\frac{N}{2}, \frac{N}{2}-1\right]$ and $\left[-\frac{M}{2}, \frac{M}{2}-1\right]$, respectively. In this case, the maximum frequency along each axis is $\pi(N-2)/N$ and $\pi(M-2)/M$, respectively. In either case, let us call the maximum of these two frequencies $\omega_{\max}$. First we want to see how many octaves we can fit between frequencies ω_0 and $\omega_{\max}$. This is easily computed as:

$$\tilde{K} \equiv \log_2 \frac{\omega_{\max}}{\omega_0}. \tag{4.144}$$

This number tells us how many radial bands we need in order to cover the whole frequency space. In general, this is not an integer, so we may round it up to the nearest integer, say K. Each radial band must be one octave in width. From the definition of an octave, we infer that the width of each band is equal to the lower frequency boundary of the band. If ω_1 is the upper limit of the first band, since $\omega_1 = 2\omega_0$, the width of the first band is:

$$\Delta\omega_1 = \omega_1 - \omega_0 = \omega_0. \tag{4.145}$$

We choose Σ_{x1} so that at the boundary of the band the value of the Gaussian is 0.5. (At the centre of the band the value of the Gaussian is 1.) This means that:

$$e^{-\frac{(\omega_0/2)^2}{2\Sigma_{x1}^2}} = 0.5 \Rightarrow -\frac{(\omega_0/2)^2}{2\Sigma_{x1}^2} = \ln 0.5 = -\ln 2 \Rightarrow \Sigma_{x1} = \frac{\omega_0}{2\sqrt{2\ln 2}}. \tag{4.146}$$

The centre of the first band is at polar radius:

$$\rho_1 = \frac{\omega_1 + \omega_0}{2} = \frac{3\omega_0}{2}. \tag{4.147}$$

If ω_2 is the upper limit of the second band, we have:

$$\omega_2 = 2\omega_1 = 2^2\omega_0. \tag{4.148}$$

The width of the second band, therefore, is

$$\Delta\omega_2 = \omega_2 - \omega_1 = \omega_1 = 2\omega_0 \tag{4.149}$$

and its centre is at:

$$\rho_2 = \frac{\omega_2 + \omega_1}{2} = \frac{3\omega_1}{2} = 3\omega_0. \tag{4.150}$$

In general, the upper limit of band i is

$$\omega_{\mathrm{I}} = 2^i\omega_0 \tag{4.151}$$

and so the bandwidth of band i is

$$\Delta\omega_{\mathrm{I}} = \omega_{\mathrm{I}} - \omega_{i-1} = 2^i\omega_0 - 2^{i-1}\omega_0 = 2^{i-1}\omega_0. \tag{4.152}$$

and the corresponding standard deviation is:

$$\Sigma_{xi} = \frac{2^{i-2}\omega_0}{\sqrt{2\ln 2}} \tag{4.153}$$

The centre of band i is at:

$$\rho_{\mathrm{I}} = \frac{\omega_{\mathrm{I}} + \omega_{i-1}}{2} = \frac{1}{2}(2^i\omega_0 + 2^{i-1}\omega_0) = 2^{i-2}3\omega_0. \tag{4.154}$$

Let us decide that we are going to use L filters along the tangential direction. Then the bandwidth $\Delta\Phi$ in the tangential direction is $\Delta\Phi = 360^\circ/L$. In practice this means that the centre of the first filter is placed at $\phi = 0^\circ$, the centre of the second at $\Delta\Phi$, the centre of the next at $2\Delta\Phi$, and so on, until the last one at $(L-1)\Delta\Phi$ degrees with respect to the positive ω_x axis.

Let us consider the Gaussian window shown in Figure 4.30. It is rotated by an angle ϕ_j with respect to the ω_x axis and centred at point A. Length AB is equal to half the spectral bandwidth of the filter along the ω_y axis (i.e. $AB = \Delta\omega_{y;i}/2$). Length OA is equal to ρ_{I}. Then we have:

$$\tan\frac{\Delta\Phi}{2} = \frac{AB}{OA} = \frac{\Delta\omega_{y;i}}{2\rho_{\mathrm{I}}} \Rightarrow \Delta\omega_{y;i} = 2\rho_{\mathrm{I}}\tan\frac{\Delta\Phi}{2}. \tag{4.155}$$

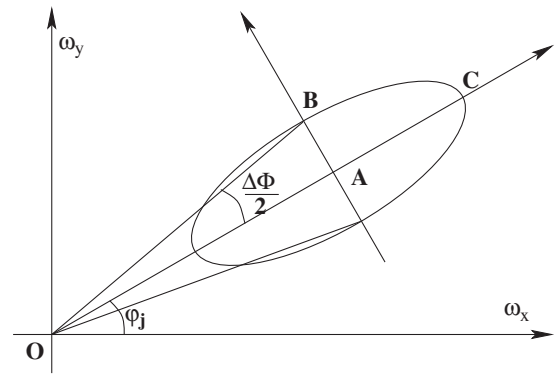


Figure 4.30 The ellipse represents an isocontour of a Gaussian window in the frequency space, such that length *AB* is equal to half the bandwidth of the function.

We choose then:

$$\Sigma_{yi} = \frac{\rho_{\rm I}}{\sqrt{2\ln 2}} \tan\frac{\Delta\Phi}{2}. \tag{4.156}$$

How do we perform the analysis of an image in terms of Gabor functions in practice?

We use the following algorithm.

Step 0: Select the radius ω_0 of the frequencies you will treat as low. Select also the number L of azimuth bands you will use.

Step 1: Assuming that your image I is $N \times M$ in size, calculate the highest frequency present in the image along each axis, as follows.
If N is odd, set $\omega_x = \frac{\pi(N-1)}{N}$.
If N is even, set $\omega_x = \frac{\pi(N-2)}{N}$.
If M is odd, set $\omega_y = \frac{\pi(M-1)}{M}$.
If M is even, set $\omega_y = \frac{\pi(M-2)}{M}$.

Step 2: Select the maximum frequency in the image: $\omega_{\max} = \max\{\omega_x, \omega_y\}$.

Step 3: Calculate the number of filters you will use along the radial direction:

$$K = \left\lceil \log_2 \frac{\omega_{\max}}{\omega_0} \right\rceil. \tag{4.157}$$

Step 4: Calculate the positions of the centres of the bands along the radial direction:

$$\rho_k = 2^{k-2} 3\omega_0 \qquad \text{for } k = 1, \ldots, K. \tag{4.158}$$

Step 5: Calculate the standard deviation of the Gaussian filters along the radial direction:

$$\Sigma_{\rho_k} = \frac{2^{k-2}\omega_0}{\sqrt{2\ln 2}}. \tag{4.159}$$

Step 6: Calculate the positions of the centres of the bands along the azimuthal direction:

$$\Delta\Phi = \frac{360^\circ}{L} \qquad \phi_l = (l-1)\Delta\Phi \qquad \text{for } l = 1, \ldots, L. \tag{4.160}$$

Step 7: Calculate the standard deviation of the Gaussian filters along the azimuthal direction:

$$\Sigma_{\phi_l k} = \frac{\rho_k}{\sqrt{2\ln 2}} \tan\frac{\Delta\Phi}{2} \qquad \text{for } k = 1, \ldots, K, \quad \forall l. \tag{4.161}$$

Note that this standard deviation is the same for all values of l and simply varies with the radial position of the centre of the Gaussian.

Step 8: If L is odd, set $T \equiv KL$, while if L is even, set $T \equiv KL/2$ and create $T+1$ arrays the same size as the image. Call them F_t, for $t = 0, 1, \ldots, T$. Define the elements of these arrays as follows.

$$F_0(u,v) = \begin{cases} 1 \text{ if } \left(\frac{2\pi u}{N}\right)^2 + \left(\frac{2\pi v}{M}\right)^2 \le \omega_0^2 \\ 0 \text{ if } \left(\frac{2\pi u}{N}\right)^2 + \left(\frac{2\pi v}{M}\right)^2 > \omega_0^2 \end{cases} \tag{4.162}$$

$$\begin{aligned} F_t(u,v) \;=\; & \exp\left\{ -\frac{\left(\frac{2\pi u}{N}\cos\phi_l + \frac{2\pi v}{M}\sin\phi_l - \rho_k\right)^2}{2\Sigma_{\rho_k}^2} - \frac{\left(-\frac{2\pi u}{N}\sin\phi_l + \frac{2\pi v}{M}\cos\phi_l\right)^2}{2\Sigma_{\phi_l k}^2} \right\} \\ & + \exp\left\{ -\frac{\left(-\frac{2\pi u}{N}\cos\phi_l - \frac{2\pi v}{M}\sin\phi_l - \rho_k\right)^2}{2\Sigma_{\rho_k}^2} - \frac{\left(\frac{2\pi u}{N}\sin\phi_l - \frac{2\pi v}{M}\cos\phi_l\right)^2}{2\Sigma_{\phi_l k}^2} \right\}. \end{aligned} \tag{4.163}$$

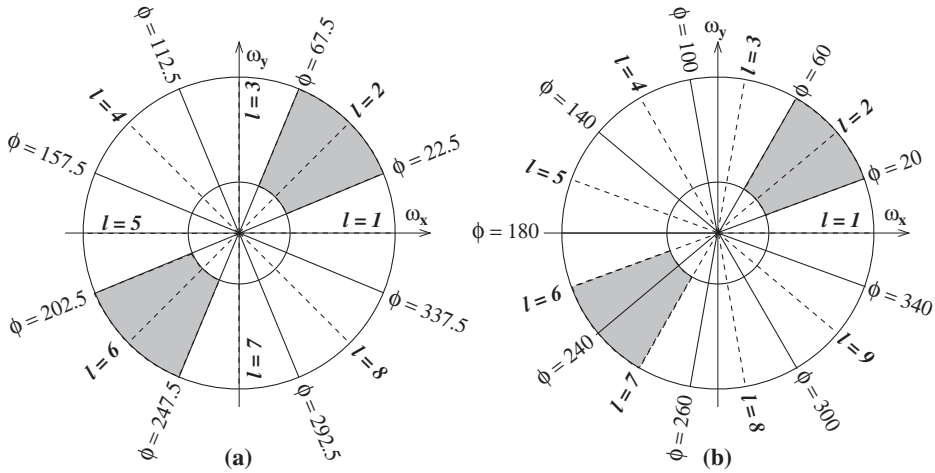


Figure 4.31 (a) When the number of azimuthal bands is even (e.g. $L = 8$), pairing the bands in order to obtain a real function in the image domain means that we may consider only values of $l = 1, \ldots, 4$ to cover the frequency space. (b) When the number of bands is odd (e.g. $L = 9$), the second band we pair with the originally chosen one ($l = 2$) strands two half bands ($l = 5$ and $l = 6$), so we have to consider all values of $l = 1, 2, \ldots, 9$. This means that the frequency spaced will be covered twice with densely overlapping pairs of bands.

Here $t = kl$ and $t = 1, 2, \ldots, T$. The addition of the two terms guarantees that F_t is symmetric about the origin. Indeed, the second term in (4.163) was produced from the first by replacing ϕ_l with $\phi_l + 180°$, to create a Gaussian window in a symmetrically opposite location from the first Gaussian, with respect to position $(0, 0)$. If L is even, these two Gaussians are placed in exactly two chosen bands, and that is why in that case we need only consider $L/2$ azimuthal bands. If L is odd, however, the second Gaussian does not cover a chosen band, but it strands across two azimuthal bands, and that is why we need to consider all L azimuthal bands (see Figure 4.31). In that case, of course, we have better coverage of the azimuthal frequencies, as there is a twofold redundancy. As the Gaussians give different weights to the different frequencies, all these redundant representations may turn out to be useful since they will cover the frequency space more densely. Indices u and v take the following values.

$$\begin{aligned}
&u \in \left[-\tfrac{N-1}{2}, \tfrac{N-1}{2}\right] \quad v \in \left[-\tfrac{M-1}{2}, \tfrac{M-1}{2}\right] \quad \text{if } N, M \text{ odd} \\
&u \in \left[-\tfrac{N-1}{2}, \tfrac{N-1}{2}\right] \quad v \in \left[-\tfrac{M}{2}, \tfrac{M}{2} - 1\right] \quad \text{if } N \text{ odd, } M \text{ even} \\
&u \in \left[-\tfrac{N}{2}, \tfrac{N}{2} - 1\right] \quad v \in \left[-\tfrac{M}{2}, \tfrac{M}{2} - 1\right] \quad \text{if } N, M \text{ even} \\
&u \in \left[-\tfrac{N}{2}, \tfrac{N}{2} - 1\right] \quad v \in \left[-\tfrac{M-1}{2}, \tfrac{M-1}{2}\right] \quad \text{if } N \text{ even, } M \text{ odd.}
\end{aligned} \tag{4.164}$$

Step 9: Take the DFT of the image. Call it $\hat{I}$. Make sure the indices of $\hat{I}$ take values in the ranges defined in (4.164).

Step 10: Set the dc component of $\hat{I}$ to 0. Call the result $\hat{I}_0$. Create the products: $\check{I}_t = \hat{I}_0 F_t$, where the matrices are multiplied point by point.

Step 11: Take the inverse DFT of each $\check{I}_t$, to create I_t. These matrices are the Gabor components of the original image. From here we have various options. If we want to create some form of reconstruction of the original image, we have to add the following steps.

Step 12: Create the product $\check{I}_0 = \hat{I} F_0$, where the matrices are multiplied point by point.

Step 13: Take the inverse DFT of $\check{I}_0$. Call it I_0.

Step 14: Create a reconstruction of the image as $\tilde{I} = \sum_{t=0}^{T} I_t$, if L is even and only half of the azimuth bands have been used, or as $\tilde{I} = \frac{1}{2}\sum_{t=0}^{T} I_t$, if L is odd and the frequency space has been covered with redundancy.

Step 15: You may compute the error of this reconstruction by summing up the squares of the values of matrix $\tilde{I} - I$.

Note that the reconstruction is not expected to be perfect, as the different frequencies of the image have been multiplied with different weights.

If you want to create texture features from the Gabor expansion of the image, you have to compute the local energy around each pixel in representations I_t. When we compute local energy, we find the sum of the squares of the values of the pixels in the local neighbourhood of each pixel and assign it to the focal pixel. However, some researchers have found that better results may be obtained by first applying the following transformation to the values of representations I_t:

$$\psi(g) = \left| \frac{1 - e^{-2\alpha g}}{1 + e^{-2\alpha g}} \right|. \tag{4.165}$$

Here $\alpha > 0$ is some parameter and g is the value of a pixel in representation I_t. If $g \to \pm\infty$, $\psi(g) \to 1$ and if $g = 0$, $\psi(g) = 0$. Figure 4.32 shows a plot of one cycle of a sinusoid, which is assumed to represent how g changes with distance x, and superimposed upon it, the corresponding values of $\psi(g)$ for $\alpha = 1.25$, 2.50 and 5.00. We note that the effect of this transformation is to rectify the wave, by making both its lobes positive, and to broaden them into stripes.

Care should be taken when choosing a value for the processing of the real data, as Figure 4.32 refers to g values in the range $[-1, 1]$, while real data will be in a totally different range. We have the choice either to scale each I_t separately in the range $[-1, 1]$ before taking its ψ transform, or to scale all I_t together in the range $[-1, 1]$ before taking their ψ transforms. The former is appropriate for visualisation. However, individual scaling loses all information on the **relative** strength of the values a pixel has in the different representations I_t, i.e. in the different bands, and it will cause problems when we need that information for segmentation. That is why, in order to compute the local energy from these representations, we scale all of them together in the range $[-1, 1]$ before taking their ψ transform.

Then in order to compute the local energy we average the $\psi(g)$ values inside a local window around each pixel. It is better if we perform this averaging using a Gaussian window, the size of which is related to the central frequency of the band we are analysing.

We know that the standard deviation of a Gaussian in the frequency domain is the inverse of the standard deviation of the corresponding Gaussian in the image domain. So, a Gaussian that has standard deviation Σ_{ρ_k} along the radial direction in the frequency domain, corresponds to a Gaussian with standard deviation $\sigma_k = 1/\Sigma_{\rho_k}$ in the image space. Using (4.159), we obtain

$$\sigma_k = \frac{\sqrt{2 \ln 2}}{2^{k-2}\omega_0} = \frac{3\sqrt{2 \ln 2}}{\rho_k} = \frac{3.532}{\rho_k} \tag{4.166}$$

where we replaced $2^{k-2}\omega_0$ from (4.158).

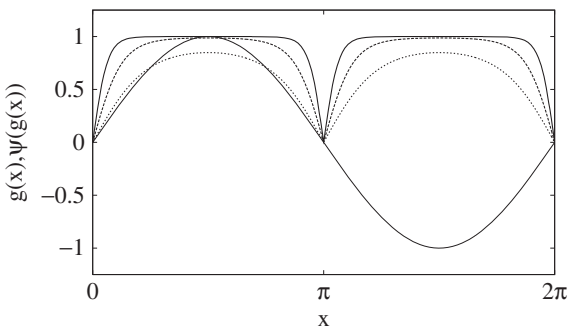


Figure 4.32 Function $\psi(g(x))$ for $\alpha = 1.25$, 2.5, and 5.0, from bottom to top respectively, when $g(x) = \sin x$, versus x. The sine wave is the plot of $g(x)$.

A difference between the filters we use in the frequency domain and those we use in the real domain for smoothing is that the filters in the frequency domain must have value 1 in their centres and, therefore, their weights do not sum up to 1. This is because they have to preserve certain frequencies intact. However, the weights of the smoothing convolution filters we use here to compute local averages, must sum up to 1. So the values of the truncated Gaussian filter we create have to be divided with their sum to fulfil this condition.

So, to produce texture features from representations I_t, we have to perform the following steps.

Step 16: Scale all representations I_t to have values in the range $[-1, 1]$. Call the resultant representations $\tilde{I}_t$.

Step 17: Take the ψ transform of every $\tilde{I}_t$, using Equation (4.165), applied to each pixel value, and thus produce representations $I_{\psi t}$.

Step 18: For each value of k, work out the Gaussian weights you will use to compute the local energy, using $e^{-(x^2+y^2)/(2\sigma_k^2)}$ with σ_k given by (4.166). To decide the size of such a window, let us consider it to be $(2S+1)\times(2S+1)$. To specify the value of S, we accept that at $x = \pm S$ and $y = 0$, the value of the window must be 0.1. Then:

$$e^{-\frac{S^2}{2\sigma_k^2}} = 0.1 \Rightarrow -\frac{S^2}{2\sigma_k^2} = \ln 0.1 \Rightarrow S = \sigma_k\sqrt{-2\ln 0.1} = 2.146\sigma_k. \tag{4.167}$$

Compute the values of the $(2S+1)\times(2S+1)$ mask by allowing x and y in $e^{-(x^2+y^2)/(2\sigma_k^2)}$ to take all possible integer values in the range $[-S, S]$. At the end, add all elements of the mask you created and divide each element by this sum, so that the mask is normalised to consist of elements that sum up to 1. Create one such mask for every value of k and call it G_k.

Step 19: Square each element of each representation $I_{\psi t}$. Call the results I_{t2}.

Step 20: Each t value corresponds to a particular pair of indices (k, l). Convolve image I_{t2} with the corresponding G_k mask, to work out the weighted local energy values for each pixel in each representation I_{t2}. Call the resultant representations E_t. These are the feature maps of the original image.

Step 21: To create the feature vector for each pixel, read the sequence of values it has in the feature maps E_t, thus creating a T-dimensional feature vector for each pixel of the original image.

Step 22: To visualise the feature maps, scale either each one individually or all together using the same scale, to the range [0,255], rounding the scaled values to the nearest integer.

Steps 16 and 17 are optional. One may not wish to apply the ψ transform to representations I_t, but take the local energy of them directly. In that case, step 19 should be applied to representations I_t and not $I_{\psi t}$.

One may also wish to avoid using a Gaussian weighting to compute the local energy. In that case, step 18 may be omitted and when performing step 20 one may use a flat averaging window instead of a Gaussian window.

Example B4.35

Work out the value mask F_t, defined by Equation (4.163), has at the centre of each of the two Gaussians it consists of.

The centre of the Gaussian represented by the first term of (4.163) is at (ρ_k, ϕ_l). The components of the radial frequency ρ_k along the two axes are $(\rho_k\cos\phi_l, \rho_k\sin\phi_l)$. We have to use these values in

(Continued)

Example B4.35 (Continued)

place of $(2\pi u/N, 2\pi v/M)$ *in (4.163) to work out the value of* F_t *at the centre of the first Gaussian:*

$$\begin{aligned} F_t(\rho_k, \phi_l) &= \exp\left\{ -\frac{(\rho_k \cos^2\phi_l + \rho_k \sin^2\phi_l - \rho_k)^2}{2\Sigma^2_{\rho_k}} - \frac{(-\rho_k \cos\phi_l \sin\phi_l + \rho_k \sin\phi_l \cos\phi_l)^2}{2\Sigma^2_{\phi_l k}} \right\} \\ &\quad + \exp\left\{ -\frac{(-\rho_k \cos^2\phi_l - \rho_k \sin^2\phi_l - \rho_k)^2}{2\Sigma^2_{\rho_k}} - \frac{(\rho_k \cos\phi_l \sin\phi_l - \rho_k \sin\phi_l \cos\phi_l)^2}{2\Sigma^2_{\phi_l k}} \right\} \\ &= 1 + \exp\left\{ -\frac{\rho_k^2}{\Sigma^2_{\rho_k}} \right\}. \end{aligned} \tag{4.168}$$

The centre of the second Gaussian is at $(\rho_k, \phi_l + 180°)$*. So, the radial frequency along the two axes has components* $(-\rho_k \cos\phi_l, -\rho_k \sin\phi_l)$*. If we substitute into (4.163), we shall eventually work out a value equal to* $F_t(\rho_k, \phi_l)$*.*

So, if we want the F_t *mask to have value 1 at the central frequency of each Gaussian, we shall have to divide all its values with* $F_t(\rho_k, \phi_l)$ *given by (4.168).*

Example 4.36

How will you modify step 8 of the algorithm so that F_t has 0 value outside the band of its definition?

We must follow these sub-steps.

Step 9.0: *Identify the k and l indices to which the t index of* F_t *corresponds. Set all elements of* F_t *equal to 0.*

Step 9.1: *For* $u \in [u_{\min}, u_{\max}]$*, where* $u_{\min}$ *and* $u_{\max}$ *are defined according to (4.164), and for each* $v \in [v_{\min}, v_{\max}]$*, where* $v_{\min}$ *and* $v_{\max}$ *are defined according to (4.164), compute:*

$$\rho = \sqrt{u^2 + v^2} \qquad \tilde{\phi} = \tan^{-1}\frac{|v|}{\rho}. \tag{4.169}$$

The use of the absolute value in the calculation of $\tilde{\phi}$ *ensures that the computed value is in the range* $[0, 90°]$*. So, we must then put it in the range* $[0, 360°]$ *using the following rules.*

$$\begin{aligned} &\text{If } u \geq 0 \text{ and } v \geq 0 \text{ Set } \phi = \tilde{\phi} \\ &\text{If } u \geq 0 \text{ and } v < 0 \text{ Set } \phi = 360° - \tilde{\phi} \\ &\text{If } u < 0 \text{ and } v \geq 0 \text{ Set } \phi = 180° - \tilde{\phi} \\ &\text{If } u < 0 \text{ and } v < 0 \text{ Set } \phi = 180° + \tilde{\phi}. \end{aligned} \tag{4.170}$$

Step 9.2: *If* $\{2^{k-1}\omega_0 < \rho < 2^k\omega_0$ *and* $l > 1$ *and* $\Delta\Phi(l - 1.5) < \phi < \Delta\Phi(l - 0.5)\}$*, or* $\{2^{k-1}\omega_0 < \rho < 2^k\omega_0$ *and* $l = 1$ *and* $0 < \phi < 0.5\Delta\Phi\}$*, or* $\{2^{k-1}\omega_0 < \rho < 2^k\omega_0$ *and* $l = 1$ *and* $360° - 0.5\Delta\Phi < \phi < 360°\}$*, set*

$$F_t(u, v) = \exp\left\{ -\frac{\left(\frac{2\pi u}{N}\cos\phi_l + \frac{2\pi v}{M}\sin\phi_l - \rho_k\right)^2}{2\Sigma^2_{\rho_k}} - \frac{\left(-\frac{2\pi u}{N}\sin\phi_l + \frac{2\pi v}{M}\cos\phi_l\right)^2}{2\Sigma^2_{\phi_l k}} \right\}. \tag{4.171}$$

If $2^{k-1}\omega_0 < \rho < 2^k\omega_0$ *and* $\Delta\Phi(l-1.5)+180° < \phi < \Delta\Phi(l-0.5)+180°$, *set*

$$F_t(u,v) = \exp\left\{ -\frac{\left(-\frac{2\pi u}{N}\cos\phi_l - \frac{2\pi v}{M}\sin\phi_l - \rho_k\right)^2}{2\Sigma_{\rho_k}^2} - \frac{\left(\frac{2\pi u}{N}\sin\phi_l - \frac{2\pi v}{M}\cos\phi_l\right)^2}{2\Sigma_{\phi_l k}^2} \right\}. \quad (4.172)$$

Example 4.37

Use Gabor functions to compute feature maps for the image of Figure 4.33a.
This image is 640×480 *in size. So, the frequency indices that are present in it are in the ranges* $[-320,319]$ *and* $[-240,239]$.

The basic frequencies along the two axes are $2\pi/640$ *and* $2\pi/480$ *respectively. Let us say that we choose* ω_0 *to be* $10 \times 2\pi/480$, *that is,* $\omega_0 = 0.131$.

The highest frequencies along the two axes are $\pi 638/640$ *and* $\pi 478/480$. *The highest of these two is* $\pi 638/640$, *and so we set* $\omega_{\max} = \pi 638/640 = 3.132$.

Next, we compute:

$$K = \left\lfloor \log_2 \frac{\omega_{\max}}{\omega_0} \right\rfloor = \left\lfloor \log_2 \frac{3.132}{0.131} \right\rfloor = \lceil 4.58 \rceil = 5. \quad (4.173)$$

From these values and by making use of Equations (4.152) and (4.158), we obtain the values of $\Delta\omega_k$ *and* ρ_k *for all the radial bands, shown under the corresponding columns of Table 4.1. The values for* Σ_{ρ_k} *and* $\Sigma_{\phi_l k}$ *are computed from Equations (4.159) and (4.161), respectively.*

Let us choose $L = 8$ *orientations for our directional filters. Then* $\Delta\Phi = 360°/8 = 45° = 0.785$ *rad. Table 4.2 shows the orientations along which the centre of each Gaussian window is placed. The parameters of Tables 4.1 and 4.2 are then used in equation (4.163) to construct the Gaussian masks. Note, that as we consider pairs of masks in symmetric positions about the centre, in reality we use only* $L/2 = 4$ *inverse Fourier transforms to extract the local frequency content of the image.*

An important detail concerns the symmetric placement of the windows about the origin $(0,0)$. *The image is of even size along each axis, so there is no central point. It is of size* 640×480 *pixels, i.e. the ranges of indices used are* $[-320,319]$ *and* $[-240,239]$. *To make sure that the reconstructed images are real (as opposed to complex), the pairs of filters used for each band should be symmetric and therefore row* -320 *and column* -240 *are not considered.*

Table 4.1 Values of ρ_k, $\Delta\omega_k$, Σ_{ρ_k}, and $\Sigma_{\phi_l k}$ used to segment the image in Figure 4.33.

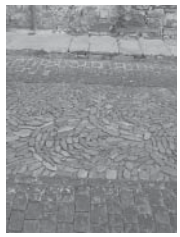
Radial band	ρ_k	$\Delta\omega_k$	Σ_{ρ_k}	$\Sigma_{\phi_l k}$
1	0.196	0.131	0.055	0.070
2	0.393	0.262	0.111	0.139
3	0.785	0.524	0.223	0.277
4	1.571	1.047	0.445	0.553
5	3.142	2.094	0.890	1.107

(Continued)

Example 4.37 (Continued)

Table 4.2 Orientations of the angular bands of the Gaussian masks.

Angular band	ϕ_l
$l = 1$	0°
$l = 2$	45°
$l = 3$	90°
$l = 4$	135°



(a)

(b)

Figure 4.33 (a) A Girona pavement to be segmented and (b) the magnitude of its Fourier transform (appropriately scaled for visualisation purposes). Source: Maria Petrou.

The top rows of Figures 4.34–4.38 show the results of multiplying the Gaussian masks with the Fourier transform of the original image shown in Figure 4.33b. Note that, the same scaling to the range [0,255] *has been used for all the results to allow visual comparison between the different panels. The second rows of the same figures show the inverse Fourier transforms of the filtered data. A strong visual characteristic of each reconstructed image shown in Figures 4.34–4.38 is its ripple-like appearance. This indicates the presence of positive and negative values, both of which should contribute to the image energy. When we compute the energy we take the square of the values, so we take care of the fluctuations in sign. However, we choose to apply the ψ transform first, given by Equation (4.165) with $\alpha = 2.50$. Figures 4.34–4.38 show the $\psi(g)$ transforms of the reconstructed images, each one scaled separately for visualisation purposes. These are the uniformly scaled versions of representations $I_{\psi t}$ of step 17 of the algorithm. Then we apply steps 18 and 19 of the same algorithm to the unscaled representations $I_{\psi t}$.*

Finally, to visualise all these local energy representations, we scale them all together in the range [0,255]*, so that grey values in different panels may be compared. The results are shown in the bottom rows of Figures 4.34–4.38.*

Now, we have 20 feature maps. The most useful ones are those with the most energy.

We may find the total energy of each representation I_t, by squaring and adding all its values.

One way of identifying which channels should be retained is to reconstruct the original image from the retained channels. To avoid having pixels with negative values, we include in all reconstructions the reconstruction produced by taking the inverse Fourier transform of the frequencies inside the central disk of radius ω_0, shown in Figure 4.40. This central band is not used for texture analysis. Figures 4.41–4.42 show the reconstructions obtained by simply adding an extra channel

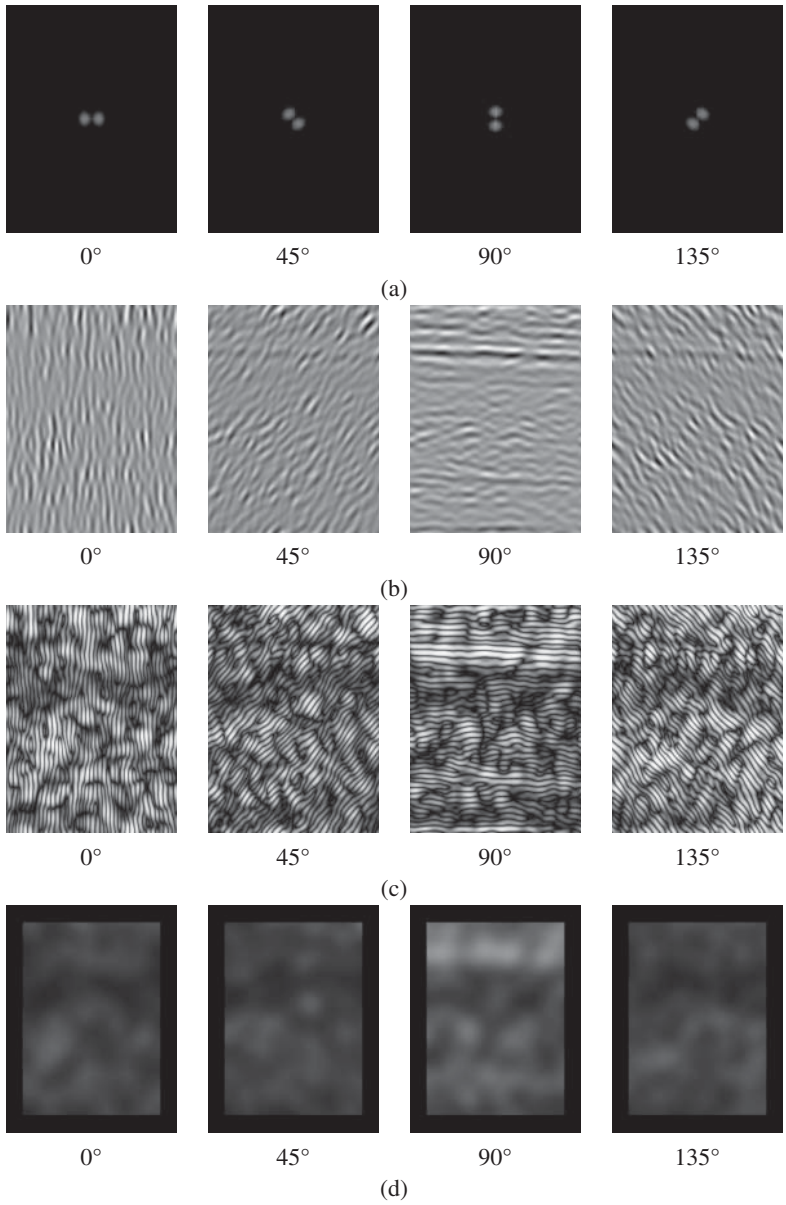


Figure 4.34 Top row: magnitude of the Fourier transform of image 4.33a multiplied with the Gaussian masks for ρ_1. Second row: inverse Fourier transforms of the panels above. Third row: the $\psi(g)$ transforms of the panels above. Fourth row: local energy feature maps computed from the panels above. Source: Maria Petrou.

(Continued)

Example 4.37 (Continued)

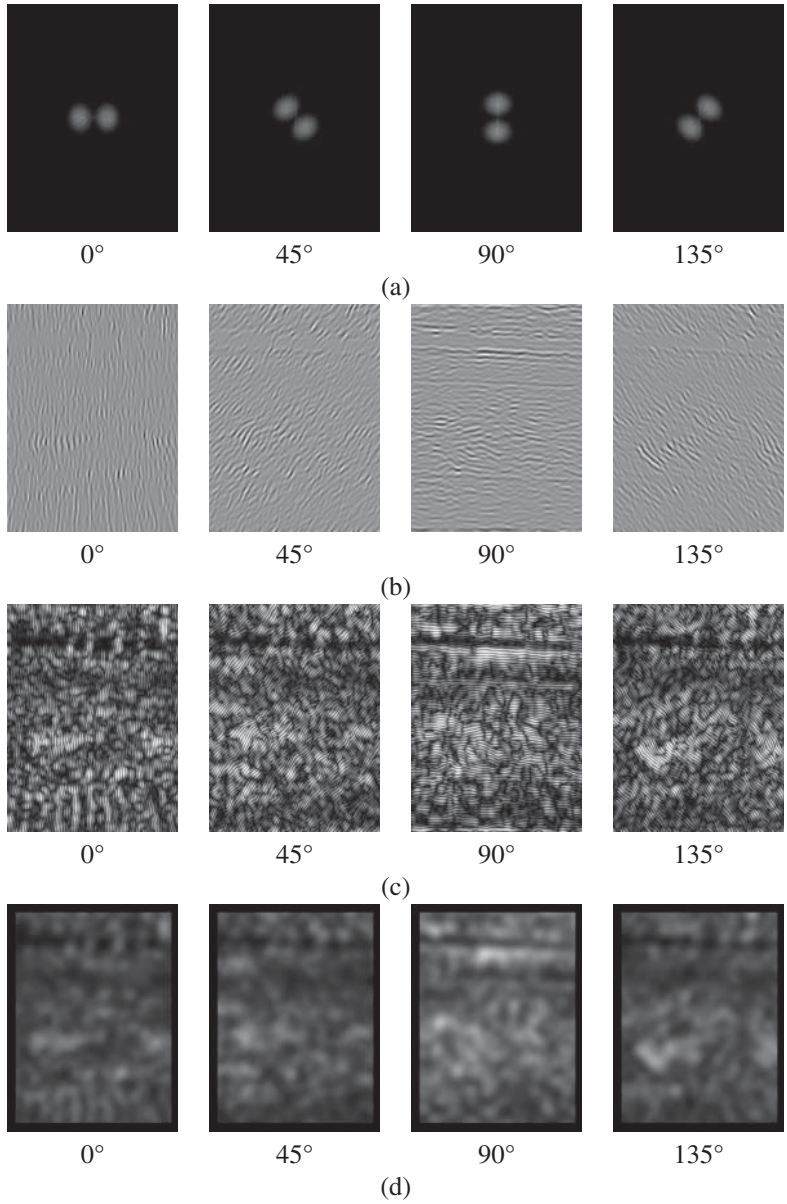


Figure 4.35 Top row: magnitude of the Fourier transform of image 4.33a multiplied with the Gaussian masks for ρ_2. Second row: inverse Fourier transforms of the panels above. Third row: the $\psi(g)$ transforms of the panels above. Fourth row: local energy feature maps computed from the panels above. Source: Maria Petrou.

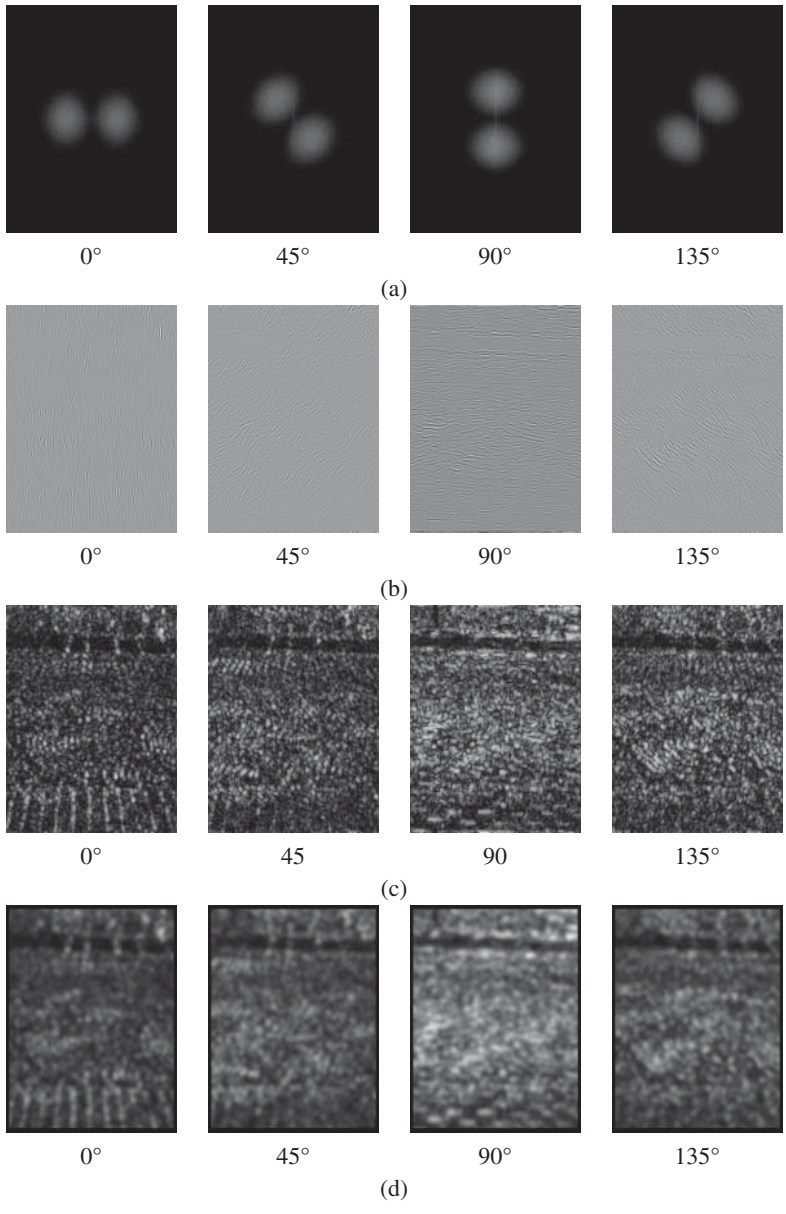


Figure 4.36 Top row: magnitude of the Fourier transform of image 4.33a multiplied with the Gaussian masks for ρ_3. Second row: inverse Fourier transforms of the panels above. Third row: the $\psi(g)$ transforms of the panels above. Fourth row: local energy feature maps computed from the panels above. Source: Maria Petrou.

(Continued)

Example 4.37 (Continued)

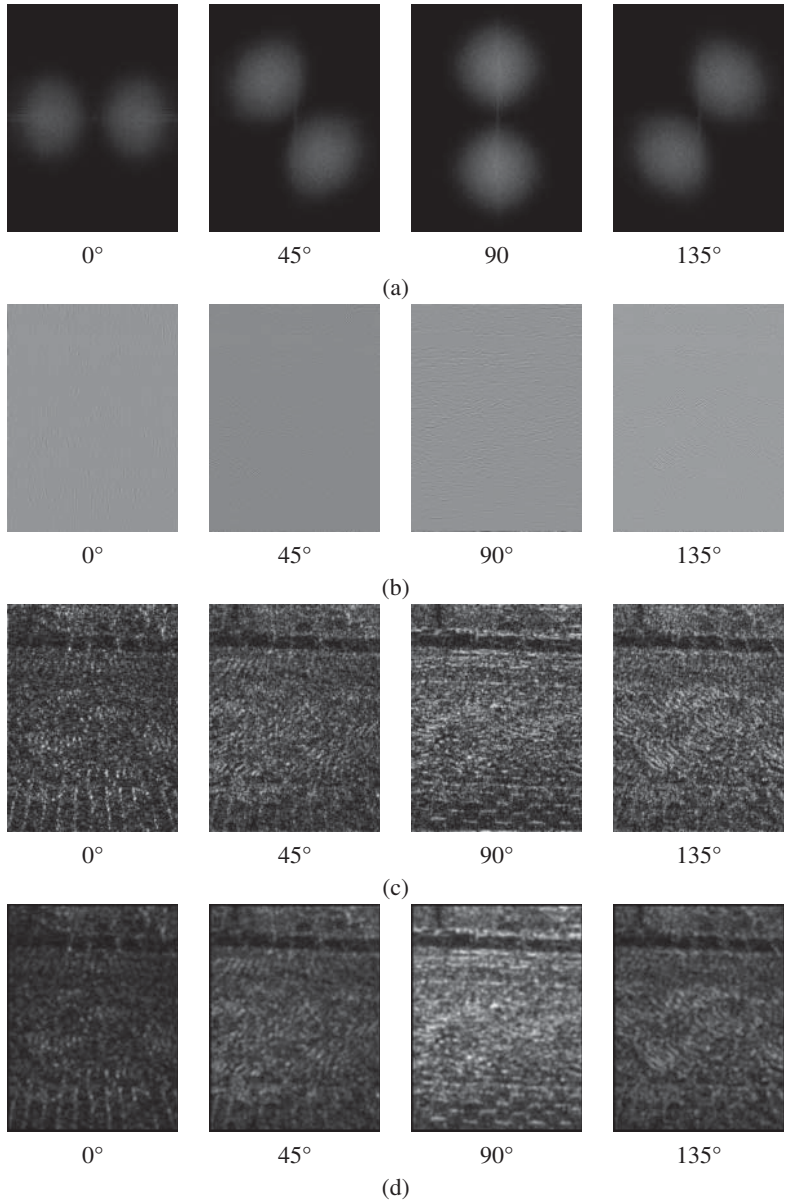


Figure 4.37 Top row: magnitude of the Fourier transform of image 4.33a multiplied with the Gaussian masks for ρ_4. Second row: inverse Fourier transforms of the panels above. Third row: the $\psi(g)$ transforms of the panels above. Fourth row: local energy feature maps computed from the panels above. Source: Maria Petrou.

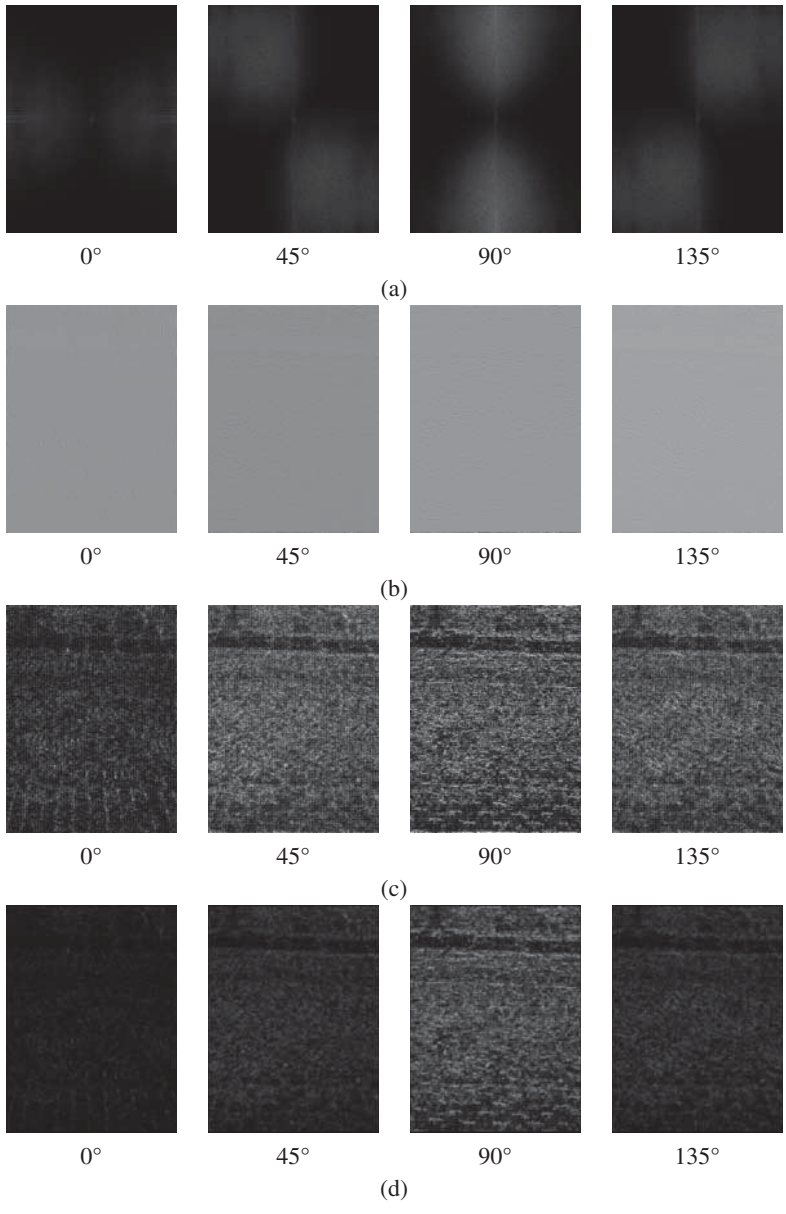


Figure 4.38 Top row: magnitude of the Fourier transform of image 4.33a multiplied with the Gaussian masks for ρ_5. Second row: inverse Fourier transforms of the panels above. Third row: the $\psi(g)$ transforms of the panels above. Fourth row: local energy feature maps computed from the panels above. Source: Maria Petrou.

(Continued)

Example 4.37 (Continued)

at a time in the order they appear along the horizontal axis of Figure 4.39. These cumulative reconstructions are produced by simply adding the reconstructions we created by taking the inverse Fourier transforms of the individual channels. To evaluate how good each reconstruction is, we may compute the sum of the squares of the differences of the values of individual pixels in the reconstructed image and in the original image. Figure 4.43a shows the value of this reconstruction error as a function of the number of channels used, always ranking the channels in order of decreasing energy. Note that this error cannot possibly be zero due to the distortions created by multiplying the various frequency amplitudes with the corresponding values of the Gaussian windows we use. The values of the reconstruction errors are also shown in the second column of Table 4.3. The results of this study indicate that the first 10 channels are adequate in describing the textures in this image.

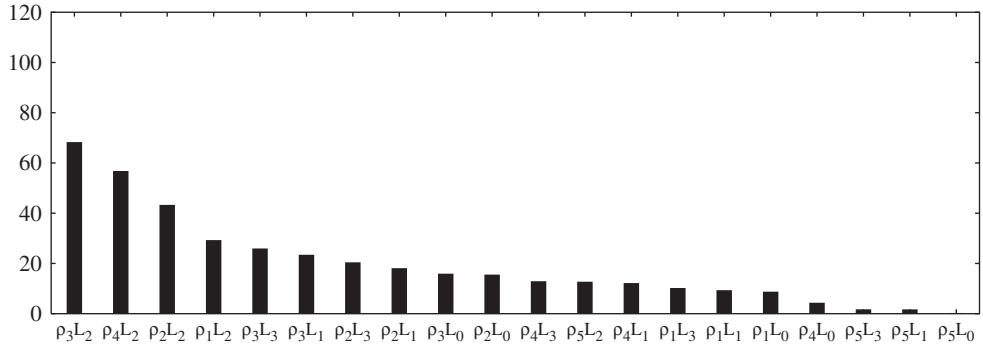


Figure 4.39 Energies of the 20 bands used in the Gabor expansion of the image, ranked in decreasing order. Abscissa $\rho_i L_j$ means the ith radial and jth angular band.

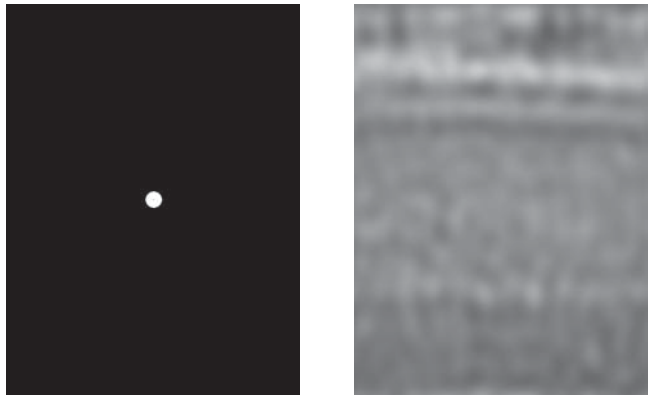


Figure 4.40 The central band containing the dc component of the image and the reconstruction obtained by retaining only those frequencies. Source: Maria Petrou.

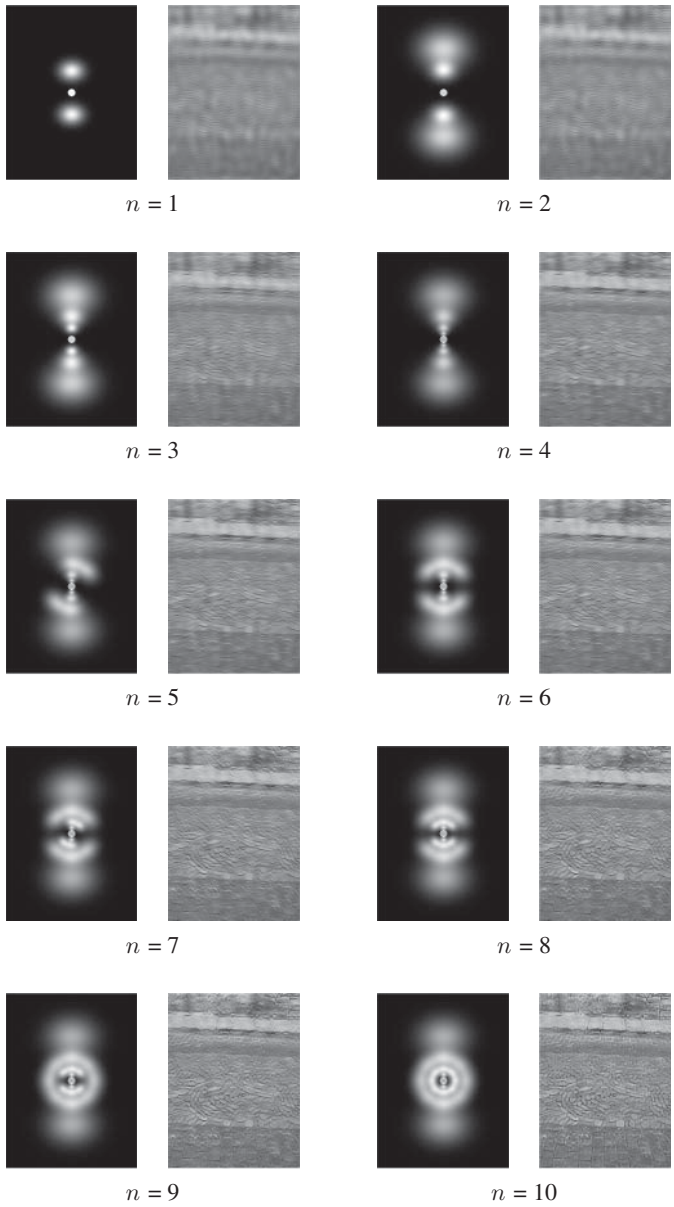


Figure 4.41 For each pair of panels: on the left the Gaussian filters that correspond to the *n* channels with the highest energy, plus the low pass channel, and on the right the image reconstruction produced by summing up the inverse Fourier transforms of these channels. Values of *n* from 1 to 10. Source: Maria Petrou.

(Continued)

Example 4.37 (Continued)

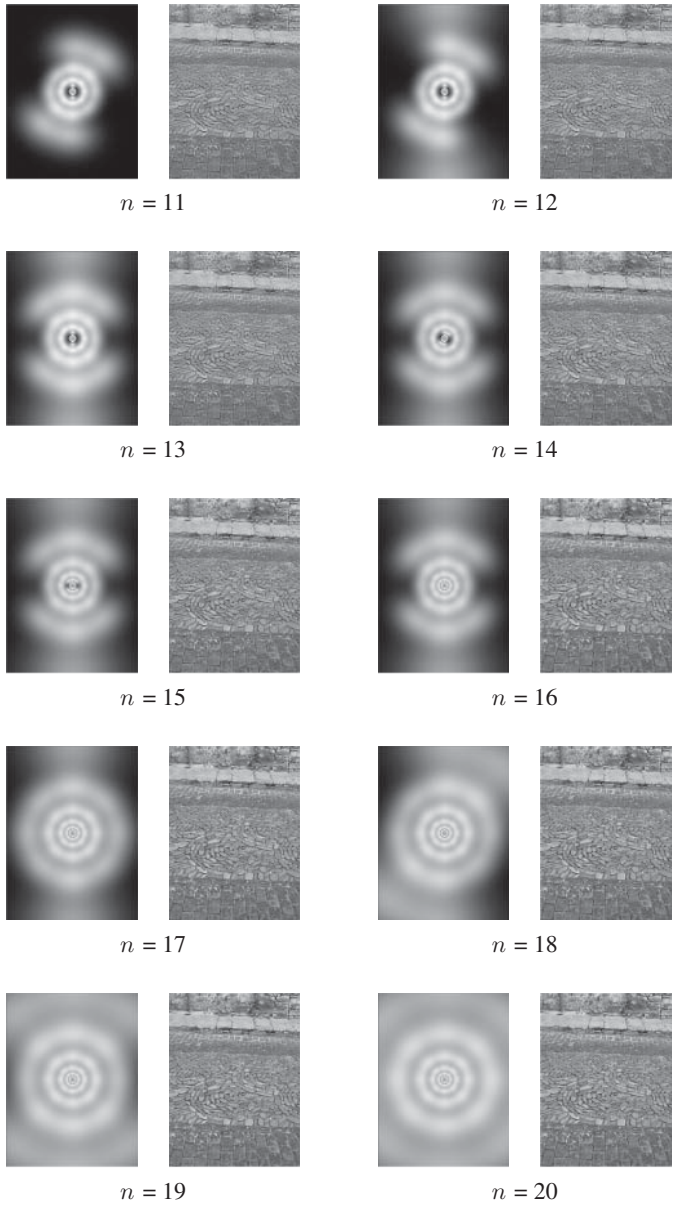


Figure 4.42 For each pair of panels: on the left the Gaussian filters that correspond to the n channels with the highest energy, plus the low pass channel, and on the right the image reconstruction produced by summing up the inverse Fourier transforms of these channels. Values of n from 11 to 20. Source: Maria Petrou.

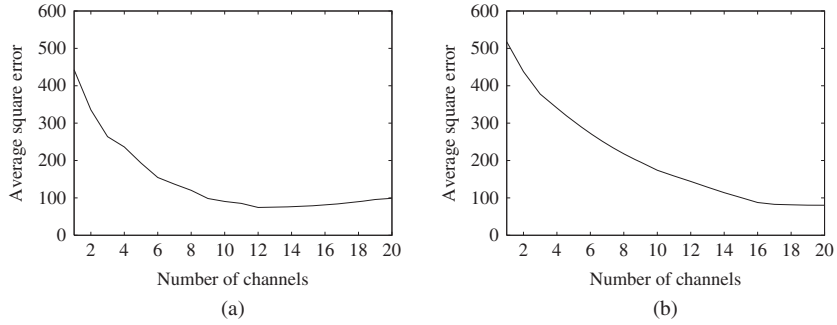


Figure 4.43 Reconstruction error as a function of the channels used for the images in Figures (a) 4.41 and 4.42, and (b) 4.45 and 4.46.

Example 4.38

Reconstruct the image of 4.33a from its bands with the most energy, using the same bands as those used in Example 4.37, but with the Gaussians truncated beyond the limits of the bands. Show the error of the reconstruction as a function of the number of channels used and compare it with the corresponding error in Example 4.37.

In Figure 4.44 we plot the channel energies in decreasing order. We note that there is no change in the order of the first 10 channels from Example 4.37.

The results of the reconstructions are shown in Figures 4.45 and 4.46. The errors of the reconstruction are given in the third column of Table 4.3, next to the errors produced when the tails of the Gaussians outside the bands are not truncated. These errors are plotted in Figure 4.43b.

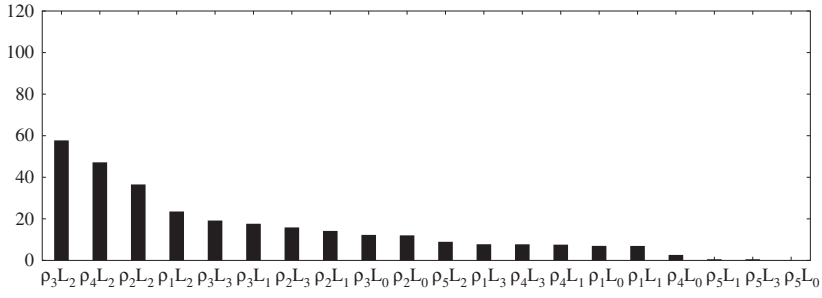


Figure 4.44 Energies of the 20 bands created by truncating the Gaussian masks beyond the band limits, ranked in decreasing order.

(Continued)

Example 4.38 (Continued)

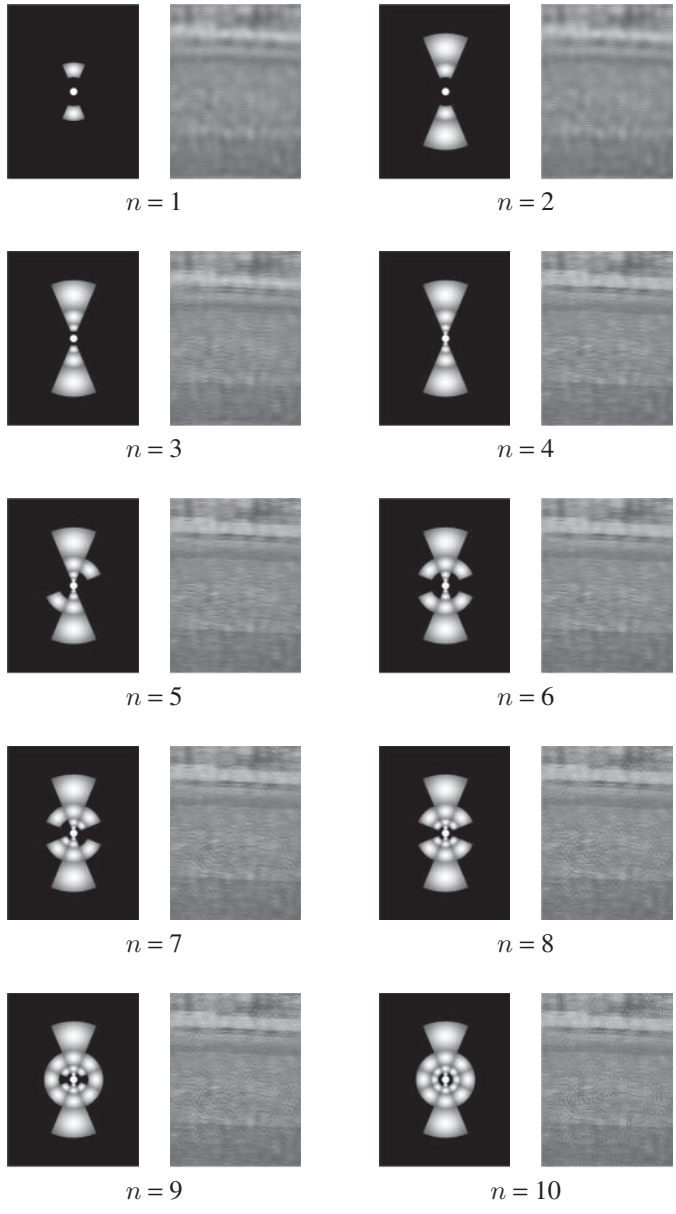


Figure 4.45 For each pair of panels: on the left the truncated Gaussian filters that correspond to the n channels with the highest energy, plus the low pass channel, and on the right the image reconstruction produced by summing up the inverse Fourier transforms of these channels. Values of n from 1 to 10. Source: Maria Petrou.

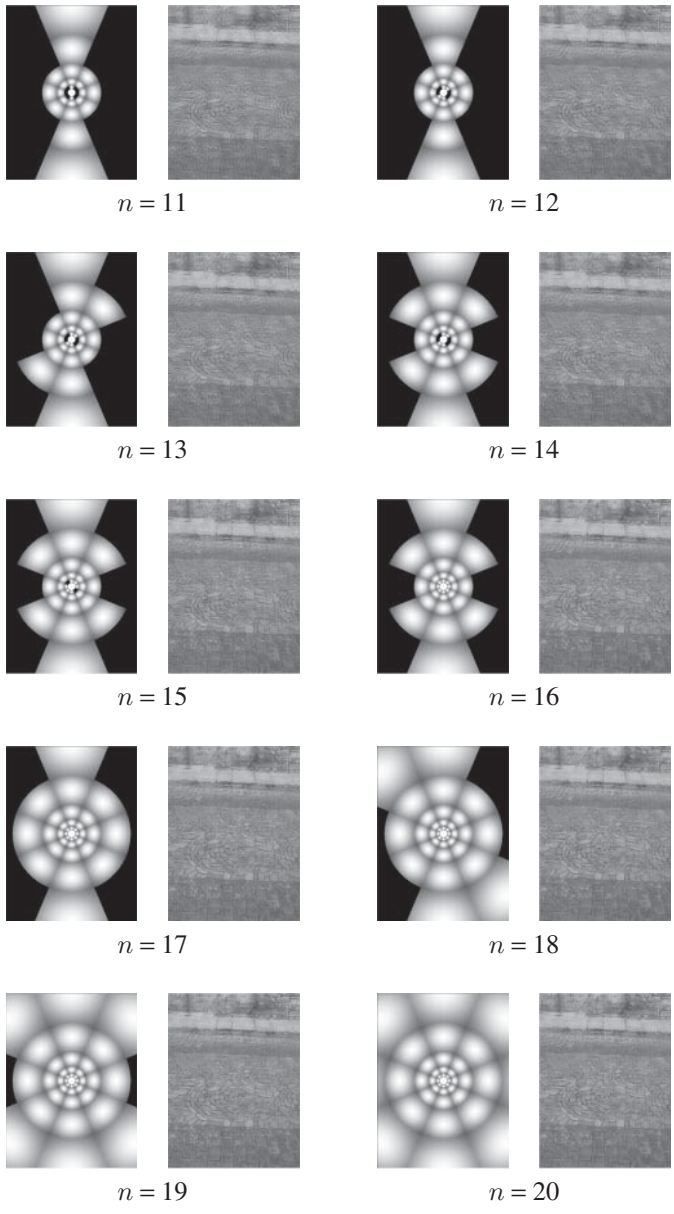


Figure 4.46 For each pair of panels: on the left the truncated Gaussian filters that correspond to the *n* channels with the highest energy, plus the low pass channel, and on the right the image reconstruction produced by summing up the inverse Fourier transforms of these channels. Values of *n* from 11 to 20. Source: Maria Petrou.

Example 4.39

Use the same process as in Example 4.37 to extract feature maps from the image of Figure 4.33a, except now truncate the tails of the Gaussians outside the bands, and keep only the 10 bands with the most energy.

The results of the features constructed from the 10 bands with the highest energy values are shown in Figure 4.47, next to the corresponding ψ transformed bands. All $I_{\psi t}$ representations are shown here scaled globally to the range [0,255]*, so that cross-panel comparisons may be made. The same is true for all energy feature maps.*

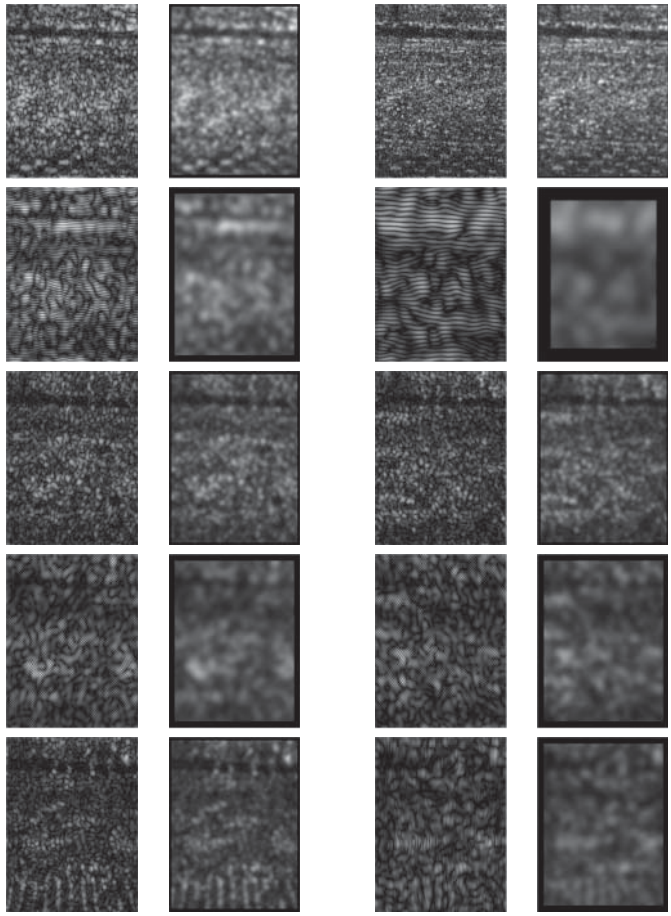


Figure 4.47 Truncated Gaussian windows. The $\psi(g)$ and the corresponding local energy maps extracted for the 10 channels with the highest energy, presented as pairs in decreasing energy order, from top to bottom and left to right. Source: Maria Petrou.

Example 4.40

Reconstruct the image of 4.33a from its bands with the most energy, using the same bands as those used in Example 4.37, but with flat windows that have value 1 inside their bands and 0 beyond the limits of their bands. Show the error of the reconstruction as a function of the number of channels used and compare it with the corresponding error in Examples 4.37 and 4.38.

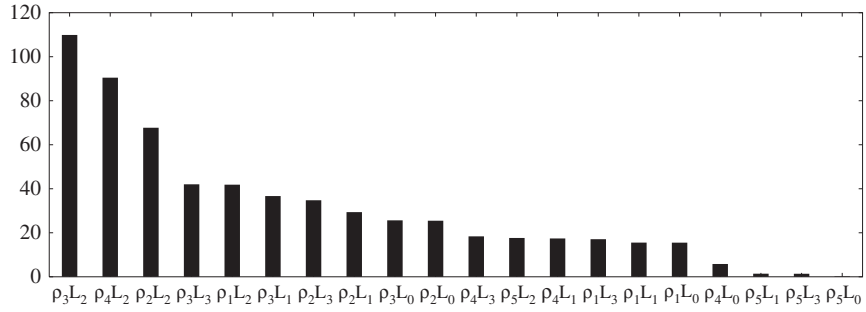


Figure 4.48 Energies of the 20 bands into which the Fourier transform of the image was divided, ranked in decreasing order.

The energy of each channel in decreasing order is shown in Figure 4.48.

We note that the first 10 most significant channels are in the same order as in the case of the Gaussian masks, but this is not true for the remaining channels. This is because the use of the Gaussian masks obviously distorts the computed local energies.

The results of the reconstructions are shown in Figures 4.50 and 4.51.

The errors of the reconstruction are given in the fourth column of Table 4.3, next to the errors produced when the tails of the Gaussians outside the bands are truncated. These errors are also shown in Figure 4.49a. We can see that when no Gaussian window is used, the reconstruction error tends to 0 as more bands are included.

Figure 4.49b shows all three reconstruction errors plotted together (i.e. bands created by multiplying the Fourier transform of the image with Gaussian masks, bands created by multiplying the Fourier transform of the image with truncated Gaussian masks, and bands created by using flat masks).

From this example, and Examples 4.37 and 4.38, we conclude that only the bands with significant energy contain reliable information. In all three examples the image could be adequately reproduced from the first 10 bands only. In fact in Example 4.37 the quality of the image worsened when information beyond the 12th band was included in the reconstruction. This may be used as a rule of thumb to select which channels to keep for image segmentation: the minimum error of reconstruction in the second column of Table 4.3 is at the 12th channel, indicating that only the 12 channels with the most energy should be used to construct features.

(Continued)

Example 4.40 (Continued)

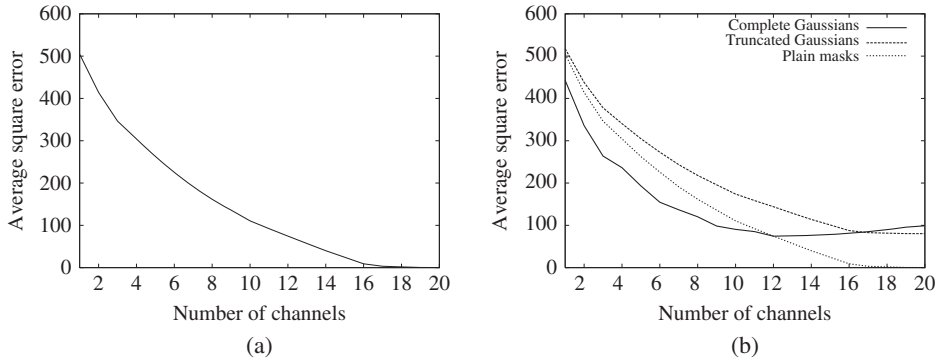


Figure 4.49 (a) Reconstruction error as a function of the channels used for the images in Figures 4.50 and 4.51. (b) Reconstruction errors as functions of the channels used for the three studied cases: complete Gaussian masks, truncated Gaussian masks and flat masks.

Table 4.3 Reconstruction errors for the three studied cases: complete Gaussian masks, truncated Gaussian masks, and flat masks.

Number of channels	Complete Gaussians	Truncated Gaussians	Flat masks
1	442.65	517.59	504.59
2	335.51	437.83	414.13
3	263.79	377.86	346.41
4	236.30	340.50	304.39
5	193.18	304.71	262.53
6	154.58	272.97	225.83
7	136.72	243.40	191.07
8	120.29	218.04	161.70
9	98.39	195.88	136.04
10	90.42	174.02	110.57
11	85.24	158.57	92.20
12	74.34	144.01	74.55
13	75.28	128.66	57.15
14	76.59	113.99	40.07
15	78.15	100.85	24.54
16	81.39	87.69	9.05
17	84.66	82.75	3.24
18	90.11	81.60	1.83
19	95.75	80.47	0.46
20	98.94	80.31	0.25

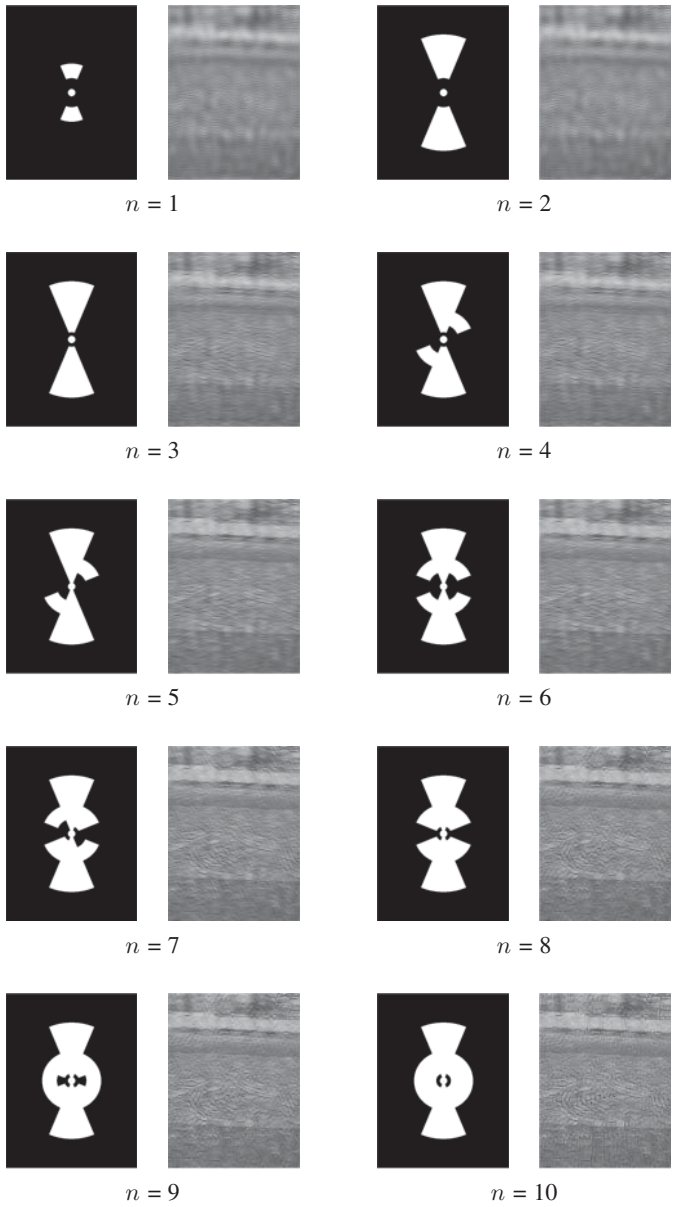
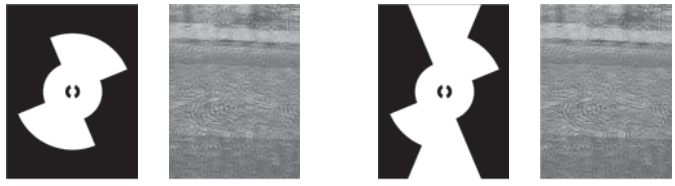


Figure 4.50 For each pair of panels: on the left the flat filters that correspond to the *n* channels with the highest energy, plus the low pass channel, and on the right the image reconstruction produced by summing up the inverse Fourier transforms of these channels. Values of *n* from 1 to 10. Source: Maria Petrou.

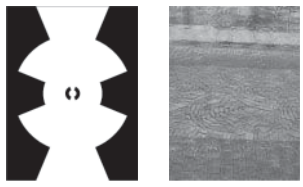
(Continued)

Example 4.40 (Continued)

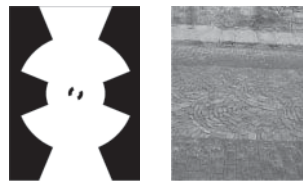


$n = 11$

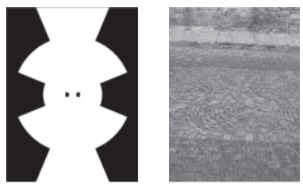
$n = 12$



$n = 13$

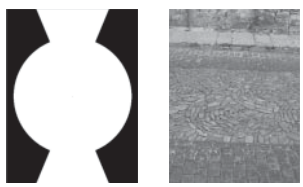


$n = 14$



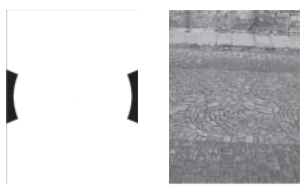
$n = 15$

$n = 16$



$n = 17$

$n = 18$



$n = 19$

$n = 20$

Figure 4.51 For each pair of panels: on the left the flat filters that correspond to the n channels with the highest energy, plus the low pass channel, and on the right the image reconstruction produced by summing up the inverse Fourier transforms of these channels. Values of n from 11 to 20. Source: Maria Petrou.

Example 4.41

Use the same process as in Example 4.37 to extract feature maps from the image of Figure 4.33a, except now use flat windows to isolate the 10 bands with the highest energies.

The results are shown in 4.52. As in the previous example, all $I_{\psi t}$ representations have been scaled together and the same is true for all local energy feature maps.

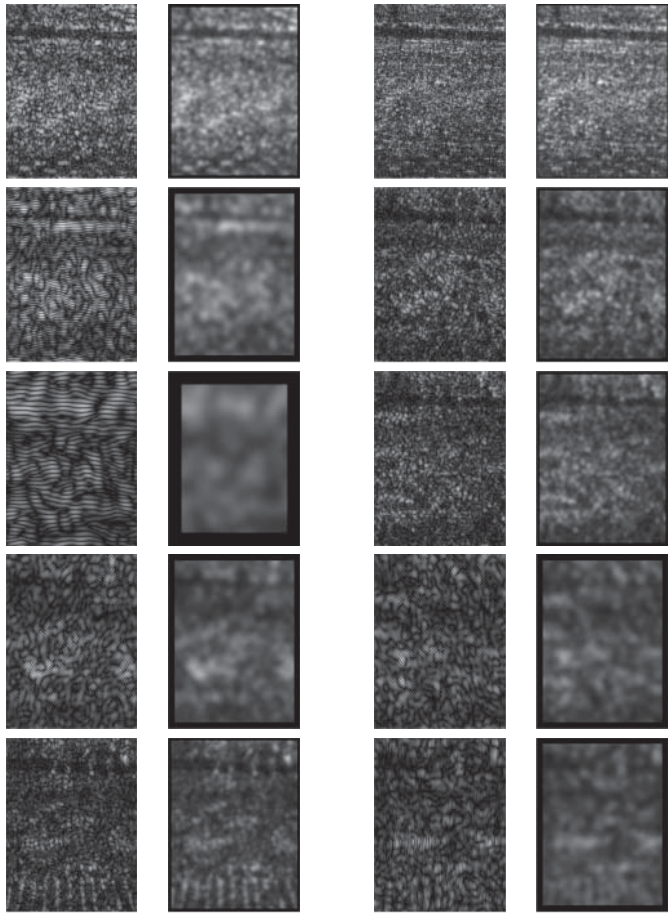


Figure 4.52 Flat windows: The $\psi(g)$ and the corresponding local energy maps extracted for the 10 channels with the highest energy, presented as pairs in decreasing energy order, from top to bottom and left to right. Source: Maria Petrou.

4.3 Prolate Spheroidal Sequence Functions

Is it possible to have a window with sharp edges in one domain that has minimal side ripples in the other domain?

Yes, provided that we do not insist on the window being flat. Let us say that we are interested in a band-limited window and that we allow its Fourier transform to take any values inside its sharp edges. We want to determine these values so that in the real domain the window is maximally concentrated with minimum energy in its side ripples. Solving this problem leads to the **prolate spheroidal sequence functions** which may be used as windows.

Example B4.42

Show that

$$\sum_{n=n_1}^{n_2} \mathrm{e}^{-2A\mathrm{j}n} = \mathrm{e}^{-A\mathrm{j}(n_1+n_2)} \frac{\sin(A(n_2-n_1+1))}{\sin A} \tag{4.174}$$

where A is a constant.

We start by changing summation variable on the left-hand side of this expression to $\tilde{n} \equiv n - n_1 + 1$*:*

$$\sum_{n=n_1}^{n_2} \mathrm{e}^{-2A\mathrm{j}n} = \sum_{\tilde{n}=1}^{n_2-n_1+1} \mathrm{e}^{-2A\mathrm{j}(\tilde{n}+n_1-1)}. \tag{4.175}$$

The sum on the right-hand side is a geometric progression, with first term

$$a \equiv \mathrm{e}^{-2A\mathrm{j}n_1} \tag{4.176}$$

and ratio:

$$q \equiv \mathrm{e}^{-2A\mathrm{j}}. \tag{4.177}$$

The sum of the first N terms of a geometric progression is given by:

$$\sum_{k=1}^{N} aq^{k-1} = \frac{a(q^N-1)}{q-1} \qquad \text{for } q \neq 1. \tag{4.178}$$

If we apply this formula for a and q given by (4.176) and (4.177), respectively, we obtain:

$$\sum_{n=n_1}^{n_2} \mathrm{e}^{-2A\mathrm{j}n} = \frac{\mathrm{e}^{-2A\mathrm{j}n_1}(\mathrm{e}^{-2A\mathrm{j}(n_2-n_1+1)}-1)}{\mathrm{e}^{-2A\mathrm{j}}-1}. \tag{4.179}$$

As q must not be 1, this expression is valid only if $A \neq 0$*.*

Equation (4.179) may further be written as:

$$\sum_{n=n_1}^{n_2} \mathrm{e}^{-2A\mathrm{j}n} = \mathrm{e}^{-2A\mathrm{j}n_1} \frac{\mathrm{e}^{-A\mathrm{j}(n_2-n_1+1)}(\mathrm{e}^{-A\mathrm{j}(n_2-n_1+1)} - \mathrm{e}^{A\mathrm{j}(n_2-n_1+1)})}{\mathrm{e}^{-A\mathrm{j}}(\mathrm{e}^{-A\mathrm{j}} - \mathrm{e}^{A\mathrm{j}})}. \tag{4.180}$$

Upon simplification, we have

$$\sum_{n=n_1}^{n_2} \mathrm{e}^{-2A\mathrm{j}n} = \mathrm{e}^{-A\mathrm{j}(2n_1+n_2-n_1+1-1)} \frac{\mathrm{e}^{-A\mathrm{j}(n_2-n_1+1)} - \mathrm{e}^{A\mathrm{j}(n_2-n_1+1)}}{\mathrm{e}^{-A\mathrm{j}} - \mathrm{e}^{A\mathrm{j}}} \tag{4.181}$$

which further may be simplified to

$$\sum_{n=n_1}^{n_2} \mathrm{e}^{-2\mathrm{j}An} = \mathrm{e}^{-A\mathrm{j}(n_1+n_2)} \frac{-2\mathrm{j}\sin(A(n_2-n_1+1))}{-2\mathrm{j}\sin A} \tag{4.182}$$

which yields (4.174) as long as $A \neq 0$.

If $A = 0$, *the sum is:*

$$\sum_{n=n_1}^{n_2} \mathrm{e}^{-2\mathrm{j}An} = \sum_{n=n_1}^{n_2} 1 = n_2 - n_1 + 1. \tag{4.183}$$

If we set $A = 0$ *on the right-hand side of (4.182), the exponential factor becomes 1, and we have to take the limit of the trigonometric factor using de L'Hospital's rule:*

$$\lim_{A\to 0} \frac{\sin(A(n_2-n_1+1))}{\sin A} = \lim_{A\to 0} \frac{(n_2-n_1+1)\cos(A(n_2-n_1+1))}{\cos A} = n_2 - n_1 + 1. \tag{4.184}$$

If this result is substituted into (4.182) we obtain again (4.183). Therefore, we may say that Equation (4.174) is valid even when $A = 0$, *provided we realise that in this case the ratio of the two trigonometric factors is equal to* $n_2 - n_1 + 1$.

Box 4.2 Of all the band-limited sequences one can define, which sequence has the maximum energy concentration between a given set of indices?

Let us say that the unknown sequence we wish to identify consists of numbers $w(n)$. The Fourier transform of this sequence is given by:

$$W(\omega) = \sum_{n=-\infty}^{\infty} w(n)\mathrm{e}^{-\mathrm{j}\omega n}. \tag{4.185}$$

The inverse Fourier transform of $W(\omega)$ is given by:

$$w(n) = \frac{1}{2\pi}\int_{-\infty}^{\infty} W(\omega)\mathrm{e}^{\mathrm{j}\omega n}\mathrm{d}\omega. \tag{4.186}$$

Even if we assume that $W(\omega)$ is band-limited with bandwidth 2Ω, in which case $w(n) = \frac{1}{2\pi}\int_{-\Omega}^{\Omega} W(\omega)\mathrm{e}^{\mathrm{j}\omega n}\mathrm{d}\omega$, $w(n)$ may be defined for any value of n in $(-\infty, \infty)$. So sequence $w(n)$ is infinite. Its total energy is given by

$$E_\infty \equiv \sum_{n=-\infty}^{\infty} |w(n)|^2 = \frac{1}{2\pi}\int_{-\Omega}^{\Omega} |W(\omega)|^2 \mathrm{d}\omega \tag{4.187}$$

where we made use of Parseval's theorem (see Example 4.4.

The energy between two specific indices n_1 and n_2 is obviously equal to:

$$E_{n_1,n_2} \equiv \sum_{n=n_1}^{n_2} |w(n)|^2. \tag{4.188}$$

The idea is to choose the values of $w(n)$ (or $W(\omega)$) so that the energy between indices n_1 and n_2 is as large a fraction of the total energy of the sequence as possible. Choose $W(\omega)$ so that

(Continued)

Box 4.2 (Continued)

we maximise:

$$R \equiv \frac{E_{n_1,n_2}}{E_\infty}. \tag{4.189}$$

Upon substitution from (4.187) and (4.188), we obtain

$$R = \frac{\sum_{n=n_1}^{n_2} |w(n)|^2}{\sum_{n=-\infty}^{\infty} |w(n)|^2} = \frac{\sum_{n=n_1}^{n_2} w(n)w^*(n)}{\frac{1}{2\pi}\int_{-\Omega}^{\Omega} |W(\omega)|^2 \mathrm{d}\omega} \tag{4.190}$$

where $w^*(n)$ means the complex conjugate of $w(n)$. We may substitute $w(n)$ and $w^*(n)$ in the numerator from Equation (4.186)

$$R = \frac{\sum_{n=n_1}^{n_2} \int_{-\Omega}^{\Omega} W(\omega)\mathrm{e}^{\mathrm{j}\omega n}\mathrm{d}\omega \int_{-\Omega}^{\Omega} W^*(\omega')\mathrm{e}^{-\mathrm{j}\omega' n}\mathrm{d}\omega'}{2\pi\int_{-\Omega}^{\Omega} |W(\omega)|^2 \mathrm{d}\omega} \tag{4.191}$$

where we changed the variable of integration in the second integral of the numerator so that there is no confusion with the variable of integration in the first integral.

We notice that in the numerator the factors that depend on the summed index are only the two exponentials. So, we may write:

$$R = \frac{\int_{-\Omega}^{\Omega}\int_{-\Omega}^{\Omega} W(\omega)W^*(\omega')\sum_{n=n_1}^{n_2} \mathrm{e}^{\mathrm{j}(\omega-\omega')n}\mathrm{d}\omega\mathrm{d}\omega'}{2\pi\int_{-\Omega}^{\Omega} |W(\omega)|^2 \mathrm{d}\omega}. \tag{4.192}$$

Here we make use of (4.174) with $A \equiv \pi(f - f')$, to substitute for the sum in the numerator. We obtain

$$R = \frac{\int_{-f_0}^{f_0}\int_{-f_0}^{f_0} W(f)W^*(f')\mathrm{e}^{-\pi\mathrm{j}(f-f')(n_1+n_2)}\frac{\sin(\pi(f-f')(n_2-n_1+1))}{\sin(\pi(f-f'))}\mathrm{d}f\mathrm{d}f'}{\int_{-f_0}^{f_0} |W(f)|^2 \mathrm{d}f} \tag{4.193}$$

where we also replaced integration over ω with integration over $f \equiv \omega/(2\pi)$, and defined $f_0 \equiv \Omega/(2\pi)$.

Let us define:

$$\psi(f) \equiv W(f)\mathrm{e}^{-\pi\mathrm{j}f(n_1+n_2)}. \tag{4.194}$$

Then

$$\psi^*(f) = W^*(f)\mathrm{e}^{\pi\mathrm{j}f(n_1+n_2)} \tag{4.195}$$

and upon multiplication we deduce that:

$$\psi(f)\psi^*(f) = |W(f)|^2. \tag{4.196}$$

Using (4.194)–(4.196) in (4.193), we obtain:

$$R = \frac{\int_{-f_0}^{f_0}\int_{-f_0}^{f_0} \psi(f)\psi^*(f')\frac{\sin(\pi(f-f')(n_2-n_1+1))}{\sin(\pi(f-f'))}\mathrm{d}f'\mathrm{d}f}{\int_{-f_0}^{f_0} |\psi(f)|^2 \mathrm{d}f}. \tag{4.197}$$

Let us call:

$$g(f) \equiv \int_{-f_0}^{f_0} \psi^*(f')\frac{\sin(\pi(f-f')(n_2-n_1+1))}{\sin(\pi(f-f'))}\mathrm{d}f'. \tag{4.198}$$

Then:

$$R = \frac{\int_{-f_0}^{f_0} \psi(f) g(f) \mathrm{d}f}{\int_{-f_0}^{f_0} |\psi(f)|^2 \mathrm{d}f}. \tag{4.199}$$

Our problem now is to choose $\psi(f)$ so that R is maximal. This type of optimisation problem, where we have an integral in the numerator and an integral in the denominator, both with the same limits, is called an **isoperimetric problem**. We may find the extreme value of the ratio of the two integrals by trying to extremise one of the integrals while keeping the other constant.

Our problem then may be expressed as: minimise $\int_{-f_0}^{f_0} |\psi(f)|^2 \mathrm{d}f$ with the constraint that $\int_{-f_0}^{f_0} \psi(f) g(f) \mathrm{d}f$ is a constant.

This problem can be solved with **Euler's method of Lagrange multipliers**: define a function $z \equiv |\psi|^2 + \mu\psi g$; the solution of the constrained minimisation problem then is given by the solution of the partial differential equation

$$\frac{\partial z}{\partial \psi} = 0 \Rightarrow 2\psi^* + \mu g = 0 \Rightarrow g = -\frac{2}{\mu}\psi^* \Rightarrow g = \lambda\psi^* \tag{4.200}$$

where for simplicity we called $-2/\mu \equiv \lambda$.

If we substitute into (4.199), we obtain:

$$R = \frac{\lambda \int_{-f_0}^{f_0} \psi(f)\psi^*(f) \mathrm{d}f}{\int_{-f_0}^{f_0} |\psi(f)|^2 \mathrm{d}f} = \lambda. \tag{4.201}$$

And if we substitute from (4.198) into (4.200) we obtain:

$$\int_{-f_0}^{f_0} \psi^*(f') \frac{\sin(\pi(f - f')(n_2 - n_1 + 1))}{\sin(\pi(f - f'))} \mathrm{d}f' = \lambda\psi^*(f). \tag{4.202}$$

The solutions of this equation are called **prolate spheroidal wave functions**. This is an eigenvalue type problem, with as many solutions as the length of the index sequence $n_2 - n_1 + 1$, each one of them corresponding to a different value of λ. Let us order these eigenvalues λ in decreasing order. The first one of them will be the largest and as we see from (4.201) it will maximise R.

Therefore, the solution of our problem is as follows.

Step 1: From (4.202) find the $\psi(f)$ that corresponds to the largest eigenvalue.
Step 2: From (4.194) find the corresponding $W(f)$.
Step 3: From (4.186) compute the weights $w(n)$ of the window in the image domain.
Step 4: The window produced this way is exactly band limited and in the image domain has side-lobes of minimum amplitude outside the range of the chosen indices $[n_1, n_2]$, so that it may be truncated with minimum error to produce a finite window of size $n_2 - n_1 + 1$.

Box 4.3 Do prolate spheroidal wave functions exist in the digital domain?

Yes, and they are called **prolate spheroidal sequence functions**.

Let us consider a digital signal $w(n)$, where n takes values in the range $\left[-\frac{N-1}{2}, \ldots, 0, \ldots, \frac{N-1}{2}\right]$, with N being odd and indicating the total number of samples of the signal. If $W(k)$ is the

(Continued)

Box 4.3 (Continued)

discrete Fourier transform of the signal, then we may write

$$W(k) = \sum_{n=-\frac{N-1}{2}}^{\frac{N-1}{2}} w(n) \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}kn} \tag{4.203}$$

and

$$w(n) = \frac{1}{N} \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} W(k) \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}kn}. \tag{4.204}$$

The total energy of this signal, after making use of Parseval's theorem, may be written as:

$$E_{\infty} = \sum_{n=-\frac{N-1}{2}}^{\frac{N-1}{2}} |w(n)|^2 = \frac{1}{N} \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} |W(k)|^2. \tag{4.205}$$

The energy of the signal between two indices n_1 and n_2 is

$$E_{n_1 n_2} = \sum_{n=n_1}^{n_2} |w(n)|^2 = \sum_{n=n_1}^{n_2} w(n) w^*(n) \tag{4.206}$$

where $w^*(n)$ is the complex conjugate of signal $w(n)$ and by using Equation (4.204) it may be written as:

$$w^*(n) = \frac{1}{N} \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} W^*(k) \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}kn}. \tag{4.207}$$

The fraction of the energy of the signal between indices n_1 and n_2 is:

$$\begin{aligned} R &\equiv \frac{\sum_{n=n_1}^{n_2} w(n) w^*(n)}{\frac{1}{N} \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} |W(k)|^2} \\ &= \frac{\sum_{n=n_1}^{n_2} \frac{1}{N} \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} W(k) \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}kn} \frac{1}{N} \sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} W^*(l) \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}ln}}{\frac{1}{N} \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} |W(k)|^2} \\ &= \frac{\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} \sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} W(k) W^*(l) \sum_{n=n_1}^{n_2} \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}(k-l)n}}{N \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} |W(k)|^2}. \end{aligned} \tag{4.208}$$

We may make use of (4.174) with $A \equiv \frac{\pi}{N}(k-l)$ in order to substitute for sum $\sum_{n=n_1}^{n_2} \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}(k-l)n}$:

$$R = \frac{\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} \sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} W(k) W^*(l) \mathrm{e}^{\mathrm{j}\frac{\pi}{N}(k-l)(n_1+n_2)} \frac{\sin\left(\frac{\pi}{N}(k-l)(n_2-n_1+1)\right)}{\sin\left(\frac{\pi}{N}(k-l)\right)}}{N \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} |W(k)|^2}. \tag{4.209}$$

We define next

$$\psi(k) \equiv W(k)\mathrm{e}^{\mathrm{j}\frac{\pi}{N}k(n_1+n_2)} \tag{4.210}$$

and as a consequence:

$$\psi^*(k) = W^*(k)\mathrm{e}^{-\mathrm{j}\frac{\pi}{N}k(n_1+n_2)}. \tag{4.211}$$

We observe that $\psi(k)\psi^*(k) = W(k)W^*(k) = |W(k)|^2$. Then:

$$R = \frac{\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} \sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} \psi(k)\psi^*(l)\frac{\sin\left(\frac{\pi}{N}(k-l)(n_2-n_1+1)\right)}{\sin\left(\frac{\pi}{N}(k-l)\right)}}{N\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} |\psi(k)|^2}. \tag{4.212}$$

Let us call:

$$g(k) \equiv \sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} \psi^*(l)\frac{\sin\left(\frac{\pi}{N}(k-l)(n_2-n_1+1)\right)}{\sin\left(\frac{\pi}{N}(k-l)\right)}. \tag{4.213}$$

Then:

$$R = \frac{\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} \psi(k)g(k)}{N\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} |\psi(k)|^2}$$

$$\Rightarrow RN\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} \psi(k)\psi^*(k) = \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} \psi(k)g(k)$$

$$\Rightarrow \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} \psi(k)[RN\psi^*(k) - g(k)] = 0. \tag{4.214}$$

This is true for all $\psi(k)$ only if:

$$g(k) = RN\psi^*(k). \tag{4.215}$$

If we substitute $g(k)$ from (4.213) into (4.215), we derive:

$$\sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} \psi^*(l)\frac{\sin\left(\frac{\pi}{N}(k-l)(n_2-n_1+1)\right)}{\sin\left(\frac{\pi}{N}(k-l)\right)} = RN\psi^*(k). \tag{4.216}$$

We may eliminate the asterisk from this expression by taking its complex conjugate and observing that, apart from $\psi(k)$, all other quantities involved are real. Let us define:

$$S(k,l) \equiv \frac{\sin\left(\frac{\pi}{N}(k-l)(n_2-n_1+1)\right)}{\sin\left(\frac{\pi}{N}(k-l)\right)}. \tag{4.217}$$

Then in order to define $\psi(k)$, and as a consequence $W(k)$, we must solve the equation:

$$\sum_{l=-\frac{N-1}{2}}^{\frac{N-1}{2}} \psi(l)S(k,l) = RN\psi(k) \Rightarrow S\boldsymbol{\psi} = RN\boldsymbol{\psi}. \tag{4.218}$$

(Continued)

Box 4.3 (Continued)

This is an eigenvalue problem: $\psi(k)$, which maximises R, is the eigenvector ψ of matrix S, which corresponds to the largest eigenvalue RN. The solutions of this equation constitute the **prolate spheroidal sequence functions**.

Note that matrix S is much smaller than the ranges of values of indices k and l indicate above: as we wish to examine the local frequency content of a signal in a particular band, we fix $W(k)$ to be 0 for all other indices k, apart from those that correspond to the band of interest. Then we try to specify the values of $W(k)$ (i.e. $\psi(k)$) only inside the band of interest. So, in practice, the range of values of k and l is not from $-(N-1)/2$ to $(N-1)/2$, but from, say, k_1 to k_2, and, therefore, matrix S is of size $(k_2-k_1+1)\times(k_2-k_1+1)$. So, the equation we have to solve is

$$\sum_{l=k_1}^{k_2}\psi(l)\frac{\sin\left(\frac{\pi}{N}(k-l)(n_2-n_1+1)\right)}{\sin\left(\frac{\pi}{N}(k-l)\right)}=RN\psi(k) \tag{4.219}$$

with $k\in[k_1,k_2]$.

We also observe that in the calculation of ψ, all that matters is the length of the sub-sequence (i.e. n_2-n_1+1) that has most of the energy and it does not really matter where exactly within the whole sequence this sub-sequence is. So, as long as the length of the sub-sequence with the most energy is fixed, the value of ψ is also fixed. The exact location of the sub-sequence enters only in the calculation of $W(k)$ from the values of ψ (see factor (n_1+n_2) in the exponent of Equation (4.210)). The exact location of the sub-sequence, therefore, appears as phase change of the DFT of the sequence. This is as expected, since shifting the sub-sequence should indeed result in a phase shift of its Fourier transform.

The implication of the above observation is that we may compute $W(k)$, and from it $w(n)$, only once, truncate $w(n)$ by keeping only the sub-sequence with the most energy, and use it as a convolution filter with any other sequence, in order to extract the local frequency content of the input signal within the chosen band.

In summary: to construct a band-limited digital filter in the frequency range $[k_1,k_2]$, with most of its energy in indices in the range $[n_1,n_2]$, suitable for the analysis of a digital signal of length N, with N being odd, we follow these steps.

Step 1: Construct matrix S using Equation (4.217), for $k,l\in[k_1,k_2]$.
Step 2: Find the largest eigenvalue of S.
Step 3: Compute the corresponding eigenvector ψ, with elements $\psi(k)$.
Step 4: Use Equation (4.210) to compute $W(k)$.
Step 5: Use Equation (4.204) to compute $w(n)$.
Step 6: Sequence $w(n)$ is a digital convolution filter, the same size as the signal. If an unknown signal is convolved with it, the DFT of the signal will be multiplied point by point with the DFT ($W(k)$) of this filter. The result will be the frequency content of the signal in the band where $W(k)$ is non-zero. To reduce the computational cost, we may truncate the filter and keep only its values in the range $[n_1,n_2]$.

Example B4.43

Consider a sequence $w(-5), w(-4), \ldots, w(5)$**, and its discrete Fourier transform** $W(-5), W(-4), \ldots, W(5)$**. Assume that** $W(0), W(1), \ldots, W(4)$ **are free to take any values, and that** $W(-5) = W(-4) = W(-3) = W(-2) = W(-1) = W(5) = 0$**. What values** $W(0), W(1), \ldots, W(4)$ **should be taken so that sequence** $w(-5), w(-4), \ldots, w(5)$ **has most of its energy in indices** $-4, -3$ **and** -2**?**

Here $N = 11$*,* $n_1 = -4$*,* $n_2 = -2$*,* $k_1 = 0$ *and* $k_2 = 4$*. Therefore* $n_2 - n_1 + 1 = 3$*, and Equation (4.219) takes the form:*

$$\sum_{l=0}^{4} \psi(l) \frac{\sin\left(\frac{3\pi}{11}(k-l)\right)}{\sin\left(\frac{\pi}{11}(k-l)\right)} = 11R\psi(k). \tag{4.220}$$

We may write this equation more explicitly as:

$$\begin{bmatrix} 3 & \frac{\sin(3\pi/11)}{\sin(\pi/11)} & \frac{\sin(6\pi/11)}{\sin(2\pi/11)} & \frac{\sin(9\pi/11)}{\sin(3\pi/11)} & \frac{\sin(12\pi/11)}{\sin(4\pi/11)} \\ \frac{\sin(3\pi/11)}{\sin(\pi/11)} & 3 & \frac{\sin(3\pi/11)}{\sin(\pi/11)} & \frac{\sin(6\pi/11)}{\sin(2\pi/11)} & \frac{\sin(9\pi/11)}{\sin(3\pi/11)} \\ \frac{\sin(6\pi/11)}{\sin(2\pi/11)} & \frac{\sin(3\pi/11)}{\sin(\pi/11)} & 3 & \frac{\sin(3\pi/11)}{\sin(\pi/11)} & \frac{\sin(6\pi/11)}{\sin(2\pi/11)} \\ \frac{\sin(9\pi/11)}{\sin(3\pi/11)} & \frac{\sin(6\pi/11)}{\sin(2\pi/11)} & \frac{\sin(3\pi/11)}{\sin(\pi/11)} & 3 & \frac{\sin(3\pi/11)}{\sin(\pi/11)} \\ \frac{\sin(12\pi/11)}{\sin(4\pi/11)} & \frac{\sin(9\pi/11)}{\sin(3\pi/11)} & \frac{\sin(6\pi/11)}{\sin(2\pi/11)} & \frac{\sin(3\pi/11)}{\sin(\pi/11)} & 3 \end{bmatrix} \begin{bmatrix} \psi(0) \\ \psi(1) \\ \psi(2) \\ \psi(3) \\ \psi(4) \end{bmatrix}$$

$$= 11R \begin{bmatrix} \psi(0) \\ \psi(1) \\ \psi(2) \\ \psi(3) \\ \psi(4) \end{bmatrix}. \tag{4.221}$$

The above matrix turns out to have only three non-zero eigenvalues, which are $\lambda_1 = 10.236$*,* $\lambda_2 = 4.479$ *and* $\lambda_3 = 0.285$*. The eigenvector that corresponds to the largest eigenvalue is:* $\psi = (0.348, 0.485, 0.536, 0.485, 0.348)^T$.

Having computed $\psi(k)$*, we use now Equation (4.210) to compute* $W(k)$*:*

$$W(k) = \psi(k) e^{j\frac{\pi}{11}6k}. \tag{4.222}$$

We obtain: $W(k) = (0.0, 0.0, 0.0, 0.0, 0.0, 0.348, -0.069 + 0.480j, -0.514 - 0.151j, 0.201 - 0.441j, 0.293 + 0.188j, 0.0)$.

Example B4.44

Verify that in Example 4.43 sequence $w(-5), \dots, w(5)$ has indeed the predicted fraction of its energy in indices -4, -3 and -2.

Using the derived values of $W(k)$ and Equation (4.204) we derive first sequence $w(n)$:

$$w(n) = \frac{1}{11}\sum_{k=-5}^{5} W(k)e^{\frac{2\pi}{11}kn} \qquad \text{for } n = -5, -4, \dots, 5. \tag{4.223}$$

We obtain:

$w(n) = (-0.316 - 0.365j, 0.682 - 1.493j, 2.202, 0.682 + 1.493j, -0.316 + 0.365j, 0.260 + 0.076j, 0.028 + 0.196j, 0.161 - 0.103j, 0.161 + 0.103j, 0.028 - 0.196j, 0.260 - 0.076j)$.

The total energy of this sequence is

$$E_{-5,5} = w(-5)w^*(-5) + w(-4)w^*(-4) + \dots + w(5)w^*(5) = 11. \tag{4.224}$$

The energy in indices -4, -3 and -2 is

$$E_{-4,-2} = w(-4)w^*(-4) + w(-3)w^*(-3) + w(-2)w^*(-2) = 10.236. \tag{4.225}$$

Therefore, the fraction of the total energy in these indices is

$$R = \frac{E_{-4,-2}}{E_{-5,5}} = 0.931. \tag{4.226}$$

Note that the largest eigenvalue of matrix S, which we computed in Example 4.43, was $\lambda_1 = 10.236$, and it is equal to $11R$. From this we derive $R = 10.236/11 = 0.931$ which verifies the above result.

Example B4.45

Consider a sequence $w'_{-5}, w'_{-4}, \dots, w'_5$, and its discrete Fourier transform $W'(-5)$, $W'(-4), \dots, W'(5)$. Assume that $W'(0), W'(1), \dots, W'(4)$ are free to take any values and that $W'(-5) = W'(-4) = W'(-3) = W'(-2) = W'(-1) = W'(5) = 0$. What values should $W'(0), W'(1), \dots, W'(4)$ take so that sequence $w'_{-5}, w'_{-4}, \dots, w'_5$ has most of its energy in indices 2, 3 and 4? Compute sequence w'_n and compare it with the sequence of Example 4.44.

This problem is very similar to that of Example 4.43, except here we are interested in having most of the energy of the constructed sequence in indices 2, 3 and 4 instead of -4, -3 and -2. The lengths of the two sub-sequences are identical, so nothing changes in the calculation of ψ. The change comes in the calculation of $W'(k)$, which now is given by

$$W'(k) = \psi(k)e^{-j\frac{\pi}{11}6k} \tag{4.227}$$

since we have to set in (4.210) $n_1 + n_2 = 6$, instead of -6 we used in Example 4.44.

We obtain:

$W'(k) = (0.0, 0.0, 0.0, 0.0, 0.0, 0.348, -0.069 - 0.480j, -0.514 + 0.151j, 0.201 + 0.441j, 0.293 - 0.188j, 0.0)$. *From this we derive sequence w'_n using Equation (4.223) as:* $w'_n = (0.260 + 0.076j, 0.028 + 0.196j, 0.161 - 0.103j, 0.161 + 0.103j, 0.028 - 0.196j, 0.260 - 0.076j, -0.316 - 0.365j, 0.682 - 1.493j, 2.202, 0.682 + 1.493j, -0.316 + 0.365j)$.

In Figure 4.53 we plot this sequence and the one derived in Example 4.44, one under the other, as well as their corresponding energies. The new sequence is simply a shifted version of the previous one. By looking at Equations (4.222) and (4.227) we observe that:

$$\begin{aligned} W'(k) &= \psi(k)e^{-j\frac{\pi}{11}6k} \\ &= \underbrace{\psi(k)e^{j\frac{\pi}{11}6k}}_{W(k)}e^{-j\frac{\pi}{11}12k} \\ &= W(k)e^{-j\frac{2\pi}{11}6k}. \end{aligned} \tag{4.228}$$

This result is as expected, according to the shifting property of DFT. Indeed, when we shifted the sequence from starting at index -4 to starting at index 2, i.e. by $6(=2-(-4))$ positions to the right (see Figure 4.53), its DFT was multiplied with $e^{-j\frac{2\pi}{11}6k}$.

This example shows that once we fix the band we are interested in and the size of the filter we want to construct, we may solve the problem of filter construction only once, and not separately for each range of indices in the signal domain. This is as expected, according to the convolution theorem (see Book I [75]): when we select a frequency band inside which we want to examine the signal contents, we imply that we multiply the DFT of the signals with the window that isolates this band; multiplication in the frequency domain corresponds to convolution in the signal domain. So, sequence $w(n)$, which corresponds to frequency filter $W(k)$, is a convolution filter. By construction, it has the same size as the signal we wish to analyse, with most of its energy within a certain range of indices. When used for convolution, this range shifts along the signal, each time picking up the frequency content in the corresponding band at a different locality.

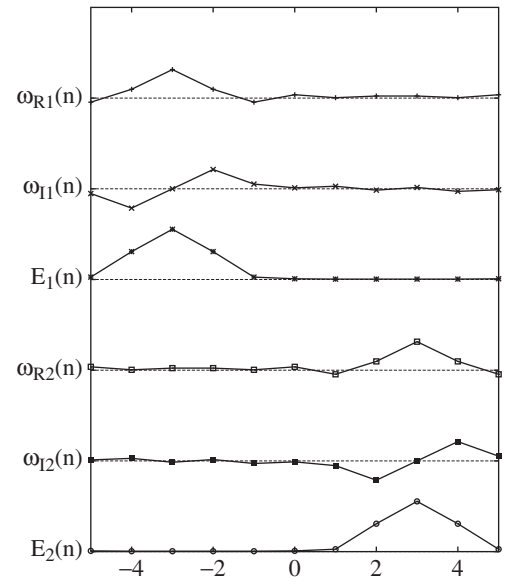


Figure 4.53 Top three lines: real part, imaginary part and energy of sequence $w(n)$ derived in Example 4.44. Bottom three lines: real part, imaginary part and energy of sequence $w(n)$ derived in this example. The latter is a shifted version of the former.

Example B4.46

Use the method of Examples 4.43 and 4.45 to construct an approximate convolution filter of size 3, appropriate for extracting the frequency content of a signal of 11 samples, in the band with indices 0, 1, 2, 3 and 4.

First we construct a sequence that is 11 samples long and has most of its energy in the middle three indices, namely indices −1, 0 and 1. The Fourier transform of this sequence is given by

$$W(k) = \psi(k) \tag{4.229}$$

since $n_1 + n_2 = -1 + 1 = 0$. *Function* $\psi(k)$ *is what was computed in Example 4.43.*

From this we derive sequence $w(n)$ *using Equation (4.223) as* $w(n) = (0.161 + 0.103\text{j}, 0.028 - 0.196\text{j}, 0.260 - 0.076\text{j}, -0.316 - 0.365\text{j}, 0.682 - 1.493\text{j}, 2.202, 0.682 + 1.493\text{j}, -0.316 + 0.365\text{j}, 0.260 + 0.076\text{j}, 0.028 + 0.196\text{j}, 0.161 - 0.103\text{j})$*, with corresponding energies* $E_n = (0.037, 0.039, 0.073, 0.233, 2.694, 4.849, 2.694, 0.233, 0.073, 0.039, 0.037)$.

We note that this sequence has its highest energy values in the middle three indices. We truncate it by keeping only those three coefficients and thus we derive an approximate convolution filter: $(0.682 - 1.493\text{j}, 2.202, 0.682 + 1.493\text{j})$.

Example B4.47

Use the approximate convolution filter derived in Example 4.46 on signal

$$2, -2, 2, -2, 2, 0, -2, 0, 2, 0, -2 \tag{4.230}$$

and show that the output signal contains the frequency content of the input signal in the band defined by indices 0-4.

Care should be taken here when performing the convolution, as the filters we use are not symmetric. Remembering the definition of convolution of a signal with a filter of length $2M + 1$ *as*

$$\text{output}(i) = \sum_{k=-M}^{M} \text{input}(i-k) \times \text{filter}(k) \tag{4.231}$$

we note that to obtain the output at, say, position 5, using a filter of length 3 ($M = 1$*), we must use*

$$\begin{aligned}\text{output}(5) &= \text{input}(6) \times \text{filter}(-1) + \text{input}(5) \times \text{filter}(0) + \text{input}(4) \times \text{filter}(1) \\ &= \text{input}(4) \times \text{filter}(1) + \text{input}(5) \times \text{filter}(0) + \text{input}(6) \times \text{filter}(-1)\end{aligned}$$

i.e. the filter is read back to front before we slide it across the signal to perform the necessary multiplications and additions.

Figure 4.54 shows the original signal, its Fourier transform, the signal convolved with the filter of Example 4.46, and the discrete Fourier transform of the output signal. We note that the convolved signal has indeed most of its energy in the chosen frequency band.

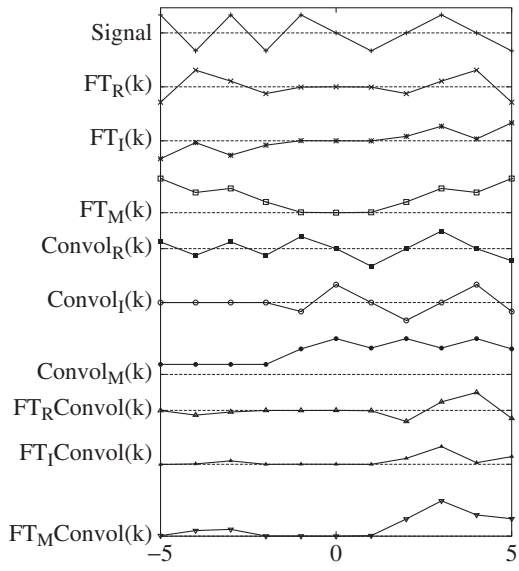


Figure 4.54 Top line: original signal. Next three lines: Fourier transform of the original signal (real part, imaginary part and magnitude). Next three lines: signal convolved with the approximate filter derived in Example 4.46 (real part, imaginary part and magnitude). Three bottom lines: Fourier transform of the convolved signal (real part, imaginary part and magnitude).

Example B4.48

Consider a sequence $w(-5), w(-4), \ldots, w(5)$**, and its discrete Fourier transform** $W(-5), W(-4), \ldots, W(5)$**. Assume that** $W(-5), W(-4), \ldots, W(-1)$ **are free to take any values and that** $W(0) = W(1) = W(2) = W(3) = W(4) = W(5) = 0$**. What values** $W(-5), W(-4), \ldots, W(-1)$ **should take so that sequence** $w(-5), w(-4), \ldots, w(5)$ **has most of its energy in indices** $-1, 0,$ **and** 1**?**

Here $N = 11$*,* $n_1 = -1$*,* $n_2 = 1$*,* $k_1 = -5$ *and* $k_2 = -1$*. Therefore* $n_2 - n_1 + 1 = 3$*, and the equation we have to solve is:*

$$\sum_{l=-5}^{-1} \psi(l) \frac{\sin\left(\frac{3\pi}{11}(k-l)\right)}{\sin\left(\frac{\pi}{11}(k-l)\right)} = 11R\psi(k). \tag{4.232}$$

Note that this equation is similar to Equation (4.221) of Example 4.43, except that there the unknown vector ψ *was called* $(\psi(0), \psi(1), \psi(2), \psi(3), \psi(4))$*, while here it has to be called* $(\psi(-5), \psi(-4), \psi(-3), \psi(-2), \psi(-1))$*. So, the solution of Equation (4.232) is:* $\psi = (0.348, 0.485, 0.536, 0.485, 0.348, 0, 0, 0, 0, 0, 0)^T$.

From this we compute $W(k)$ *using Equation (4.229), which corresponds to* $n_1 = -1$ *and* $n_2 = 1$*, and which obviously is identical to* $\psi(k)$ *since* $n_1 + n_2 = 0$.

Example B4.49

Use the result of Example 4.48 to construct an approximate convolution filter that returns the local frequency content of a signal in the frequency band $[-5, -1]$**.**

(Continued)

Example B4.49 (Continued)

Using $W(k)$ computed in Example 4.48 in Equation (4.223) we derive sequence $w(n) = (-0.125 + 0.145\text{j}, -0.167 - 0.107\text{j}, -0.112 + 0.246\text{j}, -0.463 - 0.136\text{j}, -0.234 + 1.624\text{j}, 2.202, -0.234 - 1.624\text{j}, -0.463 + 0.136\text{j}, -0.112 - 0.246\text{j}, -0.167 + 0.107\text{j}, -0.125 - 0.145\text{j})$ *which we truncate by keeping its central three elements only, to form the approximate convolution filter we need:* $(-0.234 + 1.624\text{j}, 2.202, -0.234 - 1.624\text{j})$.

Example 4.50

What is the relationship of two band-limited functions, the Fourier transforms of which are given by functions $F(\omega_x, \omega_y)$, and $F^*(-\omega_x, -\omega_y)$, respectively?

Let us say that $F(\omega_x, \omega_y)$ is the Fourier transform of function $f_1(x, y)$, and $F^*(-\omega_x, -\omega_y)$ is the Fourier transform of function $f_2(x, y)$.

We may express the two functions in terms of their Fourier transforms as follows:

$$f_1(x,y) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\omega_x,\omega_y)\mathrm{e}^{\mathrm{j}(\omega_x x+\omega_y y)}\mathrm{d}\omega_x\mathrm{d}\omega_y \tag{4.233}$$

$$f_2(x,y) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F^*(-\omega_x,-\omega_y)\mathrm{e}^{\mathrm{j}(\omega_x x+\omega_y y)}\mathrm{d}\omega_x\mathrm{d}\omega_y. \tag{4.234}$$

We change variables of integration in the last integral, from (ω_x, ω_y), to $(\tilde{\omega}_x, \tilde{\omega}_y)$, defined as $\tilde{\omega}_x \equiv -\omega_x$ and $\tilde{\omega}_y \equiv -\omega_y$, respectively. Then (4.234) may be written as:

$$f_2(x,y) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F^*(\tilde{\omega}_x,\tilde{\omega}_y)\mathrm{e}^{-\mathrm{j}(\tilde{\omega}_x x+\tilde{\omega}_y y)}\mathrm{d}\tilde{\omega}_x\mathrm{d}\tilde{\omega}_y. \tag{4.235}$$

Comparison of (4.233) and (4.235) yields that $f_2(x, y) = f_1^*(x, y)$. So, two band-limited functions, with Fourier transforms symmetric about the origin of the frequency axes, are complex conjugate of each other.

How can we construct a filter that is band-limited in two bands that are symmetrically placed about the origin of the axes in the frequency domain?

Figure 4.55 shows schematically two such frequency bands. We are interested in having a single filter that has non-zero response in those two bands only.

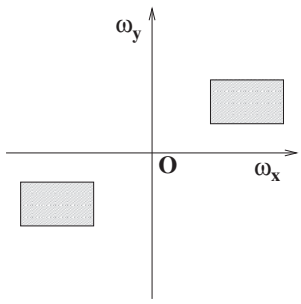


Figure 4.55 We wish to have a filter with non-zero values inside the two marked bands.

If $f(x,y)$ is a filter with frequency response in one of the two bands given by function $F(\omega_x, \omega_y)$, then $f^*(x,y)$ will have non-zero frequency response in the other band, given by $F^*(-\omega_x, -\omega_y)$ (see Example 4.50). The sum of the two functions will be equal to $2Real[f(x,y)]$ and it will have non-zero frequency response inside the pair of these two bands. So, if we want to construct a filter with non-zero frequency response inside a pair of bands, such as those shown in Figure 4.55, we must construct a filter with non-zero frequency response in one of the bands, discard its imaginary component, and multiply its real part with 2. Since usually these filters (or their outputs) are scaled before making any use of them, this multiplication with 2 is superfluous. Alternatively, if we have constructed a filter with frequency response in one of the bands, we do not need to repeat the process of constructing a filter with frequency response in the other band. All we have to do is to take the complex conjugate of the filter we have already constructed. Given that these filters are used to process real signals, the outputs of the convolution of the signal with these two filters will also form a complex conjugate pair.

Example B4.51

Use the method of Examples 4.43 and 4.45 to construct an approximate convolution filter of size 3, appropriate for extracting the frequency content of a sequence of 11 samples, in the bands with indices $-5, -4, -3$, **and** $3, 4$ **and** 5.

We shall first construct a filter with Fourier transform $W(-5) = W(-4) = \ldots = W(2) = 0$. *The filter will be a sequence that is* 11 *samples long with most of its energy in the middle three indices, namely indices* $-1, 0$ *and* 1. *We have here* $N = 11$, $n_1 = -1$, $n_2 = 1$, $k_1 = 3$ *and* $k_2 = 5$. *Therefore* $n_2 - n_1 + 1 = 3$. *The Fourier transform of this sequence is given by*

$$W(k) = \psi(k) \tag{4.236}$$

since $n_1 + n_2 = 0$ *in Equation (4.210). Function* $\psi(k)$ *is the first eigenvector of matrix:*

$$\begin{bmatrix} 3 & \frac{\sin(3\pi/11)}{\sin(\pi/11)} & \frac{\sin(6\pi/11)}{\sin(2\pi/11)} \\ \frac{\sin(3\pi/11)}{\sin(\pi/11)} & 3 & \frac{\sin(3\pi/11)}{\sin(\pi/11)} \\ \frac{\sin(6\pi/11)}{\sin(2\pi/11)} & \frac{\sin(3\pi/11)}{\sin(\pi/11)} & 3 \end{bmatrix}. \tag{4.237}$$

The eigenvalues of this matrix are $\lambda_1 = 7.818$, $\lambda_2 = 1.170$ *and* $\lambda_3 = 0.013$. *The eigenvector that corresponds to the largest eigenvalue is:* $\psi = (0.556, 0.619, 0.556)^T$. *The corresponding sequence* $w(n)$ *is given by Equation (4.204):*

$$w(n) = \frac{1}{11}\sum_{k=3}^{5} W(k) e^{j\frac{2\pi}{11}kn} \qquad \text{for } n = -5, -4, \ldots, 4, 5. \tag{4.238}$$

This sequence should be used as a convolution filter. To construct an approximate convolution filter, of size 3, from this, we keep only the central three values that contain most of the energy. We then keep only the real part, multiply it with 2, and so we have filter $(-2.034, 3.460, -2.034)$ *which has most of its non-zero frequency response in bands* $[-5, -3]$ *and* $[3, 5]$. *(Note that truncation reduced the concentration of the filter in the frequency domain, so the truncated filter has some energy even in frequencies outside the originally defined band.)*

Example B4.52

Use the method of Examples 4.43 and 4.45 to construct an approximate convolution filter of size 3**, appropriate for extracting the frequency content of a signal of** 11 **samples in the band with indices** $-2, -1, 0, 1, 2$**.**

Since the bandwidth of the sequence is five samples long, and the sequence is 11 *samples long with most of its energy in the middle three indices,* $W(k) = \psi(k)$ *where* $\psi(k)$ *is what was computed in Example 4.43. From this we derive convolution filter* $w(n)$ *using Equation (4.223) as:* $(0.191, -0.198, -0.271, 0.482, 1.641, 2.202, 1.641, 0.482, -0.271, -0.198, 0.191)$*. The three central values of this filter may be used as an approximate convolution filter to extract the frequency content of the signal in the band* $[-2, 2]$*.*

Example B4.53

Use the approximate convolution filters derived in Examples 4.46 and 4.49 to segment the signal of Example 4.47.

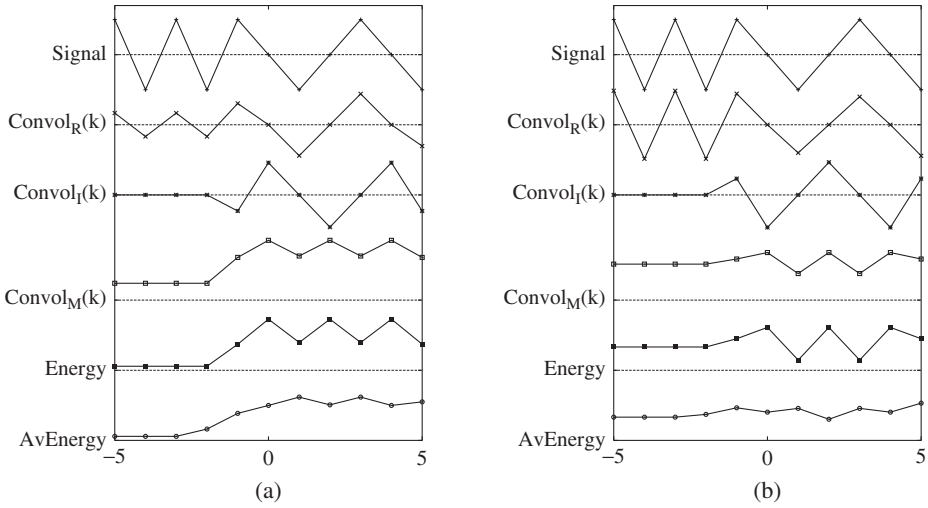


Figure 4.56 (a) Results after applying the filter obtained in Example 4.46 (frequency band $[0, 4]$). (b) Results after applying the filter obtained in Example 4.49 (frequency band $[-5, -1]$).

Figure 4.56 shows the original sequence and the two outputs, obtained by convolving it with the two filters. The real and imaginary parts of the convolved signal are shown in each case, as well as the magnitude, the energy, and the energy averaged over every three successive samples. For the purpose of convolution, and in order to avoid boundary effects, the signal was assumed to be repeated periodically in both directions.

We note that by thresholding the local energy output produced by using the first filter, we may segment the signal. These two filters do not really perform a good tessellation of the frequency space: the first contains all positive frequencies, apart from frequency 5, plus the dc component, while the second contains all negative frequencies. The right part of the signal, which is of lower

frequency than the left part, clearly shows up in the output of the first filter which contains the low frequencies. The second filter is a very broad band filter and so it does not allow the differentiation of the two components of the signal: it produces a more or less flat response.

Example B4.54

Use the approximate convolution filters derived in Examples 4.51 and 4.52 to segment the signal of Example 4.47.

The results are shown in Figure 4.57. One may achieve a segmentation by thresholding either of these outputs, although it is easier to select a good threshold for the output of the first filter. This is not surprising: the first filter is a high pass filter, while the second is a low pass filter. The first part of the signal is of higher frequency than the second part, so it has more energy in the output of the high pass filter, as expected. This is obvious from the shape of the average energy graph. The second half of the signal is of lower frequency and its average energy is higher than that of the left part of the signal when the low pass filter is used.

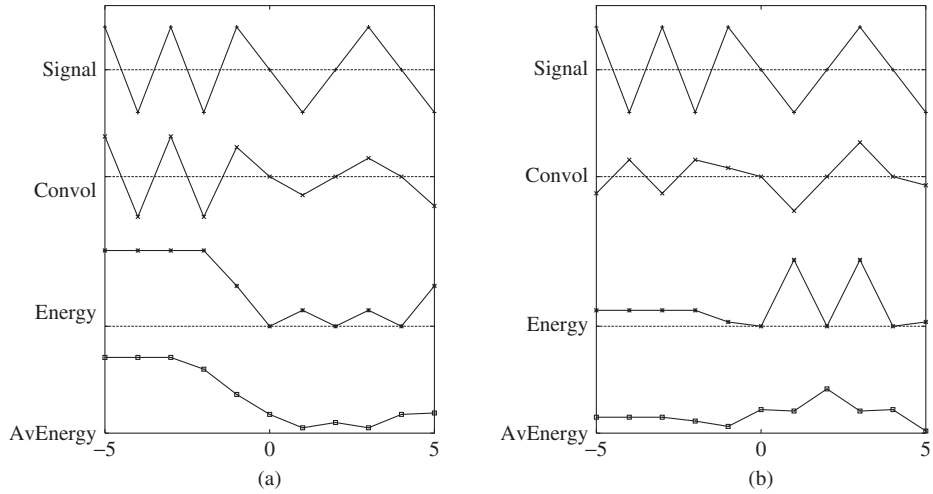


Figure 4.57 (a) Results after applying the filter obtained in Example 4.51 (high pass). (b) Results after applying the filter obtained in Example 4.52 (low pass).

Example B4.55

Use the approximate convolution filters derived in Examples 4.51 and 4.52 to segment signal $(2, -2, 2, -2, 2, -1, 1, -1, 1, -1, 1)$**.**

Figure 4.58 shows similar plots to those shown in Figure 4.56. We see that by thresholding the energy produced by the first filter we may segment the signal reasonably well. Both parts of the signal now are of the same high frequency, and both produce responses when processed by the high pass filter, but the left part has higher amplitude, i.e. more energy, than the right part, and

(Continued)

Example B4.55 (Continued)

this allows us to perform the segmentation. Neither of the two parts of the signal has significant energy in the low frequencies, so the output of the second filter is more or less flat and low, not suitable for segmentation.

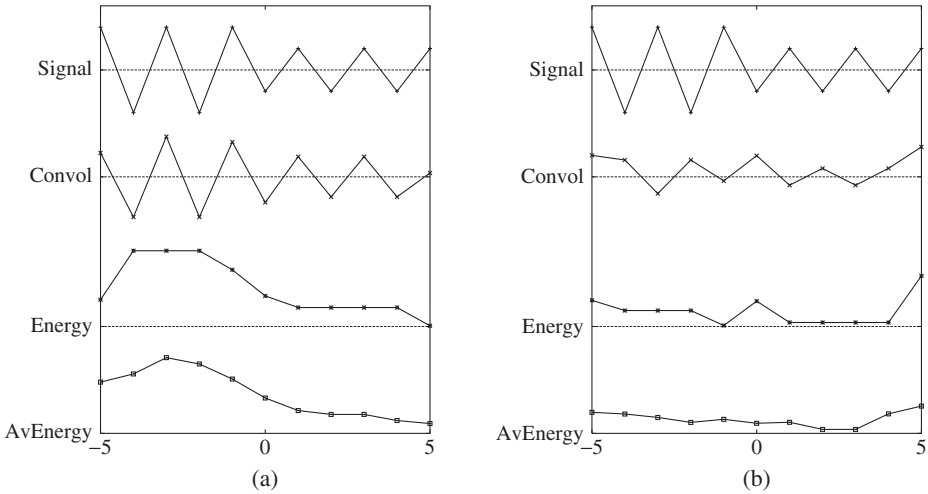


Figure 4.58 (a) Results after applying the filter obtained in Example 4.51 (high pass). (b) Results after applying the filter obtained in Example 4.52 (low pass).

Example 4.56

Segment with the help of prolate spheroidal sequence functions the signal of Example 4.14.

This signal is 128 samples long. All the theory we have developed refers to odd-sized signals. To avoid complications, we add a 0 at the end of the signal, and make it 129 samples long. Then we apply the methodology we have used so far to perform the segmentation. Note that due to the wrap-round boundary conditions we assume, there will always be a problem at the edges of the signal we segment, and we shall always have to ignore a few boundary samples on either side when producing the final result.

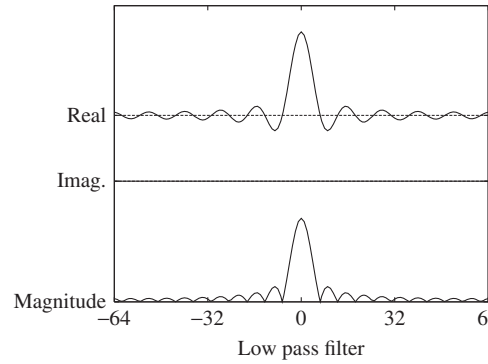
We must first decide upon the bands we wish to use. Let us say that we shall use a low frequencies band, corresponding to indices in the range $[-10, 10]$*, a medium frequencies band, corresponding to indices in the ranges* $[11, 30]$ *and* $[-30, -11]$*, and a high frequencies band, corresponding to indices in the ranges* $[31, 64]$ *and* $[-64, -31]$.

First we have to construct the three filters we want to use. Let us say that we wish the filters to be five samples long. So, we shall construct them with indices $n_1 = -2$ *and* $n_2 = 2$*. The signal is 129 samples long, and so in this case* $N = 129$.

For the low frequencies filter, the largest eigenvalue of matrix S is $\lambda_1 = 89.235$*, yielding eigenvector:* $\psi = (0.187, 0.196, 0.204, 0.211, 0.218, 0.223, 0.228, 0.231, 0.234, 0.236, 0.236, 0.236, 0.234, 0.231, 0.228, 0.223, 0.218, 0.211, 0.204, 0.196, 0.187)^T$*. From this, we construct sequence*

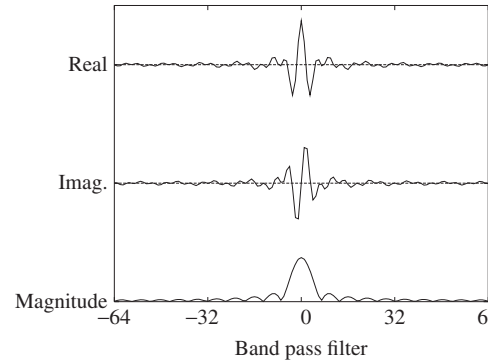
$W(k)$ and, using it, we compute sequence $w(n)$, shown in Figure 4.59, which, upon truncation, yields the low pass convolution filter: (3.864, 4.387, 4.570, 4.387, 3.864). *We normalise this filter so that the sum of its weights is 1. (Remember that low pass convolution filters should have their weights sum up to 1 so that they do not alter a flat input signal, while high pass convolution filters should have their weights sum up to 0 so that they do not respond to flat signals.) We do that by dividing each weight with the sum of all weights. As shown in Table 4.4, the final filter we shall use is:* (0.183, 0.208, 0.217, 0.208, 0.183).

Figure 4.59 Sequence $w(n)$ for the low pass convolution filter.



For the medium frequencies filter, the largest eigenvalue of matrix S is $\lambda_1 = 86.202$, yielding eigenvector: $\psi =$ $(0.194, 0.203, 0.211, 0.218, 0.224, 0.230, 0.234, 0.237, 0.239, 0.240, 0.240, 0.239, 0.237, 0.234, 0.230, 0.224, 0.218, 0.211, 0.203, 0.194)^T$. *From this, we construct sequence $W(k)$ and, using it, we compute sequence $w(n)$, shown in Figure 4.60, which, upon truncation, yields the band pass convolution filter* $(-1.583 - 3.488\text{j}, 2.328 - 3.613\text{j}, 4.462, 2.328 + 3.613\text{j}, -1.583 + 3.488\text{j})$. *We consider only the real part of this filter. Note that we do not really need to multiply it with 2, as this is only a scaling factor that plays no role in the feature construction and segmentation process. So, the real filter with frequency response in the symmetric bands is:* $(-1.583, 2.328, 4.462, 2.328, -1.583)$. *To remove any bias introduced by the process of truncation, we normalise it so that its weights sum up to 0. We may achieve this by finding the sum of its values, dividing it with 5, and subtracting the result from every single filter value. The filter we obtain this way is:* $(-2.7734, 1.1376, 3.2716, 1.1376, -2.7734)$. *Finally, we may scale the filter so that its positive weights sum up to $+1$ and its negative ones sum up to -1:* $(-0.500, 0.205, 0.590, 0.205, -0.500)$.

Figure 4.60 Sequence $w(n)$ for the band pass convolution filter.



(Continued)

Example 4.56 (Continued)

For the high frequencies filter, the largest eigenvalue of matrix S is $\lambda_1 = 116.013$*, yielding eigenvector:* $\psi = (0.109, 0.119, 0.128, 0.137, 0.146, 0.154, 0.162, 0.169, 0.176, 0.182, 0.188, 0.192, 0.196, 0.200, 0.202, 0.204, 0.205, 0.205, 0.204, 0.202, 0.200, 0.196, 0.192, 0.188, 0.182, 0.176, 0.169, 0.162, 0.154, 0.146, 0.137, 0.128, 0.119, 0.109)^T$*. From this, we construct sequence* $W(k)$ *and, using it, we compute sequence* $w(n)$*, shown in Figure 4.61, which, upon truncation, yields the high pass convolution filter:* $(-0.324 + 3.788j, -3.521 - 3.834j, 5.737, -3.521 + 3.834j, -0.324 - 3.788j)$*. To include also the band with the negative frequencies, we consider only the real part of the filter, that is:* $(-0.324, -3.521, 5.737, -3.521, -0.324)$*. Again we remove its bias so that the sum of its weights is zero, and we obtain:* $(0.0666, -3.1304, 6.1276, -3.1304, 0.0666)^T$*. Finally, we may scale the filter so that its positive weights sum up to* $+1$ *and its negative ones sum up to* -1*:* $(0.010, -0.500, 0.980, -0.500, 0.010)$*.*

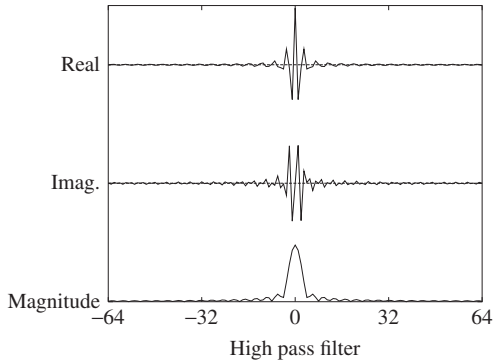


Figure 4.61 The $w(n)$ sequence that corresponds to the high pass convolution filter.

Figures 4.62a, 4.63a and 4.64a show the results obtained using the low pass, band pass and high pass filters, respectively. Note that we used the same scale for each signal in all three graphs, except for the averaged energies which were scaled independently. We can see that the averaged energies obtained allow the correct segmentation of the signal in all cases.

Table 4.4 Normalised values obtained for the low, band and high pass filters.

n	Low	Band	High
−2	0.183	−0.500	0.010
−1	0.208	0.205	−0.500
0	0.218	0.590	0.980
1	0.208	0.205	−0.500
2	0.183	−0.500	0.010

It is obvious from these figures that the filters we constructed are not very concentrated. This is particularly obvious for the low pass and the bands pass filters: there is still significant energy left in the signal outside the designated bands, as one can see from Figures 4.62a and 4.63a. This should not be surprising. The eigenvalue for the low pass filter is 89.235 which yields a concentration factor of $R = 89.235/129 = 0.696$*. (We used here equation* $\lambda = RN$ *where N is the length of*

the signal.) For the band pass filter the concentration factor is $R = 86.202/129 = 0.668$, which is equally low. For the high pass filter the concentration is much higher, $R = 116.013/129 = 0.899$ and this manifests itself in the strong suppression of the frequencies between the vertical lines of Figure 4.64a. This point is further demonstrated by Figures 4.62b, 4.63b and 4.64b, which were produced by truncating the constructed filters so that nine instead of five of their values were retained: we have much better suppression of the unwanted frequencies in each case now.

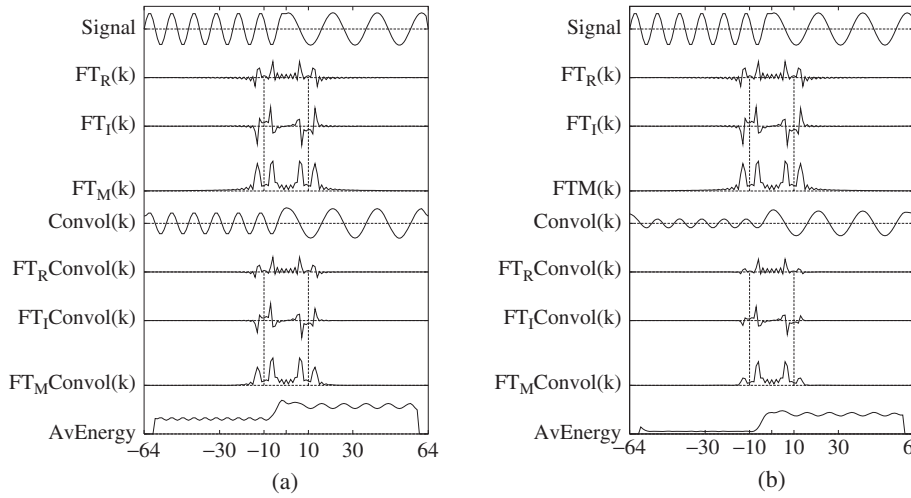


Figure 4.62 (a) Results obtained for the segmentation of the signal defined in Example 4.14 using a low pass filter of size 5 designed to pick up the frequencies between the two vertical dotted lines. From top to bottom: original signal; real part, imaginary part and magnitude of the Fourier transform of the original signal; signal convolved with the low pass filter; real part, imaginary part and magnitude of the Fourier transform of the convolved signal; average energy of the convolved signal, using an averaging window of 11 pixels long. (b) The same as (a) but now nine filter values have been retained. We have better suppression of the frequencies outside the chosen band now.

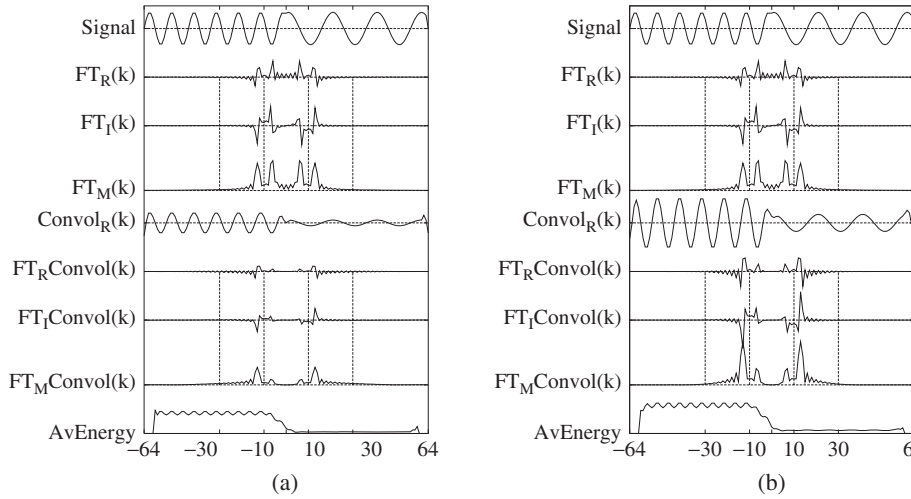


Figure 4.63 As Figure 4.62, but this figure concerns the band pass filter designed to have non-zero response between the two pairs of vertical dotted lines.

(Continued)

Example 4.56 (Continued)

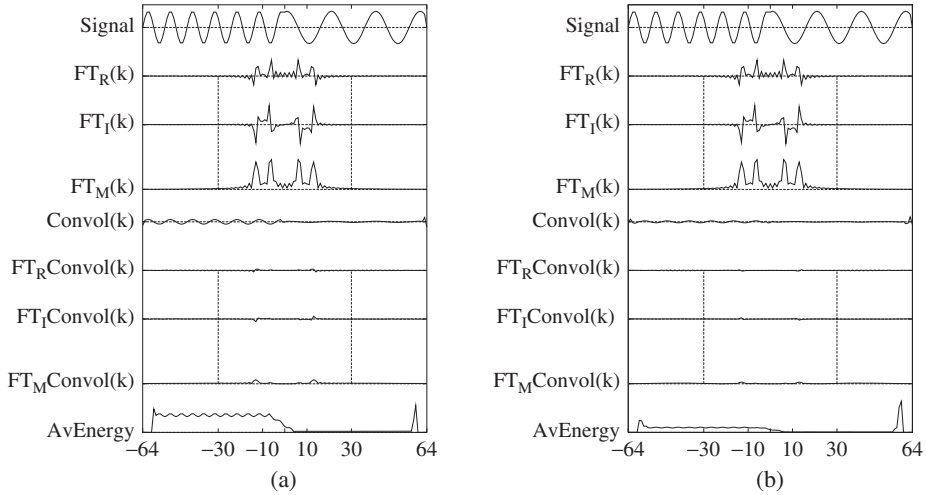


Figure 4.64 (a) Results obtained for the segmentation of the signal defined in Example 4.14 using a high pass filter of size 5 designed to have non-zero response outside the two vertical dotted lines. From top to bottom: original signal; real part, imaginary part and magnitude of the Fourier transform of the original signal; signal convolved with the high pass filter; real part, imaginary part and magnitude of the Fourier transform of the convolved signal; average energy of the convolved signal, using an averaging window of 11 pixels long. (b) The same as (a) but now nine filter values have been retained.

Box 4.4 How may we generalise the prolate spheroidal sequence functions to 2D?

Let us consider a 2D digital function $w(n,m)$, where n takes values in the range $\left[-\frac{N-1}{2},\dots,0,\dots,\frac{N-1}{2}\right]$, and m takes values in the range $\left[-\frac{M-1}{2},\dots,0,\dots,\frac{M-1}{2}\right]$, and $N\times M$ is the size of the domain of the function, with N and M being odd. If $W(k,l)$ is the DFT of the function, we may write

$$W(k,l)=\sum_{n=-\frac{N-1}{2}}^{\frac{N-1}{2}}\sum_{m=-\frac{M-1}{2}}^{\frac{M-1}{2}}w(n,m)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}kn-\mathrm{j}\frac{2\pi}{M}ml} \tag{4.239}$$

and

$$w(n,m)=\frac{1}{NM}\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}}\sum_{l=-\frac{M-1}{2}}^{\frac{M-1}{2}}W(k,l)\mathrm{e}^{\mathrm{j}2\pi\left(\frac{kn}{N}+\frac{ml}{M}\right)}. \tag{4.240}$$

The problem is: specify $W(k,l)$ so that the energy function $w(n,m)$ has between indices n_1 and n_2, and m_1 and m_2 is maximised.

Let us call, for simplicity:

$$N_0\equiv\frac{N-1}{2}\qquad M_0\equiv\frac{M-1}{2}. \tag{4.241}$$

The total energy of this function, after making use of Parseval's theorem, may be written as:

$$E_\infty = \sum_{n=-N_0}^{N_0}\sum_{m=-M_0}^{M_0}|w(n,m)|^2 = \frac{1}{NM}\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}|W(k,l)|^2. \tag{4.242}$$

The energy of the function between indices n_1 and n_2, and m_1 and m_2 is

$$E_{n_1n_2,m_1m_2} = \sum_{n=n_1}^{n_2}\sum_{m=m_1}^{m_2}|w(n,m)|^2 = \sum_{n=n_1}^{n_2}\sum_{m=m_1}^{m_2}w(n,m)w^*(n,m) \tag{4.243}$$

where $w^*(n,m)$ is the complex conjugate of function $w(n,m)$ and by using Equation (4.240) it may be written as:

$$w^*(n,m) = \frac{1}{NM}\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}W^*(k,l)\mathrm{e}^{-\mathrm{j}2\pi\left(\frac{kn}{N}+\frac{ml}{M}\right)}. \tag{4.244}$$

The fraction of the energy of the function between indices n_1 and n_2, and m_1 and m_2 is:

$$\begin{aligned}
R \equiv \frac{\sum_{n=n_1}^{n_2}\sum_{m=m_1}^{m_2}w(n,m)w^*(n,m)}{\frac{1}{NM}\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}|W(k,l)|^2} = \\
\frac{1}{\frac{1}{NM}\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}|W(k,l)|^2}\times \\
\sum_{n=n_1}^{n_2}\sum_{m=m_1}^{m_2}\frac{1}{NM}\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}W(k,l)\mathrm{e}^{\mathrm{j}2\pi\left(\frac{kn}{N}+\frac{ml}{M}\right)}\times \\
\frac{1}{NM}\sum_{\tilde{k}=-N_0}^{N_0}\sum_{\tilde{l}=-M_0}^{M_0}W^*(\tilde{k},\tilde{l})\mathrm{e}^{-\mathrm{j}2\pi\left(\frac{\tilde{k}n}{N}-\frac{\tilde{l}m}{M}\right)} = \\
\frac{1}{NM\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}|W(k,l)|^2}\times \\
\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}\sum_{\tilde{k}=-N_0}^{N_0}\sum_{\tilde{l}=-M_0}^{M_0}W(k,l)W^*(\tilde{k},\tilde{l})\sum_{n=n_1}^{n_2}\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}(k-\tilde{k})n}\sum_{m=m_1}^{m_2}\mathrm{e}^{\mathrm{j}\frac{2\pi}{M}(l-\tilde{l})m}.
\end{aligned} \tag{4.245}$$

We make use of (4.174) with $A = \frac{\pi}{N}(k-\tilde{k})$ and $A = \frac{\pi}{M}(l-\tilde{l})$ in order to substitute for sums $\sum_{n=n_1}^{n_2}\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}(k-\tilde{k})n}$ and $\sum_{m=m_1}^{m_2}\mathrm{e}^{\mathrm{j}\frac{2\pi}{M}(l-\tilde{l})m}$, respectively

$$\begin{aligned}
R = \frac{1}{NM\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}|W(k,l)|^2}\times \\
\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}\sum_{\tilde{k}=-N_0}^{N_0}\sum_{\tilde{l}=-M_0}^{M_0}W(k,l)W^*(\tilde{k},\tilde{l})\mathrm{e}^{\mathrm{j}\frac{\pi}{N}(k-\tilde{k})(n_1+n_2)}\times \\
\mathrm{e}^{\mathrm{j}\frac{\pi}{M}(l-\tilde{l})(m_1+m_2)}\frac{\sin\left(\frac{\pi}{N}(k-\tilde{k})n_0\right)}{\sin\left(\frac{\pi}{N}(k-\tilde{k})\right)}\frac{\sin\left(\frac{\pi}{M}(l-\tilde{l})m_0\right)}{\sin\left(\frac{\pi}{M}(l-\tilde{l})\right)}
\end{aligned} \tag{4.246}$$

(Continued)

Box 4.4 (Continued)

where for simplicity we defined:

$$n_0 \equiv n_2 - n_1 + 1 \qquad m_0 \equiv m_2 - m_1 + 1. \tag{4.247}$$

We define next

$$\psi(k,l) \equiv W(k,l) e^{j\frac{\pi}{N}k(n_1+n_2)+j\frac{\pi}{M}l(m_1+m_2)} \tag{4.248}$$

and as a consequence:

$$\psi^*(k,l) = W^*(k,l) e^{-j\frac{\pi}{N}k(n_1+n_2)-j\frac{\pi}{M}l(m_1+m_2)}. \tag{4.249}$$

We observe that $\psi(k,l)\psi^*(k,l) = W(k,l)W^*(k,l) = |W(k,l)|^2$. Then:

$$R = \frac{\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}\sum_{\tilde{k}=-N_0}^{N_0}\sum_{\tilde{l}=-M_0}^{M_0}\psi(k,l)\psi^*(\tilde{k},\tilde{l})\frac{\sin\left(\frac{\pi}{N}(k-\tilde{k})n_0\right)\sin\left(\frac{\pi}{M}(l-\tilde{l})m_0\right)}{\sin\left(\frac{\pi}{N}(k-\tilde{k})\right)\sin\left(\frac{\pi}{M}(l-\tilde{l})\right)}}{NM\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}|\psi(k,l)|^2}. \tag{4.250}$$

Let us call:

$$g(k,l) \equiv \sum_{\tilde{k}=-N_0}^{N_0}\sum_{\tilde{l}=-M_0}^{M_0}\psi^*(\tilde{k},\tilde{l})\frac{\sin\left(\frac{\pi}{N}(k-\tilde{k})(n_2-n_1+1)\right)\sin\left(\frac{\pi}{M}(l-\tilde{l})(m_2-m_1+1)\right)}{\sin\left(\frac{\pi}{N}(k-\tilde{k})\right)\sin\left(\frac{\pi}{M}(l-\tilde{l})\right)}. \tag{4.251}$$

Then:

$$R = \frac{\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}\psi(k,l)g(k,l)}{NM\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}|\psi(k,l)|^2}$$

$$\Rightarrow RNM\sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}\psi(k,l)\psi^*(k,l) = \sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}\psi(k,l)g(k,l)$$

$$\Rightarrow \sum_{k=-N_0}^{N_0}\sum_{l=-M_0}^{M_0}\psi(k,l)[RNM\psi^*(k,l) - g(k,l)] = 0. \tag{4.252}$$

This is true for all $\psi(k,l)$ only if:

$$g(k,l) = RNM\psi^*(k,l). \tag{4.253}$$

If we substitute $g(k,l)$ from (4.251) we derive:

$$\sum_{\tilde{k}=-N_0}^{N_0}\sum_{\tilde{l}=-M_0}^{M_0}\psi^*(\tilde{k},\tilde{l})\frac{\sin\left(\frac{\pi}{N}(k-\tilde{k})n_0\right)\sin\left(\frac{\pi}{M}(l-\tilde{l})m_0\right)}{\sin\left(\frac{\pi}{N}(k-\tilde{k})\right)\sin\left(\frac{\pi}{M}(l-\tilde{l})\right)} = RNM\psi^*(k,l). \tag{4.254}$$

We may drop the asterisk from this expression, by taking the complex conjugate of both sides. Let us go back to the original variables by replacing the auxiliary variables n_0, m_0, N_0 and M_0, and define

$$S(k,l,\tilde{k},\tilde{l}) \equiv \frac{\sin\left(\frac{\pi}{N}(k-\tilde{k})(n_2-n_1+1)\right)\sin\left(\frac{\pi}{M}(l-\tilde{l})(m_2-m_1+1)\right)}{\sin\left(\frac{\pi}{N}(k-\tilde{k})\right)\sin\left(\frac{\pi}{M}(l-\tilde{l})\right)}. \tag{4.255}$$

Then in order to work out $\psi(k,l)$, and from it $W(k,l)$, we must solve equation:

$$\sum_{\tilde{k}=-\frac{N-1}{2}}^{\frac{N-1}{2}} \sum_{\tilde{l}=-\frac{M-1}{2}}^{\frac{M-1}{2}} \psi(\tilde{k},\tilde{l})S(k,l,\tilde{k},\tilde{l}). = RNM\psi(k,l) \tag{4.256}$$

Once we know $W(k,l)$, we can work out $w(n,m)$, which may be used as a 2D convolution filter to process an image and extract its frequency content in the band inside which $W(k,l)$ is non-zero. As $w(n,m)$ will have its energy concentrated within some range of its indices, it will be possible to truncate it and form a much smaller approximate convolution filter.

Matrix S in this case is a four-dimensional matrix, and this makes the solution of the above equation somehow awkward. We can reduce the problem to that we solved for the 1D signals, if we express the 2D function $\psi(k,l)$ as a 1D vector, by writing its columns one under the other. Matrix S then is also reduced to 2D and the steps of solving the problem become identical to those we used to deal with the 1D signals. All these become much clearer in Example 4.57. However, later on (see Box 4.6) we shall see that this problem may actually be solved in a much simpler way than that of Example 4.57.

Example B4.57

Use Equation (4.256) to construct an approximate convolution filter of size 3×3 that may be used to extract local information for a band of size 5×5, from an image of size 15×15.

In this problem $N=M=15$. We choose $n_1=m_1=-1$ and $n_2=m_2=1$, so that the convolution filter we construct has most of its energy in a window of size $(n_2-n_1+1)\times(m_2-m_1+1)=3\times 3$, and so it is possible to truncate it to create an approximate convolution filter of size 3×3. We also choose $k_1=l_1=-2$ and $k_2=l_2=2$, so that the size of the band we choose to concentrate on is $(k_2-k_1+1)\times(l_2-l_1+1)=5\times 5$.

To construct this filter we must first solve Equation (4.256), which in this case has the form:

$$\sum_{\tilde{k}=-7}^{7} \sum_{\tilde{l}=-7}^{7} \psi(\tilde{k},\tilde{l})S(k,l,\tilde{k},\tilde{l}) = 225R\psi(k,l) \qquad \text{for } k,l\in\{-7,-6,\ldots,6,7\}. \tag{4.257}$$

As our filter is band limited, only the values of ψ in the chosen band will be non-zero, so this equation is further simplified to:

$$\sum_{\tilde{k}=-2}^{2} \sum_{\tilde{l}=-2}^{2} \psi(\tilde{k},\tilde{l})S(k,l,\tilde{k},\tilde{l}) = 225R\psi(k,l) \qquad \text{for } k,l\in\{-2,-1,0,1,2\}. \tag{4.258}$$

The above equation is an eigenvalue problem for a 4D array, namely matrix S. We may reduce it to an eigenvalue problem for a 2D array, if we read the columns of matrix $\psi(\tilde{k},\tilde{l})$ one under the other to form a 1D vector. To identify the elements of this vector, we introduce a single index $\hat{K}$ defined as:

$$\hat{K} \equiv (\tilde{k}-k_1)(l_2-l_1+1)+\tilde{l}-l_1. \tag{4.259}$$

The convention we use is that if we write $\psi(\tilde{k},\tilde{l})$ as an image, index $\tilde{k}$ increases along the horizontal direction and index $\tilde{l}$ increases along the vertical direction. For the particular case we are

(Continued)

Example B4.57 (Continued)

dealing with here, we have:

$$\hat{K} = (\tilde{k}+2)5 + \tilde{l} + 2 = 5\tilde{k} + \tilde{l} + 12. \tag{4.260}$$

For example, element $\psi(-1,2)$ *will appear as having index:* $\hat{K} = 9$.

The minimum value of $\hat{K}$ *is obtained for the element* $\psi(k_1, l_1)$, *i.e. for* $\tilde{k} = k_1$ *and* $\tilde{l} = l_1$, *and it is* $\hat{K}_{\min} = 0$. *The maximum value is obtained for* $\tilde{k} = k_2$ *and* $\tilde{l} = l_2$, *and it is:*

$$\begin{aligned}\hat{K}_{\max} &= (k_2 - k_1)(l_2 - l_1 + 1) + l_2 - l_1 \\ &= (k_2 - k_1)(l_2 - l_1 + 1) + l_2 - l_1 + 1 - 1 \\ &= (k_2 - k_1 + 1)(l_2 - l_1 + 1) - 1.\end{aligned}$$

So, in our particular example, $\hat{K}$ *takes values in the range* $[0, 24]$. *We would like it to take values in the range* $[-12, 12]$, *and so we define from it an index*

$$\tilde{K} \equiv \hat{K} - \frac{\hat{K}_{\max}}{2}. \tag{4.261}$$

So, in our example, element $(-1,2)$, *with* $\hat{K} = 9$, *will have index* $\tilde{K} = 9 - 12 = -3$.

There is a one-to-one correspondence between pair $(\tilde{k}, \tilde{l})$ *and index* $\hat{K}$, *as Equation (4.259) is invertible:*

$$\begin{aligned}\tilde{l} &= \hat{K}_{\text{modulo } (l_2 - l_1 + 1)} + l_1 \\ \tilde{k} &= \frac{\hat{K} - \hat{K}_{\text{modulo } (l_2 - l_1 + 1)}}{l_2 - l_1 + 1} + k_1.\end{aligned} \tag{4.262}$$

For example, for $\hat{K} = 9$, $l_1 = k_1 = -2$ *and* $l_2 = 2$, *we obtain* $\tilde{l} = 9_{\text{modulo } 5} - 2 = 4 - 2 = 2$ *and* $\tilde{k} = (9 - 9_{\text{modulo } 5})/5 - 2 = 5/5 - 2 = -1$, *i.e. index 9 corresponds to element* $(-1, 2)$.

By replacing $\hat{K}$ *in terms of* $\tilde{K}$ *from (4.261) we may write these equations directly in terms of* $\tilde{K}$, *i.e. the index we shall be using for the calculations:*

$$\begin{aligned}\tilde{l} &= \left(\tilde{K} + \frac{\hat{K}_{\max}}{2}\right)_{\text{modulo } (l_2 - l_1 + 1)} + l_1 \\ \tilde{k} &= \frac{\left(\tilde{K} + \frac{\hat{K}_{\max}}{2}\right) - \left(\tilde{K} + \frac{\hat{K}_{\max}}{2}\right)_{\text{modulo } (l_2 - l_1 + 1)}}{l_2 - l_1 + 1} + k_1.\end{aligned} \tag{4.263}$$

Therefore, wherever pair $(\tilde{k}, \tilde{l})$ *appears, it may be replaced by index* $\tilde{K}$ *and vice versa. For our particular example, these formulae have the form:*

$$\begin{aligned}\tilde{l} &= (\tilde{K} + 12)_{\text{modulo } 5} - 2 \\ \tilde{k} &= \frac{(\tilde{K} + 12) - (\tilde{K} + 12)_{\text{modulo } 5}}{5} - 2.\end{aligned} \tag{4.264}$$

For example, index $\tilde{K} = -3$, *yields* $\tilde{l} = 9_{\text{modulo } 5} - 2 = 2$ *and* $\tilde{k} = \frac{9 - 9_{\text{modulo } 5}}{5} - 2 = -1$.

Similarly, we define

$$K \equiv (k - k_1)(l_2 - l_1 + 1) + l - l_1 - \frac{\hat{K}_{\max}}{2} \tag{4.265}$$

and its inverse:

$$l = \left(K + \frac{\hat{K}_{\max}}{2}\right)_{\text{modulo } (l_2 - l_1 + 1)} + l_1$$
$$k = \frac{\left(K + \frac{\hat{K}_{\max}}{2}\right) - \left(K + \frac{\hat{K}_{\max}}{2}\right)_{\text{modulo } (l_2 - l_1 + 1)}}{l_2 - l_1 + 1} + k_1. \tag{4.266}$$

The minimum and maximum values of both $\tilde{K}$ and K are -12 and $+12$, respectively.

In view of the above, Equation (4.258) may be written as

$$\sum_{\tilde{K}=-12}^{12} \psi(\tilde{K}) S(K, \tilde{K}) = 225 R\psi(K) \qquad \text{for } K = -12, \ldots, 12 \tag{4.267}$$

where the elements of matrix $S(K, \tilde{K})$ are given by (4.255) which in this case takes the form

$$S(K, \tilde{K}) = \frac{\sin\left(\frac{\pi}{15}(k - \tilde{k})3\right) \sin\left(\frac{\pi}{15}(l - \tilde{l})3\right)}{\sin\left(\frac{\pi}{15}(k - \tilde{k})\right) \sin\left(\frac{\pi}{15}(l - \tilde{l})\right)} = \frac{\sin\left(\frac{\pi}{5}(k - \tilde{k})\right) \sin\left(\frac{\pi}{5}(l - \tilde{l})\right)}{\sin\left(\frac{\pi}{15}(k - \tilde{k})\right) \sin\left(\frac{\pi}{15}(l - \tilde{l})\right)} \tag{4.268}$$

with the understanding that for each pair $(K, \tilde{K})$ we have to compute the corresponding quadruple $(k, l, \tilde{k}, \tilde{l})$ using Equations (4.263) and (4.266), in order to compute the right-hand side of (4.268).

The non-zero eigenvalues of matrix S are: $\lambda_1 = 145.220$, $\lambda_2 = \lambda_3 = 34.595$, $\lambda_4 = 8.241$, $\lambda_5 = \lambda_6 = 0.946$, $\lambda_7 = \lambda_8 = 0.225$ and $\lambda_9 = 0.006$.

The eigenvector that corresponds to the largest eigenvalue is: $(0.155, 0.185, 0.196, 0.185, 0.155, 0.185, 0.221, 0.234, 0.221, 0.185, 0.196, 0.234, 0.247, 0.234, 0.196, 0.185, 0.221, 0.234, 0.221, 0.185, 0.155, 0.185, 0.196, 0.185, 0.155)^T$.

In matrix form this is:

$$\begin{bmatrix} 0.155 & 0.185 & 0.196 & 0.185 & 0.155 \\ 0.185 & 0.221 & 0.234 & 0.221 & 0.185 \\ 0.196 & 0.234 & 0.247 & 0.234 & 0.196 \\ 0.185 & 0.221 & 0.234 & 0.221 & 0.185 \\ 0.155 & 0.185 & 0.196 & 0.185 & 0.155 \end{bmatrix}. \tag{4.269}$$

From this we can compute the non-zero elements of $W(k, l)$ using Equation (4.248), which in this case takes the form

$$W(k, l) = \begin{cases} \psi(k, l) & \text{for } k, l \in \{-2, \ldots, 2\} \\ 0 & \text{otherwise} \end{cases} \tag{4.270}$$

since $n_1 + n_2 = m_1 + m_2 = 0$.

From Equation (4.240) then we may construct the 15×15 matrix:

$$w(n, m) = \frac{1}{225} \sum_{k=-7}^{7} \sum_{l=-7}^{7} W(k, l) e^{j\frac{2\pi}{15}(kn+ml)} = \frac{1}{225} \sum_{k=-2}^{2} \sum_{l=-2}^{2} W(k, l) e^{j\frac{2\pi}{15}(kn+ml)}. \tag{4.271}$$

(Continued)

Example B4.57 (Continued)

This matrix turns out to be:

$$w = \begin{bmatrix}
0.088 & -0.006 & -0.109 & -0.110 & 0.045 & 0.310 & 0.559 & 0.661 \\
-0.006 & 0.000 & 0.007 & 0.008 & -0.003 & -0.021 & -0.038 & -0.045 \\
-0.109 & 0.007 & 0.135 & 0.136 & -0.055 & -0.383 & -0.691 & -0.817 \\
-0.110 & 0.008 & 0.136 & 0.138 & -0.056 & -0.388 & -0.700 & -0.827 \\
0.045 & -0.003 & -0.055 & -0.056 & 0.023 & 0.157 & 0.284 & 0.335 \\
0.310 & -0.021 & -0.383 & -0.388 & 0.157 & 1.091 & 1.967 & 2.324 \\
0.559 & -0.038 & -0.691 & -0.700 & 0.284 & 1.967 & 3.548 & 4.193 \\
0.661 & -0.045 & -0.817 & -0.827 & 0.335 & 2.324 & 4.193 & 4.954 \\
0.559 & -0.038 & -0.691 & -0.700 & 0.284 & 1.967 & 3.548 & 4.193 \\
0.310 & -0.021 & -0.383 & -0.388 & 0.157 & 1.091 & 1.967 & 2.324 \\
0.045 & -0.003 & -0.055 & -0.056 & 0.023 & 0.157 & 0.284 & 0.335 \\
-0.110 & 0.008 & 0.136 & 0.138 & -0.056 & -0.388 & -0.700 & -0.827 \\
-0.109 & 0.007 & 0.135 & 0.136 & -0.055 & -0.383 & -0.691 & -0.817 \\
-0.006 & 0.000 & 0.007 & 0.008 & -0.003 & -0.021 & -0.038 & -0.045 \\
0.088 & -0.006 & -0.109 & -0.110 & 0.045 & 0.310 & 0.559 & 0.661
\end{bmatrix}$$

$$\begin{bmatrix}
0.559 & 0.310 & 0.045 & -0.110 & -0.109 & -0.006 & 0.088 \\
-0.038 & -0.021 & -0.003 & 0.008 & 0.007 & 0.000 & -0.006 \\
-0.691 & -0.383 & -0.055 & 0.136 & 0.135 & 0.007 & -0.109 \\
-0.700 & -0.388 & -0.056 & 0.138 & 0.136 & 0.008 & -0.110 \\
0.284 & 0.157 & 0.023 & -0.056 & -0.055 & -0.003 & 0.045 \\
1.967 & 1.091 & 0.157 & -0.388 & -0.383 & -0.021 & 0.310 \\
3.548 & 1.967 & 0.284 & -0.700 & -0.691 & -0.038 & 0.559 \\
4.193 & 2.324 & 0.335 & -0.827 & -0.817 & -0.045 & 0.661 \\
3.548 & 1.967 & 0.284 & -0.700 & -0.691 & -0.038 & 0.559 \\
1.967 & 1.091 & 0.157 & -0.388 & -0.383 & -0.021 & 0.310 \\
0.284 & 0.157 & 0.023 & -0.056 & -0.055 & -0.003 & 0.045 \\
-0.700 & -0.388 & -0.056 & 0.138 & 0.136 & 0.008 & -0.110 \\
-0.691 & -0.383 & -0.055 & 0.136 & 0.135 & 0.007 & -0.109 \\
-0.038 & -0.021 & -0.003 & 0.008 & 0.007 & 0.000 & -0.006 \\
0.559 & 0.310 & 0.045 & -0.110 & -0.109 & -0.006 & 0.088
\end{bmatrix}.$$

Figure 4.65 shows the corresponding energies as a grey image.

We notice that its most significant elements are the central 3×3*, so we truncate* w *to form the approximate convolution filter:*

$$\begin{bmatrix}
3.548 & 4.193 & 3.548 \\
4.193 & 4.954 & 4.193 \\
3.548 & 4.193 & 3.548
\end{bmatrix}. \tag{4.272}$$

Finally, we normalise this by dividing each element with the sum of all elements:

$$\begin{bmatrix}
0.099 & 0.117 & 0.099 \\
0.117 & 0.138 & 0.117 \\
0.099 & 0.117 & 0.099
\end{bmatrix}.$$

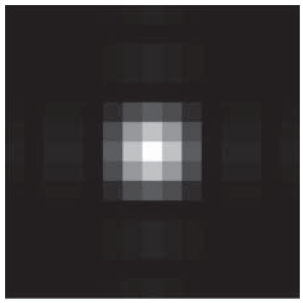


Figure 4.65 The squared values of $w(n,m)$, shown as a grey image. Source: Maria Petrou.

Example B4.58

Use the result of Example 4.57 to construct a set of orthogonal basis functions that fully span the frequency domain of a 15×15 image.

The Fourier domain of a 15×15 image is also 15×15 in size, and it looks like the grid in Figure 4.66. It can be divided into nine distinct bands of size 5×5 each, as shown in Figure 4.66.

The filter constructed in Example 4.57 spans the grey band in the middle. We may use Equation (4.271) to construct similar filters that span the other bands. All we have to do is to shift the non-zero values of $\psi(k,l)$ so that they cover a different band each time. This is shown schematically in Figure 4.67, where in each case the grey part of the frequency domain is filled with the 5×5 matrix (4.269) while the white part is filled with zeros.

Explicitly, these frequency domain masks are defined as:

$$W^{11}(k,l)=\begin{cases}\psi(k,l) & \text{for } k\in[-7,-3], l\in[-7,-3]\\ 0 & \text{elsewhere}\end{cases} \tag{4.273}$$

$$W^{12}(k,l)=\begin{cases}\psi(k,l) & \text{for } k\in[-2,2], l\in[-7,-3]\\ 0 & \text{elsewhere}\end{cases} \tag{4.274}$$

$$W^{13}(k,l)=\begin{cases}\psi(k,l) & \text{for } k\in[3,7], l\in[-7,-3]\\ 0 & \text{elsewhere}\end{cases} \tag{4.275}$$

$$W^{21}(k,l)=\begin{cases}\psi(k,l) & \text{for } k\in[-7,-3], l\in[-2,2]\\ 0 & \text{elsewhere}\end{cases} \tag{4.276}$$

$$W^{22}(k,l)=\begin{cases}\psi(k,l) & \text{for } k\in[-2,2], l\in[-2,2]\\ 0 & \text{elsewhere}\end{cases} \tag{4.277}$$

$$W^{23}(k,l)=\begin{cases}\psi(k,l) & \text{for } k\in[3,7], l\in[-2,2]\\ 0 & \text{elsewhere}\end{cases} \tag{4.278}$$

$$W^{31}(k,l)=\begin{cases}\psi(k,l) & \text{for } k\in[-7,-3], l\in[3,7]\\ 0 & \text{elsewhere}\end{cases} \tag{4.279}$$

$$W^{32}(k,l)=\begin{cases}\psi(k,l) & \text{for } k\in[-2,2], l\in[3,7]\\ 0 & \text{elsewhere}\end{cases} \tag{4.280}$$

$$W^{33}(k,l)=\begin{cases}\psi(k,l) & \text{for } k\in[3,7], l\in[3,7]\\ 0 & \text{elsewhere}\end{cases}. \tag{4.281}$$

(Continued)

Example B4.58 (Continued)

Note that these functions cover the whole frequency space. In addition, if we multiply any two of them point by point and add the results, we get 0. So, they constitute a complete and orthogonal set of basis functions, in terms of which the full frequency content of the sub-image of size 3×3*, which sits in the middle of a* 15×15 *image, may be expressed.*

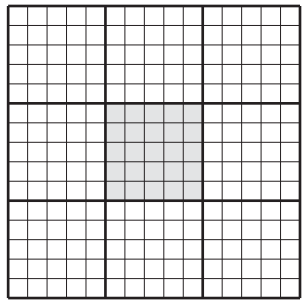


Figure 4.66 A tessellation of the 2D frequency domain of a 15×15 image into frequency bands of size 5×5. The zero frequency point is in the middle of the grid.

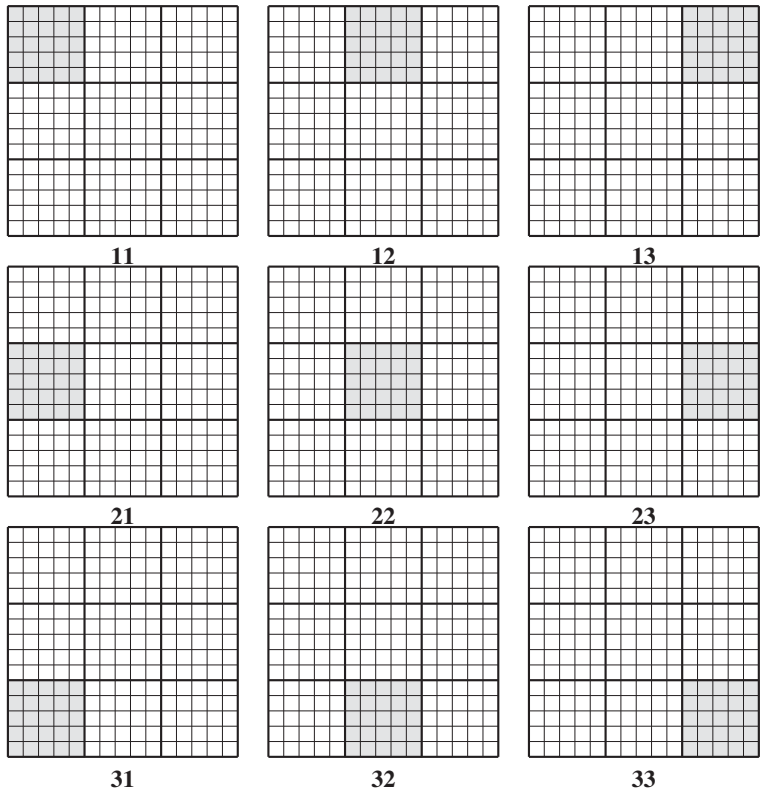


Figure 4.67 We shift the non-zero values of array (4.269) in the grey area of the frequency domain, setting all other values to zero, in order to create filters that respond to each one of the identified bands separately.

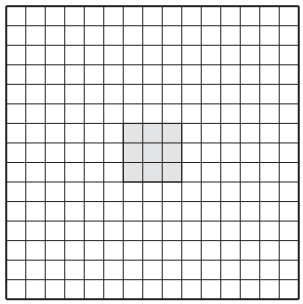


Figure 4.68 The tessellation of the 2D frequency domain of a 15×15 image shown in Figure 4.66 allows us to produce nine different arrays of size 15×15, which may be approximated by setting all their elements outside their central 3×3 patch equal to 0. The central parts with the non-zero values of the nine filters are given by Equations (4.282)–(4.290). If we multiply an input 15×15 image with each of the approximated 15×15 filters, point by point, and add the 225 products (only 9 of which will be non-zero), we shall obtain information about the frequency content of the grey patch of the image shown here in each of the frequency bands identified in Figure 4.67. This amounts to projecting the image on each one of the 9 constructed arrays.

Each of these functions corresponds to a different filter defined in the image space. We may compute these filters by using Equation (4.240) with the appropriate $W^f(k,l)$, where $f \in \{11, 12, 13, 21, 22, 23, 31, 32, 33\}$. Figure 4.68 shows schematically the image space, with the region where these filters have most of their energy highlighted in grey.

After the filters are truncated to size 3×3, to produce smaller approximate convolution filters, they have to be normalised. Normalisation for the low pass filter (w^{22}) means that its weights are scaled so they sum up to 1. The weights of the high pass filters have to sum up to 0. To achieve that, we consider each filter in turn. We first add all its elements and produce a complex number $a + \mathrm{j}b$, remembering that the high pass filters have complex weights. The elements of the filter are nine (a 3×3 filter). We divide $a + \mathrm{j}b$ by 9. This way we produce a number, say, $c + \mathrm{j}d$ for the filter under consideration. We subtract $c + \mathrm{j}d$ from all elements of the filter, so its weights now sum up to 0. We also scale these filters by the same scaling factor used to scale the low pass filter, just for consistency. In reality, no scaling is necessary, as the results have meaning only in relative and not in absolute terms. The filters obtained this way are listed below:

$$w^{11} = \begin{bmatrix} -0.050 - 0.086\mathrm{j} & -0.059 + 0.101\mathrm{j} & 0.098 \\ -0.059 + 0.101\mathrm{j} & 0.138 & -0.059 - 0.101\mathrm{j} \\ 0.098 & -0.059 - 0.101\mathrm{j} & -0.050 + 0.086\mathrm{j} \end{bmatrix} \tag{4.282}$$

$$w^{12} = \begin{bmatrix} -0.056 + 0.086\mathrm{j} & -0.065 + 0.101\mathrm{j} & -0.056 + 0.086\mathrm{j} \\ 0.110 & 0.132 & 0.110 \\ -0.056 - 0.086\mathrm{j} & -0.065 - 0.101\mathrm{j} & -0.056 - 0.086\mathrm{j} \end{bmatrix} \tag{4.283}$$

$$w^{13} = \begin{bmatrix} 0.098 & -0.059 + 0.101\mathrm{j} & -0.050 - 0.086\mathrm{j} \\ -0.059 - 0.101\mathrm{j} & 0.138 & -0.059 + 0.101\mathrm{j} \\ -0.050 + 0.086\mathrm{j} & -0.059 - 0.101\mathrm{j} & 0.098 \end{bmatrix} \tag{4.284}$$

$$w^{21} = \begin{bmatrix} -0.056 + 0.086\mathrm{j} & 0.110 & -0.056 - 0.086\mathrm{j} \\ -0.065 + 0.101\mathrm{j} & 0.132 & -0.065 - 0.101\mathrm{j} \\ -0.056 + 0.086\mathrm{j} & 0.110 & -0.056 - 0.086\mathrm{j} \end{bmatrix} \tag{4.285}$$

(Continued)

Example B4.58 (Continued)

$$w^{22} = \begin{bmatrix} 0.099 & 0.117 & 0.099 \\ 0.117 & 0.138 & 0.117 \\ 0.099 & 0.117 & 0.099 \end{bmatrix} \tag{4.286}$$

$$w^{23} = \begin{bmatrix} -0.056 - 0.086\mathrm{j} & 0.110 & -0.056 + 0.086\mathrm{j} \\ -0.065 - 0.101\mathrm{j} & 0.132 & -0.065 + 0.101\mathrm{j} \\ -0.056 - 0.086\mathrm{j} & 0.110 & -0.056 + 0.086\mathrm{j} \end{bmatrix} \tag{4.287}$$

$$w^{31} = \begin{bmatrix} 0.098 & -0.059 - 0.101\mathrm{j} & -0.050 + 0.086\mathrm{j} \\ -0.059 + 0.101\mathrm{j} & 0.138 & -0.059 - 0.101\mathrm{j} \\ -0.050 - 0.086\mathrm{j} & -0.059 + 0.101\mathrm{j} & 0.098 \end{bmatrix} \tag{4.288}$$

$$w^{32} = \begin{bmatrix} -0.056 - 0.086\mathrm{j} & -0.065 - 0.101\mathrm{j} & -0.056 - 0.086\mathrm{j} \\ 0.110 & 0.132 & 0.110 \\ -0.056 + 0.086\mathrm{j} & -0.065 + 0.101\mathrm{j} & -0.056 + 0.086\mathrm{j} \end{bmatrix} \tag{4.289}$$

$$w^{33} = \begin{bmatrix} -0.050 + 0.086\mathrm{j} & -0.059 - 0.101\mathrm{j} & 0.098 \\ -0.059 - 0.101\mathrm{j} & 0.138 & -0.059 + 0.101\mathrm{j} \\ 0.098 & -0.059 + 0.101\mathrm{j} & -0.050 - 0.086\mathrm{j} \end{bmatrix}. \tag{4.290}$$

We note that $w^{11} = w^{33*}$, $w^{12} = w^{32*}$, $w^{13} = w^{31*}$ *and* $w^{21} = w^{23*}$. *This is in accordance with the discussion on the relationship between two functions that have Fourier transforms non-zero at bands symmetric about the dc component (see Example 4.50).*

From these filters we may produce real valued filters that may return the frequency content of a 15×15 *image in pairs of bands symmetric about the centre. Obviously, there are only five distinct such real filters, the low pass filter* w^{22} *and the four high pass ones. After normalisation, so that the positive weights sum up to* $+1$ *and the negative weights to* -1, *these filters are:*

$$w_R^{11} = \begin{bmatrix} -0.148 & -0.176 & 0.294 \\ -0.176 & 0.412 & -0.176 \\ 0.294 & -0.176 & -0.148 \end{bmatrix} \quad w_R^{12} = \begin{bmatrix} -0.158 & -0.184 & -0.158 \\ 0.313 & 0.374 & 0.313 \\ -0.158 & -0.184 & -0.158 \end{bmatrix}$$

$$w_R^{13} = \begin{bmatrix} 0.294 & -0.176 & -0.148 \\ -0.176 & 0.412 & -0.176 \\ -0.148 & -0.176 & 0.294 \end{bmatrix} \quad w_R^{21} = \begin{bmatrix} -0.158 & 0.313 & -0.158 \\ -0.184 & 0.374 & -0.184 \\ -0.158 & 0.313 & -0.158 \end{bmatrix}. \tag{4.291}$$

Example B4.59

Use the results of Example 4.57 to construct a set of functions that cover fully the space of an image of size 15×15**.**

A 15×15 *image may be divided into 25 disjoint sets of* 3×3 *sub-images. We may use the* $\psi(k, l)$ *function constructed in Example 4.53 to construct functions each of which focuses in a different such sub-image. The 25 sub-images correspond to different values of indices* n_1, n_2, m_1 *and* m_2. *Table 4.5 lists the ranges of indices n and m for all 25 sub-images.*

These functions cover the whole 15×15 *image space, in the sense that each function has its energy concentrated in a different* 3×3 *sub-image. Two of the 25 functions derived this way are:*

$$\begin{bmatrix}
3.548 & 4.193 & 3.548 & 1.967 & 0.284 & -0.700 & -0.691 & -0.038 \\
4.193 & 4.954 & 4.193 & 2.324 & 0.335 & -0.827 & -0.817 & -0.045 \\
3.548 & 4.193 & 3.548 & 1.967 & 0.284 & -0.700 & -0.691 & -0.038 \\
1.967 & 2.324 & 1.967 & 1.091 & 0.157 & -0.388 & -0.383 & -0.021 \\
0.284 & 0.335 & 0.284 & 0.157 & 0.023 & -0.056 & -0.055 & -0.003 \\
-0.700 & -0.827 & -0.700 & -0.388 & -0.056 & 0.138 & 0.136 & 0.008 \\
-0.691 & -0.817 & -0.691 & -0.383 & -0.055 & 0.136 & 0.135 & 0.007 \\
-0.038 & -0.045 & -0.038 & -0.021 & -0.003 & 0.008 & 0.007 & 0.000 \\
0.559 & 0.661 & 0.559 & 0.310 & 0.045 & -0.110 & -0.109 & -0.006 \\
0.559 & 0.661 & 0.559 & 0.310 & 0.045 & -0.110 & -0.109 & -0.006 \\
-0.038 & -0.045 & -0.038 & -0.021 & -0.003 & 0.008 & 0.007 & 0.000 \\
-0.691 & -0.817 & -0.691 & -0.383 & -0.055 & 0.136 & 0.135 & 0.007 \\
-0.700 & -0.827 & -0.700 & -0.388 & -0.056 & 0.138 & 0.136 & 0.008 \\
0.284 & 0.335 & 0.284 & 0.157 & 0.023 & -0.056 & -0.055 & -0.003 \\
1.967 & 2.324 & 1.967 & 1.091 & 0.157 & -0.388 & -0.383 & -0.021
\end{bmatrix}$$

$$\begin{bmatrix}
0.559 & 0.559 & -0.038 & -0.691 & -0.700 & 0.284 & 1.967 \\
0.661 & 0.661 & -0.045 & -0.817 & -0.827 & 0.335 & 2.324 \\
0.559 & 0.559 & -0.038 & -0.691 & -0.700 & 0.284 & 1.967 \\
0.310 & 0.310 & -0.021 & -0.383 & -0.388 & 0.157 & 1.091 \\
0.045 & 0.045 & -0.003 & -0.055 & -0.056 & 0.023 & 0.157 \\
-0.110 & -0.110 & 0.008 & 0.136 & 0.138 & -0.056 & -0.388 \\
-0.109 & -0.109 & 0.007 & 0.135 & 0.136 & -0.055 & -0.383 \\
-0.006 & -0.006 & 0.000 & 0.007 & 0.008 & -0.003 & -0.021 \\
0.088 & 0.088 & -0.006 & -0.109 & -0.110 & 0.045 & 0.310 \\
0.088 & 0.088 & -0.006 & -0.109 & -0.110 & 0.045 & 0.310 \\
-0.006 & -0.006 & 0.000 & 0.007 & 0.008 & -0.003 & -0.021 \\
-0.109 & -0.109 & 0.007 & 0.135 & 0.136 & -0.055 & -0.383 \\
-0.110 & -0.110 & 0.008 & 0.136 & 0.138 & -0.056 & -0.388 \\
0.045 & 0.045 & -0.003 & -0.055 & -0.056 & 0.023 & 0.157 \\
0.310 & 0.310 & -0.021 & -0.383 & -0.388 & 0.157 & 1.091
\end{bmatrix} \tag{4.292}$$

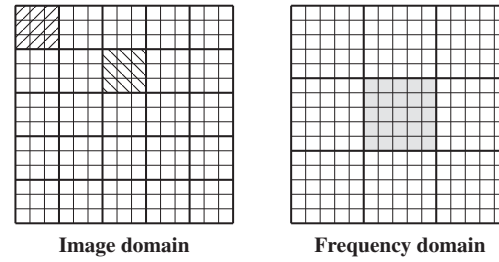


Figure 4.69 Left: tessellation of the 2D 15×15 image into sub-images of size 3×3. The highlighted sub-images are those that correspond to the two example functions. Right: the low-frequency band where the example functions have non-zero response.

(Continued)

Example B4.59 (Continued)

$$\begin{bmatrix}
-0.110 & 0.008 & 0.136 & 0.138 & -0.056 & -0.388 & -0.700 & -0.827 \\
0.045 & -0.003 & -0.055 & -0.056 & 0.023 & 0.157 & 0.284 & 0.335 \\
0.310 & -0.021 & -0.383 & -0.388 & 0.157 & 1.091 & 1.967 & 2.324 \\
0.559 & -0.038 & -0.691 & -0.700 & 0.284 & 1.967 & 3.548 & 4.193 \\
0.661 & -0.045 & -0.817 & -0.827 & 0.335 & 2.324 & 4.193 & 4.954 \\
0.559 & -0.038 & -0.691 & -0.700 & 0.284 & 1.967 & 3.548 & 4.193 \\
0.310 & -0.021 & -0.383 & -0.388 & 0.157 & 1.091 & 1.967 & 2.324 \\
0.045 & -0.003 & -0.055 & -0.056 & 0.023 & 0.157 & 0.284 & 0.335 \\
-0.110 & 0.008 & 0.136 & 0.138 & -0.056 & -0.388 & -0.700 & -0.827 \\
-0.109 & 0.007 & 0.135 & 0.136 & -0.055 & -0.383 & -0.691 & -0.817 \\
-0.006 & 0.000 & 0.007 & 0.008 & -0.003 & -0.021 & -0.038 & -0.045 \\
0.088 & -0.006 & -0.109 & -0.110 & 0.045 & 0.310 & 0.559 & 0.661 \\
0.088 & -0.006 & -0.109 & -0.110 & 0.045 & 0.310 & 0.559 & 0.661 \\
-0.006 & 0.000 & 0.007 & 0.008 & -0.003 & -0.021 & -0.038 & -0.045 \\
-0.109 & 0.007 & 0.135 & 0.136 & -0.055 & -0.383 & -0.691 & -0.817
\end{bmatrix}$$

$$\begin{bmatrix}
-0.700 & -0.388 & -0.056 & 0.138 & 0.136 & 0.008 & -0.110 \\
0.284 & 0.157 & 0.023 & -0.056 & -0.055 & -0.003 & 0.045 \\
1.967 & 1.091 & 0.157 & -0.388 & -0.383 & -0.021 & 0.310 \\
3.548 & 1.967 & 0.284 & -0.700 & -0.691 & -0.038 & 0.559 \\
4.193 & 2.324 & 0.335 & -0.827 & -0.817 & -0.045 & 0.661 \\
3.548 & 1.967 & 0.284 & -0.700 & -0.691 & -0.038 & 0.559 \\
1.967 & 1.091 & 0.157 & -0.388 & -0.383 & -0.021 & 0.310 \\
0.284 & 0.157 & 0.023 & -0.056 & -0.055 & -0.003 & 0.045 \\
-0.700 & -0.388 & -0.056 & 0.138 & 0.136 & 0.008 & -0.110 \\
-0.691 & -0.383 & -0.055 & 0.136 & 0.135 & 0.007 & -0.109 \\
-0.038 & -0.021 & -0.003 & 0.008 & 0.007 & 0.000 & -0.006 \\
0.559 & 0.310 & 0.045 & -0.110 & -0.109 & -0.006 & 0.088 \\
0.559 & 0.310 & 0.045 & -0.110 & -0.109 & -0.006 & 0.088 \\
-0.038 & -0.021 & -0.003 & 0.008 & 0.007 & 0.000 & -0.006 \\
-0.691 & -0.383 & -0.055 & 0.136 & 0.135 & 0.007 & -0.109
\end{bmatrix}. \quad (4.293)$$

These functions are versions of the same 2D convolution filter placed at two different positions of the image, since they both have been designed to have non-zero DFTs in the same frequency band. As they are the same size as the image, they may be thought of as basis functions on which the image may be projected to yield one projection coefficient per function. Such a projection is performed by point by point multiplication and addition of the results to obtain the projection coefficient. The basis these functions constitute is not orthogonal, since, if we multiply any two of them point by point and add the results, we shall not get 0. If each one of these functions is truncated by setting equal to 0 all its values outside the 3×3 *partition where most of its energy is concentrated, they will constitute an orthogonal basis, since they have been constructed so that each has its maximum energy in a different* 3×3 *partition of the image grid. Clearly, we can create one such set of functions for each one of the different frequency bands of example 4.58, if we use the corresponding* $\psi(k, l)$ *for its construction.*

Table 4.5 If we use these ranges of indices in Equation (4.248) with $N = M = 15$ and the values of $\psi(k, l)$ as derived in Example 4.57, we may derive the DFTs of the 25 functions, each of which peaks in a separate sub-image, and may be used to return the frequency content of that image in a 5×5 low frequency band centred around the dc component. From these DFTs and by using Equation (4.240) we may derive the corresponding functions in the image space.

Sub-image	n_1	n_2	m_1	m_2
1	−7	−5	−7	−5
2	−4	−2	−7	−5
3	−1	1	−7	−5
4	2	4	−7	−5
5	5	7	−7	−5
6	−7	−5	−4	−2
7	−4	−2	−4	−2
8	−1	1	−4	−2
9	2	4	−4	−2
10	5	7	−4	−2
11	−7	−5	−1	1
12	−4	−2	−1	1
13	−1	1	−1	1
14	2	4	−1	1
15	5	7	−1	1
16	−7	−5	2	4
17	−4	−2	2	4
18	−1	1	2	4
19	2	4	2	4
20	5	7	2	4
21	−7	−5	5	7
22	−4	−2	5	7
23	−1	1	5	7
24	2	4	5	7
25	5	7	5	7

The approximate convolution filter that all functions constructed in this example correspond to is:

$$\begin{bmatrix} 3.548 & 4.193 & 3.548 \\ 4.193 & 4.954 & 4.193 \\ 3.548 & 4.193 & 3.548 \end{bmatrix}. \tag{4.294}$$

The set of orthogonal basis functions we can create from them has the above 3×3 matrix placed in all tessellated 3×3 patches of the image space.

(Continued)

Example B4.59 (Continued)

Matrix (4.294) may be used as a convolution filter to extract the frequency content of any 3×3 *image patch in the* 5×5 *low frequency band centred around the dc component of the image. As the filter was truncated, its response will not be exactly zero outside the* 5×5 *band, so some leakage is expected. Before use, this filter must be normalised so that its elements add up to 1. The sum of the elements of (4.294) is 35.702. Once we divide the elements of (4.294) with 35.702, we recover again filter* w^{22} *we constructed in example 4.58.*

Box 4.5 What is the difference between projection onto basis functions and convolution with the corresponding filters?

A projection is performed by point by point multiplication of the basis function/image and the image we wish to analyse, and summation of the results. So, to perform the projection, the basis function on which we project the signal has to be of the same size as the signal. The information on different localities is extracted by having a different basis function for each locality. This is shown schematically in Figure 4.70. If the localities that interest us are not non-overlapping, but rather dense, shifted by a single sample one from the other, then we can replace the idea of projection with the idea of pseudo-convolution; from the basis function we cut off the parts that are 0 or nearly 0, and use the remaining part as a convolution filter. To preserve the idea of projection, we do not read the values of the filter backwards when we perform the convolution, but we simply place the centre of the filter at every sample, multiply the corresponding values and add (see Figure 4.70).

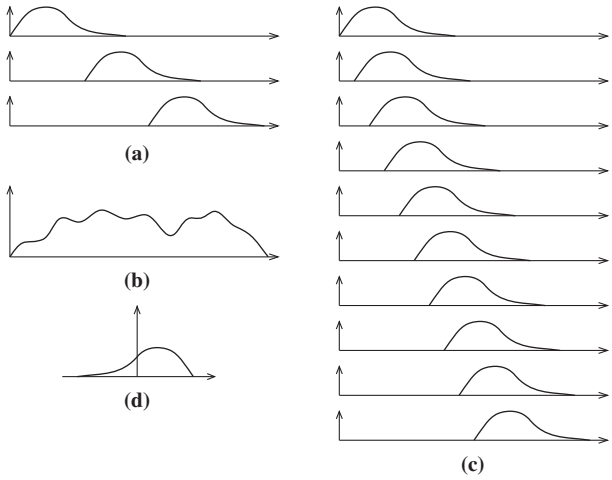


Figure 4.70 (a) Basis functions on which the signal shown in (b) can be projected to yield its content in a particular frequency band, at non-overlapping localities. (c) If we are interested in the frequency content of the signal at the vicinity of each sample, the basis functions are more, with the significant (non-zero) part of each one densely placed. (d) In such a case, we may cut off the parts of these functions that are near 0 and create an approximate convolution filter that we can use to process the signal so its frequency content in the particular band is extracted (approximately) at the locality of each sample. Note that as convolution uses the filter back to front, we reverse it before using it for convolution, so the two reversals cancel each other and the results of projection and convolution are the same. Alternatively, we may perform pseudo-convolution, i.e. we neither reverse the filter beforehand, nor during convolution.

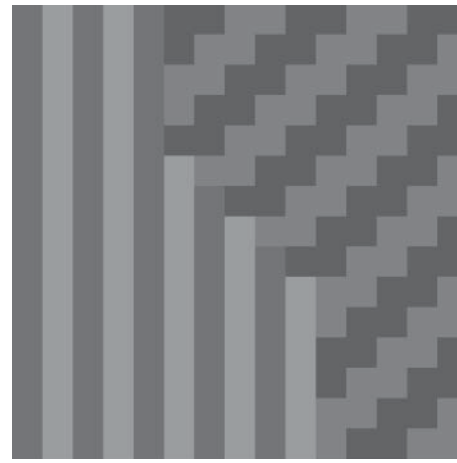
Alternatively, from a convolution filter one may construct projection functions by reversing the filter, padding it with 0s, to make it the same size as the signal, and shifting its non-zero part to start from different sample positions. In Figure 4.70 this corresponds to going from (d) to (c). Set of functions (c) constitutes an over-complete and non-orthogonal basis for expanding signal (b). If the filter is placed in non-overlapping sub-segments of the signal, we may construct a complete and orthogonal basis. In Figure 4.70 this corresponds to going from (d) to (a).

Example B4.60

Use the nine 3×3 filters constructed in Example 4.58 as convolution filters to construct feature maps for the image of Figure 4.71.

100	150	100	150	100	80	80	120	120	80	80	120	120	80	80
100	150	100	150	100	80	120	120	80	80	120	120	80	80	120
100	150	100	150	100	120	120	80	80	120	120	80	80	120	120
100	150	100	150	100	120	80	80	120	120	80	80	120	120	80
100	150	100	150	100	80	80	120	120	80	80	120	120	80	80
100	150	100	150	100	150	120	120	80	80	120	120	80	80	120
100	150	100	150	100	150	100	80	80	120	120	80	80	120	120
100	150	100	150	100	150	100	150	120	120	80	80	120	120	80
100	150	100	150	100	150	100	150	100	80	80	120	120	80	80
100	150	100	150	100	150	100	150	100	150	120	120	80	80	120
100	150	100	150	100	150	100	150	100	150	120	80	80	120	120
100	150	100	150	100	150	100	150	100	150	80	80	120	120	80
100	150	100	150	100	150	100	150	100	150	80	120	120	80	80
100	150	100	150	100	150	100	150	100	150	120	120	80	80	120
100	150	100	150	100	150	100	150	100	150	120	80	80	120	120

(a)



(b)

Figure 4.71 An image consisting of two different textures. (a) The pixel values. (b) The grey image. Source: Maria Petrou.

We convolve the image with each one of the filters of Equations (4.282)–(4.290). We assume that the image is repeated ad infinitum in all directions, so that we avoid boundary problems. Figures 4.72–4.74 show the magnitude, real part and imaginary part, respectively, of the nine outputs we produce this way. From each output we may compute the local energy of the image in each band. The averaged energies inside windows of size 3×3 around each pixel are shown in Figure 4.75.

We note that the feature maps constructed from the eight high pass filter outputs are pairwise identical, as expected. So, we need only half of them. From these results we can see that the two textures are easily distinguished, particularly from the outputs of filters 11 (or 33) and 23 (or 21).

Identical results (apart from a multiplicative factor arising due to the normalisation of the filters) would have been obtained if we were using the real part of each of the four pairs of filters.

(Continued)

Example B4.60 (Continued)

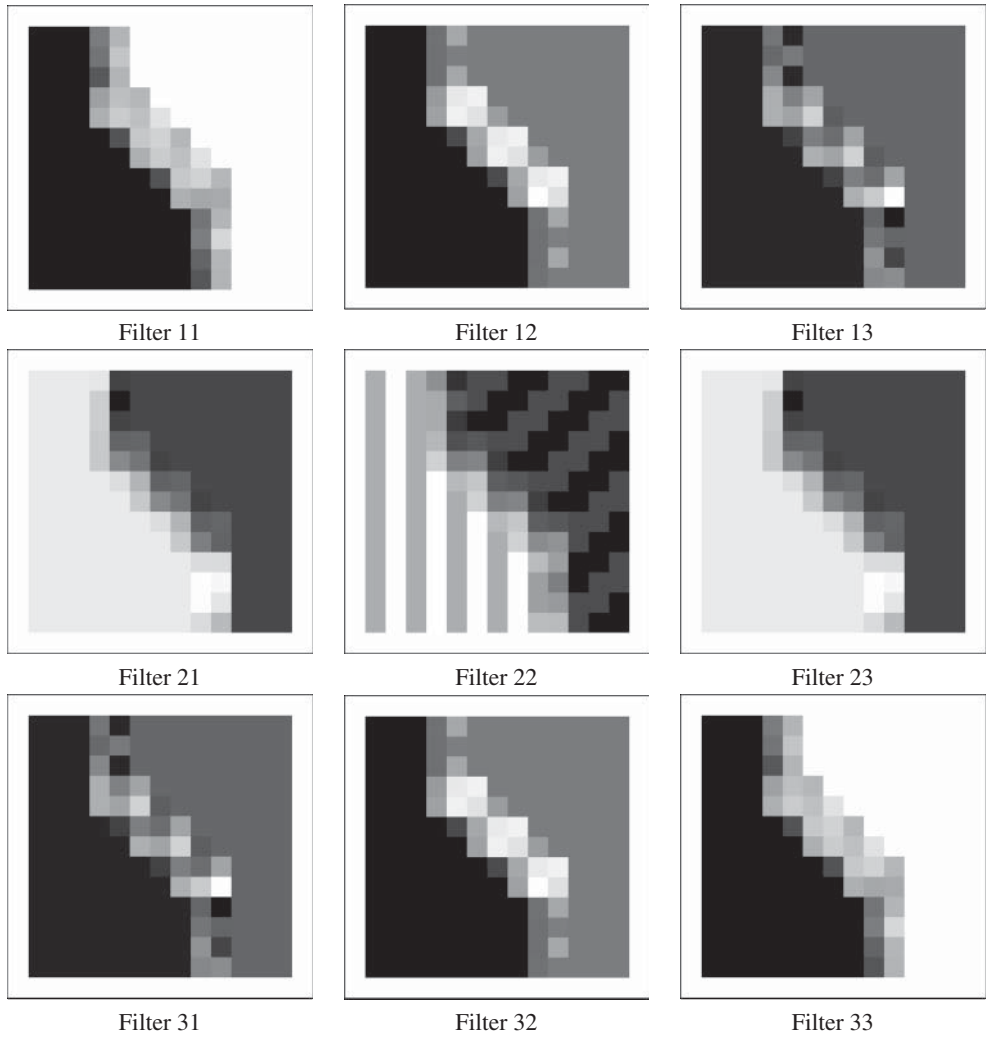


Figure 4.72 Magnitude of the outputs of the nine convolution filters designed in Example 4.58 when applied to the image of Figure 4.71. A one-pixel wide border of pixels with unreliable and discarded values exists around each output. Source: Maria Petrou.

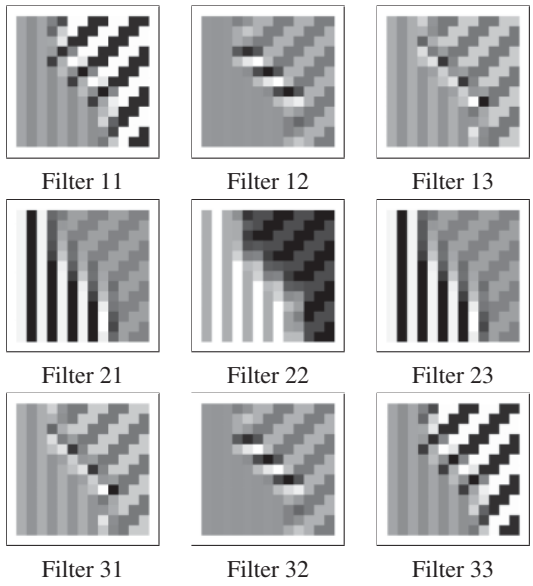
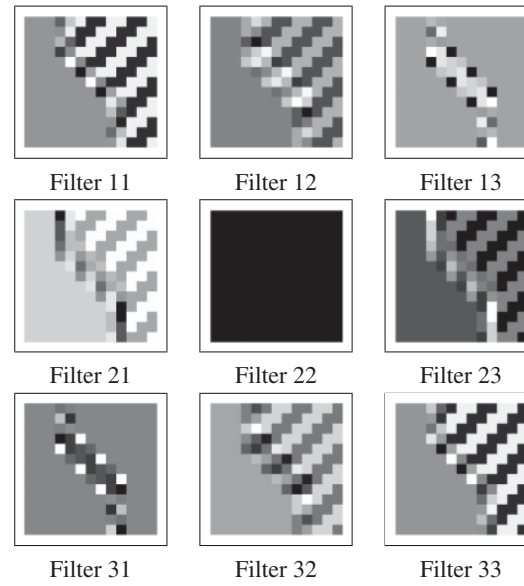


Figure 4.73 Real part of the outputs of the nine convolution filters designed in Example 4.58, when applied to the image of Figure 4.71. Source: Maria Petrou.

Figure 4.74 Imaginary part of the outputs of the nine convolution filters designed in Example 4.58, when applied to the image of Figure 4.71. Source: Maria Petrou.



(Continued)

Example B4.60 (Continued)

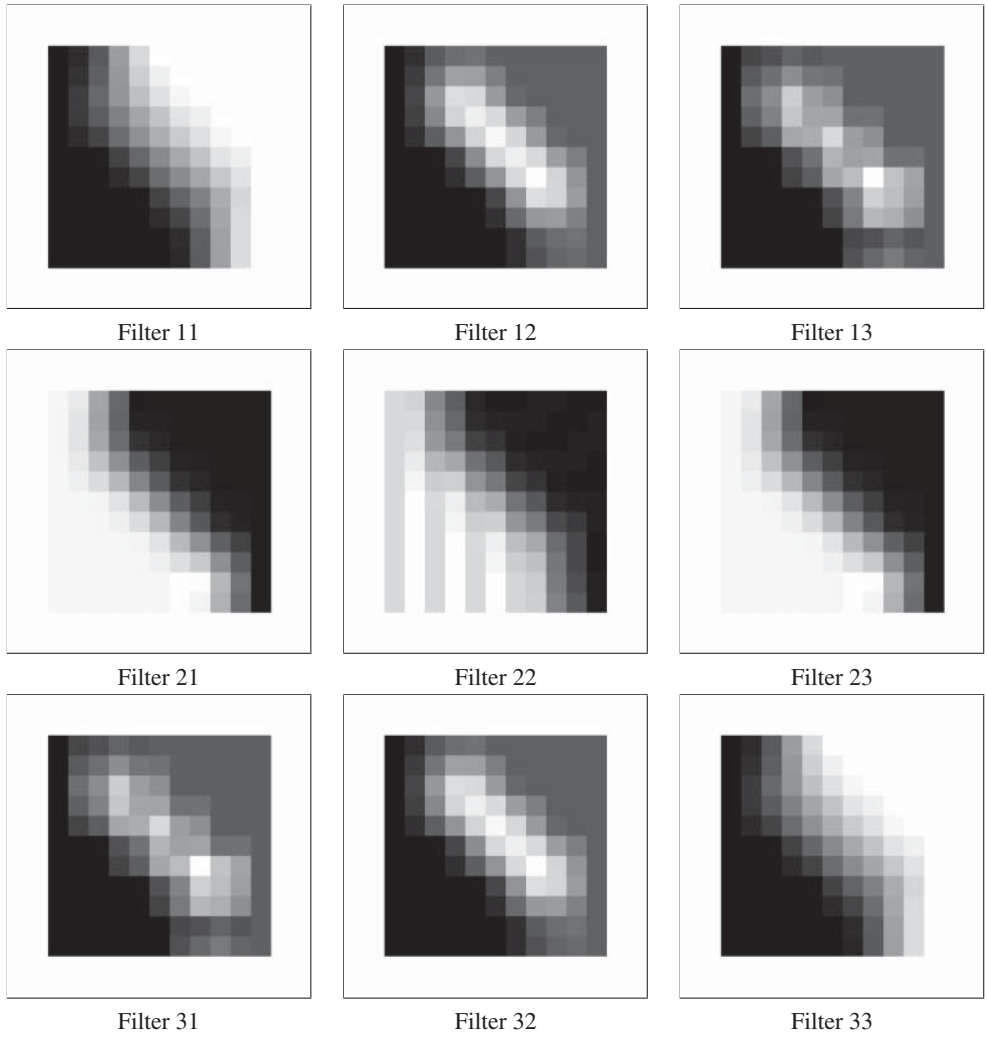


Figure 4.75 Averaged energies inside a window of size 3×3 of the outputs of the nine filters designed in Example 4.58, when applied to the image of Figure 4.71. A two-pixel wide border of unprocessed pixels exists around each output. Source: Maria Petrou.

Example B4.61

Show that the necessary and sufficient conditions for a 3×3 filter w, with weights w_{ij}, to be separable are:

$$\frac{w_{11}}{w_{12}} = \frac{w_{21}}{w_{22}} = \frac{w_{31}}{w_{32}}$$
$$\frac{w_{12}}{w_{13}} = \frac{w_{22}}{w_{23}} = \frac{w_{32}}{w_{33}}$$

$$\frac{w_{11}}{w_{21}} = \frac{w_{12}}{w_{22}} = \frac{w_{13}}{w_{23}}$$
$$\frac{w_{21}}{w_{31}} = \frac{w_{22}}{w_{32}} = \frac{w_{23}}{w_{33}}. \tag{4.295}$$

An $M \times M$ filter is **separable** *if it can be written as the outer product of two $M \times 1$ vectors. Let us call these vectors* $\mathbf{u}$ *and* $\mathbf{v}$*, and let us set $M = 3$. The outer product of vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *is:*

$$\boldsymbol{uv}^T = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} = \begin{pmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \end{pmatrix}. \tag{4.296}$$

We shall prove first that the satisfaction of conditions (4.295) is necessary for a 3×3 filter w to be separable.

If a 3×3 filter w is separable, there must exist two vectors $\mathbf{u}$ *and* $\mathbf{v}$ *so that we may write:*

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} = \begin{pmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \end{pmatrix}. \tag{4.297}$$

By equating the corresponding elements, we obtain:

$$\begin{aligned} w_{11} &= u_1v_1 & w_{12} &= u_1v_2 & w_{13} &= u_1v_3 \\ w_{21} &= u_2v_1 & w_{22} &= u_2v_2 & w_{23} &= u_2v_3 \\ w_{31} &= u_3v_1 & w_{32} &= u_3v_2 & w_{33} &= u_3v_3. \end{aligned} \tag{4.298}$$

By simple inspection, we see that from these expressions by pairwise divisions we may obtain:

$$\begin{aligned} \frac{w_{11}}{w_{12}} &= \frac{w_{21}}{w_{22}} = \frac{w_{31}}{w_{32}} = \frac{v_1}{v_2} \\ \frac{w_{12}}{w_{13}} &= \frac{w_{22}}{w_{23}} = \frac{w_{32}}{w_{33}} = \frac{v_2}{v_3} \\ \frac{w_{11}}{w_{21}} &= \frac{w_{12}}{w_{22}} = \frac{w_{13}}{w_{23}} = \frac{u_1}{u_2} \\ \frac{w_{21}}{w_{31}} &= \frac{w_{22}}{w_{32}} = \frac{w_{23}}{w_{33}} = \frac{u_2}{u_3}. \end{aligned} \tag{4.299}$$

So if the filter is separable, conditions (4.295) are fulfilled. Now we shall show that these conditions are also sufficient, i.e. we shall show that if conditions (4.295) are fulfilled, the filter is separable.

Let us start by setting the right-hand sides of Equations (4.295) equal to some constants, a, b, c and d, from top to bottom, respectively. Then we may write:

$$\begin{aligned} w_{11} &= aw_{12} & w_{21} &= aw_{22} & w_{31} &= aw_{32} \\ w_{12} &= bw_{13} & w_{22} &= bw_{23} & w_{32} &= bw_{33} \\ w_{11} &= cw_{21} & w_{12} &= cw_{22} & w_{13} &= cw_{23} \\ w_{21} &= dw_{31} & w_{22} &= dw_{32} & w_{23} &= dw_{33}. \end{aligned} \tag{4.300}$$

From the first two lines of the above expressions we infer that $w_{11} = abw_{13}$, $w_{21} = abw_{23}$ and $w_{31} = abw_{33}$. So, we may write:

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} = \begin{pmatrix} abw_{13} & bw_{13} & w_{13} \\ abw_{23} & bw_{23} & w_{23} \\ abw_{33} & bw_{33} & w_{33} \end{pmatrix} = \begin{pmatrix} w_{13} \\ w_{23} \\ w_{33} \end{pmatrix} \begin{pmatrix} ab & b & 1 \end{pmatrix}. \tag{4.301}$$

This concludes the proof of the theorem.

Example B4.62

Show that filter (4.272) is separable and write it as a vector outer product. Is your answer unique?

We must check first whether conditions (4.295) are fulfilled. We note that this filter has the following structure:

$$\begin{bmatrix} A & B & A \\ B & C & B \\ A & B & A \end{bmatrix}. \tag{4.302}$$

Here $A = 3.548$, $B = 4.193$ *and* $C = 4.954$. *It is obvious then that the conditions are fulfilled if* $A/B = B/C$. *In the particular case* $3.548/4.193 = 4.193/4.954 = 0.846$ *and indeed the filter is separable, with ratios a, b, c and d, as defined in Example 4.61 equal to* $a = 0.846$, $b = 1.182$, $c = 0.846$ *and* $d = 1.182$. *So, substituting into (4.301), we may write the filter as*

$$\begin{pmatrix} 3.548 \\ 4.193 \\ 3.548 \end{pmatrix} \begin{pmatrix} 1 & 1.182 & 1 \end{pmatrix}. \tag{4.303}$$

In general, this answer is not unique (see also Example 4.63). For example, if in Example 4.61 we had used the expressions in the last two rows of (4.300), we would have deduced that

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} = \begin{pmatrix} cdw_{13} & cdw_{13} & cdw_{13} \\ dw_{23} & dw_{23} & dw_{23} \\ w_{33} & w_{33} & w_{33} \end{pmatrix} = \begin{pmatrix} cd \\ d \\ 1 \end{pmatrix} \begin{pmatrix} w_{31} & w_{32} & w_{33} \end{pmatrix}. \tag{4.304}$$

In our case, because of the symmetric structure of the filter, $c = a$ *and* $d = b$, *the two vectors deduced this way are the same vectors deduced by applying (4.301), but used in reverse order.*

Box 4.6 Could we construct the 2D prolate spheroidal sequence filters as separable filters?

Yes, because we may write matrix $S(k, l, \tilde{k}, \tilde{l})$, defined by Equation (4.255) as the product of two matrices, one of which depends on k and $\tilde{k}$, and one which depends on l and $\tilde{l}$:

$$S_1(k, \tilde{k}) \equiv \frac{\sin\left(\frac{\pi}{N}(k - \tilde{k})(n_2 - n_1 + 1)\right)}{\sin\left(\frac{\pi}{N}(k - \tilde{k})\right)}$$
$$S_2(l, \tilde{l}) \equiv \frac{\sin\left(\frac{\pi}{M}(l - \tilde{l})(m_2 - m_1 + 1)\right)}{\sin\left(\frac{\pi}{M}(l - \tilde{l})\right)}. \tag{4.305}$$

Note that the term "product" above does not mean the matrix product of S_1 and S_2, but rather their product when they are considered as simple functions. Then Equation (4.256) may be written as:

$$\sum_{\tilde{k}=-\frac{N-1}{2}}^{\frac{N-1}{2}} \sum_{\tilde{l}=-\frac{M-1}{2}}^{\frac{M-1}{2}} \psi(\tilde{k}, \tilde{l}) S_1(k, \tilde{k}) S_2(l, \tilde{l}) = RNM\psi(k, l)$$
$$\Rightarrow \sum_{\tilde{k}=-\frac{N-1}{2}}^{\frac{N-1}{2}} S_1(k, \tilde{k}) \sum_{\tilde{l}=-\frac{M-1}{2}}^{\frac{M-1}{2}} \psi(\tilde{k}, \tilde{l}) S_2(l, \tilde{l}) = RNM\psi(k, l). \tag{4.306}$$

According to Examples 4.63 and 4.64, the solution $\psi(k,l)$ of the above equation is the outer product of the eigenvectors of matrices S_1 and S_2. The solution then of Equation (4.256) may be found by solving two 1D eigenvalue problems

$$\sum_{\tilde{l}=-\frac{M-1}{2}}^{\frac{M-1}{2}} S_2(l,\tilde{l})\psi(\tilde{k},\tilde{l}) = \mu\psi(\tilde{k},l) \tag{4.307}$$

and

$$\sum_{\tilde{k}=-\frac{N-1}{2}}^{\frac{N-1}{2}} S_1(k,\tilde{k})\psi(\tilde{k},l) = \frac{RNM}{\mu}\psi(k,l) \tag{4.308}$$

to produce two 1D filters, the vector outer product of which is the same as the 2D filter produced by solving Equation (4.256) directly.

Example B4.63

Show that if the outer product of 3×1 vectors u and v is equal to the outer product of vectors a and b, vector u is equal to vector a and vector v is equal to vector b apart from some scaling factors.

Equating the outer products $\mathbf{uv}^T$ *and* $\mathbf{ab}^T$ *we obtain:*

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \end{pmatrix} = \begin{pmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{pmatrix}. \tag{4.309}$$

By equating the corresponding elements of the two matrices and taking appropriate ratios from them, we can easily verify that

$$\frac{u_1}{a_1} = \frac{u_2}{a_2} = \frac{u_3}{a_3} \qquad \text{and} \qquad \frac{v_1}{b_1} = \frac{v_2}{b_2} = \frac{v_3}{b_3} \tag{4.310}$$

which yield $(u_1,u_2,u_3)^T = \lambda(a_1,a_2,a_3)^T$ *and* $(v_1,v_2,v_3)^T = \mu(b_1,b_2,b_3)^T$ *where* λ *and* μ *are some scaling constants.*

Example B4.64

Show that Equation (4.306) implies that the eigenfunctions $\psi(k,l)$ of matrix $S(k,l,\tilde{k},\tilde{l})$ may be obtained by taking the vector outer products of the eigenvectors of matrices $S_1(k,\tilde{k})$ and $S_2(l,\tilde{l})$.

We start by writing Equation (4.306) in matrix form, representing function $\psi(k,l)$ *as a 2D matrix* Ψ *and noticing that the inner sum on the left-hand side of (4.306) is nothing else than the matrix*

(Continued)

Example B4.64 (Continued)

product of matrix Ψ and the transpose of matrix S_2

$$S_1 \Psi S_2^T = \beta \Psi \tag{4.311}$$

where we used β to represent all scaling constants on the right-hand side of the equation.

In Book I [75] it was shown that **any** *matrix Ψ may be analysed by performing singular value decomposition (SVD) into the sum of the vector outer products of the eigenvalues of two matrices, $U \equiv \Psi\Psi^T$ and $V \equiv \Psi^T\Psi$. These two matrices were shown to have the same eigenvalues, the square roots of which are the so-called* **singular values** *of matrix Ψ. Let us call these eigenvalues λ_I and the corresponding eigenvectors $\mathbf{u_I}$ and $\mathbf{v_I}$. So, according to the SVD theory, we may write:*

$$\Psi = \sum_i \sqrt{\lambda_I} \mathbf{u_I} \mathbf{v_I}^T. \tag{4.312}$$

We substitute this expansion into (4.311) to obtain:

$$\begin{aligned} & S_1 \sum_i \sqrt{\lambda_I} \mathbf{u_I} \mathbf{v_I}^T S_2^T = \beta \sum_i \sqrt{\lambda_I} \mathbf{u_I} \mathbf{v_I}^T \\ \Rightarrow & \sum_i \sqrt{\lambda_I} S_1 \mathbf{u_I} (S_2 \mathbf{v_I})^T = \beta \sum_i \sqrt{\lambda_I} \mathbf{u_I} \mathbf{v_I}^T \\ \Rightarrow & \sum_i \sqrt{\lambda_I} (S_1 \mathbf{u_I} (S_2 \mathbf{v_I})^T - \beta \mathbf{u_I} \mathbf{v_I}^T) = 0. \end{aligned} \tag{4.313}$$

The last equation is valid if all elements of the sum are equal to zero, i.e. if the outer product of vectors $S_1\mathbf{u_I}$ and $S_2\mathbf{v_I}$ is equal to the outer product of vectors $\beta\mathbf{u_I}$ and $\mathbf{v_I}$. According to Example 4.63 this is true when vector $S_1\mathbf{u_I}$ is equal to vector $\beta\mathbf{u_I}$, and vector $S_2\mathbf{v_I}$ is equal to vector $\mathbf{v_I}$ apart from some scaling constants

$$S_1 \mathbf{u_I} = \mu_1 \mathbf{u_I} \qquad S_2 \mathbf{v_I} = \mu_2 \mathbf{v_I} \tag{4.314}$$

where μ_1 and μ_2 are some scalars. These two equations tell us that vectors $\mathbf{u_I}$ and $\mathbf{v_I}$ are the eigenvectors of matrices S_1 and S_2 respectively, and so Equation (4.312) tells us that matrix Ψ (i.e. function $\psi(k,l)$) can be expressed as the vector outer product of these eigenvectors. If we choose vectors $\mathbf{u_I}$ and $\mathbf{v_I}$ to correspond to the largest eigenvalues of matrices S_1 and S_2, respectively, then $\Psi = \mathbf{u_I}\mathbf{v_I}^T$ will correspond to the first eigenvector of S. Substituting in (4.311), we get

$$S_1 \Psi S_2^T = S_1 \mathbf{u_I} \mathbf{v_I}^T S_2^T = S_1 \mathbf{u_I} (S_2 \mathbf{v_I})^T = \mu_{1max} \mathbf{u_I} \mu_{2max} \mathbf{v_I}^T = \beta \mathbf{u_I} \mathbf{v_I}^T \tag{4.315}$$

which makes β maximum.

Example B4.65

Produce the separable form of filters (4.282)–(4.290) by solving the 1D eigenvalue problem.

For the 1D case we have $N = 15$, $n_1 = -1$, $n_2 = 1$, while $(k_1, k_2) \in \{(-7,-3),(-2,2),(3,7)\}$. We derive the following three filters, presented here as vectors, rather than as sequences of numbers:

$$\begin{aligned} \mathbf{w_1} &\equiv (-0.942 + 1.631\mathrm{j}, 2.226, -0.942 - 1.631\mathrm{j})^T \\ \mathbf{w_2} &\equiv (1.884, 2.226, 1.884)^T \\ \mathbf{w_3} &\equiv (-0.942 - 1.631\mathrm{j}, 2.226, -0.942 + 1.631\mathrm{j})^T. \end{aligned} \tag{4.316}$$

We take all possible vector outer products of them to produce:

$$\mathbf{w_1}\mathbf{w_1}^T = \begin{bmatrix} -1.773 - 3.073\mathrm{j} & -2.097 + 3.631\mathrm{j} & 3.548 \\ -2.097 + 3.631\mathrm{j} & 4.955 & -2.097 - 3.631\mathrm{j} \\ 3.548 & -2.097 - 3.631\mathrm{j} & -1.773 + 3.073\mathrm{j} \end{bmatrix} \tag{4.317}$$

$$\mathbf{w}_1\mathbf{w}_2^T = \begin{bmatrix} -1.775 + 3.073\mathrm{j} & -2.097 + 3.631\mathrm{j} & -1.775 + 3.073\mathrm{j} \\ 4.194 & 4.955 & 4.194 \\ -1.775 - 3.073\mathrm{j} & -2.097 - 3.631\mathrm{j} & -1.775 - 3.073\mathrm{j} \end{bmatrix} \tag{4.318}$$

$$\mathbf{w}_1\mathbf{w}_3^T = \begin{bmatrix} 3.548 & -2.097 + 3.631\mathrm{j} & -1.773 - 3.073\mathrm{j} \\ -2.097 - 3.631\mathrm{j} & 4.955 & -2.097 + 3.631\mathrm{j} \\ -1.773 + 3.073\mathrm{j} & -2.097 - 3.631\mathrm{j} & 3.548 \end{bmatrix} \tag{4.319}$$

$$\mathbf{w}_2\mathbf{w}_1^T = \begin{bmatrix} -1.775 + 3.073\mathrm{j} & 4.194 & -1.775 - 3.073\mathrm{j} \\ -2.097 + 3.631\mathrm{j} & 4.955 & -2.097 - 3.631\mathrm{j} \\ -1.775 + 3.073\mathrm{j} & 4.194 & -1.775 - 3.073\mathrm{j} \end{bmatrix} \tag{4.320}$$

$$\mathbf{w}_2\mathbf{w}_2^T = \begin{bmatrix} 3.549 & 4.194 & 3.549 \\ 4.194 & 4.955 & 4.194 \\ 3.549 & 4.194 & 3.549 \end{bmatrix} \tag{4.321}$$

$$\mathbf{w}_2\mathbf{w}_3^T = \begin{bmatrix} -1.775 - 3.073\mathrm{j} & 4.194 & -1.775 + 3.073\mathrm{j} \\ -2.097 - 3.631\mathrm{j} & 4.955 & -2.097 + 3.631\mathrm{j} \\ -1.775 - 3.073\mathrm{j} & 4.194 & -1.775 + 3.073\mathrm{j} \end{bmatrix} \tag{4.322}$$

$$\mathbf{w}_3\mathbf{w}_1^T = \begin{bmatrix} 3.548 & -2.097 - 3.631\mathrm{j} & -1.773 + 3.073\mathrm{j} \\ -2.097 + 3.631\mathrm{j} & 4.955 & -2.097 - 3.631\mathrm{j} \\ -1.773 - 3.073\mathrm{j} & -2.097 + 3.631\mathrm{j} & 3.548 \end{bmatrix} \tag{4.323}$$

$$\mathbf{w}_3\mathbf{w}_2^T = \begin{bmatrix} -1.775 - 3.073\mathrm{j} & -2.097 - 3.631\mathrm{j} & -1.775 - 3.073\mathrm{j} \\ 4.194 & 4.955 & 4.194 \\ -1.775 + 3.073\mathrm{j} & -2.097 + 3.631\mathrm{j} & -1.775 + 3.073\mathrm{j} \end{bmatrix} \tag{4.324}$$

$$\mathbf{w}_3\mathbf{w}_3^T = \begin{bmatrix} -1.773 + 3.073\mathrm{j} & -2.097 - 3.631\mathrm{j} & 3.548 \\ -2.097 - 3.631\mathrm{j} & 4.955 & -2.097 + 3.631\mathrm{j} \\ 3.548 & -2.097 + 3.631\mathrm{j} & -1.773 - 3.073\mathrm{j} \end{bmatrix}. \tag{4.325}$$

These filters after normalisation become identical to the filters given by equations (4.282)–(4.290).

Example B4.66

Show how you will use the 1D filters given by (4.316) to construct 2D separable filters that, if used to convolve the image, will produce its frequency output in pairs of bands symmetrically placed about the dc component.

We know that to have the frequency content of the image in a pair of bands symmetrically placed about the dc component is enough to use for the convolution the real part only of the filters we construct. For this purpose, we may use the real parts of the filters given by equations (4.317)–(4.325). However, these filters are not separable. Let us say that one such filter was constructed by taking

(Continued)

Example B4.66 (Continued)

the vector outer product of the 1D filters $(a_1 + \mathrm{j}b_1, a_2 + \mathrm{j}b_2)^T$ *and* $(a_3 + \mathrm{j}b_3, a_4 + \mathrm{j}b_4)^T$*:*

$$\begin{pmatrix} a_1 + \mathrm{j}b_1 \\ a_2 + \mathrm{j}b_2 \end{pmatrix} \begin{pmatrix} a_3 + \mathrm{j}b_3 & a_4 + \mathrm{j}b_4 \end{pmatrix} = \begin{pmatrix} a_1a_3 - b_1b_3 + \mathrm{j}(a_1b_3 + b_1a_3) & a_1a_4 - b_1b_4 + \mathrm{j}(a_1b_4 + b_1a_4) \\ a_2a_3 - b_2b_3 + \mathrm{j}(a_2b_3 + b_2a_3) & a_2a_4 - b_2b_4 + \mathrm{j}(a_2b_4 + b_2a_4) \end{pmatrix}. \tag{4.326}$$

If we take only the real part of the filter, we shall have:

$$\begin{aligned} \begin{pmatrix} a_1a_3 - b_1b_3 & a_1a_4 - b_1b_4 \\ a_2a_3 - b_2b_3 & a_2a_4 - b_2b_4 \end{pmatrix} &= \begin{pmatrix} a_1a_3 & a_1a_4 \\ a_2a_3 & a_2a_4 \end{pmatrix} - \begin{pmatrix} b_1b_3 & b_1b_4 \\ b_2b_3 & b_2b_4 \end{pmatrix} \\ &= \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \begin{pmatrix} a_3 & a_4 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \begin{pmatrix} b_3 & b_4 \end{pmatrix}. \end{aligned} \tag{4.327}$$

So, to construct the separable 2D filters we need, we must use the following two sets of filters:

$$\begin{aligned} \mathbf{w_{1R}} &\equiv (-0.942, 2.226, -0.942)^T \\ \mathbf{w_{2R}} &\equiv (1.884, 2.226, 1.884)^T \\ \mathbf{w_{3R}} &\equiv (-0.942, 2.226, -0.942)^T \end{aligned} \tag{4.328}$$

$$\begin{aligned} \mathbf{w_{1I}} &\equiv (1.631, 0, -1.631)^T \\ \mathbf{w_{2I}} &\equiv (0, 0, 0)^T \\ \mathbf{w_{3I}} &\equiv (-1.631, 0, 1.631)^T. \end{aligned} \tag{4.329}$$

What we have to do, therefore, is to convolve the image with all possible combinations of filters (4.328) in a cascaded way, using one filter along the horizontal axis and one along the vertical, do the same with filters (4.329), and then subtract the second lot of outputs from the corresponding first ones.

Note that there will be only five distinct outputs this way, due to the regularities presented by the filters. This is shown schematically in Figure 4.76.

	1	2	3
1	A	C	A
2	D	B	D
3	A	C	A

—

	1	2	3
1	E	0	−E
2	0	0	0
3	−E	0	E

=

A−E	C	A+E
D	B	D
A+E	C	A−E

Figure 4.76 The numbers on the top and left of the grids identify the indices *i* and *j* of the filters $\mathbf{w_{iR}}\mathbf{w_{jR}}^T$ or $\mathbf{w_{iI}}\mathbf{w_{jI}}^T$ that are multiplied to produce the result that is denoted by a capital letter (or a $\mathbf{0}$, if it is a matrix made up from 0s) in the corresponding cell. The distinct outcomes in each case are depicted with different letters. For example, outer product $\mathbf{w_{1I}}\mathbf{w_{1I}}^T$ is the same as outer product $\mathbf{w_{3I}}\mathbf{w_{3I}}^T$, depicted with the letter $\mathbf{E}$. It is noted that in the final output, when the results of the two 2D separable convolutions are combined, there are only 5 distinct outputs, out of the possible 9.

Box 4.7 What is the advantage of using separable filters?

The convolution of an image with an $M \times M$ filter implies M^2 operations per pixel. If the filter can be written as the outer product of two vectors of size $M \times 1$ each, these two vectors may be used as two 1D filters applied one after the other to the image, in a cascaded way, along two orthogonal directions, to produce the same output as the 2D filter would have produced. This means that the M^2 operations per pixel are replaced by $2M$ operations: a very significant gain for large filters.

How do we perform the analysis of an image using prolate spheroidal sequence functions in practice?

The algorithm that follows is for a square image of size $N \times N$, with N being odd.

Step 1: Specify the size of the convolution filters you wish to use. Call it $(2M+1) \times (2M+1)$. Specify also the size K of the bands along each axis in the frequency domain, that should be selected by each filter. It is more convenient if K is odd.

Step 2: In the frequency domain, the indices along each axis will take values in the range $\left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$. Divide this range in non-overlapping sub-ranges of length K each. Make sure that one of the ranges is $\left[-\frac{K-1}{2}, \frac{K-1}{2}\right]$ so it includes the 0 index that corresponds to the dc component. We expect to have an odd number of ranges, one containing the 0 frequency index and the others being in pairs of positive and negative indices. Let us say that there are $2V+1$ ranges along one of the axes. We shall use all bands along one axis in combination with all positive bands (V of them) of the other axis (i.e. $(2V+1) \times V$ 2D bands), plus the 0 frequency band of the second axis in combination with all the positive bands of the first axis (V of them) and the 0 frequency band of the first axis. This means that the distinct 2D filter bands we shall use will be $(2V+1)V + V + 1$ (see Figure 4.76, which corresponds to the case $K = 3$). Let us say that they are L distinct 2D bands in total ($L \equiv (2V+1)V + V + 1$).

Step 3: Construct a matrix S of size $N \times N$, with elements

$$S(k, \tilde{k}) = \frac{\sin \frac{\pi(k-\tilde{k})(2M+1)}{N}}{\sin \frac{\pi(k-\tilde{k})}{N}} \tag{4.330}$$

where k and $\tilde{k}$ take all integer values in the range $\left[-\frac{K-1}{2}, \frac{K-1}{2}\right]$.

Step 4: Work out the largest eigenvalue of S and the corresponding eigenvector. Call its elements $\psi(k)$, for $k \in \left[-\frac{K-1}{2}, \frac{K-1}{2}\right]$.

Step 5: Compute

$$w(n) = \frac{1}{N} \sum_{k=-\frac{K-1}{2}}^{\frac{K-1}{2}} \psi(k) e^{j\frac{2\pi}{N}kn} \tag{4.331}$$

for $n \in \left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$.

Step 6: Keep only the values of $w(n)$ for $n = -M, -M+1, \ldots, 0, \ldots, M-1, M$. Normalise them by dividing them with their sum to create the 1D low pass filter of size $2M+1$, the elements of which sum up to 1. Call this filter f_0.

Step 7: For each of the positive ranges of indices you constructed in step 2, construct matrix S using (4.330) allowing k to take values in the selected range, but $\tilde{k}$ to take values in $\left[-\frac{K-1}{2}, \frac{K-1}{2}\right]$. For each such matrix S constructed, compute its largest eigenvector. Call its elements $\psi(k)$, for $k \in \left[-\frac{K-1}{2}, \frac{K-1}{2}\right]$. From each $\psi(k)$ compute $w(n)$ using Equation (4.331). Truncate each $w(n)$ to keep only its central $2M+1$ values. Call the results $\tilde{a}(n) + \mathrm{j}\tilde{b}(n)$, where $n = -M, \ldots, 0, \ldots, M$, and $\tilde{a}(n)$ and $\tilde{b}(n)$ are real. Sum up all the values of $\tilde{a}(n) + \mathrm{j}\tilde{b}(n)$ and divide them with $2M+1$. Call the result $c + \mathrm{j}d$. Subtract $c + \mathrm{j}d$ from all values of $\tilde{a}(n) + \mathrm{j}\tilde{b}(n)$. Call the result $a(n) + \mathrm{j}b(n)$. The result will be the convolution band pass filter designed to pick up frequencies in the image in the corresponding frequency band. We are going to have $2V+1$ such filters. To distinguish between them, and between their real and imaginary parts, let us denote them in a systematic way: f_{vR}, for $v = -V, \ldots, 0, \ldots, V$, identifies the real parts of the filters ($a(n)$) for the various bands, and f_{vI}, for $v = -V, \ldots, 0, \ldots, V$, identifies the corresponding imaginary parts of the filters ($b(n)$) for the same bands.

Step 8: Convolve the image with filters f_{vR}, for $v \in [-V, V]$, along the horizontal direction and each output with all filters f_{vR}, for $v \in [1, V]$, along the vertical direction. In addition, convolve the outputs of the horizontal convolutions with filters f_{vR}, for $v \in [0, V]$, with filter f_{0R} along the vertical direction. This way, you will have L distinct outputs.

Step 9: Repeat step 8 using filters f_{vI} this time. Due to symmetries of the filters you may be able to make this step even faster by omitting certain convolutions, but this may be confusing (see Figure 4.76, for the case $K = 3$, where only one pair of convolutions (horizontal followed by vertical) is necessary).

Step 10: Subtract the results of step 9 from the corresponding results of step 8. What you produce this way is the expansion of the image in terms of the prolate spheroidal sequence functions. From these expansions feature maps may be produced.

Step 11: Square the values of all L output matrices you produced in step 10.

Step 12: Convolve the arrays you produced in step 11 with a flat averaging window. The outputs will be the local energy feature maps of the original input image. To visualise these maps you may scale them collectively or individually to have values in the range [0,255].

Step 13: The sequence of values a pixel has in each one of the feature maps (without any scaling done for visualisation) constitutes the feature vector of the pixel and it may be used to classify the pixel.

Note that steps 1–7 are independent of the image. They concern the construction of the filters that will pick up different frequencies in the image. So, the constructed filters may be used for any image of the same size.

Example 4.67

Use prolate spheroidal sequence functions to construct 9×9 convolution filters with which to analyse the frequency content of a 255×255 image.

The problem only specifies the size of the convolution filters but not the size of the frequency band inside which each filter will be designed to have non-zero response. We choose to tessellate the frequency space into nine areas of size 85×85 each. The corresponding 1D problem we have to solve is for $N = 255$, $n_1 = -42$, $n_2 = 42$ and $(k_1, k_2) \in \{(-127, -43), (-42, 42), (43, 127)\}$. The filters we construct this way are:

$$\mathbf{w_1} = (-0.618 + 1.070\mathrm{j}, 3.146, -2.721 - 4.712\mathrm{j}, -3.668 + 6.353\mathrm{j}, 8.070,$$
$$-3.668 - 6.353\mathrm{j}, -2.721 + 4.712\mathrm{j}, 3.146, -0.618 - 1.070\mathrm{j})^T$$

$$\mathbf{w_2} = (1.236, 3.146, 5.441, 7.335, 8.070, 7.335, 5.441, 3.146, 1.236)^T$$
$$\mathbf{w_3} = (-0.618 - 1.070\text{j}, 3.146, -2.721 + 4.712\text{j}, -3.668 - 6.353\text{j}, 8.070,$$
$$-3.668 + 6.353\text{j}, -2.721 - 4.712\text{j}, 3.146, -0.618 + 1.070\text{j})^T. \quad (4.332)$$

If we take all vector outer products of these filters in all possible pairs, we shall obtain the nine 2D convolution filters shown in Figures 4.77 (real part) and 4.78 (imaginary part), arranged according to the frequency band in which they have their maximum response.

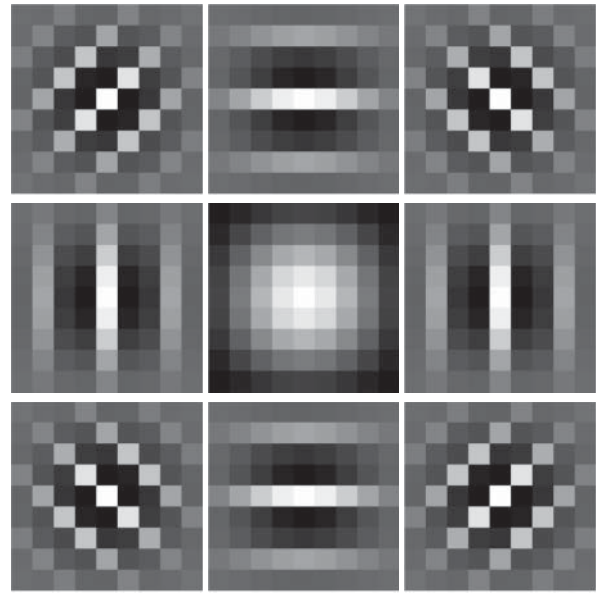


Figure 4.77 Real part of the nine filters designed. Each filter is of size 9×9 and it was designed to cover an area of size 85×85 in the Fourier domain. Source: Maria Petrou.

These filters are rather gross. In effect all they do is to measure the strength of horizontal lines and edges (filter w^{21} or w^{23}), of vertical lines and edges (filter w^{12} or w^{31}), and of round blobs.

We may try to characterise the image content in a little more detail if we tessellate the frequency space into 25 bands instead of nine only. Then each band will be of size 51×51. To divide the frequency space into 25 bands, we must choose for the 1D problem $(k_1, k_2) \in \{(-127, -77), (-76, -26), (-25, 25), (26, 76), (77,127)\}$. The filters we derive this way are:

$$\mathbf{w_1} = (-2.273 - 1.651\text{j}, 1.326 + 4.080\text{j}, 1.733 - 5.333\text{j}, -5.273 + 3.831\text{j},$$
$$6.842, -5.273 - 3.831\text{j}, 1.733 + 5.333\text{j}, 1.326 - 4.080\text{j}, -2.273 + 1.651\text{j})^T$$
$$\mathbf{w_2} = (0.868 - 2.672\text{j}, -3.471 - 2.522\text{j}, -4.537 + 3.296\text{j}, 2.014 + 6.198\text{j},$$
$$6.842, 2.014 - 6.198\text{j}, -4.537 - 3.296\text{j}, -3.471 + 2.522\text{j}, 0.868 + 2.672\text{j})^T$$
$$\mathbf{w_3} = (2.810, 4.290, 5.608, 6.517, 6.842, 6.517, 5.608, 4.290, 2.810)^T$$
$$\mathbf{w_4} = (0.868 + 2.672\text{j}, -3.471 + 2.522\text{j}, -4.537 - 3.296\text{j}, 2.014 - 6.198\text{j},$$
$$6.842, 2.014 + 6.198\text{j}, -4.537 + 3.296\text{j}, -3.471 - 2.522\text{j}, 0.868 - 2.672\text{j})^T$$
$$\mathbf{w_5} = (-2.273 + 1.651\text{j}, 1.326 - 4.080\text{j}, 1.733 + 5.333\text{j}, -5.273 - 3.831\text{j}, 6.842,$$
$$-5.273 + 3.831\text{j}, 1.733 - 5.333\text{j}, 1.326 + 4.080\text{j}, -2.273 - 1.651\text{j})^T. \quad (4.333)$$

(Continued)

Example 4.67 (Continued)

Figure 4.79 shows the real parts and Figure 4.80 the imaginary parts of the filters constructed by taking the vector outer products of these 1D filters, scaled individually to the range [0,255] *each, so they can be presented as images.*

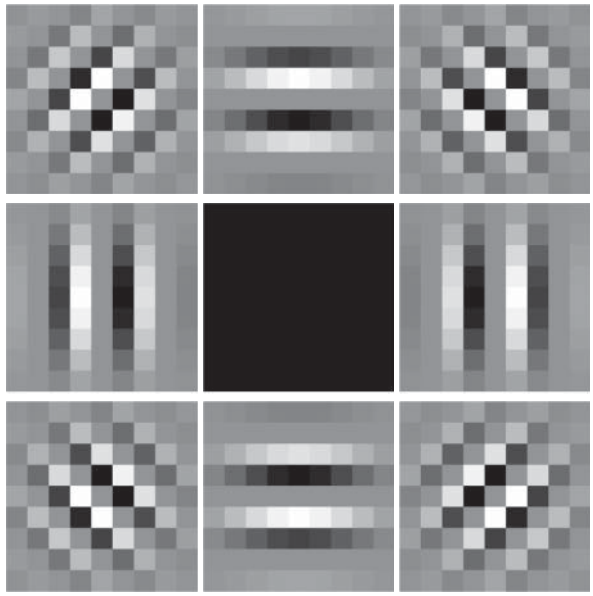


Figure 4.78 Imaginary part of the nine filters designed. Each filter is of size 9×9 and it was designed to cover an area of size 85×85 in the Fourier domain. Source: Maria Petrou.

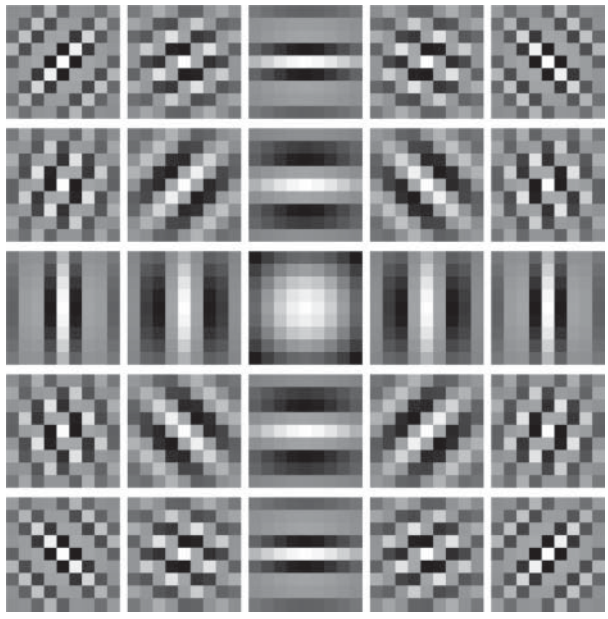


Figure 4.79 Real part of the 25 filters designed. Each filter is of size 9×9 and it was designed to cover and area of size 51×51 in the Fourier domain. Source: Maria Petrou.

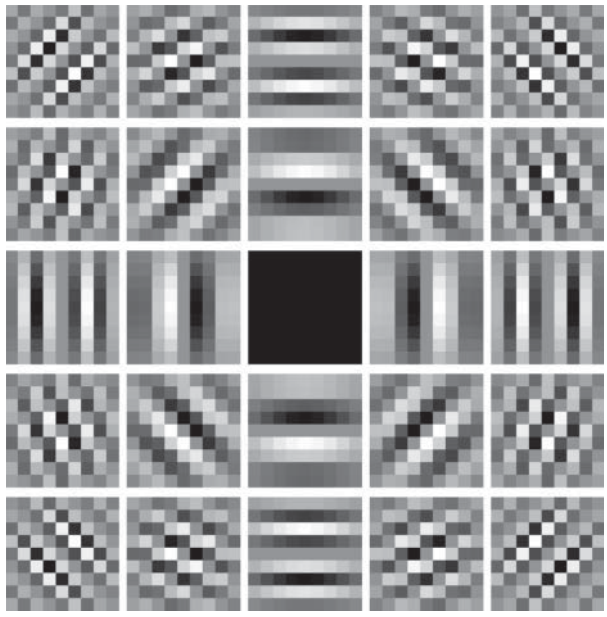


Figure 4.80 Imaginary part of the 25 filters designed. Each filter is of size 9×9 and it was designed to cover and area of size 51×51 in the Fourier domain. Source: Maria Petrou.

Example 4.68

Use the filters you constructed in Example 4.67 to construct feature maps for the image of Figure 4.81.

The outputs of the filters are shown in Figures 4.82–4.87. Figures 4.82 and 4.87 show the features constructed by either taking the magnitude of the complex filter outputs or by using the real filters directly. The empty panels in Figure 4.82 are put there to emphasise that filters tuned to those bands would have produced the same features as those already presented. We may see that vertical and horizontal edges and lines are emphasised as we move along the axial directions starting from the central panel, with the amplitude of the detected ripples becoming finer and finer as we move towards higher frequencies (see the details shown in Figure 4.88).

Figure 4.81 The Girona pavement in size 255×255. We wish to construct features that might be used for its segmentation. Source: Maria Petrou.

(Continued)

Example 4.68 (Continued)

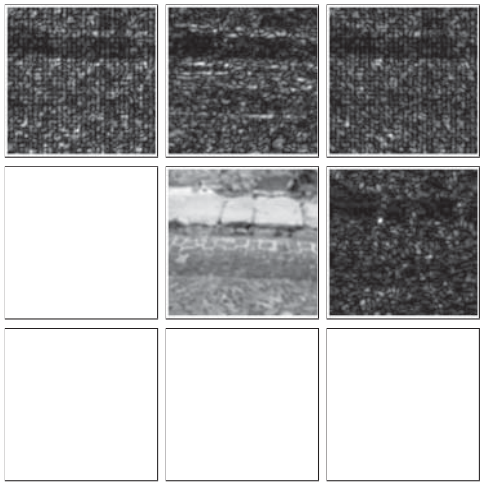


Figure 4.82 Magnitude of the outputs of the nine-filter bank. Source: Maria Petrou.

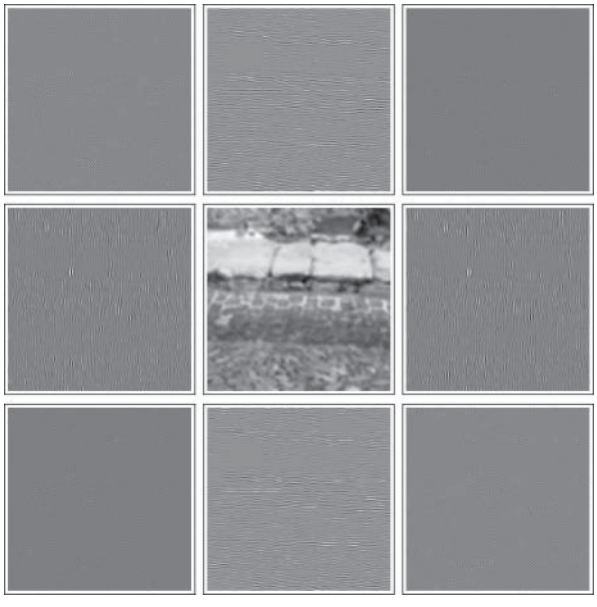


Figure 4.83 Real part of the outputs of the nine-filter bank. Source: Maria Petrou.

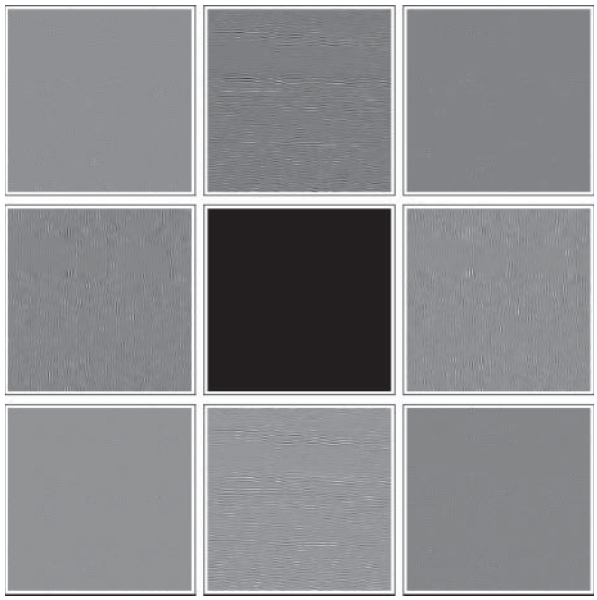


Figure 4.84 Imaginary part of the outputs of the nine-filter bank. Source: Maria Petrou.

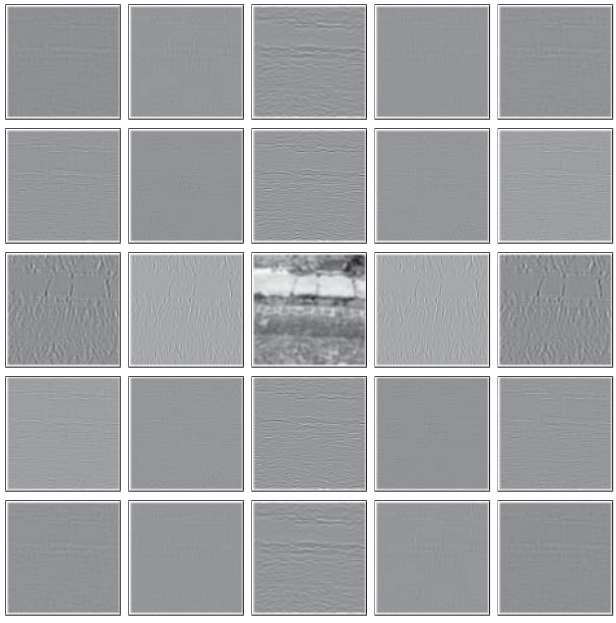


Figure 4.85 Real part of the outputs of the 25-filter bank. Source: Maria Petrou.

(Continued)

Example 4.68 (Continued)

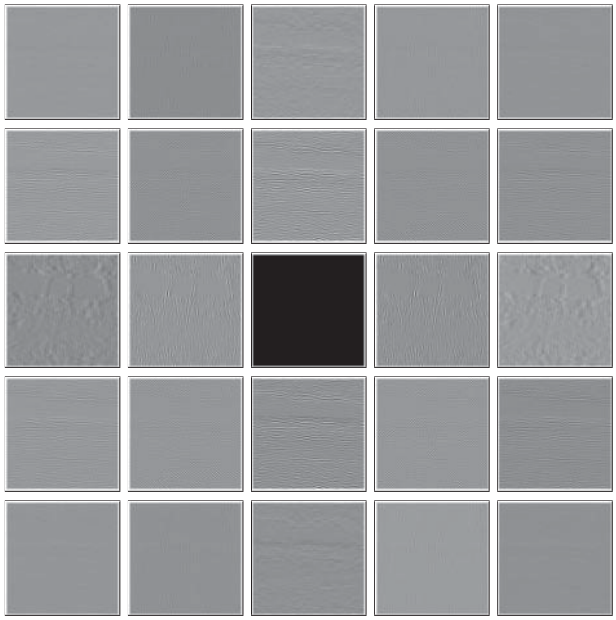


Figure 4.86 Imaginary part of the outputs of the 25-filter bank. Source: Maria Petrou.

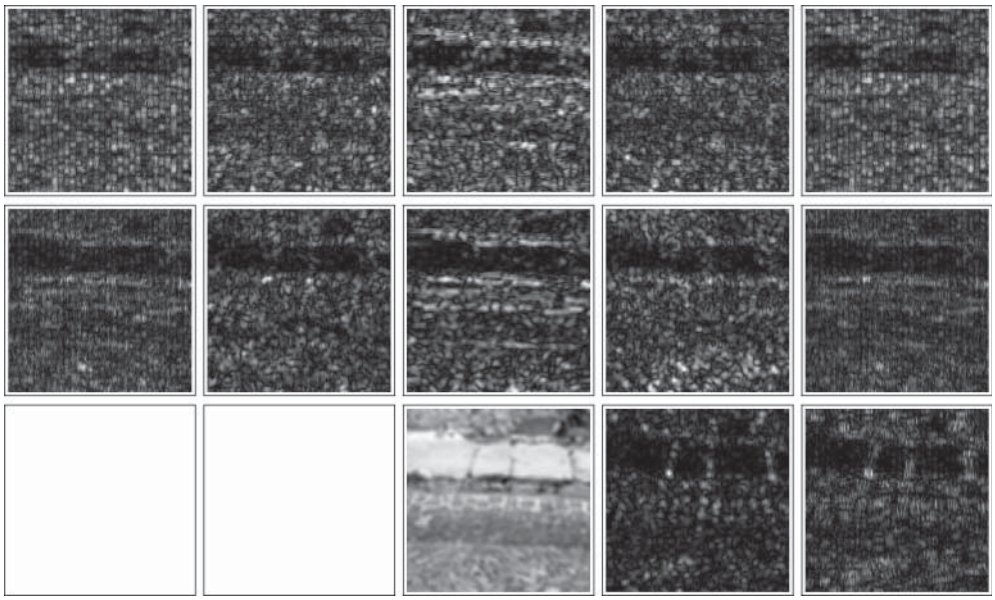


Figure 4.87 Magnitude of the outputs of the 25-filter bank. The two empty panels and the ten missing ones are omitted because they are identical to those shown. Source: Maria Petrou.

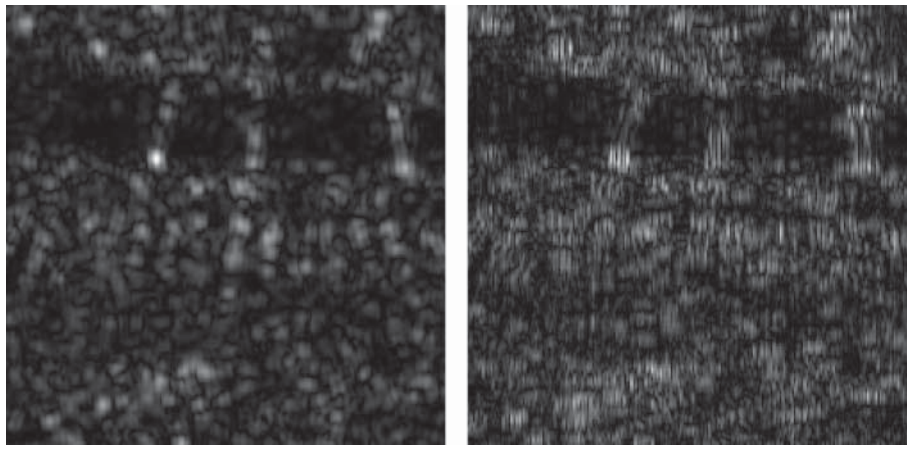


Figure 4.88 The last two panels of Figure 4.87. They correspond to low frequencies along the vertical axis and moderate and high frequencies, from left to right respectively, along the horizontal axis. Note how finer vertical details are picked as we move towards the high frequencies. Source: Maria Petrou.

4.4 Local Phase Features

Can we use the phase of the DFT to compute from it texture features?

Yes, but not in a straightforward way. The problem is that the phase of the signal tends to be wrapped to the range $[0, 2\pi]$ and it does not take continuous values. Phase unwrapping is an option, but algorithms that use phase unwrapping are not very robust (see Figure 4.139). So, phase tends to be estimated in an indirect way, by projecting the signal onto an orthonormal basis made up from two filters that constitute a **Hilbert transform** pair. Phase then is estimated from the ratio of these projections.

What is a Hilbert transform pair?

Two functions $f(x)$ and $g(x)$, with Fourier transforms $\hat{f}(\omega)$ and $\hat{g}(\omega)$, respectively, constitute a Hilbert transform pair, if both have 0 dc component and $\hat{g}(\omega) = \text{sign}(\omega)\text{j}\hat{f}(\omega)$. In other words, the Fourier transform of one is equal to the Fourier transform of the other times j, with a positive sign for positive frequencies and with a negative sign for negative frequencies. This is shown schematically in Figure 4.89.

Why may two filters that constitute a Hilbert transform pair be used for local phase estimation?

To answer this question we must first understand what the physical meaning of the local phase of a signal is. A purely symmetric function has real Fourier transform, so all its harmonics have phase 0°. A purely antisymmetric function has imaginary Fourier transform, so all its harmonics have phase 90°. A function that is neither symmetric nor antisymmetric will have harmonics with phases in between these two values. So, the phase of the Fourier transform of a function is related to the degree of symmetry of the function.

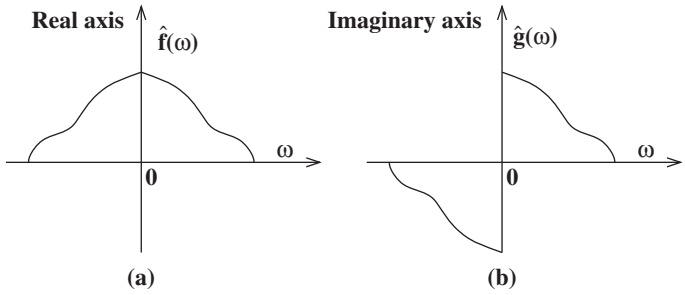


Figure 4.89 (a) The Fourier transform of a symmetric band limited filter. (b) The Fourier transform of its Hilbert transform pair.

If we project a function on an orthogonal basis, made up from a symmetric and an antisymmetric function, from the ratio of the two projections we can tell how symmetric or antisymmetric the function is. This statement may be rephrased as: if we project a function on an orthogonal basis made up from a function with purely real Fourier transform and a function with purely imaginary Fourier transform, from the ratio of the two projections we may assign a nominal phase to the function. We may use a Hilbert transform pair of functions to construct the basis we need: if one of the functions is symmetric, it will have a purely real Fourier transform, and its Hilbert pair will be antisymmetric (see Figure 4.89), as it will have a purely imaginary Fourier transform. Indeed if we multiply two such functions point by point and add the results, we shall get 0, either we perform the multiplication in the signal or in the frequency domain. So, the two functions constitute an orthogonal basis. If further we normalise the first function so the sum of the squares of its values is 1, this will be the case for the sum of the squares of the values of the other function too, and thus the basis will be orthonormal.

What if the harmonics of a non-symmetric and non-antisymmetric function all have different phases?

It is true that the harmonics of a function that is neither symmetric nor antisymmetric will all have different phases. The phase computed from the projection of the signal on the two basis signals is only a nominal phase that is meaningful when the phases of all harmonics tend to agree. When this happens, we say that we have **phase congruency** (see Book I [75]). Phase congruency really means that there is a specific structural characteristic of the signal at that point, that may be a line, a stripe, an edge, or something similar. If phase congruence is low, the calculation of phase information is meaningless, as the signal is almost flat and amorphous at that locality. So, before we pay attention to the local phase, we examine first the local energy content of the signal. This can be computed very easily: we simply square and add the two projections to have the local energy content. Once this is above a certain threshold, we realise that the signal has a structural feature there, and we can calculate the local phase to characterise the feature. This is shown schematically in Figure 4.90.

How can we construct two filters so they constitute a Hilbert transform pair?

We can always select the symmetric filter and then from it construct its Hilbert transform pair by simply taking its Fourier transform, multiplying it with sign(ω)j and taking the inverse Fourier transform (see Book I [75]). However, the key idea is that we would like the filter to be at the same time frequency selective. In other words, we would like our filter to:

i) Return local information

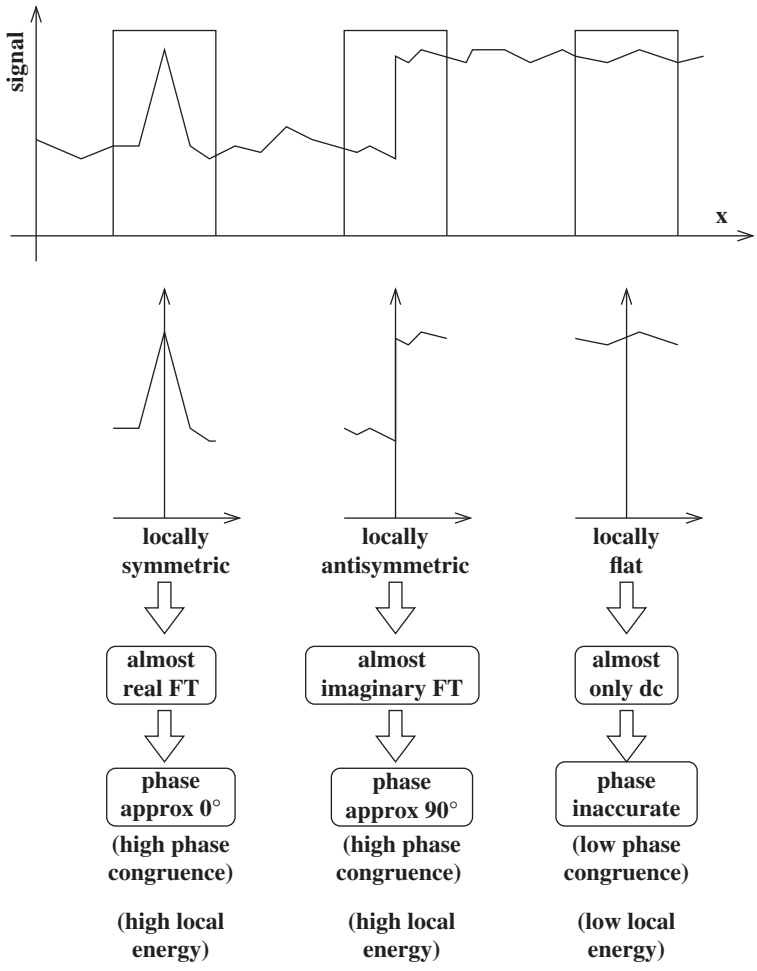


Figure 4.90 A local window may isolate part of a signal. If the isolated segment has high energy, it represents some structural feature of the signal. This feature may be treated like a function that is approximately symmetric or antisymmetric. To decide whether it is approximately symmetric or antisymmetric, we may examine the phase of the isolated signal. As the phase is different for the different harmonics of the isolated signal, we project the isolated signal onto two basis functions, one symmetric and one antisymmetric and work out an "effective" phase, from the ratio of the two projections.

ii) Return information that concerns the detection of signal characteristics in a pre-selected frequency band
iii) Allow the detection of significant local features by allowing the calculation of the local energy
iv) Allow us to decide whether the feature manifested by a high level of local energy is locally symmetric or antisymmetric, through the estimation of the local signal phase.

All the above requirements are fulfilled if we use the

i) **Monogenic signal** computed with the help of the **Riesz transform**, which is discussed in Book I [75]; or
ii) Use the prolate spheroidal sequence filters. From the monogenic signal one may compute the local phase, local orientation and local energy of the image. The prolate spheroidal sequence filters are oriented filters, from which local energy and phase may be estimated for each selected band and orientation.

How do we compute the Riesz transform of an image?

Step 0: Call the image you wish to analyse $f(x,y)$.

Step 1: Select a filter that is defined in the frequency domain, so that it has non-zero response in a selected band, and it is even with respect to the frequencies ω_x and ω_y along the x and y axes, respectively, of the image. Call this filter $H(\omega_x,\omega_y)$. An example such filter is

$$H(\omega_x,\omega_y)=\log\left(\frac{\cos^2(\omega_x-\omega_{0x})}{\cos^2\omega_{1x}}\right)\log\left(\frac{\cos^2(\omega_y-\omega_{0y})}{\cos^2\omega_{1y}}\right) \quad \text{for } |\omega_x-\omega_{0x}|<\omega_{1x} \text{ and } |\omega_y-\omega_{0y}|<\omega_{1y} \tag{4.334}$$

which is non-zero in the pair of positive and negative frequency bands defined for frequencies in the ranges $[\omega_{0x}-\omega_{1x},\omega_{0x}+\omega_{1x}]$ and $\omega_{0y}-\omega_{1y},\omega_{0y}+\omega_{1y}]$.

Step 2: Define functions:

$$H_1(\omega_x,\omega_y)\equiv \mathrm{j}\frac{\omega_x}{\sqrt{\omega_x^2+\omega_y^2}} \qquad H_2(\omega_x,\omega_y)\equiv \mathrm{j}\frac{\omega_y}{\sqrt{\omega_x^2+\omega_y^2}}. \tag{4.335}$$

Step 3: Take the DFT of the image, $\hat{f}(\omega_x,\omega_y)$.

Step 4: Multiply the DFT of the image with the DFT of filter $H(\omega_x,\omega_y)$ point by point, and take the inverse DFT of the product, to produce $f_H(x,y)$.

Step 5: Take the inverse DFT of $\hat{f}(\omega_x,\omega_y)H(\omega_x,\omega_y)H_1(\omega_x,\omega_y)$ to produce $f_{H_1}(x,y)$.

Step 6: Take the inverse DFT of $\hat{f}(\omega_x,\omega_y)H(\omega_x,\omega_y)H_2(\omega_x,\omega_y)$ to produce $f_{H_2}(x,y)$.

The output $(f_H(x,y),f_{H_1}(x,y),f_{H_2}(x,y))$ is the monogenic signal of the image.

How can the monogenic signal be used?

The monogenic signal $(f_H(x,y),f_{H_1}(x,y),f_{H_2}(x,y))$, at a pixel position (x,y), may be used to compute the local image energy as:

$$E(x,y)\equiv\sqrt{f_H(x,y)^2+f_{H_1}(x,y)^2+f_{H_2}(x,y)^2}. \tag{4.336}$$

Local maxima of this quantity identify the locations of image features.

The local feature symmetry (i.e. how symmetric or antisymmetric the feature is) may be measured as:

$$\Phi(x,y)\equiv\left|\tan^{-1}\frac{f_H(x,y)}{\sqrt{f_{H_1}(x,y)^2+f_{H_2}(x,y)^2}}\right|. \tag{4.337}$$

Here we assume that function $\tan^{-1}$ yields values in the range $[-\pi/2,\pi/2]$. If the feature is purely symmetric, the numerator of this fraction is maximal and the denominator is minimal. So the calculated angle will tend to be either close to $90°$ or to $-90°$. If we do not want to distinguish the two, we take the absolute value of the result. If the feature is mostly antisymmetric, the numerator approaches 0 while the denominator is large, and $\Phi(x,y)$ is close to 0. So, $\Phi(x,y)$ is a measure of local symmetry, taking values in the range $[0,\pi/2]$, with higher values indicating higher symmetry.

The local feature orientation may be computed as:

$$\Theta(x,y)\equiv\tan^{-1}\frac{f_{H_2}(x,y)}{f_{H_1}(x,y)}+\delta\left(1+\text{sign}\left[\tan^{-1}\frac{f_{H_2}(x,y)}{f_{H_1}(x,y)}\right]\right)\pi. \tag{4.338}$$

This number varies between 0 and π, as factor π is added only to negative angles computed by $\tan^{-1}$, to put them in the range $[\pi/2, \pi]$.

How can we select the even filter we use in the Riesz transform?

To make sure that we capture features that may manifest themselves in different frequency bands, we tessellate the frequency space of the image in a systematic way, like we do when we use Gabor functions or prolate spheroidal sequence functions. We note that the filters used by these two methods are symmetric functions of the two frequencies. So, we may use any of the two approaches in combination with the Riesz transform to produce for each band not just the local energy, like we usually do, but also the local orientation and a measure of the local symmetry.

Can we tessellate the frequency space using polar coordinates like we did for the Gabor functions?

There is no point in doing that. The use of polar coordinates in the frequency domain in conjunction with Gabor functions has as purpose to create filters along different orientations, so that we can detect image structure along the selected orientations. The Riesz transform allows us to work out the orientation of each feature from the outputs of the filters that are in terms of the Cartesian coordinates in the frequency domain. So, an appropriate tessellation of the frequency domain is like the one shown in figure 4.24 or in Figure 4.27.

Example 4.69

Apply the Riesz transform to the 15×15 image of Figure 4.71 with the frequency domain tessellated according to Figure 4.66. Use a Gaussian function as the even filter.

First we shall define the Gaussian filter. Let us call the indices in the frequency domain (k, l). Each takes values in the range $[-7, 7]$. The centres of the Gaussian filters associated with each band are at coordinates $(k_0, l_0) \in \{(0,0), (5,0), (-5,0), (0,5), (0,-5), (5,5), (5,-5), (-5,5), (-5,-5)\}$ (see Figure 4.66, assuming that the $(0,0)$ point is in the middle of the grid). The Gaussian filter then is defined as

$$H(k,l;k_0,l_0) = \exp\left\{-\frac{\left(\frac{2\pi(k-k_0)}{15}\right)^2 + \left(\frac{2\pi(l-l_0)}{15}\right)^2}{2\Sigma^2}\right\} + \exp\left\{-\frac{\left(\frac{2\pi(k+k_0)}{15}\right)^2 + \left(\frac{2\pi(l+l_0)}{15}\right)^2}{2\Sigma^2}\right\} \text{ for } (k_0,l_0) \neq (0,0). \quad (4.339)$$

The two terms ensure that we place one Gaussian in each of the paired bands symmetrically positioned about the centre of the axes, so that the inverse DFT of H is real. For $(k_0, l_0) = (0,0)$ the second term is not needed, so we define:

$$H(k,l;0,0) = \exp\left\{-\frac{\left(\frac{2\pi k}{15}\right)^2 + \left(\frac{2\pi l}{15}\right)^2}{2\Sigma^2}\right\}. \quad (4.340)$$

(Continued)

Example 4.69 (Continued)

Let us select Σ so that at point $k = k_0 + 2$ and $l = l_0$, the value of the filter is 0.5:

$$0.5 = \exp\left\{-\frac{\left(\frac{2\pi 2}{15}\right)^2}{2\Sigma^2}\right\} \Rightarrow \ln 0.5 = -\frac{\frac{16\pi^2}{225}}{2\Sigma^2} \Rightarrow 2\Sigma^2 = \frac{\frac{16\pi^2}{225}}{\ln 2} \Rightarrow 2\Sigma^2 = 1.01254. \tag{4.341}$$

For each filter $H(k,l;k_0,l_0)$ we must calculate the corresponding antisymmetric filters

$$\begin{aligned}\tilde{H}_1(k,l;k_0,l_0) &= \mathrm{j}\frac{\frac{2\pi k}{15}}{\sqrt{\left(\frac{2\pi k}{15}\right)^2+\left(\frac{2\pi l}{15}\right)^2}}H(k,l;k_0,l_0) = \mathrm{j}\frac{k}{\sqrt{k^2+l^2}}H(k,l;k_0,l_0)\\ \tilde{H}_2(k,l;k_0,l_0) &= \mathrm{j}\frac{\frac{2\pi l}{15}}{\sqrt{\left(\frac{2\pi k}{15}\right)^2+\left(\frac{2\pi l}{15}\right)^2}}H(k,l;k_0,l_0) = \mathrm{j}\frac{l}{\sqrt{k^2+l^2}}H(k,l;k_0,l_0).\end{aligned} \tag{4.342}$$

Next, we take the DFT of the image and make sure that its dc component is at index $(k,l) = (0,0)$, i.e. that indices (k,l) take values in the range $[-7,7]$. Let us call it $\hat{I}(k,l)$.

Then for every set of filters $(H(k,l;k_0,l_0), \tilde{H}_1(k,l;k_0,l_0), \tilde{H}_2(k,l;k_0,l_0))$, we take the point by point products:

$$\begin{aligned}\hat{I}_{ek_0l_0}(k,l) &= H(k,l;k_0,l_0)\hat{I}(k,l)\\ \hat{I}_{o_1k_0l_0}(k,l) &= \tilde{H}_1(k,l;k_0,l_0)\hat{I}(k,l)\\ \hat{I}_{o_2k_0l_0}(k,l) &= \tilde{H}_2(k,l;k_0,l_0)\hat{I}(k,l).\end{aligned} \tag{4.343}$$

Finally, we take the inverse DFT of $\hat{I}_{ek_0l_0}$, $\hat{I}_{o_1k_0l_0}$ and $\hat{I}_{o_2k_0l_0}$, to produce $I_{ek_0l_0}$, $I_{o_1k_0l_0}$ and $I_{o_2k_0l_0}$, respectively. The triplet $(I_{ek_0l_0}, I_{o_1k_0l_0}, I_{o_2k_0l_0})$ is the monogenic signal of the original image.

Example 4.70

Use the monogenic signal computed in Example 4.69, to produce the local energy, local symmetry and local orientation of the original image.

We have five frequency bands, and for each band we shall have to compute the local energy as:

$$E(k,l;k_0,l_0) = \sqrt{I_{ek_0l_0}(k,l)^2 + I_{o_1k_0l_0}(k,l)^2 + I_{o_2k_0l_0}(k,l)^2}. \tag{4.344}$$

For each band we may also compute the local orientation using:

$$\Theta(k,l;k_0,l_0) = \tan^{-1}\frac{I_{o_2k_0l_0}(k,l)}{I_{o_1k_0l_0}(k,l)} + \delta\left(1+\mathrm{sign}\left[\tan^{-1}\frac{I_{o_2k_0l_0}(k,l)}{I_{o_1k_0l_0}(k,l)}\right]\right)\pi. \tag{4.345}$$

Here function tan *is assumed to produce values in the range $[-\pi/2, \pi/2]$, and the delta function is used so that a π is added when the angle computed by function* tan *is negative, in order to make the range of orientations computed to be $[0,\pi]$. For visualisation purposes, this range is scaled to the range* [0,255]*, for each panel separately.*

Finally, we may compute the local symmetry of the image using:

$$\Phi(k,l;k_0,l_0)=\left|\tan^{-1}\frac{I_{ek_0l_0}(k,l)}{\sqrt{I_{o_1k_0l_0}(k,l)^2+I_{o_2k_0l_0}(k,l)^2}}\right|. \tag{4.346}$$

Function tan *is assumed to produce values in the range* $[-\pi/2,\pi/2]$. *If the detected feature is symmetric,* $I_{ek_0l_0}(k,l) >> \max\{I_{o_1k_0l_0}(k,l), I_{o_2k_0l_0}(k,l)\}$ *and the calculated angle will be near* $-\pi/2$ *or* $\pi/2$, *depending on whether it represents a darker or brighter than its surroundings stripe. We use the absolute value so we do not distinguish between the two polarities. If the local feature is antisymmetric,* $I_{ek_0l_0}(k,l) << \max\{I_{o_1k_0l_0}(k,l), I_{o_2k_0l_0}(k,l)\}$ *and the calculated angle will be near 0. So, the range of the local symmetry measure is* $[0,\pi/2]$.

Can we use prolate spheroidal sequence functions instead of the monogenic signal to extract local symmetry and local orientation?

Yes. Let us consider the filter we construct for a particular band. It is complex (see the section How do we perform the analysis of an image using prolate spheroidal sequence functions in practice?). We saw that if we take only its real part, we construct a filter that has response in a paired set of bands, symmetrically placed about the dc component. It turns out that the imaginary part is the Hilbert transform pair of the real part of such a filter, and so if we use the real and the imaginary parts as a pair of filters, they will provide us local information for the selected frequency band and the option to compute from their outputs local energy and local phase.

Example 4.71

Examine whether the real and the imaginary parts of the filter constructed in Example 4.46 constitute a Hilbert transform pair.

The real and the imaginary parts of this filter are: $w_R(n)=(0.161, 0.028, 0.260, -0.316, 0.682, 2.202, 0.682, -0.316, 0.260, 0.028, 0.161)$, $w_{\rm I}(n)=(0.103, -0.196, -0.076, -0.365, -1.493, 0, 1.493, 0.365, 0.076, 0.196, -0.103)$. *We note that the elements of* $w_R(n)$ *do not sum up to 0, so it does not have 0 dc component. So, first we remove the average value from all elements of* $w_R(n)$. *The sum of its elements is* $\sum_n w_R(n)=3.832$ *and this divided with 11 yields 0.348. After removing this value from all elements of* $w_R(n)$ *we obtain:* $w'_{\rm R}(n)=(-0.187, -0.32, -0.088, -0.664, 0.334, 1.854, 0.334, -0.664, -0.088, -0.32, -0.187)$.

To show that $w'_{\rm R}(n)$ *and* $w_{\rm I}(n)$ *constitute a Hilbert transform pair, we shall compute their Fourier transforms* $\hat{w}'_{\rm R}(k)$ *and* $\hat{w}_{\rm I}(k)$ *and show that* $\hat{w}_{\rm I}(k)=\text{sign}(k)\text{j}\hat{w}'_{\rm R}(k)$.

$$\begin{aligned}\hat{w}'_{\rm R}(k)&=\sum_{n=-5}^{5}w'_{\rm R}(n)e^{-\text{j}\frac{2\pi}{11}nk}\\&=-0.187e^{\text{j}\frac{2\pi}{11}5k}-0.32e^{\text{j}\frac{2\pi}{11}4k}-0.088e^{\text{j}\frac{2\pi}{11}3k}-0.664e^{\text{j}\frac{2\pi}{11}2k}+0.334e^{\text{j}\frac{2\pi}{11}k}\\&\quad+1.854+0.334e^{-\text{j}\frac{2\pi}{11}k}-0.664e^{-\text{j}\frac{2\pi}{11}2k}-0.088e^{-\text{j}\frac{2\pi}{11}3k}-0.32e^{-\text{j}\frac{2\pi}{11}4k}\\&\quad-0.187e^{-\text{j}\frac{2\pi}{11}5k}\end{aligned}$$

(Continued)

Example 4.71 (Continued)

$$\begin{aligned}
&= -0.187 \times 2\cos\frac{2\pi 5k}{11} - 0.32 \times 2\cos\frac{2\pi 4k}{11} - 0.088 \times 2\cos\frac{2\pi 3k}{11} \\
&\quad -0.664 \times 2\cos\frac{2\pi 2k}{11} + 0.334 \times 2\cos\frac{2\pi k}{11} + 1.854 \\
&= -0.374\cos\frac{10\pi k}{11} - 0.64\cos\frac{8\pi k}{11} - 0.176\cos\frac{6\pi k}{11} - 1.328\cos\frac{4\pi k}{11} \\
&\quad +0.668\cos\frac{2\pi k}{11} + 1.854
\end{aligned} \tag{4.347}$$

$$\begin{aligned}
\hat{w}_{\mathrm{I}}(k) &= \sum_{n=-5}^{5} w_{\mathrm{I}}(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{11}nk} \\
&= 0.103\mathrm{e}^{\mathrm{j}\frac{2\pi}{11}5k} - 0.196\mathrm{e}^{\mathrm{j}\frac{2\pi}{11}4k} - 0.076\mathrm{e}^{\mathrm{j}\frac{2\pi}{11}3k} - 0.365\mathrm{e}^{\mathrm{j}\frac{2\pi}{11}2k} - 1.493\mathrm{e}^{\mathrm{j}\frac{2\pi}{11}k} \\
&\quad +1.493\mathrm{e}^{-\mathrm{j}\frac{2\pi}{11}k} + 0.365\mathrm{e}^{-\mathrm{j}\frac{2\pi}{11}2k} + 0.076\mathrm{e}^{-\mathrm{j}\frac{2\pi}{11}3k} + 0.196\mathrm{e}^{-\mathrm{j}\frac{2\pi}{11}4k} - 0.103\mathrm{e}^{-\mathrm{j}\frac{2\pi}{11}5k} \\
&= 0.103 \times 2\mathrm{j}\sin\frac{2\pi 5k}{11} - 0.196 \times 2\mathrm{j}\sin\frac{2\pi 4k}{11} - 0.076 \times 2\mathrm{j}\sin\frac{2\pi 3k}{11} \\
&\quad -0.365 \times 2\mathrm{j}\sin\frac{2\pi 2k}{11} - 1.493 \times 2\mathrm{j}\sin\frac{2\pi k}{11} \\
&= 0.206\mathrm{j}\sin\frac{10\pi k}{11} - 0.392\mathrm{j}\sin\frac{8\pi k}{11} - 0.152\mathrm{j}\sin\frac{6\pi k}{11} - 0.730\mathrm{j}\sin\frac{4\pi k}{11} \\
&\quad -2.986\mathrm{j}\sin\frac{2\pi k}{11}.
\end{aligned} \tag{4.348}$$

Allowing k then to take values in the range $[-5, 5]$*, we obtain:*

$$\begin{aligned}
\hat{w}_{\mathrm{R}}(k) &= (0.001, 1.914, 2.668, 2.948, 2.667, 0.000, -2.667, -2.948, -2.668, -1.914, -0.001) \\
\hat{w}_{\mathrm{I}}(k) &= \mathrm{j}(-1.856, 0.062, 0.813, 1.092, 0.813, -1.850, 0.813, 1.092, 0.813, 0.062, -1.856)
\end{aligned} \tag{4.349}$$

We can see that the two functions constitute a Hilbert transform pair.

Example 4.72

Show how you may use the Hilbert pair filters $w'_{\mathrm{R}}(n)$ and $w_{\mathrm{I}}(n)$ of Example 4.71 to estimate the local energy and phase of an 11-sample long signal.

Filters $w'_{\mathrm{R}}(n)$ and $w_{\mathrm{I}}(n)$ are convolution filters. In order to be able to use them to estimate local energy and phase, each should have energy equal to 1. We observe that:

$$\begin{aligned}
\sum_n w'_{\mathrm{R}}(n)^2 &= 1.854^2 + 2(0.334^2 + 0.664^2 + 0.088^2 + 0.32^2 + 0.187^2) = 4.832 \\
\sum_n w_{\mathrm{I}}(n)^2 &= 2(1.493^2 + 0.365^2 + 0.076^2 + 0.196^2 + 0.103^2) = 4.834.
\end{aligned} \tag{4.350}$$

So, within the accuracy of the calculation, the two filters have the same square magnitude, i.e. the same energy. This was expected, since their Fourier transforms differ by a sign only and by a factor j*, i.e. they differ only in the phase, so Parseval's theorem guarantees that the sum of the squares of their values in the signal domain will be the same.*

So, to create an orthonormal basis out of these two functions, we divide the elements of both of them by $\sqrt{4.833} = 2.198$. *We obtain:* $\tilde{w}_R(n) = (-0.085, -0.146, -0.04, -0.302, 0.152, 0.843, 0.152, -0.302, -0.04, -0.146, -0.085)$ $\tilde{w}_I(n) = (-0.047, 0.089, 0.035, 0.166, 0.679, 0, -0.679, -0.166, -0.035, -0.089, 0.047)$

Now if we have an 11-sample long signal, we shall have to convolve it with each one of these filters. The ratio of the two outputs at each sample will be the tangent of the local signal phase.

Example 4.73

Construct three-sample long filters that will allow you to estimate local signal phase starting from the prolate spheroidal filter of Example 4.46.

The real and the imaginary parts of the truncated original convolution filters are:

$w_R(n) = (0.682, 2.202, 0.682)$

$w_I(n) = (-1.493, 0, 1.493)$.

The dc component of the first filter is 1.189. Subtracting it from all elements of $w_R(n)$ *yields:* $w'_R(n) = (-0.507, 1.013, -0.507)$.

The sum of the squares of these values is 1.540, while the sum of the squares of $w_I(n)$ *is 4.458. To be able to normalise both filters together, we select the values of* $w_I(n)$ *so their squares sum up to 1.540. Indeed, if we set* $\tilde{w}_I(n) = (-0.878, 0, 0.878)$ *both filters will have the squares of their elements sum up to 1.540. So, if we divide both of them by* $\sqrt{1.540} = 1.241$, *we obtain:*

$F_R(n) = (-0.408, 0.816, -0.408)$

$F_I(n) = (-0.707, 0, 0.707)$.

The DFTs of these two filters are:

$$\hat{f}_R(k) = -0.408 \times 2\cos\frac{2\pi k}{3} + 0.816 = (1.225, 0, 1.225)$$

$$\hat{f}_I(k) = -0.707 \times 2\sin\frac{2\pi k}{3} = \mathrm{j}(1.225, 0, -1.225). \tag{4.351}$$

So, these two filters constitute a Hilbert pair and an orthonormal basis for the symmetry/antisymmetry space. To demonstrate how we can use them, let us analyse the signal:

$s(n) = (1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 2, 3, 3, 3, -2, 3, 3, 3, 3)$, *for* $n = 1, \ldots, 19$.

This signal has a symmetric feature at position $n = 5$, *an antisymmetric one at* $n = 11$ *and another symmetric at* $n = 15$. *Its convolution with the two filters yields:*

$(., 0, 0, -0.816, 1.632, -0.816, 0, 0, 0, -0.408, 0, 0.408, 0, 2.04, -4.08, 2.04, 0, 0, .)$

$(., 0, 0, -1.414, 0, 1.414, 0, 0, 0, -0.707, -1.414, -0.707, 0, 3.535, 0, -3.535, 0, 0, .)$.

Note that we used a dot to represent the border samples for which there is no output.

The local energy at each sample is given by the sum of the squares of the two outputs:

$(., 0, 0, 2.666, 2.663, 2.666, 0, 0, 0, 0.666, 1.999, 0.666, 0, 16.658, 16.646, 16.658, 0, 0, .)$.

At the places where the local energy is non-zero, we compute the local phase, as the angle of the inverse tangent of the ratio of the second output over the first. An angle close to $90°$ *indicates local antisymmetry, while an angle close to* $0°$ *indicates local symmetry:*

$(., ., ., 60°, 0°, -60°, ., ., ., 60°, -90°, -60^o, ., 60°, 0°, -60°, ., ., .)$.

(Continued)

Example 4.73 (Continued)

We note that there are clear symmetric features indicated at positions $n = 5$ and $n = 15$, where the phase is 0, and an antisymmetric feature is indicated at $n = 11$, where the phase is -90^o.

Example B4.74

Show that the filters constructed by the prolate spheroidal sequence functions have their real part symmetric and their imaginary part antisymmetric.

The Fourier transform of the constructed filter $W(k)$ is given by Equation (4.210). In this equation $\psi(k)$ is the eigenvector of a real and symmetric matrix. So, $\psi(k)$ is real. The factor $n_1 + n_2$ in the complex exponential comes from the centre of the band to which the filter we are constructing refers. When the band is selected to contain the centre of the signal axis, i.e. when $n_1 = -n_2$, the complex exponential is 0 and $W(k) = \psi(k)$, i.e. it is real. Since the filter we are constructing is a convolution filter, it does not really matter how we choose n_1 and n_2. So, we might as well stick to $n_1 + n_2 = 0$ for convenience. We shall show now that filter $w(n)$ obtained by taking the inverse DFT of $W(k)$ is the complex conjugate of $w(-n)$. Let us say that the function has non-zero response in the band with indices in the range $[k_1, k_2]$. We may write:

$$w(n) = \frac{1}{N}\sum_{k=k_1}^{k_2} W(k)e^{j\frac{2\pi}{N}kn}. \tag{4.352}$$

Then:

$$w(-n) = \frac{1}{N}\sum_{k=k_1}^{k_2} W(k)e^{-j\frac{2\pi}{N}kn}. \tag{4.353}$$

Since $W(k)$ is real, obviously $w(n) = w^*(-n)$. Let us write $w(n)$ explicitly in terms of its real and imaginary parts: $w(n) \equiv a(n) + jb(n)$. We shall have:

$$a(n) + jb(n) = a(-n) - jb(-n) \Rightarrow a(n) = a(-n) \text{and } b(n) = -b(-n). \tag{4.354}$$

So, the convolution filters constructed by the prolate spheroidal sequence functions method are:
(i) Complex
(ii) The real part is symmetric about its centre
(iii) The imaginary part is antisymmetric about its centre.

Example B4.75

Show that the real and the imaginary parts of the filters constructed from the prolate spheroidal sequence functions constitute a Hilbert transform pair.

From the way we select the bands, and since the filter we construct is complex, we may assume that it has non-zero response either to only positive or to only negative frequencies. Let us assume for simplicity that it has non-zero response for positive frequencies only. Let us assume that the filter may be written as $w(n) = a(n) + jb(n)$.

In Example 4.74 it was shown that $a(n)$ is symmetric and $b(n)$ is antisymmetric. This means that the DFT $\hat{a}(k)$ of $a(n)$ is real and the DFT $\hat{b}(k)$ of $b(n)$ is imaginary. Let us call it $\mathrm{j}\tilde{b}(k)$, where $\tilde{b}(k)$ is real. The DFT of $w(n)$ obviously must be $W(k) = \hat{a}(k) + \mathrm{j}\hat{b}(k) = \hat{a}(k) - \tilde{b}(k)$

We know that $W(k) = 0$ for $k \leq 0$. So, for negative frequencies, $\hat{a}(k) = \tilde{b}(k) = -\mathrm{j}\hat{b}(k)$.

Now let us consider the complex conjugate of $w(n)$, $w^*(n) = a(n) - \mathrm{j}b(n)$. We know from Example 4.50 that two functions that are complex conjugate of each other have their DFTs in bands that are symmetric about the origin of the frequency axes. In other words, the DFT of $w^*(n)$ will be non-zero in the corresponding negative frequencies band of that in which $W(k)$ is non-zero. The DFT of $w^*(n)$ will be $\hat{a}(k) - \mathrm{j}\hat{b}(k)$ and for $k > 0$ it will be 0. In other words, $\hat{a}(k) = \mathrm{j}\hat{b}(k)$ for $k > 0$. Combining the two results, we may write $\hat{a}(k) = \mathrm{sign}(k)\mathrm{j}\hat{b}(k)$, i.e. that the real and the imaginary parts of filter $w(n)$ constitute a Hilbert transform pair.

Example B4.76

Show that the 2D DFT of the $(2N+1) \times (2N+1)$ matrix created by taking the vector outer product of two vectors of length $2N+1$ is equal to the product of the 1D DFTs of the individual vectors.

Consider vectors **a** *and* **b**, *which will be treated as signals with $2N+1$ elements each:*
$(a(-N), a(-N+1), \ldots, a(0), \ldots, a(N-1), a(N))$
$(b(-N), b(-N+1), \ldots, b(0), \ldots, b(N-1), b(N))$
Their DFTs are given by:

$$\hat{a}(k) = \sum_{n=-N}^{N} a(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{2N+1}} \qquad \hat{b}(l) = \sum_{n=-N}^{N} b(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi ln}{2N+1}}. \tag{4.355}$$

Their vector outer product is:

$$A \equiv \begin{pmatrix} a(-N)b(-N) & \cdots & a(-N)b(0) & \cdots & a(-N)b(N) \\ a(-N+1)b(-N) & \cdots & a(-N+1)b(0) & \cdots & a(-N+1)b(N) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a(0)b(-N) & \cdots & a(0)b(0) & \cdots & a(0)b(N) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a(N-1)b(-N) & \cdots & a(N-1)b(0) & \cdots & a(N-1)b(N) \\ a(N)b(-N) & \cdots & a(N)b(0) & \cdots & a(N)b(N) \end{pmatrix}. \tag{4.356}$$

The DFT of this array, when treated as a 2D function, is:

$$\begin{aligned} \hat{A}(k,l) &= \sum_{n=-N}^{N}\sum_{m=-N}^{N} A(n,m)\mathrm{e}^{-\mathrm{j}\frac{2\pi(kn+lm)}{2N+1}} = \sum_{n=-N}^{N}\sum_{m=-N}^{N} a(n)b(m)\mathrm{e}^{-\mathrm{j}\frac{2\pi(kn+lm)}{2N+1}} \\ &= \sum_{n=-N}^{N} a(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi kn}{2N+1}} \sum_{m=-N}^{N} b(m)\mathrm{e}^{-\mathrm{j}\frac{2\pi lm}{2N+1}} = \hat{a}(k)\hat{b}(l). \end{aligned} \tag{4.357}$$

Example B4.77

Assume that we have two 1D complex filters, such that the real and the imaginary parts of each one constitute a Hilbert pair. We create a 2D filter by taking their vector outer product. Work out the relationship between the real and the imaginary parts of the 2D filter you create this way.

Let us call the two 1D filters $a_1(n)+\mathrm{j}b_1(n)$ and $a_2(n)+\mathrm{j}b_2(n)$. Their DFTs are $\hat{a}_1(k)+\mathrm{j}\hat{b}_1(k)$ and $\hat{a}_2(k)+\mathrm{j}\hat{b}_2(k)$, and they obey the following relation:

$$\hat{a}_1(k)=\begin{cases}-\mathrm{j}\hat{b}_1(k) & \text{for } k<0\\ \hat{b}_1(k)=0 & \text{for } k=0\\ \mathrm{j}\hat{b}_1(k) & \text{for } k>0\end{cases}\qquad \hat{a}_2(k)=\begin{cases}-\mathrm{j}\hat{b}_2(k) & \text{for } k<0\\ \hat{b}_2(k)=0 & \text{for } k=0\\ \mathrm{j}\hat{b}_2(k) & \text{for } k>0\end{cases}. \tag{4.358}$$

The filter we create by taking the vector outer product of these two filters is $s(n,m)+\mathrm{j}t(n,m)$, with:

$$s(n,m)=a_1(n)a_2(m)-b_1(n)b_2(m)\qquad t(n,m)=a_1(n)b_2(m)+b_1(n)a_2(m). \tag{4.359}$$

The DFT of this function, according to Example 4.76, is $\hat{s}(k,l)+\mathrm{j}\hat{t}(k,l)$, with:

$$\hat{s}(k,l)=\hat{a}_1(k)\hat{a}_2(l)-\hat{b}_1(k)\hat{b}_2(l)\qquad \hat{t}(k,l)=\hat{a}_1(k)\hat{b}_2(l)+\hat{b}_1(k)\hat{a}_2(l). \tag{4.360}$$

Let us use (4.358) to express $\hat{s}(k,l)$ and $\hat{t}(k,l)$ in terms of the independent functions $\hat{b}_1(k)$ and $\hat{b}_2(l)$.

$$\hat{s}(k,l)=\begin{cases}-\mathrm{j}\hat{b}_1(k)(-1)\mathrm{j}\hat{b}_2(l)-\hat{b}_1(k)\hat{b}_2(l)=-2\hat{b}_1(k)\hat{b}_2(l) & \text{for } k<0,l<0\\ -\mathrm{j}\hat{b}_1(k)\mathrm{j}\hat{b}_2(l)-\hat{b}_1(k)\hat{b}_2(l)=0 & \text{for } k<0,l>0\\ 0 & \text{for } k=0 \text{ or } l=0\\ \mathrm{j}\hat{b}_1(k)(-1)\mathrm{j}\hat{b}_2(l)-\hat{b}_1(k)\hat{b}_2(l)=0 & \text{for } k>0,l<0\\ \mathrm{j}\hat{b}_1(k)\mathrm{j}\hat{b}_2(l)-\hat{b}_1(k)\hat{b}_2(l)=-2\hat{b}_1(k)\hat{b}_2(l) & \text{for } k>0,l>0\end{cases} \tag{4.361}$$

$$\hat{t}(k,l)=\begin{cases}-\mathrm{j}\hat{b}_1(k)\hat{b}_2(l)+\hat{b}_1(k)(-1)\mathrm{j}\hat{b}_2(l)=-2\mathrm{j}\hat{b}_1(k)\hat{b}_2(l) & \text{for } k<0,l<0\\ -\mathrm{j}\hat{b}_1(k)\hat{b}_2(l)+\hat{b}_1(k)\mathrm{j}\hat{b}_2(l)=0 & \text{for } k<0,l>0\\ 0 & \text{for } k=0 \text{ or } l=0\\ \mathrm{j}\hat{b}_1(k)\hat{b}_2(l)+\hat{b}_1(k)(-1)\mathrm{j}\hat{b}_2(l)=0 & \text{for } k>0,l<0\\ \mathrm{j}\hat{b}_1(k)\hat{b}_2(l)+\hat{b}_1(k)\mathrm{j}\hat{b}_2(l)=2\mathrm{j}\hat{b}_1(k)\hat{b}_2(l) & \text{for } k>0,l>0\end{cases}. \tag{4.362}$$

We observe that:

$$\hat{s}(k,l)=\begin{cases}-\mathrm{j}\hat{t}(k,l) & \text{for } k<0,l<0\\ \mathrm{j}\hat{t}(k,l) & \text{for } k>0,l>0\\ \hat{t}(k,l)=0 & \text{otherwise}\end{cases}. \tag{4.363}$$

Example B4.78

Consider the 2D filter given by Equation (4.319). Construct from it two filters that will allow you to work out the local symmetry of the image feature enhanced by this filter.

The real and the imaginary parts of this filter are:

$$w_{13R} \equiv \begin{bmatrix} 3.548 & -2.097 & -1.773 \\ -2.097 & 4.955 & -2.097 \\ -1.773 & -2.097 & 3.548 \end{bmatrix} \tag{4.364}$$

$$w_{13I} \equiv \begin{bmatrix} 0 & 3.631 & -3.073 \\ -3.631 & 0 & 3.631 \\ 3.073 & -3.631 & 0 \end{bmatrix}. \tag{4.365}$$

We note that these filters are oriented so that they will enhance features that are oriented from top left to bottom right. The first of the two will enhance symmetric such features, while the second will enhance antisymmetric features. Starting from these filters, we would like to create an orthonormal basis for the symmetry/antisymmetry space, which will allow us to assign a phase to the feature that will be picked up by these filters. First, we need to remove the dc component from the first filter. The sum of the values of w_{13R} *is* 0.117*. This divided by* 9 *yields* 0.013*, which when subtracted from all values of* w_{13R} *yields:*

$$w'_{13R} = \begin{bmatrix} 3.535 & -2.110 & -1.786 \\ -2.110 & 4.942 & -2.110 \\ -1.786 & -2.110 & 3.535 \end{bmatrix}. \tag{4.366}$$

Further, the sum of the squares of each filter should be 1*. The sum of the squares of* w_{13I} *is* 71.623*. So, we divide all its elements by* $\sqrt{71.623} = 8.463$*:*

$$\tilde{w}_{13I} = \begin{bmatrix} 0 & 0.429 & -0.363 \\ -0.429 & 0 & 0.429 \\ 0.363 & -0.429 & 0 \end{bmatrix}. \tag{4.367}$$

The sum of the squares of w'_{13R} *is 73.604. So, we divide all elements of* w'_{13R} *with* $\sqrt{73.604} = 8.579$*:*

$$\tilde{w}_{13R} = \begin{bmatrix} 0.412 & -0.246 & -0.208 \\ -0.246 & 0.576 & -0.246 \\ -0.208 & -0.246 & 0.412 \end{bmatrix}. \tag{4.368}$$

Note that if these filters were not approximate due to truncation, they would have had the same sum of squared values. Truncation makes them approximate, and the normalisation of each one with a slightly different value damages somehow their relationship as Hilbert pair.

If we convolve an image with these two filters, we can detect the places where some features oriented along the axis of symmetry/antisymmetry of the filters exists, by squaring and summing the two outputs at each pixel to compute the local energy, which is expected to be high only in the places where such features are present. Then, from the ratio of the two outputs, we shall be able to assign a value of phase to each pixel, to express the local symmetry or antisymmetry of the detected feature.

Example 4.79

Use the filters you developed in Example 4.78 to detect the features of image 4.91 and characterise their symmetry.

(Continued)

Example 4.79 (Continued)

0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0
0.5	0	0	0	0	0	1	0	0
1	0.5	0	0	0	0	0	1	0
1	1	0.5	0	0	0	0	0	1
1	1	1	0.5	0	0	0	0	0
1	1	1	1	0.5	0	0	0	0

Figure 4.91 An image that contains a symmetric and an antisymmetric feature at 45°.

The outputs of the two convolutions are:

$$o_{\mathrm{I}} = \begin{bmatrix} 0.363 & -0.858 & 0 & 0.858 & -0.363 & 0 & 0 \\ 0 & 0.363 & -0.858 & 0 & 0.858 & -0.363 & 0 \\ -0.182 & 0 & 0.363 & -0.858 & 0 & 0.858 & -0.363 \\ 0.066 & -0.182 & 0 & 0.363 & -0.858 & 0 & 0.858 \\ 0.495 & 0.066 & -0.182 & 0 & 0.363 & -0.858 & 0 \\ 0.066 & 0.495 & 0.066 & -0.182 & 0 & 0.363 & -0.858 \\ -0.182 & 0.066 & 0.495 & 0.066 & -0.182 & 0 & 0.363 \end{bmatrix} \tag{4.369}$$

$$o_{\mathrm{R}} = \begin{bmatrix} -0.208 & -0.492 & 1.4 & -0.492 & -0.208 & 0 & 0 \\ 0 & -0.208 & -0.492 & 1.4 & -0.492 & -0.208 & 0 \\ -0.104 & 0 & -0.208 & -0.492 & 1.4 & -0.492 & -0.208 \\ -0.454 & -0.104 & 0 & -0.208 & -0.492 & 1.4 & -0.492 \\ 0 & -0.454 & -0.104 & 0 & -0.208 & -0.492 & 1.4 \\ 0.454 & 0 & -0.454 & -0.104 & 0 & -0.208 & -0.492 \\ 0.104 & 0.454 & 0 & -0.454 & -0.104 & 0 & -0.208 \end{bmatrix}. \tag{4.370}$$

The local energy, computed as the sum of the squares of the two outputs is:

$$E = \begin{bmatrix} 0.175 & 0.978 & 1.96 & 0.978 & 0.175 & 0 & 0 \\ 0 & 0.175 & 0.978 & 1.96 & 0.978 & 0.175 & 0 \\ 0.175 & 0 & 0.175 & 0.978 & 1.96 & 0.978 & 0.175 \\ 0.735 & 0.175 & 0 & 0.175 & 0.978 & 1.96 & 0.978 \\ 0.735 & 0.735 & 0.175 & 0 & 0.175 & 0.978 & 1.96 \\ 0.175 & 0.735 & 0.735 & 0.175 & 0 & 0.175 & 0.978 \\ 0 & 0.175 & 0.735 & 0.735 & 0.175 & 0 & 0.175 \end{bmatrix}. \tag{4.371}$$

The estimated local phase of the detected feature at places where the energy is non-zero, rounded to the nearest integer, is:

$$p = \begin{bmatrix} -60° & 60° & 0° & -60° & 60° & & \\ & -60° & 60° & 0° & -60° & 60° & \\ 60° & & -60° & 60° & 0° & -60° & 60° \\ -8° & 60° & & -60° & 60° & 0° & -60° \\ 90° & -8° & 60° & & -60° & 60° & 0° \\ 8° & 90° & -8° & 60° & & -60° & 60° \\ -60° & 8° & 90° & -8° & 60° & & -60° \end{bmatrix}. \tag{4.372}$$

Example 4.80

Use the 2D filters constructed in Example 4.65 to construct filters that will allow you to estimate in each band the local image phase. Use these filters then to process image 4.71.

We use the following algorithm.

Step 1: *Split the filters given by Equations (4.317)–(4.320) into their real and imaginary parts. There is no need to do the same with the other filters as they are identical. Also, there is no need to do anything with the* $\mathbf{w}_2\mathbf{w}_2{}^T$ *filter, as it is real and it captures the low frequency components of the image. So, at the end of this step you will have four pairs of filters.*

Step 2: *Remove the mean from all elements of each one of the filters you constructed from the real part of a complex filter. This way, all eight filters will have* 0 *mean.*

Step 3: *Find the sum of the squares of the elements of each filter. Divide the elements of each filter by the square root of the sum of its squared elements. This way, each filter will have the sum of squares of its elements equal to* 1.

Step 4: *Convolve the image with each one of the eight filters. You will have* 8 *outputs.*

Step 5: *Combine the outputs of each pair of filters: the sum of the squares of their values at each pixel position will give the local energy. This way you will produce four outputs. These outputs correspond to those of Figure 4.72. However, they are not identical, as the filters here have been subjected to different normalisation.*

Step 6: *For the pixels where the local energy is above* 0*, compute the local phase feature from each pair of outputs, using:*

$$\text{phase} = \left| \tan^{-1} \frac{\text{output\_antisymmetric\_filter}}{\text{output\_symmetric\_filter}} \right|. \tag{4.373}$$

Make sure that the result is in the range $[0°, 90°]$*. The higher the value, the more locally anti-symmetric the image is around the corresponding pixel. This way you will have four more outputs.*

To pixels with local energy 0*, assign a nominal phase computed, say, as the average of the phases of their neighbours. This is so that all pixels have a meaningful phase feature that will allow you to perform step* 7.

Step 7: *In total you have produced* 8 *feature maps: four local energy ones and four local phase ones. You can read the value a pixel has in each one of the outputs to produce the feature vector of the pixel. You may then cluster these feature vectors to produce the segmentation of the image.*

4.5 Wavelets

Is there a way other than using Gabor functions to span the whole spatio-frequency space?

Yes, by using **wavelets**. Gabor functions may be thought of as *modulated* and translated versions of the Gaussian function used to cover the whole spatio-frequency domain. In wavelet analysis a basic function is **scaled** and translated in order to cover the spatio-frequency domain. This makes use of the **shifting** and **scaling** properties of the Fourier transform.

The **shifting property** of the Fourier transform tells us that if $\Psi(\omega)$ is the Fourier transform of a function $\psi(t)$, the Fourier transform of the shifted function $\psi(t-b)$ is $\Psi(\omega)\mathrm{e}^{-\mathrm{j}b\omega}$ (see Example 4.81). So, shifting a function results in changing only the phase of the Fourier transform of the function.

The **scaling property** of the Fourier transform tells us that the Fourier transform of the scaled function $\psi(t/a)$ is $a\Psi(a\omega)$ (see Example 4.82).

So, scaling a function changes its frequency bandwidth and shifts the centre of the band away from its original position. Note that if we want to use this property to span the whole of the frequency axis with non-overlapping bands, we must choose a function $\psi(t)$ such that the zero frequency is not included in its frequency band, since 0 cannot be scaled (and thus it cannot be shifted away from its original position; see Example 4.86).

Example 4.81

If $F(\omega)$ is the Fourier transform of $f(t)$, show that the Fourier transform of $f(t-b)$ is $F(\omega)\mathrm{e}^{-\mathrm{j}b\omega}$.

The Fourier transform $\hat{F}(\omega)$ of $f(t-b)$ is:

$$\hat{F}(\omega) = \int_{-\infty}^{\infty} f(t-b)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t. \tag{4.374}$$

We define a new variable of integration $\tilde{t} \equiv t-b$, so that $t = \tilde{t}+b$ and $\mathrm{d}t = \mathrm{d}\tilde{t}$. Then:

$$\begin{aligned}\hat{F}(\omega) &= \int_{-\infty}^{\infty} f(\tilde{t})\mathrm{e}^{-\mathrm{j}\omega(\tilde{t}+b)}\mathrm{d}\tilde{t}\\ &= \mathrm{e}^{-\mathrm{j}b\omega}\int_{-\infty}^{\infty} f(\tilde{t})\mathrm{e}^{-\mathrm{j}\omega\tilde{t}}\mathrm{d}\tilde{t}\\ &= \mathrm{e}^{-\mathrm{j}b\omega}F(\omega).\end{aligned} \tag{4.375}$$

Example 4.82

If $F(\omega)$ is the Fourier transform of $f(t)$, show that the Fourier transform of $f(t/a)$ is $aF(a\omega)$.

The Fourier transform of $f(t)$ is:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \tag{4.376}$$

The Fourier transform $\tilde{F}(\omega)$ of $f(t/a)$ is:

$$\tilde{F}(\omega) = \int_{-\infty}^{\infty} f\left(\frac{t}{a}\right) \mathrm{e}^{-\mathrm{j}\omega t} \mathrm{d}t. \tag{4.377}$$

We define a new variable of integration $\tilde{t} \equiv t/a$ in (4.377), so that $t = a\tilde{t}$ and $\mathrm{d}t = a\mathrm{d}\tilde{t}$:

$$\tilde{F}(\omega) = \int_{-\infty}^{\infty} f(\tilde{t}) \mathrm{e}^{-\mathrm{j}\omega a\tilde{t}} a\mathrm{d}\tilde{t} = a\int_{-\infty}^{\infty} f(\tilde{t}) \mathrm{e}^{-\mathrm{j}(\omega a)\tilde{t}} \mathrm{d}\tilde{t}. \tag{4.378}$$

We recognise the integral on the right-hand side of (4.378) as the Fourier transform $F(\omega)$ of Equation (4.376) computed at frequency $a\omega$. So the conjecture has been proven.

Example 4.83

Consider a function $\psi(t)$ with Fourier transform $\Psi(\omega)$ defined as follows:

$$\Psi(\omega) = \begin{cases} 1 \text{ for } 5 < \omega < 10 \\ 0 \text{ otherwise} \end{cases}. \tag{4.379}$$

Find the frequency band inside which the Fourier transform of function $\psi(t/2)$ is non-zero.

The Fourier transform of $\psi(t/2)$ is $2\Psi(2\omega)$. According to definition (4.379), this function is non-zero when its argument is in the range $(5, 10)$, i.e. when:

$$5 < 2\omega < 10 \Rightarrow 2.5 < \omega < 5. \tag{4.380}$$

Note that, although at first glance function $\Psi(2\omega)$ appears to refer to higher frequencies than function $\Psi(\omega)$, this is deceptive: we have to examine the **range** *of the* **independent variable** *ω rather than the* **argument** *of the function to decide whether the frequency band is shifted to higher or lower frequencies by scaling.*

Example 4.84

Consider a function $\phi(t)$ with Fourier transform $\Phi(\omega)$ defined as follows:

$$\Phi(\omega) = \begin{cases} 1 \text{ for } -10 < \omega < 10 \\ 0 \text{ otherwise} \end{cases}. \tag{4.381}$$

Find the frequency band inside which the Fourier transform of function $\phi(t/2)$ is non-zero.

The Fourier transform of $\phi(t/2)$ is $2\Phi(2\omega)$. This function is non-zero when its argument is in the range $(-10, 10)$, i.e. when:

$$-10 < 2\omega < 10 \Rightarrow -5 < \omega < 5. \tag{4.382}$$

Example 4.85

Consider a band-limited function $\psi(t)$ defined in the range $[\omega_1, \omega_2]$ with central frequency $\omega_0 \equiv (\omega_1 + \omega_2)/2$ and bandwidth $\omega_2 - \omega_1$. Compute the corresponding quantities for $\psi(t/a)$, $a > 0$.

The argument of the Fourier transform of the scaled function will be $a\omega$ and it will take values in the range $[a\omega_1, a\omega_2]$. This Fourier transform is non-zero if its argument $a\omega$ is in the range $[\omega_1, \omega_2]$. We must, therefore, have $\omega_1 < a\omega < \omega_2 \Rightarrow \omega_1/a < \omega < \omega_2/a$. So, the scaled function, $\psi(t/a)$, will be non-zero over the frequency range $[\omega_1/a, \omega_2/a]$, with central frequency $(\omega_1/a + \omega_2/a)/2 = \omega_0/a$, and bandwidth $\omega_2/a - \omega_1/a = (\omega_2 - \omega_1)/a$.

Example 4.86

Show that if the 0 frequency belongs to the band $[\omega_1, \omega_2]$ of Example 4.85, it is impossible to use function $\psi(t)$ to create non-overlapping frequency bands by scaling it. The frequency bands we shall create will always be nested, i.e. if $a_1 > a_2 > 0$ are two values of a used in $\psi(t/a)$, the frequency band inside which the Fourier transform of $\psi(t/a_1)$ is non-zero will be contained in the frequency band inside which the Fourier transform of $\psi(t/a_2)$ is non-zero.

According to Example 4.85, the frequency band inside which the Fourier transform of $\psi(t/a_1)$ is non-zero is $[\omega_1/a_1, \omega_2/a_1]$, and the frequency band inside which the Fourier transform of $\psi(t/a_2)$ is non-zero is $[\omega_1/a_2, \omega_2/a_2]$. To show that the first is contained inside the second if 0 is contained in $[\omega_1, \omega_2]$, it is enough to show that always $\omega_1/a_1 \geq \omega_1/a_2$ and $\omega_2/a_1 \leq \omega_2/a_2$. We observe that since $a_1 > a_2 > 0$, $1/a_1 < 1/a_2$. There are three possible cases:

Case 1: *$\omega_1 = 0, \omega_2 > 0$: Then $\omega_2/a_1 < \omega_2/a_2$, and obviously $[0, \omega_2/a_1]$ is contained in $[0, \omega_2/a_2]$.*

Case 2: *$\omega_2 = 0, \omega_1 < 0$: Then $\omega_1/a_1 > \omega_1/a_2$, and obviously $[\omega_1/a_1, 0]$ is contained in $[\omega_1/a_2, 0]$.*

Case 3: *$\omega_1 < 0, \omega_2 > 0$: Then $\omega_1/a_1 > \omega_1/a_2$ and $\omega_2/a_1 < \omega_2/a_2$, and obviously $[\omega_1/a_1, \omega_2/a_1]$ is contained inside $[\omega_1/a_2, \omega_2/a_2]$.*

Therefore, if we want to use the scaling property of the Fourier transform to span the whole of the frequency axis with non-overlapping bands, we must choose a function $\psi(t)$ such that the zero frequency is not included inside its frequency band, since 0 cannot be scaled (and thus shifted away from its original position).

What is a wavelet?

A wavelet $\psi_{ab}(t)$ is a function produced by shifting by b and scaling by a a basic function $\psi(t)$ called **mother wavelet**:

$$\psi_{ab}(t) \equiv \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right). \tag{4.383}$$

The Fourier transform of this function is

$$\Psi_{ab}(\omega) = \sqrt{a}\Psi(a\omega)\mathrm{e}^{-\mathrm{j}b\omega} \tag{4.384}$$

where $\Psi(\omega)$ is the Fourier transform of the mother wavelet.

Example 4.87

By direct calculation show that $\Psi_{ab}(\omega)$ given by Equation (4.384) is the Fourier transform of function $\psi_{ab}(t)$ defined by Equation (4.383).

The Fourier transform of $\psi_{ab}(t)$ is:

$$\Psi_{ab}(\omega) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) \mathrm{e}^{-\mathrm{j}\omega t} \mathrm{d}t. \tag{4.385}$$

We define a new variable of integration $\tilde{t} \equiv (t-b)/a$, so that $t = a\tilde{t} + b$ and $\mathrm{d}t = a\mathrm{d}\tilde{t}$. Then:

$$\begin{aligned}\Psi_{ab}(\omega) &= \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \psi(\tilde{t}) \mathrm{e}^{-\mathrm{j}\omega(a\tilde{t}+b)} a \mathrm{d}\tilde{t} \\ &= \sqrt{a} \int_{-\infty}^{\infty} \psi(\tilde{t}) \mathrm{e}^{-\mathrm{j}(\omega a)\tilde{t}} \mathrm{d}\tilde{t} \mathrm{e}^{-\mathrm{j}\omega b} \\ &= \sqrt{a} \Psi(a\omega) \mathrm{e}^{-\mathrm{j}b\omega}. \end{aligned} \tag{4.386}$$

Example 4.88

Using the results of Examples 4.81 and 4.82, prove that $\Psi_{ab}(\omega)$ given by Equation (4.384) is the Fourier transform of function $\psi_{ab}(t)$ defined by Equation (4.383).

A confusing point is the order in which we apply scaling and shifting to the independent variable t.

Let us try shifting first and scaling afterwards, as it appears to be the order indicated by $(t-b)/a$. We start by stating that the Fourier transform of function $\psi(t)$ is $\Psi(\omega)$. Then according to Example 4.81, the Fourier transform of $\psi(t-b)$ is $\Psi(\omega)\mathrm{e}^{-\mathrm{j}b\omega}$, and according to Example 4.82, the Fourier transform of $\psi((t-b)/a)$ is $a\Psi(a\omega)\mathrm{e}^{-\mathrm{j}ba\omega}$. Therefore, the Fourier transform of $\psi((t-b)/a)/\sqrt{a}$ appears to be $\sqrt{a}\Psi(a\omega)\mathrm{e}^{-\mathrm{j}ba\omega}$. This is clearly the wrong result (see Example 4.87).

So, let us try scaling first and shifting afterwards. Then according to Example 4.82, the Fourier transform of $\psi(t/a)$ is $a\Psi(a\omega)$, and according to Example 4.81 the Fourier transform of $\psi((t-b)/a)$ is $a\Psi(a\omega)\mathrm{e}^{-\mathrm{j}b\omega}$ which leads to the correct answer.

To avoid confusion as to which operation has to be applied first and which second, we must always remember that we scale or shift the **independent** *variable t and* **not** *the* **argument** *of function ψ. When we did the shifting first and the scaling afterwards, we scaled the whole argument of ψ, which was $t-b$, and not just t. If we had scaled only t, as we should have done, we would have not created function $\psi((t-b)/a)$, which we wanted, but function $\psi(t/a - b)$, the Fourier transform of which is indeed $a\Psi(a\omega)\mathrm{e}^{-\mathrm{j}ba\omega}$.*

How can we use wavelets to analyse a signal?

By changing a and b in Equations (4.383) and (4.384) we can produce two families of functions that constitute two equivalent sets of elementary answers to the fundamental question "what is where" in a signal. If we wish to know how much of each elementary "what is where" answer is conveyed by the signal, we must project that signal onto each one of the elementary functions of the set in turn. Projection in the continuous domain is achieved by multiplication of the signal, $f(t)$, with the

complex conjugate of the wavelet function and integration over all values of t:

$$w_f(a,b) \equiv \int_{-\infty}^{\infty} \psi_{ab}^*(t) f(t) \mathrm{d}t. \tag{4.387}$$

This is the **wavelet transform** of the signal.

Alternatively, we may obtain the same answer if we project the Fourier transform $F(\omega)$ of the signal on the elementary basis made up from functions (4.384):

$$W_f(a,b) \equiv \int_{-\infty}^{\infty} \Psi_{ab}^*(2\pi\nu) F(2\pi\nu) d\nu = \frac{1}{2\pi} \int_{-\infty}^{\infty} \Psi_{ab}^*(\omega) F(\omega) \mathrm{d}\omega. \tag{4.388}$$

Note that $\omega \equiv 2\pi\nu$.

Example B4.89

Prove that $w_f(a,b)$ and $W_f(a,b)$, as defined by Equations (4.387) and (4.388), respectively, are equal.

We start from Equation (4.388) and substitute $\Psi_{ab}(\omega)$ from (4.384):

$$W_f(a,b) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \sqrt{a} \Psi^*(a\omega) \mathrm{e}^{\mathrm{j}b\omega} F(\omega) \mathrm{d}\omega. \tag{4.389}$$

From the definition of the Fourier transform we have

$$\Psi(\omega) = \int_{-\infty}^{\infty} \psi(t) \mathrm{e}^{-\mathrm{j}\omega t} \mathrm{d}t \tag{4.390}$$

and:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \mathrm{e}^{-\mathrm{j}\omega t} \mathrm{d}t. \tag{4.391}$$

Upon substitution into (4.389) we obtain

$$W_f(a,b) = \frac{\sqrt{a}}{2\pi} \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} \psi^*(t) \mathrm{e}^{\mathrm{j}a\omega t} \mathrm{d}t \right\} \mathrm{e}^{\mathrm{j}b\omega} \left\{ \int_{-\infty}^{\infty} f(\tilde{t}) \mathrm{e}^{-\mathrm{j}\omega\tilde{t}} \mathrm{d}\tilde{t} \right\} \mathrm{d}\omega \tag{4.392}$$

where in the second internal integral we changed the dummy variable of integration to $\tilde{t}$ so that there is no confusion with the variable of integration t of the first integral. We may integrate first over ω:

$$W_f(a,b) = \frac{\sqrt{a}}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \psi^*(t) f(\tilde{t}) \left\{ \int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{j}\omega(at+b-\tilde{t})} \mathrm{d}\omega \right\} \mathrm{d}t \mathrm{d}\tilde{t}. \tag{4.393}$$

We use Equation (4.47) with $y \equiv \omega$ and $x \equiv at + b - \tilde{t}$ to compute the integral over ω as $2\pi\delta(at + b - \tilde{t})$ and then perform the integration over $\tilde{t}$:

$$W_f(a,b) = \sqrt{a} \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} f(\tilde{t}) \delta(at + b - \tilde{t}) \mathrm{d}\tilde{t} \right\} \psi^*(t) \mathrm{d}t. \tag{4.394}$$

The integral over $\tilde{t}$ yields $f(at + b)$:

$$W_f(a,b) = \sqrt{a} \int_{-\infty}^{\infty} \psi^*(t) f(at + b) \mathrm{d}t. \tag{4.395}$$

We define a new variable of integration $x \equiv at + b \Rightarrow t = (x - b)/a$ and $\mathrm{d}t = \mathrm{d}x/a$, so

$$W_f(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \psi^* \left(\frac{x-b}{a} \right) f(x) \mathrm{d}x = \int_{-\infty}^{\infty} \psi_{ab}^*(x) f(x) \mathrm{d}x \tag{4.396}$$

where we used the definition of ψ_{ab} of Equation (4.383) to obtain the last equality. If we change the dummy variable of integration from x to t, we deduce, upon comparison with Equation (4.387), that:

$$W_f(a,b) = w_f(a,b). \tag{4.397}$$

In other words, the two bases of the space "what is where" defined by the sets of functions $w_f(a,b)$ and $W_f(a,b)$, respectively, are equivalent, but the first one is appropriate for the expansion of the signal itself, while the second one is appropriate for the expansion of the Fourier transform of the signal.

Box 4.8 How should we choose the mother wavelet?

First of all, since we scale the frequencies over which $\Psi(\omega)$ is defined, and since number 0 cannot be scaled, it is obvious that the bandwidth of $\Psi(\omega)$ should not contain 0, i.e. $\Psi(\omega)$ should have zero dc (direct component).

One of the drawbacks of the Gabor functions is that the expansion of a signal in terms of them is not invertible. To improve upon this, we require here the wavelet transform expressed by Equation (4.387) to be invertible. This means that from the knowledge of $w_f(a,b)$ we want to be able to recover function $f(t)$.

Let us start by multiplying both sides of Equation (4.387) with $\psi_{ab}(s)$ and integrating over all scales and all shifts of the independent variable. As $\psi_{ab}(s) = \psi((s-b)/a)$, the scaling factor is $1/a$ and the shifting parameter is b. So, in order to integrate over the scale-shift space, we must multiply both sides of the equation with the elementary area in that space, which is $|d(1/a)db| = \mathrm{d}a\mathrm{d}b/a^2$. Therefore:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty}\psi_{ab}(s)w_f(a,b)\frac{\mathrm{d}a\mathrm{d}b}{a^2} = \int_{-\infty}^{\infty}\int_{0}^{\infty}\psi_{ab}(s)\int_{-\infty}^{\infty}\psi^*_{ab}(t)f(t)\mathrm{d}t\frac{\mathrm{d}a\mathrm{d}b}{a^2}. \tag{4.398}$$

On the right-hand side let us express $\psi_{ab}(s)$ and $\psi_{ab}(t)$ in terms of their Fourier transforms

$$\int_{-\infty}^{\infty}\int_{0}^{\infty}\frac{\psi_{ab}(s)}{a^2}w_f(a,b)\mathrm{d}a\mathrm{d}b = \int_{-\infty}^{\infty}\int_{0}^{\infty}\int_{-\infty}^{\infty}\frac{1}{a^2}f(t)\times \left\{\frac{1}{2\pi}\int_{-\infty}^{\infty}\sqrt{a}\Psi(a\tilde{\omega})\mathrm{e}^{-\mathrm{j}b\tilde{\omega}}\mathrm{e}^{\mathrm{j}\tilde{\omega}s}\mathrm{d}\tilde{\omega}\right\}\left\{\frac{1}{2\pi}\int_{-\infty}^{\infty}\sqrt{a}\Psi^*(a\omega)\mathrm{e}^{\mathrm{j}b\omega}\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}\omega\right\}\mathrm{d}t\mathrm{d}a\mathrm{d}b \tag{4.399}$$

where we used $\tilde{\omega}$ instead of ω in one of the two Fourier integrals in order to avoid confusion.

Let us integrate first over b:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty}\frac{\psi_{ab}(s)}{a^2}w_f(a,b)\mathrm{d}a\mathrm{d}b = \frac{1}{(2\pi)^2}\int_{0}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{a}f(t)\Psi(a\tilde{\omega})\mathrm{e}^{\mathrm{j}\tilde{\omega}s}\Psi^*(a\omega)\mathrm{e}^{-\mathrm{j}\omega t}\underbrace{\int_{-\infty}^{\infty}\mathrm{e}^{\mathrm{j}b(\omega-\tilde{\omega})}\mathrm{d}b}_{2\pi\delta(\omega-\tilde{\omega})}\mathrm{d}\omega\mathrm{d}\tilde{\omega}\mathrm{d}t\mathrm{d}a. \tag{4.400}$$

(Continued)

Box 4.8 (Continued)

Next we integrate over $\tilde{\omega}$:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty}\frac{\psi_{ab}(s)}{a^2}w_f(a,b)\mathrm{d}a\mathrm{d}b = \frac{1}{2\pi}\int_{0}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{a}f(t)\Psi^*(a\omega)\mathrm{e}^{-\mathrm{j}\omega t}\underbrace{\int_{-\infty}^{\infty}\Psi(a\tilde{\omega})\mathrm{e}^{\mathrm{j}\tilde{\omega}s}\delta(\omega-\tilde{\omega})\mathrm{d}\tilde{\omega}}_{\Psi(a\omega)\mathrm{e}^{\mathrm{j}\omega s}}\mathrm{d}\omega\mathrm{d}t\mathrm{d}a. \tag{4.401}$$

Let us define a new variables of integration u instead of a:

$$u \equiv a\omega \Rightarrow a = \frac{u}{\omega} \Rightarrow \frac{1}{a} = \frac{\omega}{u} \quad \text{and} \quad \mathrm{d}a = \frac{\mathrm{d}u}{\omega}. \tag{4.402}$$

Then:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty}\frac{\psi_{ab}(s)}{a^2}w_f(a,b)\mathrm{d}a\mathrm{d}b = \frac{1}{2\pi}\int_{0}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{u}f(t)\Psi^*(u)\Psi(u)\mathrm{e}^{\mathrm{j}\omega(s-t)}\mathrm{d}t\mathrm{d}u\mathrm{d}\omega. \tag{4.403}$$

Next we integrate over ω:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty}\frac{\psi_{ab}(s)}{a^2}w_f(a,b)\mathrm{d}a\mathrm{d}b = \frac{1}{2\pi}\int_{0}^{\infty}\int_{-\infty}^{\infty}\frac{1}{u}f(t)\Psi(u)\Psi^*(u)\underbrace{\int_{-\infty}^{\infty}\mathrm{e}^{\mathrm{j}\omega(s-t)}\mathrm{d}\omega}_{2\pi\delta(s-t)}\mathrm{d}t\mathrm{d}u. \tag{4.404}$$

Then we integrate over t:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty}\frac{\psi_{ab}(s)}{a^2}w_f(a,b)\mathrm{d}a\mathrm{d}b = \int_{0}^{\infty}\frac{1}{u}\Psi(u)\Psi^*(u)\underbrace{\int_{-\infty}^{\infty}f(t)\delta(s-t)dt}_{f(s)}\mathrm{d}u. \tag{4.405}$$

And finally:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty}\frac{\psi_{ab}(s)}{a^2}w_f(a,b)\mathrm{d}a\mathrm{d}b = f(s)\int_{0}^{\infty}\frac{|\Psi(u)|^2}{u}\mathrm{d}u. \tag{4.406}$$

Let us define:

$$C_\psi \equiv \int_{0}^{\infty}\frac{|\Psi(\omega)|^2}{\omega}\mathrm{d}\omega. \tag{4.407}$$

For the inverse transformation to exist, this integral must exist and must have a finite value. This is the **admissibility condition**. If this condition is fulfilled, then:

$$f(t) = \frac{1}{C_\psi}\int_{-\infty}^{\infty}\int_{0}^{\infty}\frac{\psi_{ab}(t)}{a^2}w_f(a,b)\mathrm{d}a\mathrm{d}b. \tag{4.408}$$

This equation is called **the resolution of identity**. It expresses the original function as a linear superposition of the wavelet functions.

We notice that the integrand in (4.407) would have been singular for $\omega = 0$, unless $\Psi(0) = 0$. This confirms the intuitive understanding we had earlier when we deduced that **the mother wavelet should have no direct component**.

So, the mother wavelet has to be chosen so that it has zero mean and it obeys the admissibility condition expressed by the existence of the integral of Equation (4.407).

Example B4.90

Prove that the wavelet transform obeys Parseval's relation:

$$C_\psi \int_{-\infty}^{\infty} |f(t)|^2 \mathrm{d}t = \int_{-\infty}^{\infty}\int_{0}^{\infty} |w_f(a,b)|^2 \frac{\mathrm{d}a\mathrm{d}b}{a^2}. \tag{4.409}$$

Let us write explicitly the right-hand side of this expression and manipulate it so that we bring it into the form of the left-hand side

$$\int_{-\infty}^{\infty}\int_{0}^{\infty} |w_f(a,b)|^2 \frac{1}{a^2}\mathrm{d}a\mathrm{d}b =$$
$$\int_{-\infty}^{\infty}\int_{0}^{\infty} w_f^*(a,b) w_f(a,b) \frac{1}{a^2}\mathrm{d}a\mathrm{d}b =$$
$$\int_{-\infty}^{\infty}\int_{0}^{\infty} \left\{\int_{-\infty}^{\infty} \psi_{ab}(t) f(t)\mathrm{d}t\right\}\left\{\int_{-\infty}^{\infty} \psi_{ab}^*(s) f(s) ds\right\} \frac{1}{a^2}\mathrm{d}a\mathrm{d}b \tag{4.410}$$

where we substituted for $w_f(a,b)$ from Equation (4.387) and used a different dummy variable of integration for the two integrals in order to avoid confusion. Let us next substitute $\psi_{ab}(t)$ in terms of its Fourier transform given by (4.384):

$$\int_{-\infty}^{\infty}\int_{0}^{\infty} |w_f(a,b)|^2 \frac{1}{a^2}\mathrm{d}a\mathrm{d}b =$$
$$\int_{-\infty}^{\infty}\int_{0}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left\{\frac{1}{2\pi}\int_{-\infty}^{\infty} \sqrt{a}\Psi(a\omega)\mathrm{e}^{-\mathrm{j}b\omega}\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega\right\} \times$$
$$\left\{\frac{1}{2\pi}\int_{-\infty}^{\infty} \sqrt{a}\Psi^*(a\tilde{\omega})\mathrm{e}^{\mathrm{j}b\tilde{\omega}}\mathrm{e}^{-\mathrm{j}\tilde{\omega}s}\mathrm{d}\tilde{\omega}\right\} f(t)f(s)\frac{1}{a^2}\mathrm{d}t\mathrm{d}s\mathrm{d}a\mathrm{d}b. \tag{4.411}$$

We re-arrange and perform the integration over b first, making use of Equation (4.47):

$$\int_{-\infty}^{\infty}\int_{0}^{\infty} |w_f(a,b)|^2 \frac{\mathrm{d}a\mathrm{d}b}{a^2} =$$
$$\frac{1}{(2\pi)^2}\int_{0}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Psi(a\omega)\Psi^*(a\tilde{\omega}) f(t)f(s)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{e}^{-\mathrm{j}\tilde{\omega}s}\frac{1}{a}\mathrm{d}t\mathrm{d}s\mathrm{d}\omega\mathrm{d}\tilde{\omega}\times$$
$$\underbrace{\int_{-\infty}^{\infty}\mathrm{e}^{\mathrm{j}b(\tilde{\omega}-\omega)}\mathrm{d}b}_{2\pi\delta(\tilde{\omega}-\omega)}\mathrm{d}a =$$
$$\frac{1}{2\pi}\int_{0}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Psi(a\omega)\Psi^*(a\tilde{\omega}) f(t)f(s)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{e}^{-\mathrm{j}\tilde{\omega}s}\frac{1}{a}\delta(\tilde{\omega}-\omega)\mathrm{d}t\mathrm{d}s\mathrm{d}\omega\mathrm{d}\tilde{\omega}\mathrm{d}a.$$

We re-arrange and perform the integration over $\tilde{\omega}$ next:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty} |w_f(a,b)|^2 \frac{\mathrm{d}a\mathrm{d}b}{a^2} =$$
$$\frac{1}{2\pi}\int_{0}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Psi(a\omega) f(t)f(s)\mathrm{e}^{\mathrm{j}\omega t}\underbrace{\frac{1}{a}\int_{-\infty}^{\infty}\Psi^*(a\tilde{\omega})\mathrm{e}^{-\mathrm{j}\tilde{\omega}s}\delta(\tilde{\omega}-\omega)\mathrm{d}\tilde{\omega}}_{\Psi^*(a\omega)\mathrm{e}^{-\mathrm{j}\omega s}}\mathrm{d}t\mathrm{d}s\mathrm{d}\omega\mathrm{d}a.$$

(Continued)

Example B4.90 (Continued)

Before we proceed, we replace integration over variable a with integration over a new variable u, defined as:

$$u \equiv a\omega \Rightarrow a = \frac{u}{\omega} \Rightarrow \frac{1}{a} = \frac{\omega}{u} \quad \text{and} \quad \mathrm{d}a = \frac{\mathrm{d}u}{\omega}. \tag{4.412}$$

Then:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty} |w_f(a,b)|^2 \frac{\mathrm{d}a\mathrm{d}b}{a^2} =$$

$$\frac{1}{2\pi}\int_{0}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Psi(u)f(t)f(s)\mathrm{e}^{\mathrm{j}\omega t}\frac{\omega}{u}\Psi^*(u)\mathrm{e}^{-\mathrm{j}\omega s}\mathrm{d}t\mathrm{d}s\mathrm{d}\omega\frac{\mathrm{d}u}{\omega}. \tag{4.413}$$

We perform next the integration over ω and group together the factors that have to be integrated over u:

$$\int_{-\infty}^{\infty}\int_{0}^{\infty} |w_f(a,b)|^2 \frac{\mathrm{d}a\mathrm{d}b}{a^2} =$$

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t)f(s)\underbrace{\int_{0}^{\infty}\Psi(u)\Psi^*(u)\frac{1}{u}du}_{C_\psi}\underbrace{\int_{-\infty}^{\infty}\mathrm{e}^{\mathrm{j}\omega(t-s)}\mathrm{d}\omega}_{2\pi\delta(t-s)}\mathrm{d}t\mathrm{d}s =$$

$$C_\psi\int_{-\infty}^{\infty} f(t)\underbrace{\int_{-\infty}^{\infty} f(s)\delta(t-s)\mathrm{d}s}_{f(t)}\mathrm{d}t =$$

$$C_\psi\int_{-\infty}^{\infty}|f(t)|^2\mathrm{d}t. \tag{4.414}$$

The last result is exactly the left-hand side of Equation (4.409), which we had to prove.

Example B4.91

Prove that the wavelet transform preserves the energy over different scales, i.e. prove that

$$\int_{-\infty}^{\infty}|\psi_{ab}(t)|^2\mathrm{d}t \equiv \int_{-\infty}^{\infty}\frac{1}{a}\left|\psi\left(\frac{t-b}{a}\right)\right|^2\mathrm{d}t \tag{4.415}$$

is independent of the scaling factor a**.**

We change the variable of integration to $\tilde{t} \equiv (t-b)/a$ *in the integral on the right-hand side of this equation. Then:*

$$\int_{-\infty}^{\infty}|\psi_{ab}(t)|^2\mathrm{d}t = \int_{-\infty}^{\infty}\frac{1}{a}|\psi(\tilde{t})|^2 a\mathrm{d}\tilde{t} = \int_{-\infty}^{\infty}|\psi(t)|^2\mathrm{d}t. \tag{4.416}$$

We observe that the right-hand side is independent of the scaling parameter a and so the energy of $\psi_{ab}(t)$ *is the same for all values of a, i.e. at all scales.*

Example B4.92

Identify the conditions under which the centre of gravity of a mother wavelet is zero.
The centre of gravity of a signal is given by Equation (4.27). For the mother wavelet this becomes

$$\bar{t} = \frac{1}{E_\infty}\int_{-\infty}^{\infty} t|\psi(t)|^2 \mathrm{d}t \tag{4.417}$$

where

$$E_\infty \equiv \int_{-\infty}^{\infty} |\psi(t)|^2 \mathrm{d}t. \tag{4.418}$$

If $\psi(t)$ is chosen to be a symmetric ($\psi(t) = \psi(-t)$) or an antisymmetric ($\psi(t) = -\psi(-t)$) function, the integral in (4.417) is an integral of an antisymmetric integrand over a symmetric range of integration, so it must be zero. If $\psi(t)$ is neither symmetric nor antisymmetric, $\bar{t} \neq 0$.

Example B4.93

Show that the centre of gravity of function $\psi_{ab}(t)$ under certain conditions is b.
We use Equation (4.27) to compute the centre of gravity of $\psi_{ab}(t)$

$$\bar{t}_{ab} \equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty} t|\psi_{ab}(t)|^2 \mathrm{d}t \tag{4.419}$$

where E_∞ is given by (4.418).

We substitute in (4.419) from Equation (4.383):

$$\bar{t}_{ab} = \frac{1}{E_\infty}\int_{-\infty}^{\infty} t\frac{1}{a}\left|\psi\left(\frac{t-b}{a}\right)\right|^2 \mathrm{d}t. \tag{4.420}$$

We define a new variable of integration, $\tilde{t} \equiv (t-b)/a$, so that $t = a\tilde{t} + b$ and $\mathrm{d}t = a\mathrm{d}\tilde{t}$. Then:

$$\begin{aligned}\bar{t}_{ab} &= \frac{1}{E_\infty}\int_{-\infty}^{\infty} (a\tilde{t}+b)\frac{1}{a}|\psi(\tilde{t})|^2 a\mathrm{d}\tilde{t} \\ &= \frac{1}{E_\infty}a\int_{-\infty}^{\infty} \tilde{t}|\psi(\tilde{t})|^2\mathrm{d}\tilde{t} + \frac{1}{E_\infty}b\int_{-\infty}^{\infty}|\psi(\tilde{t})|^2\mathrm{d}\tilde{t}.\end{aligned} \tag{4.421}$$

If $\psi(t)$ is chosen to be a symmetric or an antisymmetric function, the first integral will vanish because it will be the integral of an antisymmetric integrand over a symmetric range of integration. The second integral is recognised to be E_∞. This will result in $\bar{t}_{ab} = b$. If $\psi(t)$ is neither symmetric nor antisymmetric, $\bar{t}_{ab} = a\bar{t} + b$, where $\bar{t}$ is the centre of gravity of the mother wavelet.

Example B4.94

Compute the spectral centre of gravity $\overline{\omega}_{ab}$ of function $\Psi_{ab}(\omega)$ in terms of the spectral centre of gravity $\overline{\omega}$ of $\Psi(\omega)$.
The spectral centre of gravity of a function is given by Equation (4.30):

$$\overline{\omega}_{ab} \equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty} \omega|\Psi_{ab}(\omega)|^2\mathrm{d}\omega. \tag{4.422}$$

(Continued)

Example B4.94 (Continued)

Upon substitution of $\Psi_{ab}(\omega)$ *from Equation (4.384)we obtain:*

$$\overline{\omega}_{ab} \equiv \frac{1}{E_\infty}\int_{-\infty}^{\infty} \omega|\sqrt{a}\Psi(a\omega)\mathrm{e}^{-\mathrm{j}b\omega}|^2\mathrm{d}\omega. \tag{4.423}$$

In the last integral we define a new variable of integration, $\tilde{\omega} \equiv a\omega$. *Then* $\omega = \tilde{\omega}/a$ *and* $\mathrm{d}\omega = \mathrm{d}\tilde{\omega}/a$*:*

$$\begin{aligned}\overline{\omega}_{ab} &= \frac{1}{E_\infty}\int_{-\infty}^{\infty}\frac{\tilde{\omega}}{a}a\Psi(\tilde{\omega})^2\frac{1}{a}\mathrm{d}\tilde{\omega}\\ &= \underbrace{\frac{1}{a}\frac{1}{E_\infty}\int_{-\infty}^{\infty}\tilde{\omega}\Psi(\tilde{\omega})^2\mathrm{d}\tilde{\omega}}_{\text{centre of gravity of }\Psi(\omega)}\\ &= \frac{1}{a}\overline{\omega}.\end{aligned} \tag{4.424}$$

Box 4.9 Does the wavelet function minimise the uncertainty inequality?

In general it does not. We can compute the left-hand side of the uncertainty inequality (4.38) for the wavelet function to answer this question more precisely.

We start by computing the time dispersion of function $\psi_{ab}(t)$ first. We use the result of Example 4.93 and Equation (4.26):

$$(\Delta t_{ab})^2 = \frac{1}{E_\infty}\int_{-\infty}^{\infty}(t - a\bar{t} - b)^2\frac{1}{a}\left|\psi\left(\frac{t-b}{a}\right)\right|^2\mathrm{d}t. \tag{4.425}$$

We define a new variable of integration $\tilde{t} \equiv (t-b)/a$, so that $t = a\tilde{t} + b$ and $dt = a\mathrm{d}\tilde{t}$

$$\begin{aligned}(\Delta t_{ab})^2 &= a^2\underbrace{\frac{1}{E_\infty}\int_{-\infty}^{\infty}(\tilde{t}-\bar{t})^2|\psi(\tilde{t})|^2\mathrm{d}\tilde{t}}_{(\Delta t_\psi)^2}\\ &= a^2(\Delta t_\psi)^2\end{aligned} \tag{4.426}$$

where $(\Delta t_\psi)^2$ is the time dispersion of the mother wavelet.

The spectral bandwidth of $\psi_{ab}(t)$ can be computed from Equation (4.29). The Fourier transform of $\psi_{ab}(t)$ is given by expression (4.384) and its spectral centre of gravity was computed in Example 4.94:

$$\begin{aligned}(\Delta\omega_{ab})^2 &= \frac{1}{2\pi E_\infty}\int_{-\infty}^{\infty}(\omega - \overline{\omega}_{ab})^2|\Psi_{ab}(\omega)|^2\mathrm{d}\omega\\ &= \frac{1}{2\pi E_\infty}\int_{-\infty}^{\infty}\left(\omega - \frac{\overline{\omega}}{a}\right)^2 a|\Psi(a\omega)|^2\mathrm{d}\omega.\end{aligned} \tag{4.427}$$

In the last expression we introduce a new variable of integration, $\tilde{\omega} \equiv a\omega$ so that $\omega = \tilde{\omega}/a$ and $\mathrm{d}\omega = \mathrm{d}\tilde{\omega}/a$, and obtain

$$(\Delta\omega_{ab})^2 = \frac{1}{a^2}\underbrace{\frac{1}{2\pi E_\infty}\int_{-\infty}^{\infty}(\tilde{\omega}-\overline{\omega})^2|\Psi(\tilde{\omega})|^2\mathrm{d}\tilde{\omega}}_{(\Delta\omega_\psi)^2}$$

$$= \frac{1}{a^2}(\Delta\omega_\psi)^2 \tag{4.428}$$

where $(\Delta\omega_\psi)^2$ is the spectral bandwidth of the mother wavelet.

Combining the results of Equations (4.426) and (4.428) we obtain the joint uncertainty with which the wavelet function $\psi_{ab}(t)$ answers the question "what is where":

$$(\Delta t_{ab})^2(\Delta\omega_{ab})^2 = (\Delta t_\psi)^2(\Delta\omega_\psi)^2. \tag{4.429}$$

We observe that this uncertainty depends only on the mother wavelet and, therefore, it is the same for all functions $\psi_{ab}(t)$ irrespective of the values of a and b. So, if the mother wavelet minimises the uncertainty inequality, so does the wavelet function $\psi_{ab}(t)$. Note that the product of the time and frequency resolutions of all cells is the same, but scaled by factor a between time and frequency, proportionally and inversely proportionally, respectively. So, the scaling factor a allows one to trade off resolution in the time domain with resolution in the frequency domain and vice versa.

Example B4.95

The function plotted in Figure 4.92 is the second derivative of a Gaussian with changed sign:

$$\psi(t) = (1-t^2)\mathrm{e}^{-\frac{t^2}{2}}. \tag{4.430}$$

Show that its Fourier transform is given by:

$$\Psi(\omega) = \sqrt{2\pi}\omega^2\mathrm{e}^{-\frac{\omega^2}{2}}. \tag{4.431}$$

Let us consider the Gaussian function $g(t)$ given by Equation (4.39) computed for $\sigma = 1$:

$$g(t) = \mathrm{e}^{-\frac{t^2}{2}}. \tag{4.432}$$

In Example 3.99 it was shown that its Fourier transform is given by:

$$G(\omega) = \sqrt{2\pi}\mathrm{e}^{-\frac{\omega^2}{2}}. \tag{4.433}$$

The first derivative of function $g(t)$ is:

$$g'(t) = -t\mathrm{e}^{-\frac{t^2}{2}} \tag{4.434}$$

(Continued)

Example B4.95 (Continued)

According to Example 4.1 the Fourier transform of $g'(t)$ is $j\omega G(\omega)$. The second derivative of $g(t)$ is:

$$g''(t) = -e^{-\frac{t^2}{2}} + t^2 e^{-\frac{t^2}{2}} = -(1-t^2)e^{-\frac{t^2}{2}}. \tag{4.435}$$

If we apply once more the result of Example 4.1, we deduce that the Fourier transform of $g''(t)$ is $-\omega^2 G(\omega)$. Function $\psi(t)$ of Equation (4.430) is the negative of $g''(t)$, so its Fourier transform must be $\omega^2 G(\omega)$, which, upon substitution from (4.433), leads to (4.431).

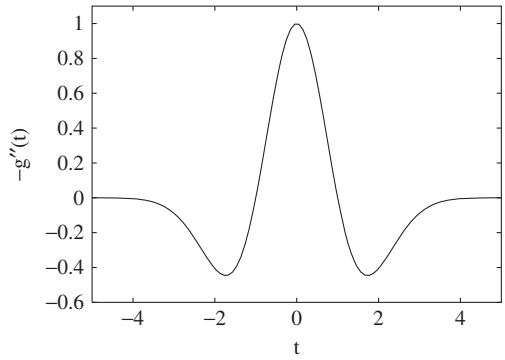


Figure 4.92 The negative of the second derivative of a Gaussian with $\sigma = 1$. In Example 4.96 it is shown that it may be used as a mother wavelet.

Example B4.96

Show that the function defined by Equation (4.430) has zero direct component and fulfils the admissibility condition so that it may be used as a mother wavelet.

The Fourier transform of this function is given by Equation (4.431). We observe that $\Psi(0) = 0$, so the function has zero direct component. We use Equation (4.431) in Equation (4.407), which is the admissibility condition:

$$\begin{aligned} C_\psi &= \int_0^\infty \frac{|\Psi(\omega)|^2}{\omega} d\omega = 2\pi \int_0^\infty \omega^3 e^{-\omega^2} d\omega \\ &= \pi \int_0^\infty \omega^2 e^{-\omega^2} d(\omega^2) = -\pi \int_0^\infty \omega^2 d(e^{-\omega^2}) \\ &= \underbrace{-\pi \omega^2 e^{-\omega^2}\Big|_0^\infty}_{=0} + \pi \int_0^\infty e^{-\omega^2} d(\omega^2) = -\pi e^{-\omega^2}\Big|_0^\infty = \pi. \end{aligned} \tag{4.436}$$

So, C_ψ is finite, and therefore function (4.430) fulfils the admissibility condition.

Example B4.97

Show that:

$$\int_{-\infty}^{\infty} x^4 e^{-x^2} dx = \frac{3}{4}\sqrt{\pi}. \tag{4.437}$$

We integrate by parts:

$$\int_{-\infty}^{\infty} x^4 e^{-x^2} dx = -\frac{1}{2}\int_{-\infty}^{\infty} x^3 d(e^{-x^2})$$
$$= \underbrace{-\frac{1}{2}x^3 e^{-x^2}\Big|_{-\infty}^{\infty}}_{=0} + \frac{1}{2}\int_{-\infty}^{\infty} e^{-x^2} 3x^2 dx$$
$$= \frac{3}{4}\sqrt{\pi}. \tag{4.438}$$

Here we made use of Equation (3.81).

Example B4.98

Compute the total energy of the mother wavelet given by Equation (4.430).

The total energy of the mother wavelet may be computed from Equation (4.416):

$$E_\infty = \int_{-\infty}^{\infty} |\psi(t)|^2 dt = \int_{-\infty}^{\infty} (1-t^2)^2 e^{-t^2} dt. \tag{4.439}$$

Therefore

$$E_\infty = \int_{-\infty}^{\infty} (1 + t^4 - 2t^2) e^{-t^2} dt = \sqrt{\pi} + \frac{3}{4}\sqrt{\pi} - \sqrt{\pi} = \frac{3}{4}\sqrt{\pi} \tag{4.440}$$

where we made use of the results of Examples 3.36, 3.37 and 4.97.

Example B4.99

Show that:

$$\int_{-\infty}^{\infty} x^6 e^{-x^2} dx = \frac{15}{8}\sqrt{\pi}. \tag{4.441}$$

We integrate by parts:

$$\int_{-\infty}^{\infty} x^6 e^{-x^2} dx = -\frac{1}{2}\int_{-\infty}^{\infty} x^5 d(e^{-x^2}) = \underbrace{-\frac{1}{2}x^5 e^{-x^2}\Big|_{-\infty}^{\infty}}_{=0} + \frac{1}{2}\int_{-\infty}^{\infty} e^{-x^2} 5x^4 dx = \frac{15}{8}\sqrt{\pi}. \tag{4.442}$$

Here we made use of Equation (4.437).

Example B4.100

Compute the time dispersion of the mother wavelet given by Equation (4.430).

This mother wavelet is a symmetric function, so according to Example 4.92 its centre of gravity is at $\bar{t} = 0$. For $\bar{t} = 0$ the time dispersion is given by Equation (4.28), which for this case has

(Continued)

Example B4.100 (Continued)

the form:

$$(\Delta t_\psi)^2 = \frac{1}{E_\infty}\int_{-\infty}^{\infty} t^2(1-t^2)^2 e^{-t^2} dt = \frac{1}{E_\infty}\int_{-\infty}^{\infty}(t^2+t^6-2t^4)e^{-t^2} dt. \tag{4.443}$$

If we make use of the results of Examples 4.98, 3.37, 4.97 and 4.99, we obtain:

$$(\Delta t_\psi)^2 = \frac{4}{3\sqrt{\pi}}\left(\frac{1}{2}\sqrt{\pi}+\frac{15}{8}\sqrt{\pi}-\frac{3}{2}\sqrt{\pi}\right) = \frac{7}{6}. \tag{4.444}$$

Example B4.101

Compute the spectral bandwidth of the mother wavelet given by Equation (4.430).
Since $\Psi(\omega)$ is a symmetric function, $\overline{\omega} = 0$. We use Equation (4.431) in the definition of the spectral bandwidth given by Equation (4.29) to write:

$$(\Delta\omega_\psi)^2 = \frac{1}{2\pi E_\infty}\int_{-\infty}^{\infty} 2\pi\omega^6 e^{-\omega^2} d\omega. \tag{4.445}$$

Making use of the results of Examples 4.98 and 4.99 we obtain:

$$(\Delta\omega_\psi)^2 = \frac{4}{3\sqrt{\pi}}\frac{15}{8}\sqrt{\pi} = \frac{5}{2}. \tag{4.446}$$

Example B4.102

Show that the wavelets produced by the mother wavelet of Equation (4.430) are sub-optimal in the joint resolution with which they cover the time-frequency domain.
The joint resolution of any wavelet produced by the mother wavelet $\psi(t)$ is the same as the joint resolution of the mother wavelet itself (see Equation (4.429)). Using the results of Examples 4.100 and 4.101, we compute the joint resolution of the mother wavelet as:

$$(\Delta t_\psi)^2(\Delta\omega_\psi)^2 = \frac{7}{6}\times\frac{5}{2} = \frac{35}{12} \simeq 3. \tag{4.447}$$

This number is much larger than the minimum value the joint resolution may take, which was shown to be $1/4$ (see Equation (4.38)). So, the set of wavelets produced by the mother wavelet of Equation (4.430) do not answer the question "what is where" with the maximum possible joint accuracy. However, the scaling parameter "a" allows us to construct wavelets that share the available joint resolution in a chosen way between the time and the frequency component.

Example B4.103

Produce a wavelet that has better time resolution than the Gabor function with $\sigma = 1$.

According to Equation (4.41) the Gabor function with $\sigma = 1$ has time dispersion $1/2$. If we use as the mother wavelet that of Equation (4.430) and produce a wavelet with scaling parameter a, its time dispersion, according to Equation (4.426) and the result of Example 4.100, will be $a^2 7/6$. Therefore, for its time resolution to be better than that of the Gabor function with $\sigma = 1$, we must have:

$$a^2 \frac{7}{6} < \frac{1}{2} \Rightarrow a < \sqrt{\frac{3}{7}} \simeq 0.65. \tag{4.448}$$

If, for example, we choose $a = 0.5$, the wavelet will have better time resolution than the corresponding Gabor function.

Example B4.104

Produce a wavelet that has better spectral resolution than the Gabor function with $\sigma = 1$.

According to Equation (4.42) the Gabor function with $\sigma = 1$ has a spectral bandwidth of $1/2$. If we use as the mother wavelet that of Equation (4.430) and produce a wavelet with scaling parameter a, its spectral bandwidth, according to Equation (4.428) and the result of Example 4.101, will be $5/(2a^2)$. Therefore, for its frequency resolution to be better than that of the Gabor function with $\sigma = 1$, we must have:

$$\frac{5}{2a^2} < \frac{1}{2} \Rightarrow a > \sqrt{5} \simeq 2.2. \tag{4.449}$$

If, for example, we choose $a = 4$, the wavelet will have better frequency resolution than the corresponding Gabor function.

Example B4.105

Use the mother wavelet of Equation (4.430) to produce the wavelet transform of the signal:

$$f(t) = \mathrm{e}^{-\frac{t^2}{2\sigma^2}}. \tag{4.450}$$

First, from the mother wavelet (4.430) we produce function $\psi_{ab}(t)$ using Equation (4.383):

$$\psi_{ab}(t) = \frac{1}{\sqrt{a}} \left[1 - \left(\frac{t-b}{a} \right)^2 \right] \mathrm{e}^{-\frac{(t-b)^2}{2a^2}}. \tag{4.451}$$

(Continued)

Example B4.105 (Continued)

Then we use $\psi_{ab}(t)$ and the definition of $f(t)$ into the definition of the wavelet transform given by Equation (4.387) to write:

$$w_f(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \left[1 - \left(\frac{t-b}{a}\right)^2\right] \mathrm{e}^{-\frac{(t-b)^2}{2a^2}} \mathrm{e}^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t. \tag{4.452}$$

To calculate this integral we must complete the square in the exponent:

$$\begin{aligned} w_f(a,b) = \\ \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \left[1 - \frac{b^2}{a^2} - \frac{t^2}{a^2} + \frac{2bt}{a^2}\right] \mathrm{e}^{-\frac{t^2}{2a^2} - \frac{b^2}{2a^2} + \frac{bt}{a^2} - \frac{t^2}{2\sigma^2}} \mathrm{d}t = \\ \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \left[1 - \frac{b^2}{a^2} - \frac{t^2}{a^2} + \frac{2bt}{a^2}\right] \mathrm{e}^{-t^2\left(\frac{1}{2a^2} + \frac{1}{2\sigma^2}\right) + \frac{bt}{a^2} - \frac{b^2}{2a^2}} \mathrm{d}t = \\ \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \left[1 - \frac{b^2}{a^2} - \frac{t^2}{a^2} + \frac{2bt}{a^2}\right] \times \\ \mathrm{e}^{-t^2 \frac{a^2+\sigma^2}{2a^2\sigma^2} + 2\left(t\frac{\sqrt{a^2+\sigma^2}}{\sqrt{2a\sigma}}\right)\left(\frac{\sqrt{2a\sigma}}{2\sqrt{a^2+\sigma^2}} \frac{b}{a^2}\right) - \left(\frac{\sqrt{2a\sigma}}{2\sqrt{a^2+\sigma^2}} \frac{b}{a^2}\right)^2 + \left(\frac{\sqrt{2a\sigma}}{2\sqrt{a^2+\sigma^2}} \frac{b}{a^2}\right)^2 - \frac{b^2}{2a^2}} \mathrm{d}t = \\ \frac{1}{\sqrt{a}} \mathrm{e}^{\left(\frac{b\sigma}{\sqrt{2a}\sqrt{a^2+\sigma^2}}\right)^2 - \frac{b^2}{2a^2}} \int_{-\infty}^{\infty} \left[1 - \frac{b^2}{a^2} - \frac{t^2}{a^2} + \frac{2bt}{a^2}\right] \mathrm{e}^{-\left(t\frac{\sqrt{a^2+\sigma^2}}{\sqrt{2a\sigma}} - \frac{b\sigma}{\sqrt{2a}\sqrt{a^2+\sigma^2}}\right)^2} \mathrm{d}t. \end{aligned} \tag{4.453}$$

We define a new variable of integration

$$\tilde{t} \equiv \frac{\sqrt{a^2+\sigma^2}}{\sqrt{2a\sigma}} t - \frac{b\sigma}{\sqrt{2a}\sqrt{a^2+\sigma^2}} \Rightarrow t = \frac{\sqrt{2a\sigma}}{\sqrt{a^2+\sigma^2}} \tilde{t} + \frac{\sigma^2 b}{a^2+\sigma^2}. \tag{4.454}$$

When substituting t in the square bracket, we note that all terms with odd powers of $\tilde{t}$ will yield zero upon integration with $\mathrm{e}^{-\tilde{t}^2}$, as odd integrands integrated over a symmetric range of integration. So, for simplicity, we omit them in the formulae that follow:

$$\begin{aligned} w_f(a,b) = \\ \frac{1}{\sqrt{a}} \mathrm{e}^{-\frac{b^2}{2(a^2+\sigma^2)}} \frac{\sqrt{2a\sigma}}{\sqrt{a^2+\sigma^2}} \times \\ \int_{-\infty}^{\infty} \left[1 - \frac{b^2}{a^2} - \frac{1}{a^2}\left(\frac{\sqrt{2a\sigma}}{\sqrt{a^2+\sigma^2}}\right)^2 \tilde{t}^2 - \frac{1}{a^2}\left(\frac{\sigma^2 b}{a^2+\sigma^2}\right)^2 + \frac{2b}{a^2}\frac{\sigma^2 b}{a^2+\sigma^2}\right] \mathrm{e}^{-\tilde{t}^2} \mathrm{d}\tilde{t} = \\ \mathrm{e}^{-\frac{b^2}{2(a^2+\sigma^2)}} \frac{\sqrt{2a\sigma}}{\sqrt{a^2+\sigma^2}} \int_{-\infty}^{\infty} \left[1 - \frac{b^2 a^2}{(a^2+\sigma^2)^2} - \frac{2\sigma^2}{a^2+\sigma^2}\tilde{t}^2\right] \mathrm{e}^{-\tilde{t}^2} \mathrm{d}\tilde{t}. \end{aligned} \tag{4.455}$$

To compute these integrals we make use of the results of Examples 3.36, 3.37 and 4.97:

$$\begin{aligned} w_f(a,b) &= \mathrm{e}^{-\frac{b^2}{2(a^2+\sigma^2)}} \frac{\sqrt{2a\sigma}}{\sqrt{a^2+\sigma^2}} \sqrt{\pi} \left[1 - \frac{b^2 a^2}{(a^2+\sigma^2)^2} - \frac{2\sigma^2}{a^2+\sigma^2}\frac{1}{2}\right] \\ &= \mathrm{e}^{-\frac{b^2}{2(a^2+\sigma^2)}} \frac{\sqrt{2\pi a\sigma}a^2}{(a^2+\sigma^2)^{3/2}} \left[1 - \frac{b^2}{a^2+\sigma^2}\right]. \end{aligned} \tag{4.456}$$

This is the wavelet transform of $f(t)$ given by Equation (4.450).

Example B4.106

Recover the original signal $f(t)$ starting from its wavelet transform given by Equation (4.456).

To recover the original signal $f(t)$ we apply the inverse wavelet transform given by Equation (4.408) in conjunction with Equation (4.451) for $\psi_{ab}(t)$, and make use of Equation (4.436) for the value of C_ψ for this particular wavelet:

$$\begin{aligned} f(t) &= \frac{1}{\pi}\int_0^\infty\int_{-\infty}^\infty \frac{1}{a^2}\frac{1}{\sqrt{a}}\left[1-\left(\frac{t-b}{a}\right)^2\right]\mathrm{e}^{-\frac{(t-b)^2}{2a^2}}\times \\ &\quad \mathrm{e}^{-\frac{b^2}{2(a^2+\sigma^2)}}\frac{\sqrt{2\pi a}\sigma a^2}{(a^2+\sigma^2)^{3/2}}\left[1-\frac{b^2}{a^2+\sigma^2}\right]\mathrm{d}b\mathrm{d}a \\ &= \frac{\sqrt{2}\sigma}{\sqrt{\pi}}\int_0^\infty\int_{-\infty}^\infty\left[1-\left(\frac{t-b}{a}\right)^2\right]\left[1-\frac{b^2}{a^2+\sigma^2}\right]\frac{1}{(a^2+\sigma^2)^{3/2}}\times \\ &\quad \mathrm{e}^{-\frac{(t-b)^2}{2a^2}-\frac{b^2}{2(a^2+\sigma^2)}}\mathrm{d}b\mathrm{d}a. \end{aligned} \tag{4.457}$$

We define an auxiliary variable $s^2 \equiv a^2+\sigma^2$, so that the above expression becomes:

$$\begin{aligned} f(t) &= \frac{\sqrt{2}\sigma}{\sqrt{\pi}}\int_0^\infty\int_{-\infty}^\infty\left[1-\left(\frac{t-b}{a}\right)^2\right]\left[1-\frac{b^2}{s^2}\right]\frac{1}{s^3}\mathrm{e}^{-\frac{(t-b)^2}{2a^2}-\frac{b^2}{2s^2}}\mathrm{d}b\mathrm{d}a \\ &= \frac{\sqrt{2}\sigma}{\sqrt{\pi}}\int_0^\infty\int_{-\infty}^\infty\left[1-\frac{t^2}{a^2}-\frac{b^2}{a^2}+\frac{2bt}{a^2}\right]\left[1-\frac{b^2}{s^2}\right]\frac{1}{s^3}\mathrm{e}^{-\frac{(t-b)^2}{2a^2}-\frac{b^2}{2s^2}}\mathrm{d}b\mathrm{d}a \\ &= \frac{\sqrt{2}\sigma}{\sqrt{\pi}}\int_0^\infty\int_{-\infty}^\infty\left[1-\frac{t^2}{a^2}-\frac{b^2}{a^2}+\frac{2bt}{a^2}-\frac{b^2}{s^2}+\frac{t^2b^2}{a^2s^2}+\frac{b^4}{a^2s^2}-\frac{2tb^3}{a^2s^2}\right]\times \\ &\quad \frac{1}{s^3}\mathrm{e}^{-\frac{(t-b)^2}{2a^2}-\frac{b^2}{2s^2}}\mathrm{d}b\mathrm{d}a. \end{aligned} \tag{4.458}$$

Let us perform first the integration over b. We observe that in order to compute the integral we must complete first the square in the exponent. We notice that the exponent is identical to that of Equation (4.453) provided we exchange the roles of b and t and we identify s with σ. So, we may write

$$\begin{aligned} f(t) &= \frac{\sqrt{2}\sigma}{\sqrt{\pi}}\int_0^\infty\int_{-\infty}^\infty\left[1-\frac{t^2}{a^2}-\frac{b^2}{a^2}+\frac{2bt}{a^2}-\frac{b^2}{s^2}+\frac{t^2b^2}{a^2s^2}+\frac{b^4}{a^2s^2}-\frac{2tb^3}{a^2s^2}\right]\times \\ &\quad \frac{1}{s^3}\mathrm{e}^{-\tilde{b}^2}\frac{\sqrt{2}as}{\sqrt{a^2+s^2}}\mathrm{d}\tilde{b}\mathrm{e}^{\left(\frac{ts}{\sqrt{2}a\sqrt{a^2+s^2}}\right)^2-\frac{t^2}{2a^2}}\mathrm{d}a \end{aligned} \tag{4.459}$$

where we defined:

$$\tilde{b} \equiv \frac{\sqrt{a^2+s^2}}{\sqrt{2}as}b - \frac{ts}{\sqrt{2}a\sqrt{a^2+s^2}} \Rightarrow b = \frac{\sqrt{2}as}{\sqrt{a^2+s^2}}\tilde{b} + \frac{s^2t}{a^2+s^2}. \tag{4.460}$$

(Continued)

Example B4.106 (Continued)

When substituting b in the square bracket, we note that all terms with odd powers of $\tilde{b}$ will yield zero upon integration with $e^{-\tilde{b}^2}$, as odd integrands integrated over a symmetric range of integration. So, for simplicity, we omit them in the formulae that follow:

$$
\begin{aligned}
f(t) = \frac{2\sigma}{\sqrt{\pi}} \int_0^\infty \frac{1}{s^3} \frac{as}{\sqrt{a^2+s^2}} e^{-\frac{t^2}{2(a^2+s^2)}} \\
\int_{-\infty}^{\infty} \Bigg[1 - \frac{t^2}{a^2} + \frac{t^2-s^2-a^2}{a^2 s^2} \left(\frac{\sqrt{2}as}{\sqrt{a^2+s^2}} \right)^2 \tilde{b}^2 + \frac{t^2-s^2-a^2}{a^2 s^2} \left(\frac{s^2 t}{a^2+s^2} \right)^2 \\
+ \frac{2t}{a^2} \frac{s^2 t}{a^2+s^2} + \frac{1}{a^2 s^2} \left(\frac{\sqrt{2}as}{\sqrt{a^2+s^2}} \right)^4 \tilde{b}^4 \\
+ \frac{1}{a^2 s^2} 6 \left(\frac{\sqrt{2}as}{\sqrt{a^2+s^2}} \right)^2 \left(\frac{s^2 t}{a^2+s^2} \right)^2 \tilde{b}^2 + \frac{1}{a^2 s^2} \left(\frac{s^2 t}{a^2+s^2} \right)^4 \\
- \frac{2t}{a^2 s^2} 3 \left(\frac{\sqrt{2}as}{\sqrt{a^2+s^2}} \right)^2 \frac{s^2 t}{a^2+s^2} \tilde{b}^2 - \frac{2t}{a^2 s^2} \left(\frac{s^2 t}{a^2+s^2} \right)^3 \Bigg] e^{-\tilde{b}^2} \mathrm{d}\tilde{b} \mathrm{d}a. \qquad (4.461)
\end{aligned}
$$

We make use of the results of Examples 3.36 and 3.37, and of Equation (4.437) to obtain:

$$
\begin{aligned}
f(t) = \frac{2\sigma}{\sqrt{\pi}} \int_0^\infty \frac{a}{s^2 \sqrt{a^2+s^2}} e^{-\frac{t^2}{2(a^2+s^2)}} \times \\
\sqrt{\pi} \Bigg[1 - \frac{t^2}{a^2} \\
+ \frac{t^2-s^2-a^2}{a^2 s^2} \left(\frac{\sqrt{2}as}{\sqrt{a^2+s^2}} \right)^2 \frac{1}{2} + \frac{t^2-s^2-a^2}{a^2 s^2} \left(\frac{s^2 t}{a^2+s^2} \right)^2 \\
+ \frac{2t}{a^2} \frac{s^2 t}{a^2+s^2} + \frac{1}{a^2 s^2} \left(\frac{\sqrt{2}as}{\sqrt{a^2+s^2}} \right)^4 \frac{3}{4} \\
+ \frac{1}{a^2 s^2} 6 \left(\frac{\sqrt{2}as}{\sqrt{a^2+s^2}} \right)^2 \left(\frac{s^2 t}{a^2+s^2} \right)^2 \frac{1}{2} + \frac{1}{a^2 s^2} \left(\frac{s^2 t}{a^2+s^2} \right)^4 \\
- \frac{2t}{a^2 s^2} 3 \left(\frac{\sqrt{2}as}{\sqrt{a^2+s^2}} \right)^2 \frac{s^2 t}{a^2+s^2} \frac{1}{2} - \frac{2t}{a^2 s^2} \left(\frac{s^2 t}{a^2+s^2} \right)^3 \Bigg] \mathrm{d}a \\
= 2\sigma \int_0^\infty \frac{a}{s^2 \sqrt{a^2+s^2}} e^{-\frac{t^2}{2(a^2+s^2)}} \times \\
\left[\frac{3s^2 a^2}{(a^2+s^2)^2} - \frac{6s^2 a^2 t^2}{(a^2+s^2)^3} + \frac{s^2 a^2 t^4}{(a^2+s^2)^4} \right] \mathrm{d}a \\
= 2\sigma \int_0^\infty \frac{a^3}{(a^2+s^2)^{5/2}} e^{-\frac{t^2}{2(a^2+s^2)}} \left[3 - \frac{6t^2}{a^2+s^2} + \frac{t^4}{(a^2+s^2)^2} \right] \mathrm{d}a. \qquad (4.462)
\end{aligned}
$$

We define a new variable of integration $x > 0$, as:

$$
x^2 \equiv \frac{1}{a^2+s^2} = \frac{1}{2a^2+\sigma^2} \Rightarrow 2a^2+\sigma^2 = \frac{1}{x^2} \Rightarrow 4ada = -2\frac{1}{x^3}\mathrm{d}x \Rightarrow ada = -\frac{1}{2x^3}\mathrm{d}x. \quad (4.463)
$$

The limits of x are from $1/\sigma$ for $a = 0$, to 0 for $a \to \infty$. Then:

$$\begin{aligned} f(t) &= -2\sigma \int_{1/\sigma}^{0} \left(\frac{1}{2x^2} - \frac{\sigma^2}{2} \right) x^5 \mathrm{e}^{-\frac{t^2x^2}{2}} [3 - 6t^2x^2 + t^4x^4] \frac{1}{2x^3} \mathrm{d}x \\ &= \frac{1}{2}\sigma \int_{0}^{1/\sigma} (1 - x^2\sigma^2) \mathrm{e}^{-\frac{t^2x^2}{2}} [3 - 6t^2x^2 + t^4x^4] \mathrm{d}x \\ &= \frac{\sigma}{2} \int_{0}^{1/\sigma} (3 - 6t^2x^2 + t^4x^4 - 3x^2\sigma^2 + 6t^2\sigma^2x^4 - \sigma^2t^4x^6) \mathrm{e}^{-\frac{t^2x^2}{2}} \mathrm{d}x. \end{aligned} \tag{4.464}$$

Let us define a new variable $y \equiv tx/\sqrt{2} \Rightarrow x = \sqrt{2}y/t$ and $\mathrm{d}x = \sqrt{2}\mathrm{d}y/t$. Upon substitution in the above expression we obtain

$$\begin{aligned} f(t) &= \frac{\sigma}{\sqrt{2}t} \int_{0}^{t/(\sqrt{2}\sigma)} \left(3 - 12y^2 + 4y^4 - 6\frac{\sigma^2}{t^2}y^2 + 24\frac{\sigma^2}{t^2}y^4 - 8\frac{\sigma^2}{t^2}y^6 \right) \mathrm{e}^{-y^2} \mathrm{d}y \\ &= \frac{3\sigma}{\sqrt{2}t} \mathrm{erf}\left(\frac{t}{\sqrt{2}\sigma} \right) \frac{\sqrt{\pi}}{2} \\ &\quad + \frac{\sigma}{\sqrt{2}t} \left[-12 - 6\frac{\sigma^2}{t^2} \right] \int_{0}^{t/(\sqrt{2}\sigma)} y^2 \mathrm{e}^{-y^2} \mathrm{d}y \\ &\quad + \frac{\sigma}{\sqrt{2}t} \left[4 + 24\frac{\sigma^2}{t^2} \right] \int_{0}^{t/(\sqrt{2}\sigma)} y^4 \mathrm{e}^{-y^2} \mathrm{d}y \\ &\quad - \frac{\sigma}{\sqrt{2}t} 8\frac{\sigma^2}{t^2} \int_{0}^{t/(\sqrt{2}\sigma)} y^6 \mathrm{e}^{-y^2} \mathrm{d}y \end{aligned} \tag{4.465}$$

where we have made use of the definition of the error function:

$$\mathrm{erf}(z) \equiv \frac{2}{\sqrt{\pi}} \int_{0}^{z} \mathrm{e}^{-y^2} \mathrm{d}y \tag{4.466}$$

Next we compute the integrals we need:

$$\begin{aligned} \int_{0}^{z} y^2 \mathrm{e}^{-y^2} \mathrm{d}y &= -\frac{1}{2} \int_{0}^{z} y d(\mathrm{e}^{-y^2}) \\ &= -\frac{1}{2} y \mathrm{e}^{-y^2} \Big|_{0}^{z} + \frac{1}{2} \int_{0}^{z} \mathrm{e}^{-y^2} \mathrm{d}y = -\frac{z}{2} \mathrm{e}^{-z^2} + \frac{1}{2} \frac{\sqrt{\pi}}{2} \mathrm{erf}(z) \end{aligned} \tag{4.467}$$

$$\begin{aligned} \int_{0}^{z} y^4 \mathrm{e}^{-y^2} \mathrm{d}y &= -\frac{1}{2} \int_{0}^{z} y^3 d(\mathrm{e}^{-y^2}) \\ &= -\frac{1}{2} y^3 \mathrm{e}^{-y^2} \Big|_{0}^{z} + \frac{3}{2} \int_{0}^{z} y^2 \mathrm{e}^{-y^2} \mathrm{d}y \\ &= -\frac{z^3}{2} \mathrm{e}^{-z^2} + \frac{3}{2} \left[-\frac{z}{2} \mathrm{e}^{-z^2} + \frac{1}{2} \frac{\sqrt{\pi}}{2} \mathrm{erf}(z) \right] \\ &= -\frac{z^3}{2} \mathrm{e}^{-z^2} + \frac{3z}{4} \mathrm{e}^{-z^2} + \frac{3\sqrt{\pi}}{8} \mathrm{erf}(z). \end{aligned} \tag{4.468}$$

(Continued)

Example B4.106 (Continued)

Here we made use of Equation (4.467).

$$\begin{aligned}\int_0^z y^6 e^{-y^2} dy &= -\frac{1}{2}\int_0^z y^5 d(e^{-y^2}) \\ &= -\frac{1}{2}y^5 e^{-y^2}\Big|_0^z + \frac{5}{2}\int_0^z y^4 e^{-y^2} dy \\ &= -\frac{z^5}{2}e^{-z^2} + \frac{5}{2}\left[-\frac{z^3}{2}e^{-z^2} + \frac{3z}{4}e^{-z^2} + \frac{3\sqrt{\pi}}{8}\text{erf}(z)\right] \\ &= -\frac{z^5}{2}e^{-z^2} - \frac{5z^3}{4}e^{-z^2} - \frac{15z}{8}e^{-z^2} + \frac{15\sqrt{\pi}}{16}\text{erf}(z). \end{aligned} \tag{4.469}$$

Here we made use of Equation (4.468). If we substitute now from Equations (4.467), (4.468) and (4.469) with $z = t/(\sqrt{2}\sigma)$ into Equation (4.465), we obtain:

$$\begin{aligned} f(t) = {} & \frac{3\sqrt{\pi}\sigma}{2\sqrt{2}t}\text{erf}\left(\frac{t}{\sqrt{2}\sigma}\right) \\ & -\frac{12\sigma}{\sqrt{2}t}\left[-\frac{t}{2\sqrt{2}\sigma}e^{-\frac{t^2}{2\sigma^2}} + \frac{\sqrt{\pi}}{4}\text{erf}\left(\frac{t}{\sqrt{2}\sigma}\right)\right] \\ & -\frac{6\sigma^3}{\sqrt{2}t^3}\left[-\frac{t}{2\sqrt{2}\sigma}e^{-\frac{t^2}{2\sigma^2}} + \frac{\sqrt{\pi}}{4}\text{erf}\left(\frac{t}{\sqrt{2}\sigma}\right)\right] \\ & +\frac{4\sigma}{\sqrt{2}t}\left[-\frac{t^3}{4\sqrt{2}\sigma^3}e^{-\frac{t^2}{2\sigma^2}} - \frac{3t}{4\sqrt{2}\sigma}e^{-\frac{t^2}{2\sigma^2}} + \frac{3\sqrt{\pi}}{8}\text{erf}\left(\frac{t}{\sqrt{2}\sigma}\right)\right] \\ & +\frac{24\sigma^3}{\sqrt{2}t^3}\left[-\frac{t^3}{4\sqrt{2}\sigma^3}e^{-\frac{t^2}{2\sigma^2}} - \frac{3t}{4\sqrt{2}\sigma}e^{-\frac{t^2}{2\sigma^2}} + \frac{3\sqrt{\pi}}{8}\text{erf}\left(\frac{t}{\sqrt{2}\sigma}\right)\right] \\ & -\frac{8\sigma^3}{\sqrt{2}t^3}\left[-\frac{t^5}{8\sqrt{2}\sigma^5}e^{-\frac{t^2}{2\sigma^2}} - \frac{5t^3}{8\sqrt{2}\sigma^3}e^{-\frac{t^2}{2\sigma^2}} - \frac{15t}{8\sqrt{2}\sigma}e^{-\frac{t^2}{2\sigma^2}} + \frac{15\sqrt{\pi}}{16}\text{erf}\left(\frac{t}{\sqrt{2}\sigma}\right)\right] \\ = {} & \text{erf}\left(\frac{t}{\sqrt{2}\sigma}\right)\left[\frac{3\sqrt{\pi}\sigma}{2\sqrt{2}t} - \frac{12\sigma\sqrt{\pi}}{\sqrt{2}t4} - \frac{6\sigma^3\sqrt{\pi}}{\sqrt{2}t^3 4} + \frac{4\sigma 3\sqrt{\pi}}{\sqrt{2}t8} + \frac{24\sigma^3 3\sqrt{\pi}}{\sqrt{2}t^3 8} - \frac{8\sigma^3 15\sqrt{\pi}}{\sqrt{2}t^3 16}\right] \\ & +e^{-\frac{t^2}{2\sigma^2}}\left[\frac{6\sigma t}{\sqrt{2}t\sqrt{2}\sigma} + \frac{3\sigma^3 t}{\sqrt{2}t^3\sqrt{2}\sigma} - \frac{\sigma t^3}{\sqrt{2}t\sqrt{2}\sigma^3} - \frac{3\sigma t}{\sqrt{2}t\sqrt{2}\sigma} - \frac{6\sigma^3 t^3}{\sqrt{2}t^3\sqrt{2}\sigma^3}\right. \\ & \left. -\frac{18\sigma^3 t}{\sqrt{2}t^3\sqrt{2}\sigma} + \frac{\sigma^3 t^5}{\sqrt{2}t^3\sqrt{2}\sigma^5} + \frac{5\sigma^3 t^3}{\sqrt{2}t^3\sqrt{2}\sigma^3} + \frac{15\sigma^3 t}{\sqrt{2}t^3\sqrt{2}\sigma}\right] \\ = {} & e^{-\frac{t^2}{2\sigma^2}}\left[3 + \frac{3\sigma^2}{2t^2} - \frac{t^2}{2\sigma^2} - \frac{3}{2} - 3 - \frac{9\sigma^2}{t^2} + \frac{t^2}{2\sigma^2} + \frac{5}{2} + \frac{15\sigma^2}{2t^2}\right] \\ = {} & e^{-\frac{t^2}{2\sigma^2}}. \end{aligned} \tag{4.470}$$

So, after all this algebra, we recovered the original signal from its wavelet transform. This example demonstrated that the continuous wavelet transform performed with mother wavelet (4.430) is invertible.

How is the wavelet transform adapted for digital signals?

The continuous wavelet transform is not very useful in practical applications. That is why a **discrete version** of it had to be introduced. To discretise the wavelet transform, we have to discretise its two parameters, namely a and b. It is **conventional** to let a take values 2^l and thus create the so-called **dyadic wavelets**. Note that a is a scaling parameter, so it does not have any units. On the contrary, b is a shifting parameter, so it must be measured in the same units as the independent variable t. Since we are going to sample the wavelet, we must choose first the sampling step $\Delta\tau$. We agree to sample each wavelet at the integer values of the independent variable t, so that $\Delta\tau = 1$. Further, we observe that for the argument of the scaled and shifted wavelet $\psi_{ab}((t-b)/a)$ to remain integer, b must be an integer multiple of a, i.e. $b = (k-1)a = (k-1)2^l$ for $k = 1, 2, \ldots$. These wavelets are identified by indices l and k, and they are denoted by ψ_{lk}. In addition, since the time variable t is sampled at discrete points, all integrals used to compute the wavelet coefficients of a discrete signal are replaced by sums (see Example 4.107).

Example 4.107

Use the Haar function defined as

$$h(t) = \begin{cases} -A & \text{for } 0 \le t < \frac{1}{2} \\ A & \text{for } \frac{1}{2} \le t < 1 \end{cases} \tag{4.471}$$

to construct a wavelet basis for the analysis of an eight-sample long signal. In the above definition, A is a positive constant the value of which must be chosen so that the wavelet is normalised, i.e. its total energy is 1.

First we choose the value of A. The total energy of the mother wavelet is:

$$\int_0^1 h(t)^2 \mathrm{d}t = (-A)^2 \int_0^{\frac{1}{2}} \mathrm{d}t + A^2 \int_{\frac{1}{2}}^1 dt = A^2 = 1 \Rightarrow A = 1. \tag{4.472}$$

For $l = 1$, i.e. $a = 2^1 = 2$, function $h(t/a)$ is:

$$h\left(\frac{t}{2}\right) = \begin{cases} -1 & \text{for } 0 \le t < 1 \\ 1 & \text{for } 1 \le t < 2 \end{cases}. \tag{4.473}$$

There are only two integer values in the definition range of the wavelet, so the simplest wavelet we may construct is:

$$\boldsymbol{H} = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right). \tag{4.474}$$

Note that the $1/\sqrt{2}$ factor corresponds to factor $1/\sqrt{a}$ of Equation (4.383) and it is there in order to make the wavelet have energy 1, i.e. the sum of the squares of its values to be 1.

For an eight-sample signal, the time axis consists of eight points, and this wavelet has to be used to cover this axis by shifts that are multiples of 2, i.e. at positions $(k-1)2$ for $k = 1, 2, 3, 4$. So, for the first wavelet vector, the filter function is placed at the first position (zero shift) and it is:

$$\boldsymbol{\psi_{11}} = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0, 0\right). \tag{4.475}$$

(Continued)

Example 4.107 (Continued)

The remaining three wavelets at this scale are:

$$\begin{aligned}
\boldsymbol{\psi}_{12} &= \left(0,0,-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0,0,0\right)\\
\boldsymbol{\psi}_{13} &= \left(0,0,0,0,-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0\right)\\
\boldsymbol{\psi}_{14} &= \left(0,0,0,0,0,0,-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right).
\end{aligned} \tag{4.476}$$

At scale $l = 2$, i.e. $a = 2^2 = 4$, the scaled mother wavelet is:

$$h\left(\frac{t}{4}\right) = \begin{cases} -1 & \text{for } 0 \le t < 2 \\ 1 & \text{for } 2 \le t < 4 \end{cases}. \tag{4.477}$$

There are only four integer values in the definition range of the wavelet, so the simplest wavelet we may construct is:

$$\tilde{\mathbf{H}} = \left(-\frac{1}{2},-\frac{1}{2},\frac{1}{2},\frac{1}{2}\right). \tag{4.478}$$

This wavelet may be placed so that it starts with shift 0 or 4, corresponding to values of $k = 1$ and $k = 2$, to produce:

$$\begin{aligned}
\boldsymbol{\psi}_{21} &= \left(-\frac{1}{2},-\frac{1}{2},\frac{1}{2},\frac{1}{2},0,0,0,0\right)\\
\boldsymbol{\psi}_{22} &= \left(0,0,0,0,-\frac{1}{2},-\frac{1}{2},\frac{1}{2},\frac{1}{2}\right).
\end{aligned} \tag{4.479}$$

For $l = 3$, i.e. $a = 2^3 = 8$, the scaled mother wavelet is:

$$h\left(\frac{t}{8}\right) = \begin{cases} -1 & \text{for } 0 \le t < 4 \\ 1 & \text{for } 4 \le t < 8 \end{cases}. \tag{4.480}$$

The normalised and discretised version of this wavelet is eight samples long, so there is no need to shift it to cover the time axis, since it already covers it fully:

$$\boldsymbol{\psi}_{31} = \left(-\frac{1}{\sqrt{8}},-\frac{1}{\sqrt{8}},-\frac{1}{\sqrt{8}},-\frac{1}{\sqrt{8}},\frac{1}{\sqrt{8}},\frac{1}{\sqrt{8}},\frac{1}{\sqrt{8}},\frac{1}{\sqrt{8}}\right). \tag{4.481}$$

Example 4.108

Assume that we have a sequence $(x_1, x_2, \ldots, x_8)$. Use the wavelets we constructed in Example 4.107 to expand this signal in the wavelet basis. Can you recover the original sequence if you know only these wavelet coefficients?

To calculate the wavelet coefficients of the digital signal we must project it onto each wavelet in turn, by taking the dot product between the signal and the wavelet. If we write the wavelets as the rows of a matrix that operates on the signal written as a column vector, the output vector will

consist of the wavelet coefficients of the signal:

$$\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{21} \\ w_{22} \\ w_{31} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}. \tag{4.482}$$

We deduce that:

$$\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{21} \\ w_{22} \\ w_{31} \end{bmatrix} = \begin{bmatrix} \frac{-x_1+x_2}{\sqrt{2}} \\ \frac{-x_3+x_4}{\sqrt{2}} \\ \frac{-x_5+x_6}{\sqrt{2}} \\ \frac{-x_7+x_8}{\sqrt{2}} \\ \frac{-x_1-x_2+x_3+x_4}{2} \\ \frac{-x_5-x_6+x_7+x_8}{2} \\ \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{\sqrt{8}} \end{bmatrix}. \tag{4.483}$$

We notice that the wavelet coefficients are seven linear combinations of the original values of the input signal. To recover eight values, we need eight independent linear equations, i.e. eight pieces of information. So, from the wavelet coefficients alone we cannot reconstruct the original signal. This is because the set of wavelets by themselves do not constitute a **complete** *basis for the eight-dimensional space of an eight-sample digital signal: they are all high pass filters.*

How do we compute the wavelet coefficients in practice?

The coefficients of the expansion of the signal in terms of wavelets are obtained by the projection of the signal on each wavelet vector in turn (see Example 4.108). The wavelet vectors at a fixed scale are created by placing a basic **filter**, the discrete version of the scaled mother wavelet, at successive positions shifted by a fixed number with respect to each other. For example, for the first scale, this operation can be expressed as the cross-correlation between the signal and the wavelet filter (given by Equation (4.474) in Example 4.107)

$$s(\tau) \equiv \sum_i \mathbf{H}(t_i) f(t_i + \tau) \tag{4.484}$$

and allowing τ to take only values $0, 2, 4, \ldots$. Alternatively, we allow τ to take all its possible values and then we remove from the output the values at the intermediate locations where they are not needed (see Example 4.110).

Many people refer to these operations as "convolution followed by decimation". This is wrong. The operation expressed by (4.484) is not a convolution. If it were a convolution, factor $f(t_i + \tau)$

would have been replaced by factor $f(\tau - t_i)$. That is, convolution reverses one of the functions before multiplying it point by point with the other function and summing the products. At best, we may say that we convolve the signal with the **reverse** of the wavelet filter, to counter-balance this intrinsic reversal a convolution imposes. For clarity, in the rest of this section we shall call this convolution **pseudo-convolution**, to make sure we do not confuse it with convolution proper.

For the dyadic wavelets, where the shifting is done in powers of 2, the sub-sampling is also done in powers of 2 (see Example 4.110).

Example B4.109

Show that the Fourier transform of the cross-correlation of two real functions is equal to the product of the Fourier transform of one of the functions times the complex conjugate of the Fourier transform of the other function.

Let us consider functions $f(t)$ and $h(t)$, with Fourier transforms $F(\omega)$ and $H(\omega)$, respectively. Their cross-correlation is:

$$s(\tau) \equiv \int_{-\infty}^{\infty} h(t)f(t+\tau)dt. \tag{4.485}$$

Taking the Fourier transform of both sides, we obtain:

$$\begin{aligned} S(\omega) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(t)f(t+\tau)\mathrm{e}^{-\mathrm{j}\omega\tau}\mathrm{d}t\mathrm{d}\tau \\ &= \int_{-\infty}^{\infty} h(t)\left\{\int_{-\infty}^{\infty} f(t+\tau)\mathrm{e}^{-\mathrm{j}\omega\tau}\mathrm{d}\tau\right\}\mathrm{d}t \\ &= \int_{-\infty}^{\infty} h(t)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}tF(\omega) \\ &= F(\omega)H^*(\omega). \end{aligned} \tag{4.486}$$

Example 4.110

Produce the Haar wavelet coefficients for level $l = 1$ of the signal of Example 4.108 by using pseudo-convolution and sub-sampling of the result.

We note that the filter function used to produce wavelets ψ_{1k} is given by Equation (4.474). If we pseudo-convolve the signal of Example 4.108 with this filter, we obtain:

$$\begin{bmatrix} \frac{-x_1+x_2}{\sqrt{2}} \\ \frac{-x_2+x_3}{\sqrt{2}} \\ \frac{-x_3+x_4}{\sqrt{2}} \\ \frac{-x_4+x_5}{\sqrt{2}} \\ \frac{-x_5+x_6}{\sqrt{2}} \\ \frac{-x_6+x_7}{\sqrt{2}} \\ \frac{-x_7+x_8}{\sqrt{2}} \end{bmatrix}. \tag{4.487}$$

This is equivalent of having projected the original signal on vectors:

$$\left(-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0,0,0,0,0\right)$$
$$\left(0,-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0,0,0,0\right)$$
$$\left(0,0,-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0,0,0\right)$$
$$\left(0,0,0,-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0,0\right)$$
$$\left(0,0,0,0,-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0\right)$$
$$\left(0,0,0,0,0,-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0\right)$$
$$\left(0,0,0,0,0,0,-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right). \tag{4.488}$$

We observe that only the first, third, fifth and last of these vectors are the wavelet vectors of scale $l=1$. So, if we keep only every other sample in output (4.487), we shall have the coefficients of the expansion of the signal on the wavelet basis at this scale:

$$\begin{bmatrix} \frac{-x_1+x_2}{\sqrt{2}} \\ \frac{-x_3+x_4}{\sqrt{2}} \\ \frac{-x_5+x_6}{\sqrt{2}} \\ \frac{-x_7+x_8}{\sqrt{2}} \end{bmatrix}. \tag{4.489}$$

Example 4.111

Show that the seven wavelets constructed in Example 4.107 constitute an orthonormal basis.

The wavelets in the discrete domain are actually vectors, the same size as the signal we wish to analyse. To show that they constitute an orthonormal basis, it is enough to show that the dot product of any two of them is 0, and the magnitude of any one of them is 1. From their definitions in Example 4.107 this is obviously true.

Example 4.112

The seven wavelets constructed in Example 4.107 constitute an orthonormal basis for the eight-dimensional space of any digital signal that consists of eight samples. This basis is obviously incomplete. Define an extra vector which, when considered

(Continued)

Example 4.112 (Continued)

with the seven wavelets, will form a complete and orthonormal basis. Is such a vector defined uniquely?

Let us call the missing vector $\boldsymbol{e} \equiv [e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8]$. *For the complete basis to be orthonormal, the values of* $e_1, \dots, e_8$ *have to be defined so that the dot product of this vector with all wavelet vectors is zero, and the magnitude of the vector is* 1. *Let us consider first the dot product of the unknown vector with each wavelet in turn, and set it to zero. We shall have a linear system of seven equations for the eight unknowns. From the first four of these equations we deduce that:*

$$\begin{aligned}
&-\frac{1}{\sqrt{2}}e_1 + \frac{1}{\sqrt{2}}e_2 = 0 \Rightarrow e_2 = e_1\\
&-\frac{1}{\sqrt{2}}e_3 + \frac{1}{\sqrt{2}}e_4 = 0 \Rightarrow e_4 = e_3\\
&-\frac{1}{\sqrt{2}}e_5 + \frac{1}{\sqrt{2}}e_6 = 0 \Rightarrow e_6 = e_5\\
&-\frac{1}{\sqrt{2}}e_7 + \frac{1}{\sqrt{2}}e_8 = 0 \Rightarrow e_8 = e_7.
\end{aligned} \tag{4.490}$$

Using these results into the next two equations, we obtain

$$\begin{aligned}
-\frac{1}{2}e_1 - \frac{1}{2}e_2 + \frac{1}{2}e_3 + \frac{1}{2}e_4 = 0 &\Rightarrow e_3 + e_4 = e_1 + e_2\\
&\Rightarrow 2e_3 = 2e_1 \Rightarrow e_3 = e_1\\
-\frac{1}{2}e_5 - \frac{1}{2}e_6 + \frac{1}{2}e_7 + \frac{1}{2}e_8 = 0 &\Rightarrow e_5 + e_6 = e_7 + e_8\\
&\Rightarrow 2e_7 = 2e_5 \Rightarrow e_7 = e_5
\end{aligned} \tag{4.491}$$

and from the last one:

$$\begin{aligned}
&-\frac{1}{\sqrt{8}}e_1 - \frac{1}{\sqrt{8}}e_2 - \frac{1}{\sqrt{8}}e_3 - \frac{1}{\sqrt{8}}e_4 + \frac{1}{\sqrt{8}}e_5 + \frac{1}{\sqrt{8}}e_6 + \frac{1}{\sqrt{8}}e_7 + \frac{1}{\sqrt{8}}e_8 = 0 \Rightarrow\\
&\qquad e_5 + e_6 + e_7 + e_8 = e_1 + e_2 + e_3 + e_4 \Rightarrow 4e_5 = 4e_1 \Rightarrow e_5 = e_1.
\end{aligned} \tag{4.492}$$

Therefore:

$$e_2 = e_3 = e_4 = e_5 = e_6 = e_7 = e_8 = e_1. \tag{4.493}$$

This vector should have length 1, so:

$$e_1^2 + e_2^2 + e_3^2 + e_4^2 + e_5^2 + e_6^2 + e_7^2 + e_8^2 = 1 \Rightarrow 8e_1^2 = 1 \Rightarrow e_1 = \pm\frac{1}{\sqrt{8}}. \tag{4.494}$$

There are, therefore, two possible vectors with which we may complete the orthonormal basis: either

$$\mathbf{e} = \left(\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}\right) \tag{4.495}$$

or

$$\left(-\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}\right). \tag{4.496}$$

By convention we would choose the first vector with the positive components. When the digital signal is projected onto this vector, we obtain a number proportional to its average (or, as it is otherwise known, to its direct component (dc)).

The answer, therefore, to the question whether the missing vector needed to make the basis complete is uniquely defined or not, is that this vector is unique, up to a sign.

Example 4.113

Using vector "e" defined in Example 4.112, re-write Equation (4.482) of the wavelet transform, so that it is invertible. Then invert it to recover the original signal from the knowledge of the vector on the left-hand side of this equation.

The augmented version of Equation (4.482) is

$$\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{21} \\ w_{22} \\ w_{31} \\ s_{31} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} \tag{4.497}$$

where we denoted by s_{31} the projection of the signal on vector **e**, *appearing in the last row of the transformation matrix. Obviously:*

$$s_{31} = \frac{x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8}{\sqrt{8}}. \tag{4.498}$$

The transformation matrix now is square 8×8, and orthogonal: all its rows are orthogonal to each other, so its inverse is its transpose. The original signal, therefore, may be recovered as follows:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{2} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{2} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{21} \\ w_{22} \\ w_{31} \\ s_{31} \end{bmatrix}. \tag{4.499}$$

(Continued)

Example 4.113 (Continued)

If we substitute the values of the wavelet coefficients from (4.483) and the value of s_{31} from (4.498), the right-hand side of the above equation becomes:

$$\begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{2} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{2} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \end{bmatrix} \begin{bmatrix} \frac{-x_1+x_2}{\sqrt{2}} \\ \frac{-x_3+x_4}{\sqrt{2}} \\ \frac{-x_5+x_6}{\sqrt{2}} \\ \frac{-x_7+x_8}{\sqrt{2}} \\ \frac{-x_1-x_2+x_3+x_4}{2} \\ \frac{-x_5-x_6+x_7+x_8}{2} \\ \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{\sqrt{8}} \\ \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8}{\sqrt{8}} \end{bmatrix}. \quad (4.500)$$

This is equal to

$$\begin{bmatrix} -\frac{-x_1+x_2}{2} - \frac{-x_1-x_2+x_3+x_4}{4} - \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{-x_1+x_2}{2} - \frac{-x_1-x_2+x_3+x_4}{4} - \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{8} \\ -\frac{-x_3+x_4}{2} + \frac{-x_1-x_2+x_3+x_4}{4} - \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{-x_3+x_4}{2} + \frac{-x_1-x_2+x_3+x_4}{4} - \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{8} \\ -\frac{-x_5+x_6}{2} - \frac{-x_5-x_6+x_7+x_8}{4} + \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{-x_5+x_6}{2} - \frac{-x_5-x_6+x_7+x_8}{4} + \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{8} \\ -\frac{-x_7+x_8}{2} + \frac{-x_5-x_6+x_7+x_8}{4} + \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{-x_7+x_8}{2} + \frac{-x_5-x_6+x_7+x_8}{4} + \frac{-x_1-x_2-x_3-x_4+x_5+x_6+x_7+x_8}{8} \end{bmatrix} + \begin{bmatrix} \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8}{8} \\ \frac{x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8}{8} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} \quad (4.501)$$

where we have simply split the array into two in order to fit it into the width of the page.

Example 4.114

Assume that we wish to use only the wavelets at scales $l = 1$ and $l = 2$ to analyse an eight-sample signal. These wavelets constitute an incomplete basis of six vectors with which we can span the eight-dimensional space of the signal. Define two more vectors so that with the six wavelet vectors they constitute a complete and orthonormal basis. Is the answer unique?

Let us call one of the two unknown vectors $(e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8)$. This vector must be orthogonal to the first six wavelet vectors, so its components must satisfy Equations (4.490) and (4.491), from which we deduce that:

$$e_2 = e_3 = e_4 = e_1 \qquad \text{and} \qquad e_6 = e_7 = e_8 = e_5. \tag{4.502}$$

The normality constraint implies that:

$$e_1^2 + e_2^2 + e_3^2 + e_4^2 + e_5^2 + e_6^2 + e_7^2 + e_8^2 = 1 \Rightarrow 4e_1^2 + 4e_5^2 = 1. \tag{4.503}$$

Let us choose for simplicity $e_5 = 0$. Then $e_6 = e_7 = e_8 = 0$ as well, and from the normality constraint we obtain $e_1 = 1/2$.

Alternatively, we may choose $e_1 = 0$. Then $e_2 = e_3 = e_4 = 0$ as well, and the normality condition leads to $e_6 = e_7 = e_8 = e_5 = 1/2$. Therefore, the two vectors, which we can use to span the sub-space left over by the use of the six wavelet vectors of $l = 1$ and 2, may be

$$\mathbf{v_1} = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0\right) \tag{4.504}$$

and

$$\mathbf{v_2} = \left(0, 0, 0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right). \tag{4.505}$$

Note that these two vectors are not only orthogonal to the six wavelet vectors, but also to each other, as they should be. This necessity to have the two unknown vectors orthogonal to each other implies that once we fix one of the two, the second is also fixed (up to a sign), because its eight unknown components have to obey $6 + 1$ linear equations plus the normality constraint. However, we have a lot of freedom in choosing the first unknown vector. For example, we could have chosen $e_5 = 1/3$. Then from (4.503) we would have had $4e_1^2 = 1 - 4/9 = 5/9 \Rightarrow e_1^2 = 5/36 \Rightarrow e_1 = \pm\sqrt{5}/6$. In this case our first vector would have been:

$$\tilde{\mathbf{v}}_1 = \left(\frac{\sqrt{5}}{6}, \frac{\sqrt{5}}{6}, \frac{\sqrt{5}}{6}, \frac{\sqrt{5}}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right). \tag{4.506}$$

The second vector should have been chosen to satisfy Equations (4.502) and to be orthogonal to $\tilde{\mathbf{v}}_1$:

$$\frac{\sqrt{5}}{6}e_1 + \frac{\sqrt{5}}{6}e_2 + \frac{\sqrt{5}}{6}e_3 + \frac{\sqrt{5}}{6}e_4 + \frac{1}{3}e_5 + \frac{1}{3}e_6 + \frac{1}{3}e_7 + \frac{1}{3}e_8 = 0 \Rightarrow$$
$$4\frac{\sqrt{5}}{6}e_1 + 4\frac{1}{3}e_5 = 0 \Rightarrow e_5 = -\frac{\sqrt{5}}{2}e_1. \tag{4.507}$$

(Continued)

Example 4.114 (Continued)

From the normality constraint we would have had:

$$4e_1^2 + 4\frac{5}{4}e_1^2 = 1 \Rightarrow 9e_1^2 = 1 \Rightarrow e_1 = \pm\frac{1}{3}. \tag{4.508}$$

If we choose the positive sign, the last vector needed to complete the basis is:

$$\tilde{\mathbf{v}}_2 = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, -\frac{\sqrt{5}}{6}, -\frac{\sqrt{5}}{6}, -\frac{\sqrt{5}}{6}, -\frac{\sqrt{5}}{6}\right). \tag{4.509}$$

Vectors $\tilde{\mathbf{v}}_1$ and $\tilde{\mathbf{v}}_2$ could have been chosen instead of vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ to form a complete and orthonormal basis with the wavelet vectors. So, the answer to this problem is not unique.

Example 4.115

Using vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ defined in Example 4.114 and the wavelet vectors for scales $l = 1$ and $l = 2$ only, re-write Equation (4.482) of the wavelet transform, so that it is invertible. Then invert it to recover the original signal from the knowledge of the vector on the left-hand side of this equation.

In this case, we make no use of the wavelet vector at scale $l = 3$. Instead, we use the two new vectors, which, with the first six wavelet vectors, constitute a complete and orthonormal basis for the eight-sample signal:

$$\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{21} \\ w_{22} \\ s_{21} \\ s_{22} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}. \tag{4.510}$$

Here we denoted by s_{21} the projection of the signal onto vector $\mathbf{v}_1$ and by s_{22} the projection of the signal onto vector $\mathbf{v}_2$, appearing in the last two rows of the transformation matrix. Obviously:

$$s_{21} = \frac{x_1 + x_2 + x_3 + x_4}{2} \qquad s_{22} = \frac{x_5 + x_6 + x_7 + x_8}{2}. \tag{4.511}$$

The transformation matrix is square 8×8, and orthogonal: all its rows are orthogonal to each other, so its inverse is its transpose. The original signal, therefore, may be recovered as follows:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{21} \\ w_{22} \\ s_{21} \\ s_{22} \end{bmatrix}. \tag{4.512}$$

If we substitute the values of the wavelet coefficients from (4.483) and the values of s_{21} and s_{22} from (4.511), the right-hand side of the above equation becomes:

$$\begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \frac{-x_1+x_2}{\sqrt{2}} \\ \frac{-x_3+x_4}{\sqrt{2}} \\ \frac{-x_5+x_6}{\sqrt{2}} \\ \frac{-x_7+x_8}{\sqrt{2}} \\ \frac{-x_1-x_2+x_3+x_4}{2} \\ \frac{-x_5-x_6+x_7+x_8}{2} \\ \frac{x_1+x_2+x_3+x_4}{2} \\ \frac{x_5+x_6+x_7+x_8}{2} \end{bmatrix} =$$

$$\begin{bmatrix} -\frac{-x_1+x_2}{2} - \frac{-x_1-x_2+x_3+x_4}{4} + \frac{x_1+x_2+x_3+x_4}{4} \\ \frac{-x_1+x_2}{2} - \frac{-x_1-x_2+x_3+x_4}{4} + \frac{x_1+x_2+x_3+x_4}{4} \\ -\frac{-x_3+x_4}{2} + \frac{-x_1-x_2+x_3+x_4}{4} + \frac{x_1+x_2+x_3+x_4}{4} \\ \frac{-x_3+x_4}{2} + \frac{-x_1-x_2+x_3+x_4}{4} + \frac{x_1+x_2+x_3+x_4}{4} \\ -\frac{-x_5+x_6}{2} - \frac{-x_5-x_6+x_7+x_8}{4} + \frac{x_5+x_6+x_7+x_8}{4} \\ \frac{-x_5+x_6}{2} - \frac{-x_5-x_6+x_7+x_8}{4} + \frac{x_5+x_6+x_7+x_8}{4} \\ -\frac{-x_7+x_8}{2} + \frac{-x_5-x_6+x_7+x_8}{4} + \frac{x_5+x_6+x_7+x_8}{4} \\ \frac{-x_7+x_8}{2} + \frac{-x_5-x_6+x_7+x_8}{4} + \frac{x_5+x_6+x_7+x_8}{4} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}. \tag{4.513}$$

Example 4.116

Assume that we wish to use only the wavelets at scale $l = 1$ to analyse an eight-sample signal. These wavelets constitute an incomplete basis of four vectors with which we can span the eight-dimensional space of the signal. Define four more vectors so that with the four wavelet vectors they constitute a complete and orthonormal basis. Is the answer unique?

Let us call one of the four unknown vectors $(e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8)$. This vector must be orthogonal to the first four wavelet vectors, so these variables must satisfy equations (4.490), from which we deduce that:

$$e_2 = e_1 \qquad e_4 = e_3 \qquad e_6 = e_5 \qquad \text{and} \qquad e_8 = e_7. \tag{4.514}$$

The normality constraint implies that:

$$e_1^2 + e_2^2 + e_3^2 + e_4^2 + e_5^2 + e_6^2 + e_7^2 + e_8^2 = 1 \Rightarrow 2e_1^2 + 2e_3^2 + 2e_5^2 + 2e_7^2 = 1. \tag{4.515}$$

Let us choose $e_3 = e_5 = e_7 = 0$. Then $e_2 = e_1 = 1/\sqrt{2}$. Thus we deduce the first such vector:

$$\mathbf{z_1} = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0, 0 \right). \tag{4.516}$$

If we choose $e_1 = e_5 = e_7 = 0$, we obtain a different vector:

$$\mathbf{z_2} = \left(0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0 \right). \tag{4.517}$$

If we choose $e_1 = e_3 = e_7 = 0$, we obtain a third vector:

$$\mathbf{z_3} = \left(0, 0, 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0 \right). \tag{4.518}$$

Finally, if we choose $e_1 = e_3 = e_5 = 0$, we obtain a fourth vector:

$$\mathbf{z_4} = \left(0, 0, 0, 0, 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right). \tag{4.519}$$

Note that these four vectors are orthogonal to each other and so with the four wavelet vectors they constitute a complete and orthonormal basis in terms of which an eight-sample signal may be analysed. Note also that the solution is not unique, as we could have chosen different values for three of the unknowns in the first step, and so produce a different first vector. Then to construct the second vector we would have the additional constraint that it should be orthogonal not only to the original four wavelet vectors, but also to the newly constructed vector. This would have meant that we could have chosen freely the values of only two of its components. For the construction of the third vector we would have had the free choice of only one component, while our choice in the construction of the fourth vector would have been restricted only to the choice of a sign. This shows that there is an infinite number of ways in which we may choose the four extra vectors with which to complete the orthonormal basis of the four wavelets of scale $l = 1$.

Example 4.117

Using vectors $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{z}_3$ and $\mathbf{z}_4$, defined in Example 4.116 and the wavelet vectors for scale $l = 1$ only, re-write Equation (4.482) of the wavelet transform, so that it is invertible. Then invert it to recover the original signal from the knowledge of the vector on the left-hand side of this equation.

In this case, we make no use of the wavelet vectors at scales $l = 2$ and $l = 3$. Instead, we use the four new vectors, which, with the first four wavelet vectors, constitute a complete and orthonormal basis for the eight-sample signal

$$\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} \tag{4.520}$$

where we denoted by s_{11} the projection of the signal on vector $\mathbf{z}_1$, by s_{12} the projection of the signal on vector $\mathbf{z}_2$, by s_{13} the projection of the signal on vector $\mathbf{z}_3$ and by s_{14} the projection of the signal on vector $\mathbf{z}_4$, appearing in the last four rows of the transformation matrix. Obviously:

$$s_{11} = \frac{x_1 + x_2}{\sqrt{2}} \qquad s_{12} = \frac{x_3 + x_4}{\sqrt{2}} \qquad s_{13} = \frac{x_5 + x_6}{\sqrt{2}} \qquad s_{14} = \frac{x_7 + x_8}{\sqrt{2}}. \tag{4.521}$$

The transformation matrix is square 8×8, and orthogonal: all its rows are orthogonal to each other. So its inverse is its transpose. The original signal, therefore, may be recovered as follows:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \end{bmatrix}. \tag{4.522}$$

(Continued)

Example 4.117 (Continued)

If we substitute the values of the wavelet coefficients from (4.483) and the values of s_{11}, s_{12}, s_{13} and s_{14} from (4.521), the right-hand side of the above equation becomes:

$$\begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{-x_1+x_2}{\sqrt{2}} \\ \frac{-x_3+x_4}{\sqrt{2}} \\ \frac{-x_5+x_6}{\sqrt{2}} \\ \frac{-x_7+x_8}{\sqrt{2}} \\ \frac{x_1+x_2}{\sqrt{2}} \\ \frac{x_3+x_4}{\sqrt{2}} \\ \frac{x_5+x_6}{\sqrt{2}} \\ \frac{x_7+x_8}{\sqrt{2}} \end{bmatrix} =$$

$$\begin{bmatrix} -\frac{-x_1+x_2}{2} + \frac{x_1+x_2}{2} \\ \frac{-x_1+x_2}{2} + \frac{x_1+x_2}{2} \\ -\frac{-x_3+x_4}{2} + \frac{x_3+x_4}{2} \\ \frac{-x_3+x_4}{2} + \frac{x_3+x_4}{2} \\ -\frac{-x_5+x_6}{2} + \frac{x_5+x_6}{2} \\ \frac{-x_5+x_6}{2} + \frac{x_5+x_6}{2} \\ -\frac{-x_7+x_8}{2} + \frac{x_7+x_8}{2} \\ \frac{-x_7+x_8}{2} + \frac{x_7+x_8}{2} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}. \tag{4.523}$$

Example 4.118

Show that the space spanned by vector "e" of Example 4.112 is a sub-space of that spanned by vectors "v" of Example 4.114 and the sub-space spanned by vectors "v" is a sub-space of that spanned by vectors "z" of Example 4.116.

There are two ways to show this. The first one is to observe that the conditions that vector **e** *in Example 4.112 had to fulfil are a superset of the conditions vectors* **v** *had to fulfil. So any basis vector chosen for the space of vector* **e** *automatically fulfils the constraints of space* **v**, *and so not only does it belong to it, but it may also serve as a basis vector for it. Similarly, the constraints that vectors* **v** *had to fulfil are a superset of the conditions vectors* **z** *had to fulfil. So any basis vector chosen for the space of vectors* **v** *automatically fulfils the constraints of space* **z**, *and so not only does it belong to it, but it may also serve as a basis vector for it.*

The second way to answer this question is to show that a basis vector chosen for the space of vector **e** *may be expressed as a linear combination of the basis vectors of space* **v**. *Indeed, we may easily see that* $\mathbf{e} = \mathbf{v_1}/\sqrt{2} + \mathbf{v_2}/\sqrt{2}$. *Similarly, vectors* **v** *may be expressed as linear combinations of vectors* **z**: $\mathbf{v_1} = \mathbf{z_1}/\sqrt{2} + \mathbf{z_2}/\sqrt{2}$ *and* $\mathbf{v_2} = \mathbf{z_3}/\sqrt{2} + \mathbf{z_4}/\sqrt{2}$. *If the basis vectors of a space A can*

*be expressed as linear combinations of the basis vectors of another space B, then **any** vector of space A can be expressed as a linear combination of the basis vectors of space B, and so it belongs to it. Therefore, space A is a sub-space of space B.*

Why is the continuous wavelet transform invertible and the discrete wavelet transform non-invertible?

In the discrete case there is a finite part of the "what happens when" space, around the dc component, which contains part of the signal and without which the full signal cannot be reconstructed. In the continuous case, this part becomes vanishingly small, and so the full signal may be recovered. Indeed, when the scaling variable a takes continuous values, we allow it to take values in the range $(0, \infty)$. Although a cannot really take the value ∞, it may become arbitrarily large. The frequency band covered by the scaled function is scaled to be non-zero in the range $\Delta\omega/a$ (see Examples 4.83–4.86). As $a \to \infty$, this band goes arbitrarily close to zero and so at the limit it picks up the dc component of the signal. The ability of the continuous transform to allow the scaling variable to become arbitrarily large allows the full representation of the signal. On the contrary, the discrete transform has a finite maximum value for the scaling parameter a. The largest value a can take determines the coarsest detail of the signal that may be captured by the wavelet function, i.e. the coarsest band pass filter used. The longer the filter in the time domain, the more it approaches the zero frequency in the frequency domain. Since there is an upper limit on the size of the filter in the time domain, there must be a lower limit on the frequencies that may be captured by the filter used to probe the signal. And since all these filters are band pass filters, there will always be a part of the signal that corresponds to low frequencies, which resides in that part of the "what happens when" space that should be spanned by wavelets with values of a above the maximum value allowed by the discretisation process. When seen that way, the discretisation process effectively carves the continuous "what happens when" space into sub-spaces of different resolutions. For each maximum value of a, we carve away part of the sub-space that is not spanned by the wavelets. The larger the maximum value of a, the more we restrict that sub-space. Thus, we may think of the successive ranges of allowed values of a as creating a nested set of sub-spaces, each one of which goes closer and closer to the direct component of the signal. The projections of the signal on the sub-spaces left out by the successive maximum values of a are called the **smooths** of the signal, while the components of the signal in the sub-space spanned by the allowed values of a constitute the **roughs** of the signal.

How can we span the part of the "what happens when" space that contains the direct component of the signal?

The "what" part may be achieved by scaling a function with non-zero dc component. The "when" part may be achieved by shifting the same function along the time axis. If we scale such a function, we simply change its bandwidth, which will always contain the 0 frequency (see Example 4.84). In the case of wavelets, scaling meant shifting also the centre of the band. Here the centre of the band is 0, so it remains 0 even when it is divided by a and it is not shifted when changing the scaling parameter. This becomes more explicit with the help of Figure 4.93.

In the discrete case, this implies the creation of nested sub-spaces of reducing resolution as a increases. Note that the cells created for a fixed resolution are expected to be orthogonal to each

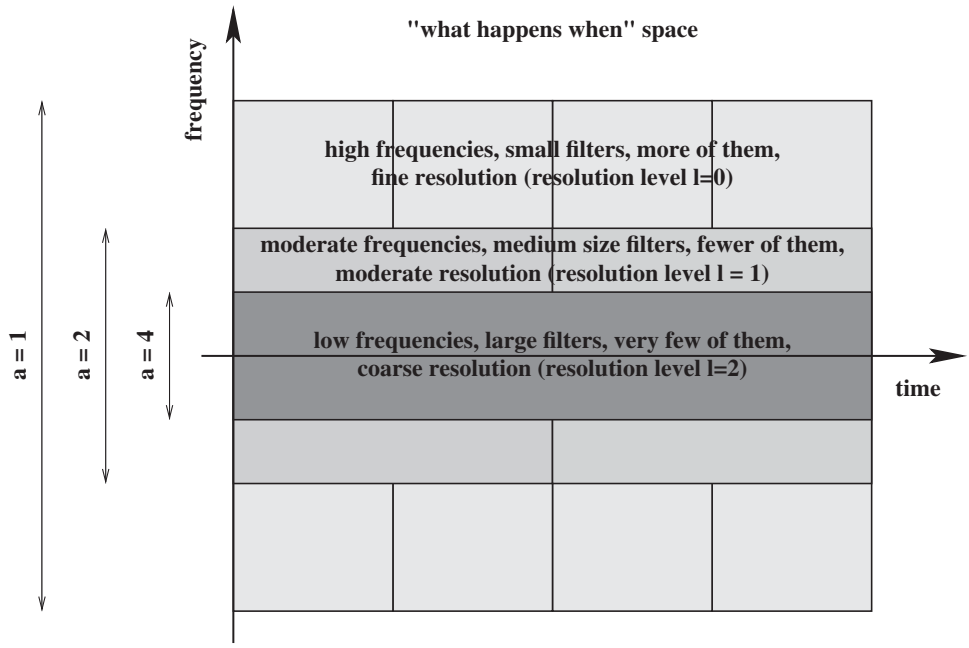


Figure 4.93 The "what happens when" space. Consider a function that contains the 0 frequency (i.e. it has non-zero direct component). Use it to span the "what happens when" space by scaling and shifting it. The frequency bands we create by scaling the time variable will be nested because their centre (being at 0 frequency) cannot be shifted. Larger values of the scaling parameter will simply cover narrower frequency bands centred about the 0 frequency. The rectangles here represent schematically the resolution cells we create by shifting and scaling the scaling function $\phi(t)$. The different shades of grey represent nested frequency bands created with different values of the scaling parameter a. The darker the shade of grey, the more bands overlap there. The time axis is covered by shifting the function.

other (they are not overlapping). However, the cells that correspond to different resolutions are nested, so they cannot possibly be orthogonal.

Let us call the basic function we use, with non-vanishing direct component, **scaling function** $\phi(t)$. The scaled and translated versions of it are:

$$\phi_{ab}(t) = \frac{1}{\sqrt{a}}\phi\left(\frac{t-b}{a}\right). \tag{4.524}$$

We can use the scaling function $\phi(t)$ to cover the discretised "what is where" space in the same way we used the mother wavelet, i.e. by choosing discrete values for the scaling and shifting parameters a and b, respectively: a should vary as 2^l, in accordance to the scale used for the mother wavelet, which is used to complete the coverage of the "what is where" space, and b should take as values integer multiples of a, i.e. $(k-1)2^l$. So, we create functions

$$\phi_{lk}(t) = \frac{1}{\sqrt{2^l}}\phi\left(\frac{t}{2^l} - k\right). \tag{4.525}$$

For fixed resolution the scaling function may be sampled at integer values of the independent variable t, to create a series of vectors that span the smooth space of a signal and thus complement the wavelet basis. Scaling such a function creates a nested sequence of sub-spaces.

Functions ψ_{lk} and ϕ_{lk} are used to create the **multi-resolution representation** of a signal, so that the contents of the signal at a coarse resolution may be extracted from the contents of the

signal at finer resolutions. Note that when we want to treat these sampled functions as vectors, we represent them with bold face. Here, as we refer to them as functions, we do not use bold face to represent them.

Example B4.119

Starting from Equation (4.525), derive the *two scale equation*:

$$\phi_{lk}(t) = \sqrt{2}\phi_{l+1,k}(2t). \tag{4.526}$$

By definition:

$$\begin{aligned}\phi_{l+1,k}(t) &= \frac{1}{\sqrt{2^{l+1}}}\phi\left(\frac{t}{2^{l+1}} - k\right)\\ &= \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2^l}}\phi\left(\frac{t}{2\times 2^l} - k\right)\\ &= \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2^l}}\phi\left(\frac{t/2}{2^l} - k\right) = \frac{1}{\sqrt{2}}\phi_{lk}\left(\frac{t}{2}\right)\\ \Rightarrow \phi_{lk}\left(\frac{t}{2}\right) &= \sqrt{2}\phi_{l+1,k}(t).\end{aligned} \tag{4.527}$$

Then by redefining the independent variable t, i.e. by setting $\tilde{t} \equiv t/2$ and then dropping the tilde, we obtain (4.526).

In a similar way, one can prove a corresponding equation for the wavelet function:

$$\psi_{lk}(t) = \sqrt{2}\psi_{l+1,k}(2t). \tag{4.528}$$

Can we span the whole "what is where" space by using only the scaling function?

Yes. This is shown schematically in Figure 4.93. This leads to the highly redundant representation of a signal by a pyramid of signals, if sub-sampling takes place, or a tower of signals, if no sub-sampling takes place. Often the scaling function used is a digital Gaussian and then we get the so-called **Gaussian pyramid**. The layers of the pyramid are versions of the signal with higher and higher levels of smoothing, and they constitute the **scale-space representation** of the signal (see Book I [75]). As the signal is smoothed, more and more of its details disappear. The more significant a detail is, the more the scales of smoothing over which it survives. This way, the significance of signal characteristics may be expressed as a function of the scale up to which the characteristic is still discernible.

If before sub-sampling we subtract one layer of the tower of signals we created from the next one, we create the so-called **Laplacian pyramid** (see the section What is the Laplacian pyramid?).

How can we extract the coarse resolution content of a signal from its content at a finer resolution?

Let us call S_l the sub-space spanned by basis vectors ϕ_{lk}. As can be seen schematically in Figure 4.93, $S_2 \subset S_1 \subset S_0$. As the scale increases ($a \uparrow, l \uparrow$), the corresponding sub-space shrinks, and the space that has to be covered by the mother wavelet function expands. This is shown schematically in Figure 4.94. Let us call the sub-space spanned by the wavelet function ψ_{lk}, D_l. From the schematic

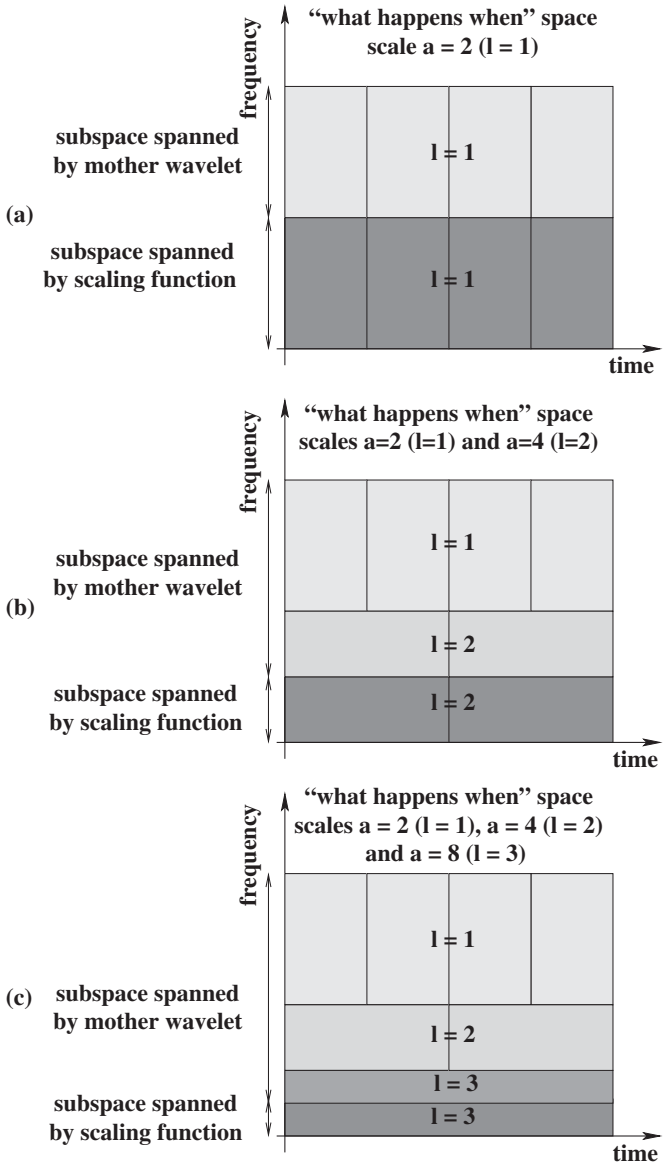


Figure 4.94 The "what happens when" space. As the sub-space spanned by the scaling function expands (from bottom to top), the sub-space spanned by the mother wavelet shrinks. As the scale doubles, the sub-space spanned by the scaling function splits into a sub-space spanned by the scaling function at the new level, plus a sub-space covered by the mother wavelet also at the new level of resolution (from top to bottom). Negative frequencies are not shown here.

representation of Figure 4.94 we can easily see that between two successive scales, sub-space S_l is replaced by sub-spaces S_{l+1} and D_{l+1}. We may indicate this by writing

$$S_l = S_{l+1} \oplus D_{l+1} \tag{4.529}$$

where $\oplus$ indicates the addition of two sub-spaces. This expression implies that $S_{l+1} \subset S_l$ and $D_{l+1} \subset S_l$. Now, if a sub-space is included inside another sub-space, all the vectors of the first sub-space

belong to the second, including the basis vectors. The basis vectors for sub-space S_{l+1} are functions $\boldsymbol{\phi}_{(\mathbf{l+1})\mathbf{k}}(t)$ and the basis vectors of sub-space D_{l+1} are functions $\boldsymbol{\psi}_{(\mathbf{l+1})\mathbf{k}}(t)$. We conclude, therefore, that these two sets of functions also belong to sub-space S_l. If they belong to sub-space S_l, it must be possible to express them as linear combinations of the basis vectors of this sub-space, namely functions $\boldsymbol{\phi}_{\mathbf{lk}}(t)$. So, we must be able to write

$$\begin{aligned}\boldsymbol{\phi}_{(\mathbf{l+1})\tilde{\mathbf{k}}}(t) &= \sum_k g_{\tilde{k}k}\boldsymbol{\phi}_{\mathbf{lk}}(t)\\ \boldsymbol{\psi}_{(\mathbf{l+1})\tilde{\mathbf{k}}}(t) &= \sum_k h_{\tilde{k}k}\boldsymbol{\phi}_{\mathbf{lk}}(t)\end{aligned} \tag{4.530}$$

where $g_{\tilde{k}k}$ and $h_{\tilde{k}k}$ are some linear coefficients. We used $\tilde{k}$ to indicate the integer translates at level $l+1$ in order to avoid confusion with the integer translates at level l.

Let us consider now a signal $\mathbf{f}(t)$ that we wish to project on the basis vectors of sub-space S_{l+1}. Let us take the dot product of both sides of Equations (4.530) with this signal:

$$\begin{aligned}\boldsymbol{\phi}_{(\mathbf{l+1})\tilde{\mathbf{k}}}(t)\cdot\mathbf{f}(t) &= \sum_k g_{\tilde{k}k}\boldsymbol{\phi}_{\mathbf{lk}}(t)\cdot\mathbf{f}(t)\\ \boldsymbol{\psi}_{(\mathbf{l+1})\tilde{\mathbf{k}}}(t)\cdot\mathbf{f}(t) &= \sum_k h_{\tilde{k}k}\boldsymbol{\phi}_{\mathbf{lk}}(t)\cdot\mathbf{f}(t).\end{aligned} \tag{4.531}$$

We observe that by definition, $\phi_{(l+1)\tilde{k}}(t)\cdot f(t)$ is the $\tilde{k}$th coefficient of the expansion of signal $f(t)$ in terms of the basis of sub-space S_{l+1}. Let us call it $s_{(l+1)\tilde{k}}$. Similarly, $\psi_{(l+1)\tilde{k}}(t)\cdot f(t)$ is the $\tilde{k}$th coefficient of the expansion of signal $f(t)$ in terms of the basis of sub-space D_{l+1}. Let us call it $w_{(l+1)\tilde{k}}$. Also, $\phi_{lk}(t)\cdot f(t)$ is the kth coefficient of the expansion of signal $f(t)$ in terms of the basis of sub-space S_l. Let us call it s_{lk}. Expressions (4.531) then become:

$$\begin{aligned}s_{(l+1)\tilde{k}} &= \sum_k g_{\tilde{k}k}s_{lk}\\ w_{(l+1)\tilde{k}} &= \sum_k h_{\tilde{k}k}s_{lk}.\end{aligned} \tag{4.532}$$

These two equations show that we may compute the coefficients of the expansion of signal $f(t)$ at a coarser level of resolution from its expansion coefficients at the previous (finer) level of resolution, if we know the coefficients $g_{\tilde{k}k}$ and $h_{\tilde{k}k}$ for all shifts k.

Example 4.120

Show that the wavelet and scaling vectors for level $l=2$ in Example 4.115 may be expressed as linear combinations of the scaling vectors at level $l=1$.

The wavelet vectors at level $l=2$ are given by Equation (4.479):

$$\begin{aligned}\boldsymbol{\psi}_{\mathbf{21}} &= \left(-\frac{1}{2},-\frac{1}{2},\frac{1}{2},\frac{1}{2},0,0,0,0\right)\\ \boldsymbol{\psi}_{\mathbf{22}} &= \left(0,0,0,0,-\frac{1}{2},-\frac{1}{2},\frac{1}{2},\frac{1}{2}\right).\end{aligned} \tag{4.533}$$

The scaling vectors at level $l=2$ were defined in Example 4.118. They are:

$$\begin{aligned}\boldsymbol{\phi}_{\mathbf{21}} &= \left(\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},0,0,0,0\right)\\ \boldsymbol{\phi}_{\mathbf{22}} &= \left(0,0,0,0,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2}\right).\end{aligned} \tag{4.534}$$

(Continued)

Example 4.120 (Continued)

The scaling vectors at level $l = 1$ were defined in Example 4.116. They are:

$$\begin{aligned}
\boldsymbol{\phi}_{11} &= \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0, 0\right) \\
\boldsymbol{\phi}_{12} &= \left(0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0\right) \\
\boldsymbol{\phi}_{13} &= \left(0, 0, 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0\right) \\
\boldsymbol{\phi}_{14} &= \left(0, 0, 0, 0, 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right).
\end{aligned} \tag{4.535}$$

We can easily verify that:

$$\begin{aligned}
\boldsymbol{\psi}_{21} &= -\frac{1}{\sqrt{2}}\boldsymbol{\phi}_{11} + \frac{1}{\sqrt{2}}\boldsymbol{\phi}_{12} \\
\boldsymbol{\psi}_{22} &= -\frac{1}{\sqrt{2}}\boldsymbol{\phi}_{13} + \frac{1}{\sqrt{2}}\boldsymbol{\phi}_{14} \\
\boldsymbol{\phi}_{21} &= \frac{1}{\sqrt{2}}\boldsymbol{\phi}_{11} + \frac{1}{\sqrt{2}}\boldsymbol{\phi}_{12} \\
\boldsymbol{\phi}_{22} &= \frac{1}{\sqrt{2}}\boldsymbol{\phi}_{13} + \frac{1}{\sqrt{2}}\boldsymbol{\phi}_{14}.
\end{aligned} \tag{4.536}$$

These equations show that the wavelet vectors, $\boldsymbol{\psi}_{21}$ and $\boldsymbol{\psi}_{22}$, and the scaling vectors, $\boldsymbol{\phi}_{21}$ and $\boldsymbol{\phi}_{22}$, of level 2, are linear combinations of the scaling vectors, $\boldsymbol{\phi}_{11}$, $\boldsymbol{\phi}_{12}$, $\boldsymbol{\phi}_{13}$ and $\boldsymbol{\phi}_{14}$, of level 1.

Example 4.121

Show that the wavelet and the scaling vectors for level $l = 3$ in Example 4.117 may be expressed as linear combinations of the scaling vectors at level $l = 2$.

The wavelet vector for level $l = 3$ is given by:

$$\boldsymbol{\psi}_{31} = \left(-\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}, -\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}\right). \tag{4.537}$$

The scaling vector at level $l = 3$ is given by Equation (4.495):

$$\boldsymbol{\phi}_{31} = \left(\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}\right). \tag{4.538}$$

The scaling vectors at level $l = 2$ are given by Equations (4.534). Then we observe that:

$$\begin{aligned}
\boldsymbol{\psi}_{31} &= -\frac{1}{\sqrt{2}}\boldsymbol{\phi}_{21} + \frac{1}{\sqrt{2}}\boldsymbol{\phi}_{22} \\
\boldsymbol{\phi}_{31} &= \frac{1}{\sqrt{2}}\boldsymbol{\phi}_{21} + \frac{1}{\sqrt{2}}\boldsymbol{\phi}_{22}.
\end{aligned} \tag{4.539}$$

How can we choose the scaling function?

The scaling function should be such that when scaled and discretised it produces vectors that are orthogonal to the wavelet vectors.

Let us concentrate on scale $l = 1$. Let us say that we are interested in analysing signals that are N samples long. The wavelet vectors are created by shifting the basic wavelet filter at positions $2(k-1)$. Let us say that the wavelet filter is M samples long. At each wavelet vector, M positions are covered by the filter values $H_1, H_2, \dots, H_M$ and the rest are zero. Let us say that the scaling filter consists of the unknown values $G_1, G_2, \dots, G_M$. The scaling vector for a particular value of k should be orthogonal to the wavelet vector for the same shift. This is shown schematically in Figure 4.95. Then the dot product between the scaling vector and the wavelet vector should be 0, so:

$$G_1H_1 + G_2H_2 + \dots + G_{M-1}H_{M-1} + G_MH_M = 0. \tag{4.540}$$

If we choose $\mathbf{G}$ to be the quadrature mirror of $\mathbf{H}$, i.e. the elements of $\mathbf{H}$ are read in reverse order and every second element is multiplied with -1, the above equation is satisfied. For filters of length M, this means:

$$G_p = (-1)^{p-1}H_{M-p+1} \qquad \text{for } p = 1, \dots, M. \tag{4.541}$$

If we substitute in (4.540) we have:

$$\begin{aligned}&(-1)^0H_MH_1 + (-1)^1H_{M-1}H_2 + \dots\\ &+(-1)^{M-2}H_{M-M+1+1}H_{M-1} + (-1)^{M-1}H_{M-M+1}H_M = 0.\end{aligned} \tag{4.542}$$

If M is even, the above equation is obviously true. At least for the dyadic wavelet analysis, i.e. the case where the scaling parameter a takes values 2^l and the shifting parameter b takes values $(k-1)2^l$, wavelet filters are always chosen to be of even length.

If we have to compute filter $\mathbf{H}$ from filter $\mathbf{G}$, we start from Equation (4.541) and multiply both sides with $(-1)^{p+1}$:

$$\begin{aligned}(-1)^{p+1}G_p = (-1)^{p+1}(-1)^{p-1}H_{M-p+1} &\Rightarrow\\ (-1)^{p+1}G_p = (-1)^{2p}H_{M-p+1} &\Rightarrow\\ H_{M-p+1} = (-1)^{p+1}G_p.&\end{aligned} \tag{4.543}$$

If we define a new variable $q \equiv M - p + 1$, then $p = M - q + 1$, and we have

$$H_q = (-1)^{M-q+2}G_{M-q+1} \Rightarrow H_q = (-1)^{M-q}G_{M-q+1} \tag{4.544}$$

which allows us to obtain filter $\mathbf{H}$ from filter $\mathbf{G}$, by reading the elements of $\mathbf{G}$ backwards and changing the sign of every first one, with q taking values $1, 2, \dots, M$.

0	0	H_1	H_2	H_3	H_4	H_5	H_6	0	0	0	0

0	0	G_1	G_2	G_3	G_4	G_5	G_6	0	0	0	0

Figure 4.95 A wavelet vector for a 12-sample long signal, for shifting parameter $k = 2$ at the top, and the corresponding scaling vector for the same shifting value k at the bottom. Their dot product must be 0.

Example 4.122

Define a scaling function appropriate for the Haar wavelet.

For the Haar wavelet, $\mathbf{H} = (-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$*. Therefore, by applying formula (4.541) we obtain* $\mathbf{G} = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$.

Example B4.123

Consider a wavelet filter of length $M = 6$**. Show that all scaling vectors produced by shifting are orthogonal to all wavelet vectors, if the scaling vectors are constructed using Equation (4.541).**

The case when the non-zero elements of one vector correspond to the zero elements of the other vector is trivially true: the dot product of the two vectors in this case will always be zero no matter what values the non-zero elements have. The problem arises when non-zero elements of one vector correspond to non-zero elements of the other. For a filter six taps long, and for the dyadic wavelet case, there are four possible cases, shown in Figure 4.96.

ψ_{1k}		0	0	H_1	H_2	H_3	H_4	H_5	H_6	0	0	0	0	
$\psi_{1(k+1)}$		0	0	0	0	H_1	H_2	H_3	H_4	H_5	H_6	0	0	
$\psi_{1(k+2)}$		0	0	0	0	0	0	H_1	H_2	H_3	H_4	H_5	H_6	
ϕ_{1k}		0	0	G_1	G_2	G_3	G_4	G_5	G_6	0	0	0	0	
$\phi_{1(k+1)}$		0	0	0	0	G_1	G_2	G_3	G_4	G_5	G_6	0	0	
$\phi_{1(k+2)}$		0	0	0	0	0	0	G_1	G_2	G_3	G_4	G_5	G_6	

Figure 4.96 Wavelet and scaling vectors constructed from a six-tap long filter for three successive values of the shifting parameter k at resolution $l = 1$. The dot product of any two of them must be 0, in order for them to constitute an orthogonal basis.

First we remember that by construction, the wavelet vectors constitute an orthonormal (but incomplete) basis. So, by construction, we know that the following equations are valid:

$$\begin{aligned} \psi_{1k}\psi_{1(k+1)} = 0 &\Rightarrow H_3H_1 + H_4H_2 + H_5H_3 + H_6H_4 = 0 \\ \psi_{1k}\psi_{1(k+2)} = 0 &\Rightarrow H_5H_1 + H_6H_2 = 0. \end{aligned} \tag{4.545}$$

From Equation (4.541) we deduce that:

$$G_1 = H_6 \quad G_2 = -H_5 \quad G_3 = H_4 \quad G_4 = -H_3 \quad G_5 = H_2 \quad G_6 = -H_1. \tag{4.546}$$

We want to prove that:

$$\begin{aligned}\phi_{1k}\phi_{1(k+1)} &= 0\\ \phi_{1k}\phi_{1(k+2)} &= 0\\ \phi_{1k}\psi_{1(k+1)} &= 0\\ \phi_{1k}\psi_{1(k+2)} &= 0.\end{aligned} \tag{4.547}$$

With the help of Figure 4.96 and Equations (4.546), the equations we have to prove become:

$$\begin{aligned}H_4H_6 + (-H_3)(-H_5) + H_2H_4 + (-H_1)(-H_3) &= 0\\ H_2H_6 + (-H_1)(-H_5) &= 0\\ H_4H_1 + (-H_3)H_2 + H_2H_3 + (-H_1)H_4 &= 0\\ H_2H_1 + (-H_1)H_2 &= 0.\end{aligned} \tag{4.548}$$

The first two of these equations are true because of Equations (4.545). The last two are trivially true as well.

So, although when we constructed filter **G** *we only ensured that the vectors produced by it were orthogonal to the wavelet vectors of the same value of the shifting parameter (the same k), it turns out that the scaling vectors are also orthogonal to each other and to all other wavelet vectors. In addition, they all have length* 1 *(since if* **H** *has length* 1*, by construction* **G** *will have too), and therefore they constitute an orthonormal basis for scale* $l = 1$.

Example 4.124

Demonstrate that in the dyadic wavelet scheme with orthogonal wavelet vectors, it is not possible to have an odd length wavelet filter.

For simplicity, let us assume that we have a wavelet filter **H** *of length* 5*. All wavelet vectors produced by placing it in shifted positions* $(k-1)2$ *at resolution level* $l = 1$ *must be orthogonal to each other. Let us consider the wavelet vectors that have non-zero overlapping elements. These are shown schematically in Figure 4.97. The orthogonality between vectors* ψ_{1k} *and* $\psi_{1(k+2)}$ *implies that* $H_5H_1 = 0$*, i.e. either* H_1 *or* H_5 *must be zero. Either way the implication is that the filter is four samples long and not five.*

ψ_{1k}		0	0	H_1	H_2	H_3	H_4	H_5	0	0	0	0	0	
$\psi_{1(k+1)}$		0	0	0	0	H_1	H_2	H_3	H_4	H_5	0	0	0	
$\psi_{1(k+2)}$		0	0	0	0	0	0	H_1	H_2	H_3	H_4	H_5	0	

Figure 4.97 A filter of odd length used to produce wavelet vectors for the dyadic wavelet case.

How do we perform the multi-resolution analysis of a signal in practice?

Let us consider Figure 4.94. Let us assume that we are given a signal $f(t)$. We assume that the given signal is at level $l = 0$, i.e. it is at full resolution. We wish first to analyse it into its projections onto the sub-spaces represented in Figure 4.94a, by creating its components in resolution level $l = 1$. Let us assume that we choose two filters, a high pass one called H, and a low pass one called G so that they constitute a mirror filter pair, i.e. they obey Equation (4.541). We may process the signal by expanding it into the basis vectors that may be created from shifts of these filters to positions $(k-1)2$. We saw earlier that we can achieve this projection onto sub-space D_1 spanned by the wavelet vectors, by pseudo-convolving the original signal with filter H and keeping only every other output sample. In a similar way, we may pseudo-convolve the original signal with filter G and keep every other output sample, in order to obtain its projection onto sub-space S_1. These pseudo-convolutions are a little different from the conventional ones: in conventional digital convolutions we usually have to deal with odd-sized filters with a central sample. The output of the multiplication of the filter weights with the corresponding signal samples and the subsequent summation of these products is assigned to the sample at which the centre of the filter is placed. In the case of wavelets, the filters are of even size, and there is no central sample. In addition, the pseudo-convolution outputs represent projections on the wavelet (or scaling) vectors, which are created by shifting the filter along a sequence of the same size as the original signal, i.e. by placing its starting sample at positions 1, 3, 5, etc. (see Examples 4.108 and 4.110). The output of each pseudo-convolution, therefore, is assigned to the first sample of the filter.

Now, once we have found the projection of the signal on the two sub-spaces S_1 and D_1, we may replace the original N-sample long signal by these two $N/2$ projections, because the two sub-spaces fully cover the original space over which the signal is defined. The only difference is that each sample of this new representation of the signal tells us how much the elementary answer to the question "what happens when", expressed by the corresponding wavelet or scaling vector, is present in the signal. This information was hidden in the original data implicitly only, and the projections we performed made it explicit.

We can take now the S_1 component of the signal and treat it as a new signal in its own right. It will be $N/2$ samples long and its frequency band will be $[0, \Omega/2]$ if the frequency band of the original signal were $[0, \Omega]$. (It will reside at the bottom half of the "what happens when" space of Figure 4.94a.) We can therefore project it again onto its own two sub-spaces, call them D_2 and S_2, by pseudo-convolving it with filters H and G, respectively, and sub-sampling as before. This split is represented by the splitting of the bottom half of the space "what happens when" of Figure 4.94a into its two halves, as shown in Figure 4.94b. This time we create two new versions of the S_1 component of the original signal, $N/4$ samples long each. Again, each of these versions may be treated as a new signal in its own right and the low-frequency one may be further analysed into two components by applying the same process. This analysis is represented by Figure 4.94c where the bottom quarter of the original space has been split into two parts, one spanned by the wavelet vectors and one by the scaling function.

The whole process of multi-resolution analysis of the original signal is shown schematically in Figure 4.98. This representation is known as the **tree wavelet** representation of the signal. This process may be continued for as many levels as the length of the signal allows it. We note that the original signal may be fully represented by the components associated with the leaves of this tree expansion. The corresponding algorithm, known as the **fast wavelet transform algorithm**, is as follows.

Step 1: Select wavelet filter **H** or scaling filter **G**. If the selected filter is **H**, construct filter **G** by reading the elements of **H** backwards and changing the sign of every second one (Equation (4.541)). If the selected filter is **G**, construct filter **H** by reading the elements of **G** backwards and changing the sign of every first one (Equation (4.544)).

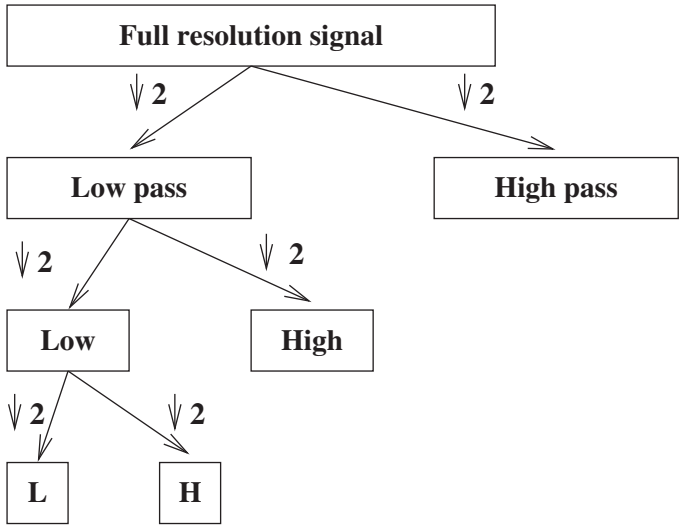


Figure 4.98 Schematic representation of the multi-resolution analysis of a signal using tree wavelet analysis. An arrow with number 2 next to it symbolises sub-sampling by a factor of 2.

Step 2: Pseudo-convolve (i.e. no filter reversal, unless you use a pre-programmed convolution function of some software package, in which case you should reverse the filter) the signal with filter **H**, assigning the output value to the first sample that is overlapped by the filter.
Step 3: Decimate the output by removing every second sample. Call the output H.
Step 4: Pseudo-convolve the signal with filter **G**, assigning the output value to the first sample that is overlapped by the filter.
Step 5: Decimate the output by removing every second sample. Call the output L.
Step 6$_{\text{tree}}$: You may apply steps 2–5 to L again.

Why in tree wavelet analysis do we always analyse the part of the signal that contains the low frequencies only?

Because wavelet theory was developed with the aim of avoiding signal representation with redundancy. If we are not interested in this aspect of the analysis, we may not analyse only the low pass component of the signal, but other components as well, creating redundancy in the representation.

We note from Figure 4.98 that if the original signal were N samples long, its multi-resolution analysis also consists of N samples, made up by the $N/2$ samples of the high frequency component of the first level of analysis, plus the $N/4$ samples of the high frequency component of the second level of analysis, plus the $N/8$ samples of the high frequency component of the third level of analysis, plus the $N/8$ samples of the low frequency component of the third level of analysis. Tree wavelets were developed with the purpose of replacing the original N samples of the signal with N new samples, each of which had a particular meaning in terms of "what happens when". In other words, each of the new replacement samples quantifies the relevance to the original signal of each elementary answer to the question "what happens when". We have constructed this set of elementary answers so that it covers the "what happens when" space with no gaps or overlaps. So, the new replacement samples constitute a compact signal representation, i.e. they contain no redundancy, since each sample gives different from any other sample information concerning the signal, and so its value cannot be predicted by the knowledge of the values of the other samples.

If we are not interested in such a compact representation of the signal, there is no reason not to analyse the high-frequency components as well at each level of resolution. For example, we note that the cells with which we cover the "what happens when" space are very broad along the ω

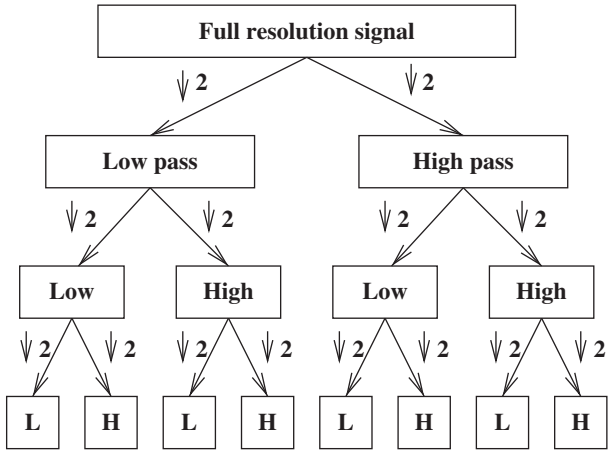


Figure 4.99 Schematic representation of the multiresolution analysis of a signal using packet wavelet analysis. An arrow with number 2 next to it symbolises sub-sampling by a factor of 2.

axis at high frequencies. We may wish to zoom into these high frequencies by analysing further those components of the signal, in the same way as we analyse the low-frequency components. This means that step 6 of the fast wavelet transform algorithm is replaced with:

Step 6$_{\text{packet}}$: Apply steps 2–5 to L and H.

Such a representation is called multi-resolution analysis in terms of **packet wavelets** and it is shown schematically in Figure 4.99.

Example 4.125

Find the wavelet and scaling coefficients for $l = 1$ for an 8-sample long signal using a wavelet filter with weights h_1, h_2, h_3, h_4.

First we must construct the wavelet vectors. These must be as long as the signal itself, i.e. eight components long. For level $l = 1$, these vectors are created by placing the start of the wavelet filter with shifts $(k-1)2$ for $k = 1, 2, 3, 4$, and filling the remaining positions of the eight-sample long vector with zeros:

$$\begin{aligned} k = 1 &: (h_1, h_2, h_3, h_4, 0, 0, 0, 0) \\ k = 2 &: (0, 0, h_1, h_2, h_3, h_4, 0, 0) \\ k = 3 &: (0, 0, 0, 0, h_1, h_2, h_3, h_4) \\ k = 4 &: (h_3, h_4, 0, 0, 0, 0, h_1, h_2). \end{aligned} \tag{4.549}$$

Note the way we wrapped round the filter for the construction of the last wavelet vector, so that we have four vectors which have to span the four-dimensional sub-space that corresponds to the upper half of the "what happens when" space of Figure 4.94a. Note also, that for these vectors to form an orthogonal basis, the dot product of any two of them must be zero. This condition holds if:

$$h_3 h_1 + h_4 h_2 = 0. \tag{4.550}$$

Next we must construct the vectors with which we will span the bottom half of the "what happens when" space of Figure 4.94a. These may be constructed by shifting in a similar way the scaling filter. Once we have the wavelet filter, the scaling filter is fully defined by using Equation (4.541): $h_4, -h_3, h_2, -h_1$. The scaling vectors are:

$$\begin{aligned} k=1 &: (h_4, -h_3, h_2, -h_1, 0, 0, 0, 0) \\ k=2 &: (0, 0, h_4, -h_3, h_2, -h_1, 0, 0) \\ k=3 &: (0, 0, 0, 0, h_4, -h_3, h_2, -h_1) \\ k=4 &: (h_2, -h_1, 0, 0, 0, 0, h_4, -h_3). \end{aligned} \tag{4.551}$$

Now we are ready to expand an eight-sample long signal $(x_1, x_2, \dots, x_8)$ into this vector basis, by taking the dot product of the signal with each one of these vectors in turn:

$$\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_1 & h_2 & h_3 & h_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_1 & h_2 & h_3 & h_4 \\ h_3 & h_4 & 0 & 0 & 0 & 0 & h_1 & h_2 \\ h_4 & -h_3 & h_2 & -h_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_4 & -h_3 & h_2 & -h_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_4 & -h_3 & h_2 & -h_1 \\ h_2 & -h_1 & 0 & 0 & 0 & 0 & h_4 & -h_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \end{bmatrix} = \begin{bmatrix} h_1x_1 + h_2x_2 + h_3x_3 + h_4x_4 \\ h_1x_3 + h_2x_4 + h_3x_5 + h_4x_6 \\ h_1x_5 + h_2x_6 + h_3x_7 + h_4x_8 \\ h_3x_1 + h_4x_2 + h_1x_7 + h_2x_8 \\ h_4x_1 - h_3x_2 + h_2x_3 - h_1x_4 \\ h_4x_3 - h_3x_4 + h_2x_5 - h_1x_6 \\ h_4x_5 - h_3x_6 + h_2x_7 - h_1x_8 \\ h_2x_1 - h_1x_2 + h_4x_7 - h_3x_8 \end{bmatrix}. \tag{4.552}$$

The first four coefficients constitute the high pass component of the signal, while the second four coefficients constitute the low-frequency component of the signal.

Example 4.126

You are given the following signal: $1, 2, 4, 1, -1, -2, -1, 1$**. Use the scaling filter**

$$0.4830, 0.8365, 0.2241, -0.1294$$

to construct the tree wavelet expansion of this signal. Present your answer in a graph similar to that of Figure 4.98.

We shall solve this problem using the fast wavelet transform algorithm, i.e. we shall solve it by using pseudo-convolutions. First we note that the filter we are given is for extracting the low-frequency component of the signal. (If it were a high pass filter, its values would have summed up to 0*.) From this filter we define the wavelet filter using Equation (4.544):*

$$0.1294, 0.2241, -0.8365, 0.4830.$$

(Continued)

Example 4.126 (Continued)

We can compute the coefficients of the expansion of the signal into the wavelet basis defined by the above filter using pseudo-convolution and sub-sampling. We start by pseudo-convolving the signal with the wavelet filter. We remember that the pseudo-convolution is done with the filter not reversed and used in a wrap round way, and that the result at each step is assigned to the first filter position:

$$\begin{aligned}
&(0.1294\times 1+0.2241\times 2-0.8365\times 4+0.4830\times 1,\\
&0.1294\times 2+0.2241\times 4-0.8365\times 1+0.4830\times(-1),\\
&0.1294\times 4+0.2241\times 1-0.8365\times(-1)+0.4830\times(-2),\\
&0.1294\times 1+0.2241\times(-1)-0.8365\times(-2)+0.4830\times(-1)\\
&0.1294\times(-1)+0.2241\times(-2)-0.8365\times(-1)+0.4830\times 1,\\
&0.1294\times(-2)+0.2241\times(-1)-0.8365\times 1+0.4830\times 1\\
&0.1294\times(-1)+0.2241\times 1-0.8365\times 1+0.4830\times 2,\\
&0.1294\times 1+0.2241\times 1-0.8365\times 2+0.4830\times 4)=\\
&(-2.285,-0.164,0.612,1.095,0.742,-0.836,0.224,0.612). \qquad (4.553)
\end{aligned}$$

The wavelet coefficients are derived by removing every second number from the above sequence, so we obtain the sequence $(-2.285, 0.612, 0.742, 0.224)$.

For the scaling coefficients we pseudo-convolve the signal with the given scaling filter, and we obtain the sequence $(2.923, 4.666, 2.803, -0.672, -2.510, -1.708, 0.319, 1.250)$. *We remove every second coefficient to obtain the low pass component of the signal as* $(2.923, 2.803, -2.510, 0.319)$.

For the next level of resolution we process the low pass sequence $(2.923, 2.803, -2.510, 0.319)$ *by pseudo-convolution again with the two filters. The output of the pseudo-convolution with the wavelet filter is* $(3.260, 0.945, -1.344, -2.861)$*, which upon down-sampling produces* $(3.260, -1.344)$. *The output of the pseudo-convolution with the scaling filter is* $(3.153, -1.052, -0.653, 3.552)$*, which upon down-sampling produces* $(3.153, -0.653)$.

Note that the last two sequences, which consist of two components each, constitute the high- and low-frequency component of the signal at level $l = 2$. *This is the final level of resolution we can use as the sequences are now shorter than the filters themselves and they cannot be subjected into further analysis. The tree wavelet analysis of this signal is presented in Figure 4.100.*

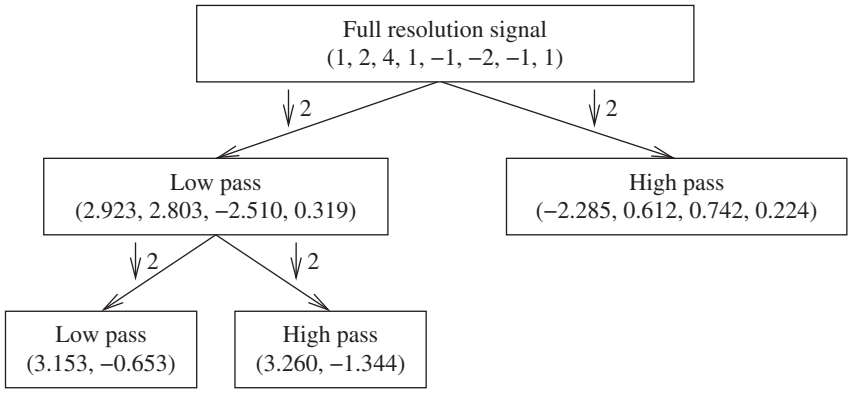


Figure 4.100 The tree wavelet analysis of a digital signal.

Example 4.127

Use the approach of Example 4.125 to compute the low and high pass components of the signal of Example 4.126 for level $l = 1$ and compare your answer with that obtained in Example 4.126.

We must apply the derived formulae on the right-hand side of (4.552) for:

$$x_1 = 1, \quad x_2 = 2, \quad x_3 = 4, \quad x_4 = 1, \quad x_5 = -1, \quad x_6 = -2, \quad x_7 = -1, \quad x_8 = 1$$
$$h_1 = 0.1294, \quad h_2 = 0.2241, \quad h_3 = -0.8365, \quad h_4 = 0.4830. \tag{4.554}$$

These formulae yield

$$w_{11} = -2.285, \quad w_{12} = 0.612, \quad w_{13} = 0.742, \quad w_{14} = 0.224$$
$$s_{11} = 2.923, \quad s_{12} = 2.803, \quad s_{13} = -2.510, \quad s_{14} = 0.319 \tag{4.555}$$

which are exactly the numbers we deduced in Example 4.126 by applying the fast wavelet transform.

Example 4.128

Perform the packet wavelet analysis of the signal of Example 4.126 using the same filter as the one used in Example 4.126. Present your answer in a graph similar to that of Figure 4.99.

To perform the packet wavelet analysis of the signal, we must also expand the high-frequency component at each level of resolution in terms of its low- and high-frequency components, in addition to expanding the low-frequency component.

This is done by pseudo-convolving the high-frequency component with the two filters and sub-sampling.

Convolution of $(-2.285, 0.612, 0.742, 0.224)$ *with the wavelet and the scaling filters produces sequences* $(-0.671, -1.046, 2.354, -0.637)$ *and* $(-0.454, 1.262, -0.045, -1.762)$*, respectively. These, upon down-sampling produce* $(-0.671, 2.354)$ *and* $(-0.454, -0.045)$*, respectively. The packet wavelet analysis of this signal is presented in Figure 4.101.*

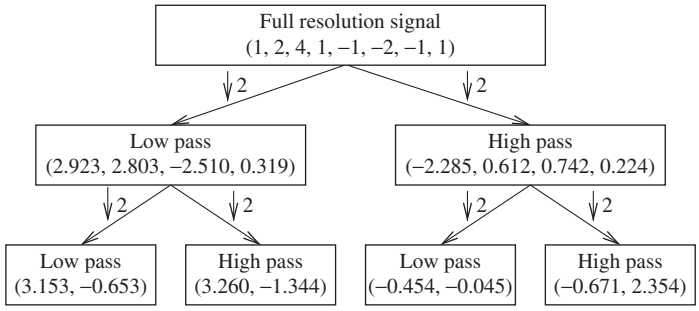


Figure 4.101 The packet wavelet analysis of a digital signal.

Example B4.129

Starting from the knowledge of vector C $\equiv (w_{11}, w_{12}, w_{13}, w_{14}, s_{11}, s_{12}, s_{13}, s_{14})^T$, recover the original signal in Example 4.125.

Let us start by calling the matrix made up by the wavelet and scaling vectors matrix A:

$$A \equiv \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_1 & h_2 & h_3 & h_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_1 & h_2 & h_3 & h_4 \\ h_3 & h_4 & 0 & 0 & 0 & 0 & h_1 & h_2 \\ h_4 & -h_3 & h_2 & -h_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_4 & -h_3 & h_2 & -h_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_4 & -h_3 & h_2 & -h_1 \\ h_2 & -h_1 & 0 & 0 & 0 & 0 & h_4 & -h_3 \end{bmatrix}. \tag{4.556}$$

If $\boldsymbol{X}$ stands for the signal vector, the expansion of the signal expressed in Example 4.125 may be written in matrix form as $\boldsymbol{C} = A\boldsymbol{X}$.

*The rows of matrix A are orthogonal to each other, so this is an orthogonal matrix. The inverse, therefore, of A is its transpose, and the inverse of the expansion is $A^T\boldsymbol{C} = \boldsymbol{X}$. So, to recover the signal we must multiply the coefficients of the expansion with the matrix made up from the wavelet and scaling vectors written next to each other as the **columns** of matrix A^T:*

$$\begin{bmatrix} h_1 & 0 & 0 & h_3 & h_4 & 0 & 0 & h_2 \\ h_2 & 0 & 0 & h_4 & -h_3 & 0 & 0 & -h_1 \\ h_3 & h_1 & 0 & 0 & h_2 & h_4 & 0 & 0 \\ h_4 & h_2 & 0 & 0 & -h_1 & -h_3 & 0 & 0 \\ 0 & h_3 & h_1 & 0 & 0 & h_2 & h_4 & 0 \\ 0 & h_4 & h_2 & 0 & 0 & -h_1 & -h_3 & 0 \\ 0 & 0 & h_3 & h_1 & 0 & 0 & h_2 & h_4 \\ 0 & 0 & h_4 & h_2 & 0 & 0 & -h_1 & -h_3 \end{bmatrix} \begin{bmatrix} h_1x_1 + h_2x_2 + h_3x_3 + h_4x_4 \\ h_1x_3 + h_2x_4 + h_3x_5 + h_4x_6 \\ h_1x_5 + h_2x_6 + h_3x_7 + h_4x_8 \\ h_3x_1 + h_4x_2 + h_1x_7 + h_2x_8 \\ h_4x_1 - h_3x_2 + h_2x_3 - h_1x_4 \\ h_4x_3 - h_3x_4 + h_2x_5 - h_1x_6 \\ h_4x_5 - h_3x_6 + h_2x_7 - h_1x_8 \\ h_2x_1 - h_1x_2 + h_4x_7 - h_3x_8 \end{bmatrix} =$$

$$\begin{bmatrix} h_1(h_1x_1 + h_2x_2 + h_3x_3 + h_4x_4) + h_3(h_3x_1 + h_4x_2 + h_1x_7 + h_2x_8) + \\ h_4(h_4x_1 - h_3x_2 + h_2x_3 - h_1x_4) + h_2(h_2x_1 - h_1x_2 + h_4x_7 - h_3x_8) \\ h_2(h_1x_1 + h_2x_2 + h_3x_3 + h_4x_4) + h_4(h_3x_1 + h_4x_2 + h_1x_7 + h_2x_8) - \\ h_3(h_4x_1 - h_3x_2 + h_2x_3 - h_1x_4) - h_1(h_2x_1 - h_1x_2 + h_4x_7 - h_3x_8) \\ h_3(h_1x_1 + h_2x_2 + h_3x_3 + h_4x_4) + h_1(h_1x_3 + h_2x_4 + h_3x_5 + h_4x_6) + \\ h_2(h_4x_1 - h_3x_2 + h_2x_3 - h_1x_4) + h_4(h_4x_3 - h_3x_4 + h_2x_5 - h_1x_6) \\ h_4(h_1x_1 + h_2x_2 + h_3x_3 + h_4x_4) + h_2(h_1x_3 + h_2x_4 + h_3x_5 + h_4x_6) - \\ h_1(h_4x_1 - h_3x_2 + h_2x_3 - h_1x_4) - h_3(h_4x_3 - h_3x_4 + h_2x_5 - h_1x_6) \\ h_3(h_1x_3 + h_2x_4 + h_3x_5 + h_4x_6) + h_1(h_1x_5 + h_2x_6 + h_3x_7 + h_4x_8) + \\ h_2(h_4x_3 - h_3x_4 + h_2x_5 - h_1x_6) + h_4(h_4x_5 - h_3x_6 + h_2x_7 - h_1x_8) \\ h_4(h_1x_3 + h_2x_4 + h_3x_5 + h_4x_6) + h_2(h_1x_5 + h_2x_6 + h_3x_7 + h_4x_8) - \\ h_1(h_4x_3 - h_3x_4 + h_2x_5 - h_1x_6) - h_3(h_4x_5 - h_3x_6 + h_2x_7 - h_1x_8) \\ h_3(h_1x_5 + h_2x_6 + h_3x_7 + h_4x_8) + h_1(h_3x_1 + h_4x_2 + h_1x_7 + h_2x_8) + \\ h_2(h_4x_5 - h_3x_6 + h_2x_7 - h_1x_8) + h_4(h_2x_1 - h_1x_2 + h_4x_7 - h_3x_8) \\ h_4(h_1x_5 + h_2x_6 + h_3x_7 + h_4x_8) + h_2(h_3x_1 + h_4x_2 + h_1x_7 + h_2x_8) - \\ h_1(h_4x_5 - h_3x_6 + h_2x_7 - h_1x_8) - h_3(h_2x_1 - h_1x_2 + h_4x_7 - h_3x_8) \end{bmatrix} = \tag{4.557}$$

$$
\begin{bmatrix}
x_1(h_1^2+h_3^2+h_4^2+h_2^2)+x_2(h_1h_2+h_3h_4-h_4h_3-h_2h_1)+\\
x_3(h_1h_3+h_4h_2)+x_4(h_1h_4-h_4h_1)+x_7(h_3h_1+h_2h_4)+x_8(h_3h_2-h_2h_3)\\
x_1(h_2h_1+h_4h_3-h_3h_4-h_1h_2)+x_2(h_2^2+h_4^2+h_3^2+h_1^2)+\\
x_3(h_2h_3-h_3h_2)+x_4(h_1h_4+h_3h_1)+x_7(h_4h_1-h_1h_4)+x_8(h_4h_2+h_1h_3)\\
x_1(h_3h_1+h_2h_4)+x_2(h_3h_2-h_2h_3)+x_3(h_3^2+h_1^2+h_2^2+h_4^2)+\\
x_4(h_3h_4+h_1h_2-h_2h_1-h_4h_3)+x_5(h_1h_3+h_4h_2)+x_6(h_1h_4-h_4h_1)\\
x_1(h_4h_1-h_1h_4)+x_2(h_4h_2+h_1h_3)+x_3(h_4h_3+h_2h_1-h_1h_2-h_3h_4)+\\
x_4(h_4^2+h_2^2+h_1^2+h_3^2)+x_5(h_2h_3-h_3h_2)+x_6(h_2h_4+h_3h_1)\\
x_3(h_3h_1+h_2h_4)+x_4(h_3h_2-h_2h_3)+x_5(h_3^2+h_1^2+h_2^2+h_4^2)+\\
x_6(h_3h_4+h_1h_2-h_2h_1-h_4h_3)+x_7(h_1h_3+h_4h_2)+x_8(h_1h_4-h_4h_1)\\
x_3(h_4h_1-h_1h_4)+x_4(h_4h_2+h_1h_3)+x_5(h_4h_3+h_2h_1-h_1h_2-h_3h_4)+\\
x_6(h_4^2+h_2^2+h_1^2+h_3^2)+x_7(h_2h_3-h_3h_2)+x_8(h_2h_4+h_3h_1)\\
x_5(h_3h_1+h_2h_4)+x_6(h_3h_2-h_2h_3)+x_7(h_3^2+h_1^2+h_2^2+h_4^2)+\\
x_8(h_3h_4+h_1h_2-h_2h_1-h_4h_3)+x_1(h_1h_3+h_4h_2)+x_2(h_1h_4-h_4h_1)\\
x_5(h_4h_1-h_1h_4)+x_6(h_4h_2+h_1h_3)+x_7(h_4h_3+h_2h_1-h_1h_2-h_3h_4)+\\
x_8(h_4^2+h_2^2+h_1^2+h_3^2)+x_1(h_2h_3-h_3h_2)+x_2(h_2h_4+h_3h_1)
\end{bmatrix}
=
\begin{bmatrix}
x_1\\x_2\\x_3\\x_4\\x_5\\x_6\\x_7\\x_8
\end{bmatrix}.
\tag{4.558}
$$

The last equality comes from the following properties of the filter we use: (i) shifted versions of it are orthogonal to each other, so $h_1h_3+h_4h_2=0$ *(see Equation (4.550)); (ii) the filter weights are normalised, so that the total energy of the filter is 1, i.e.* $h_1^2+h_2^2+h_3^2+h_4^2=1$.

Box 4.10 How do we recover the original signal from its wavelet coefficients in practice?

This is done by the **fast inverse wavelet transform**. To understand how it works, let us examine the recovery of the original signal from its expansion in Example 4.129. We write here explicitly the equation $A^T\mathbf{C}=\mathbf{X}$ and place partitions in the reconstruction matrix and in the vector of coefficients, to differentiate their parts that refer to the wavelet sub-space from the parts that refer to the scaling sub-space:

(Continued)

Box 4.10 (Continued)

$$\left[\begin{array}{cccc|cccc} h_1 & 0 & 0 & h_3 & h_4 & 0 & 0 & h_2 \\ h_2 & 0 & 0 & h_4 & -h_3 & 0 & 0 & -h_1 \\ h_3 & h_1 & 0 & 0 & h_2 & h_4 & 0 & 0 \\ h_4 & h_2 & 0 & 0 & -h_1 & -h_3 & 0 & 0 \\ 0 & h_3 & h_1 & 0 & 0 & h_2 & h_4 & 0 \\ 0 & h_4 & h_2 & 0 & 0 & -h_1 & -h_3 & 0 \\ 0 & 0 & h_3 & h_1 & 0 & 0 & h_2 & h_4 \\ 0 & 0 & h_4 & h_2 & 0 & 0 & -h_1 & -h_3 \end{array}\right]\left[\begin{array}{c} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ \hline s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \end{array}\right]=\left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array}\right]. \tag{4.559}$$

We note, therefore, that the reconstructed signal consists of two components: one obtained by the left half of the reconstruction matrix operating upon the wavelet coefficients,

$$\left[\begin{array}{cccc} h_1 & 0 & 0 & h_3 \\ h_2 & 0 & 0 & h_4 \\ h_3 & h_1 & 0 & 0 \\ h_4 & h_2 & 0 & 0 \\ 0 & h_3 & h_1 & 0 \\ 0 & h_4 & h_2 & 0 \\ 0 & 0 & h_3 & h_1 \\ 0 & 0 & h_4 & h_2 \end{array}\right]\left[\begin{array}{c} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \end{array}\right]=\left[\begin{array}{c} h_1 w_{11}+h_3 w_{14} \\ h_2 w_{11}+h_4 w_{14} \\ h_3 w_{11}+h_1 w_{12} \\ h_4 w_{11}+h_2 w_{12} \\ h_3 w_{12}+h_1 w_{13} \\ h_4 w_{12}+h_2 w_{13} \\ h_3 w_{13}+h_1 w_{14} \\ h_4 w_{13}+h_2 w_{14} \end{array}\right] \tag{4.560}$$

and one obtained by the right half of the reconstruction matrix operating upon the scaling coefficients:

$$\left[\begin{array}{cccc} h_4 & 0 & 0 & h_2 \\ -h_3 & 0 & 0 & -h_1 \\ h_2 & h_4 & 0 & 0 \\ -h_1 & -h_3 & 0 & 0 \\ 0 & h_2 & h_4 & 0 \\ 0 & -h_1 & -h_3 & 0 \\ 0 & 0 & h_2 & h_4 \\ 0 & 0 & -h_1 & -h_3 \end{array}\right]\left[\begin{array}{c} s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \end{array}\right]=\left[\begin{array}{c} h_4 s_{11}+h_2 s_{14} \\ -h_3 s_{11}-h_1 s_{14} \\ h_2 s_{11}+h_4 s_{12} \\ -h_1 s_{11}-h_3 s_{12} \\ h_2 s_{12}+h_4 s_{13} \\ -h_1 s_{12}-h_3 s_{13} \\ h_2 s_{13}+h_4 s_{14} \\ -h_1 s_{13}-h_3 s_{14} \end{array}\right]. \tag{4.561}$$

We know (see Book I [75]) that convolution of a signal with a filter may be obtained by creating a matrix made up from columns formed by the filter placed in successive positions inside a vector that is the same length as the signal, with the vacant positions filled with 0s. So, we may perform the operation indicated by (4.560) by convolution, if we notice that in the operation matrix we have, the filter is placed at every **second** position, and fill in the missing columns of the operation matrix with the filter placed at the in-between positions. At the same time, as we do not want the inserted columns to produce numbers that will interfere with the real result, we place zeros between the wavelet coefficients, so that the effect of the inserted columns is annulled:

$$\begin{bmatrix} h_1 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{h_4} & h_3 & \mathbf{h_2} \\ h_2 & \mathbf{h_1} & 0 & \mathbf{0} & 0 & \mathbf{0} & h_4 & \mathbf{h_3} \\ h_3 & \mathbf{h_2} & h_1 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{h_4} \\ h_4 & \mathbf{h_3} & h_2 & \mathbf{h_1} & 0 & \mathbf{0} & 0 & \mathbf{0} \\ 0 & \mathbf{h_4} & h_3 & \mathbf{h_2} & h_1 & \mathbf{0} & 0 & \mathbf{0} \\ 0 & \mathbf{0} & h_4 & \mathbf{h_3} & h_2 & \mathbf{h_1} & 0 & \mathbf{0} \\ 0 & \mathbf{0} & 0 & \mathbf{h_4} & h_3 & \mathbf{h_2} & h_1 & \mathbf{0} \\ 0 & \mathbf{0} & 0 & \mathbf{0} & h_4 & \mathbf{h_3} & h_2 & \mathbf{h_1} \end{bmatrix} \begin{bmatrix} w_{11} \\ \mathbf{0} \\ w_{12} \\ \mathbf{0} \\ w_{13} \\ \mathbf{0} \\ w_{14} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} h_1 w_{11} + h_3 w_{14} \\ h_2 w_{11} + h_4 w_{14} \\ h_3 w_{11} + h_1 w_{12} \\ h_4 w_{11} + h_2 w_{12} \\ h_3 w_{12} + h_1 w_{13} \\ h_4 w_{12} + h_2 w_{13} \\ h_3 w_{13} + h_1 w_{14} \\ h_4 w_{13} + h_2 w_{14} \end{bmatrix}. \tag{4.562}$$

Note that the inserted elements, marked in bold, do not affect the net result, because the elements of the inserted columns in the matrix are multiplied with the inserted 0s in the vector. We observe that, because of the wrap-round boundary condition we use, the above multiplication remains the same if we take the last three columns of the reconstruction matrix and place them in the front and at the same time we take the last three elements of the wavelet coefficient vector and place them at the top:

$$\begin{bmatrix} \mathbf{h_4} & h_3 & \mathbf{h_2} & h_1 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & h_4 & \mathbf{h_3} & h_2 & \mathbf{h_1} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{h_4} & h_3 & \mathbf{h_2} & h_1 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & h_4 & \mathbf{h_3} & h_2 & \mathbf{h_1} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{h_4} & h_3 & \mathbf{h_2} & h_1 \\ \mathbf{h_1} & 0 & \mathbf{0} & 0 & \mathbf{0} & h_4 & \mathbf{h_3} & h_2 \\ \mathbf{h_2} & h_1 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{h_4} & h_3 \\ \mathbf{h_3} & h_2 & \mathbf{h_1} & 0 & \mathbf{0} & 0 & \mathbf{0} & h_4 \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ w_{14} \\ \mathbf{0} \\ w_{11} \\ \mathbf{0} \\ w_{12} \\ \mathbf{0} \\ w_{13} \end{bmatrix} = \begin{bmatrix} h_3 w_{14} + h_1 w_{11} \\ h_4 w_{14} + h_2 w_{11} \\ h_3 w_{11} + h_1 w_{12} \\ h_4 w_{11} + h_2 w_{12} \\ h_3 w_{12} + h_1 w_{13} \\ h_4 w_{12} + h_2 w_{13} \\ h_1 w_{14} + h_3 w_{13} \\ h_2 w_{14} + h_4 w_{13} \end{bmatrix}. \tag{4.563}$$

In a similar way, we augment multiplication (4.561):

$$\begin{bmatrix} h_4 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{-h_1} & h_2 & \mathbf{-h_3} \\ -h_3 & \mathbf{h_4} & 0 & \mathbf{0} & 0 & \mathbf{0} & -h_1 & \mathbf{h_2} \\ h_2 & \mathbf{-h_3} & h_4 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{-h_1} \\ -h_1 & \mathbf{h_2} & -h_3 & \mathbf{h_4} & 0 & \mathbf{0} & 0 & \mathbf{0} \\ 0 & \mathbf{-h_1} & h_2 & \mathbf{-h_3} & h_4 & \mathbf{0} & 0 & \mathbf{0} \\ 0 & \mathbf{0} & -h_1 & \mathbf{h_2} & -h_3 & \mathbf{h_4} & 0 & \mathbf{0} \\ 0 & \mathbf{0} & 0 & \mathbf{-h_1} & h_2 & \mathbf{-h_3} & h_4 & \mathbf{0} \\ 0 & \mathbf{0} & 0 & \mathbf{0} & -h_1 & \mathbf{h_2} & -h_3 & \mathbf{h_4} \end{bmatrix} \begin{bmatrix} s_{11} \\ \mathbf{0} \\ s_{12} \\ \mathbf{0} \\ s_{13} \\ \mathbf{0} \\ s_{14} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} h_4 s_{11} + h_2 s_{14} \\ -h_3 s_{11} - h_1 s_{14} \\ h_2 s_{11} + h_4 s_{12} \\ -h_1 s_{11} - h_3 s_{12} \\ h_2 s_{12} + h_4 s_{13} \\ -h_1 s_{12} - h_3 s_{13} \\ h_2 s_{13} + h_4 s_{14} \\ -h_1 s_{13} - h_3 s_{14} \end{bmatrix} \tag{4.564}$$

and with the wrapped round rearrangement:

$$\begin{bmatrix} \mathbf{-h_1} & h_2 & \mathbf{-h_3} & h_4 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & -h_1 & \mathbf{h_2} & -h_3 & \mathbf{h_4} & 0 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{-h_1} & h_2 & \mathbf{-h_3} & h_4 & \mathbf{0} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & -h_1 & \mathbf{h_2} & -h_3 & \mathbf{h_4} & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{-h_1} & h_2 & \mathbf{-h_3} & h_4 \\ \mathbf{h_4} & 0 & \mathbf{0} & 0 & \mathbf{0} & -h_1 & \mathbf{h_2} & -h_3 \\ \mathbf{-h_3} & h_4 & \mathbf{0} & 0 & \mathbf{0} & 0 & \mathbf{-h_1} & h_2 \\ \mathbf{h_2} & -h_3 & \mathbf{h_4} & 0 & \mathbf{0} & 0 & \mathbf{0} & -h_1 \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ s_{14} \\ \mathbf{0} \\ s_{11} \\ \mathbf{0} \\ s_{12} \\ \mathbf{0} \\ s_{13} \end{bmatrix} = \begin{bmatrix} h_2 s_{14} + h_4 s_{11} \\ -h_1 s_{14} - h_3 s_{11} \\ h_2 s_{11} + h_4 s_{12} \\ -h_1 s_{11} - h_3 s_{12} \\ h_2 s_{12} + h_4 s_{13} \\ -h_1 s_{12} - h_3 s_{13} \\ h_4 s_{14} + h_2 s_{13} \\ -h_3 s_{14} - h_1 s_{13} \end{bmatrix}. \tag{4.565}$$

By examining expressions (4.563) and (4.565) we may then deduce the following procedure for recovering the original signal from its tree wavelet expansion with a filter of length M.

(Continued)

Box 4.10 (Continued)

Step 1: Add zeros after each element of the wavelet coefficients vector of the signal at the coarsest level of resolution, and after every element of the scaling coefficients vector of the signal at the same level of resolution.

Step 2: Remove the last $M-1$ elements of the augmented vectors you created in the previous step and place them at the beginning of the corresponding vector.

Step 3: Convolve each vector you created with the corresponding filter. (Note that this is a proper convolution, **not** a pseudo-convolution, so the multiplications of the successive samples have to be done with the weights of the filters read in reverse order.) The result of convolution at each step is assigned to the position where the **first** element of the convolving filter is placed.

Step 4: Add the results of the two convolutions.

Step 5: Treat the result obtained at the previous step as the scaling coefficients vector of the signal at the next level up, that of the finer resolution. Combine it with the vector of wavelet coefficients at the same finer resolution level. Use these two vectors to repeat the process from step 1.

Example B4.130

Recover the original signal from its tree wavelet expansion of Example 4.126.
The initial vectors we have are

$$\mathbf{w_1} = (-2.285, 0.612, 0.742, 0.224)^T$$
$$\mathbf{w_2} = (3.260, -1.344)^T$$
$$\mathbf{s_2} = (3.153, -0.653)^T.$$

We apply the algorithm for $l = 2$, that is, using vectors $\mathbf{w_2}$ and $\mathbf{s_2}$:

Step 1: *Add zeros after each element:*

$$(3.260, 0, -1.344, 0)^T \tag{4.566}$$
$$(3.153, 0, -0.653, 0)^T. \tag{4.567}$$

Step 2: *Put the last $M-1$ elements at the beginning. In this case the filter is four taps long, so $M = 4$, and we transfer the last three elements of each sequence at the beginning of the sequence:*

$$(0, -1.344, 0, 3.260)^T \tag{4.568}$$
$$(0, -0.653, 0, 3.153)^T. \tag{4.569}$$

Step 3: *Convolve the sequences created in step 2 with the wavelet and scaling filter respectively. This is equivalent to saying: pseudo-convolve the sequences created in step 2 with the reverse of the wavelet filter and the reverse of the scaling filter respectively. Remember to assign the computed value to the first position of placement of the convolution filter. The wavelet filter we used was* $(0.1294, 0.2241, -0.8365, 0.4830)$ *and the scaling filter we used was* $(0.4830, 0.8365, 0.2241, -0.1294)$.

The reverse filters are $(0.4830, -0.8365, 0.2241, 0.1294)$ *and* $(-0.1294, 0.2241, 0.8365, 0.4830)$*, respectively.*
The outputs of these pseudo-convolutions are:

$$(1.546, 0.081, -2.901, 1.273)^T \tag{4.570}$$

$$(1.377, 2.722, 0.391, -0.954)^T. \tag{4.571}$$

Step 4: *Add the results of the two convolutions:*

$$(2.923, 2.803, -2.510, 0.319)^T.$$

Step 5: *Treat the result as the scaling coefficients vector for level* $l = 1$*:*

$$\mathbf{s_1} = (2.923, 2.803, -2.510, 0.319)^T.$$

Next, apply the algorithm again for $l = 1$*, that is, using vectors* $\mathbf{w_1}$ *and* $\mathbf{s_1}$*:*

Step 1: *Add zeros after each element:*

$$(-2.285, 0, 0.612, 0, 0.742, 0, 0.224, 0)^T$$
$$(2.923, 0, 2.803, 0, -2.510, 0, 0.319, 0)^T.$$

Step 2: *Put the last three elements at the beginning:*

$$(0, 0.224, 0, -2.285, 0, 0.612, 0, 0.742)^T$$
$$(0, 0.319, 0, 2.923, 0, 2.803, 0, -2.510)^T.$$

Step 3: *Convolve the sequences created in step 2 with the wavelet filter and the scaling filter respectively:*

$$(-0.483, -0.404, 1.991, -0.967, -0.416, 0.462, -0.592, 0.409)^T$$
$$(1.483, 2.404, 2.009, 1.967, -0.584, -2.462, -0.408, 0.591)^T.$$

Step 4: *Add the results of the two convolutions:*

$$(1.000, 2.000, 4.000, 1.000, -1.000, -2.000, -1.000, 1.000)^T.$$

Step 5: *Treat the result as the scaling coefficients vector at level* $l = 0$*:*

$$\mathbf{s_0} = (1, 2, 4, 1, -1, -2, -1, 1)^T.$$

This is exactly the original signal of Example 4.126.

Example B4.131

Reconstruct the original signal from its tree wavelet expansion of Example 4.126 without the wavelet coefficients of level $l = 2$ **which are supposed to represent only noise. Plot the reconstructed signal to show the effect of omitting the wavelet coefficients of level** $l = 2$ **in the reconstruction.**
When we wish to reconstruct a signal without its content in a particular frequency band, we simply set all those coefficients to zero, and apply the same procedure as for the full reconstruction.

(Continued)

Example B4.131 (Continued)

In this example, this means that steps 1, 2 and 3 of the reconstruction process are identical to steps 1, 2 and 3 of Example 4.130, apart from the fact that sequences (4.566), (4.568) and (4.570) consist of zeros only. Then sequence $\mathbf{s_1}$ *of step 5 is identical to sequence (4.571). The remaining steps are straightforward repetition of the reconstruction algorithm.*

The reconstructed signal is:

$$\mathbf{s_0} = (-0.032, 0.871, 3.614, 1.132, 0.383, 0.437, -0.965, -0.440)^T. \tag{4.572}$$

In Figure 4.102 we plot the original and the approximated signal side by side.

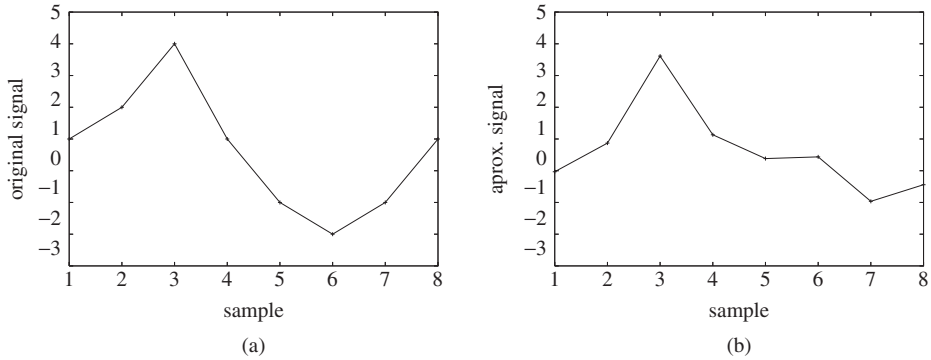


Figure 4.102 (a) Original signal. (b) Reconstructed signal after setting the wavelet coefficients of level $l = 2$ to zero.

Example B4.132

Reconstruct the original signal from its tree wavelet expansion of Example 4.126 without the weakest half coefficients, which are supposed to represent noise. Plot the reconstructed signal.

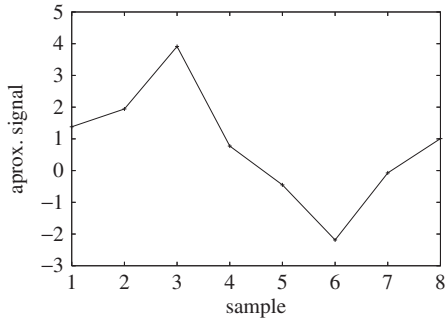


Figure 4.103 The signal of Figure 4.102a reconstructed from only the strongest half coefficients of the multi-resolution representation of the original signal.

The four weakest coefficients of the signal are identified by inspecting vectors $\mathbf{w_1}$, $\mathbf{w_2}$ *and* $\mathbf{s_2}$ *of the expansion of the signal of Example 4.126. They are:* $s_{22} = -0.653$, $w_{12} = 0.612$, $w_{13} = 0.742$, *and*

$w_{14} = 0.224$. *We set these coefficients to* 0 *and then reconstruct the signal. The reconstructed signal is* $(1.381, 1.943, 3.913, 0.773, -0.450, -2.187, -0.074, 1.008)^T$ *and is plotted in Figure 4.103. We observe that it looks quite similar to the original signal shown in Figure 4.102a, in spite of the fact that it was reconstructed from only half the coefficients of the multi-resolution representation of the original signal.*

Example B4.133

Show that if a function is produced by the cross-correlation (pseudo-convolution) of a signal with a filter, the signal may be recovered if the function is convolved with the filter, as long as the filter has unit energy at the desired frequencies.

From Example 4.109 we know that if a signal $f(t)$*, with Fourier transform* $F(\omega)$*, is cross-correlated with filter* $h(t)$*, with Fourier transform* $H(\omega)$*, to produce function* $s(\tau)$*, with Fourier transform* $S(\omega)$*, their Fourier transforms obey:*

$$S(\omega) = F(\omega)H^*(\omega). \tag{4.573}$$

Let us multiply both sides of this equation with $H(\omega)$*:*

$$S(\omega)H(\omega) = F(\omega)H^*(\omega)H(\omega) = F(\omega)|H(\omega)|^2 \simeq F(\omega). \tag{4.574}$$

The last result is because the power of the filter at the frequencies of interest is assumed to be 1*. So, in those frequencies, the signal may be recovered by multiplying the Fourier transforms of the given function* $s(\tau)$ *with the Fourier transform of the filter, point by point and taking the inverse Fourier transform. However, multiplication in the frequency domain is equivalent to convolution in the time domain, so from the knowledge of coefficients* $s(\tau)$ *one may recover the original signal by convolution with the filter used to produce the coefficients.*

Example B4.134

In view of Equations (4.530), the algorithm in Box 4.10 and Example 4.133, show that the two scale equations given by (4.526) and (4.528) can be written as

$$\phi(t) = \sqrt{2}\sum_n G(n)\phi(2t - n) \tag{4.575}$$

$$\psi(t) = \sqrt{2}\sum_n H(n)\psi(2t - n) \tag{4.576}$$

where $G(n)$ **and** $H(n)$ **are the low pass and the high pass filters that correspond to the scaling function and the associated mother wavelet. A function that obeys an equation like Equation (4.575) is called *refinable function*, while an equation of this type is known as *refinable equation*.**

First, we have to remember the conclusion we drew from Equations (4.532) that the coefficients of the expansion of a signal in a space with scaling parameter $l + 1$ *are extracted from the low pass version of the same signal in the space with scale* l*. We also remember that this is done by*

(Continued)

Example B4.134 (Continued)

cross-correlating (pseudo-convolving) the low pass version of the signal at scale l with the two filters we have, in order to produce its scaling and wavelet coefficients at level $l+1$. In Example 4.133 we show that the inverse of cross-correlation is convolution. So, the signal at level l may be recovered from the coefficients at level $l+1$ by convolution with the filter used to produce level $l+1$ from l in the first place. Then we observe that the right-hand sides of (4.575) and (4.576) represent convolutions of the scaling or wavelet functions in, say, level $l+1$, with the corresponding filters. Realising then that the sums on the right-hand sides of (4.575) and (4.576) are actually $\phi_{l+1,k}(2t)$ and $\psi_{l+1,k}(2t)$, respectively, we appreciate that (4.575) and (4.576) are equivalent to Equations (4.526) and (4.528), respectively.

How many different wavelet filters exist?

There is no limit on how many wavelet filters one may construct. However, there are a few filters that are often used in image processing. They are listed in Table 4.6.

Example 4.135

Let us consider a scaling four-tap filter g_1, g_2, g_3, g_4 that is appropriate for the wavelet expansion of a signal. Show that the sum of the filter weights is $\sqrt{2}$, i.e. $\sum_i g_i = \sqrt{2}$.
Since this filter is appropriate for wavelet analysis, a vector created by shifting it by two positions to the right should be orthogonal to the original scaling vector. Therefore, its weights must satisfy:

$$g_1 g_3 + g_2 g_4 = 0 \tag{4.577}$$

In addition, this filter is used to form a unit vector of expansion, so its norm is 1:

$$g_1^2 + g_2^2 + g_3^2 + g_4^2 = 1. \tag{4.578}$$

Finally, from this scaling filter we must be able to construct a wavelet filter which must have 0 dc. The corresponding wavelet filter is $g_4, -g_3, g_2, -g_1$, and it must satisfy:

$$g_4 - g_3 + g_2 - g_1 = 0. \tag{4.579}$$

If we multiply Equation (4.577) with 2 and add it to Equation (4.578), we obtain:

$$g_1^2 + g_2^2 + g_3^2 + g_4 + 2g_1 g_3 + 2g_2 g_4 = 1 \Rightarrow (g_1 + g_3)^2 + (g_2 + g_4)^2 = 1. \tag{4.580}$$

From Equation (4.579) we have that $g_4 + g_2 = g_3 + g_1$. If we use this in (4.580) we obtain:

$$(g_1 + g_3)^2 + (g_2 + g_4)^2 = 2(g_1 + g_3)^2 = 2(g_2 + g_4)^2 = 1 \Rightarrow g_1 + g_3 = g_2 + g_4 = \frac{1}{\sqrt{2}}. \tag{4.581}$$

Therefore

$$g_1 + g_2 + g_3 + g_4 = \frac{2}{\sqrt{2}} = \sqrt{2}. \tag{4.582}$$

Table 4.6 Some commonly used scaling filters. These filters are appropriate for image reconstruction. If the filters are to be used for texture feature extraction, all values of the Haar and Daubechies filters have to be divided by $\sqrt{2}$ so that the sum of the coefficients of each filter is 1 (see Example 4.135). The reason is that smoothing filters should leave unaltered a flat signal. This is only possible if $\sum_i g_i = 1$. The Coiflet filters already have values that sum up to 1. From these scaling filters one may easily work out the corresponding wavelet filters using Equation (4.544)

Haar	Daubechies 4	Daubechies 6	Daubechies 8
0.707107	0.482963	0.332671	0.230378
0.707107	0.836516	0.806892	0.714847
	0.224144	0.459877	0.630881
	−0.129410	−0.135011	−0.027984
		−0.085441	−0.187035
		0.035226	0.030841
			0.032883
			−0.010597
Daubechies 20	Coiflet 12	Coiflet 18	Coiflet 24
0.026670	0.011588	−0.002682	0.000631
0.188177	−0.029320	0.005503	−0.001152
0.527201	−0.047640	0.016584	−0.005195
0.688459	0.273021	−0.046508	0.011362
0.281172	0.574682	−0.043221	0.018867
−0.249846	0.294867	0.286503	−0.057464
−0.195946	−0.054086	0.561285	−0.039653
0.127369	−0.042026	0.302984	0.293667
0.093057	0.016744	−0.050770	0.553126
−0.071394	0.003968	−0.058196	0.307157
−0.029458	−0.001289	0.024434	−0.047113
0.033213	−0.000510	0.011229	−0.068038
0.003607		−0.006370	0.027814
−0.010733	Coiflet 6	−0.001820	0.017736
0.001395	−0.051430	0.000790	−0.010756
0.001992	0.238930	0.000330	−0.004001
−0.000686	0.602859	−0.000050	0.002653
−0.000116	0.272141	−0.000024	0.000896
0.000094	−0.051430		−0.000417
−0.000013	−0.011070		−0.000184
			0.000044
			0.000022
			−0.000002
			−0.000001

How may we use wavelets to process images?

Images are two-dimensional signals. So, for an image we may create four different combinations of a low and a high pass filters: use the low pass filter along both axes, use the high pass filter along both axes, and use the low pass filter along one axis and the high pass filter along the other axis. Therefore, at each level of resolution, instead of splitting the signal into two components, we split the image into four. Further, image processing is all about presenting redundant information in several different ways, something that adds robustness to the system and at the same time allows different image characteristics to become obvious by different representations. So, for texture analysis, we are not interested in image reconstruction from the minimum number of bits, so we do not, in general, use tree wavelet expansion, but packet wavelet expansion. A schematic representation of such analysis is shown in Figure 4.104. Packet wavelets allow the option to zoom in on some high frequencies if we like, by analysing at each level only the band with the highest energy. The tree we construct this way is called a **structure tree**. Figure 4.105 shows an example tessellation of the frequency space which corresponds to the structure tree shown in Figure 4.106.

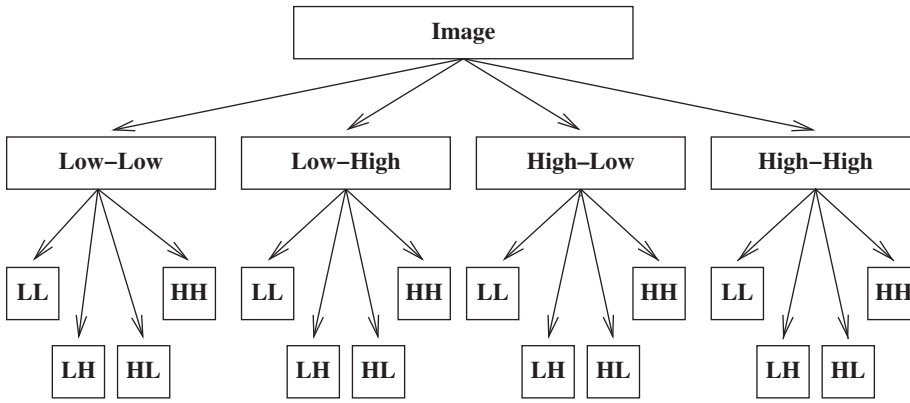


Figure 4.104 The packet wavelet representation of an image.

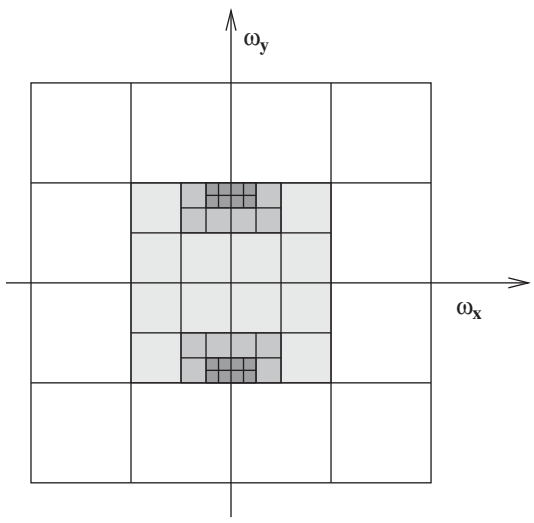


Figure 4.105 The tessellation of the frequency space that corresponds to the structure tree of Figure 4.106. The increasing darkness of the grey shade used to identify the analysed bands corresponds to the increasing level of zooming in.

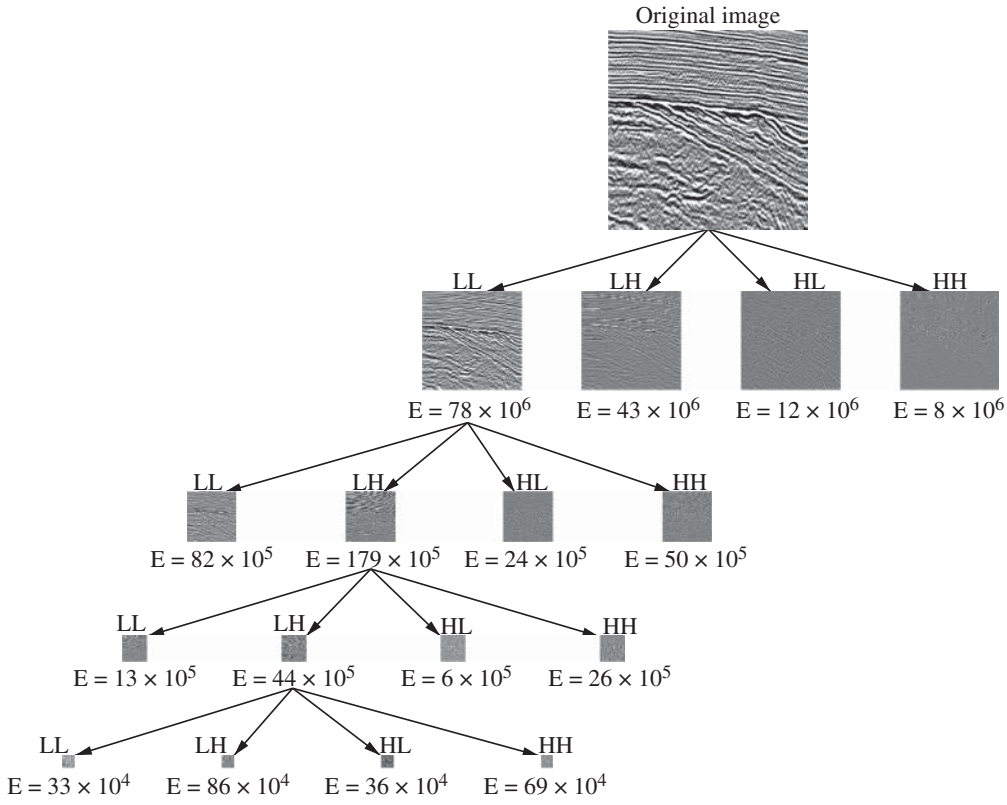


Figure 4.106 Zooming into a frequency band with the packet wavelet transform may be achieved by preferentially expanding a chosen band at each level of resolution. We usually choose the band with the most energy. The energy of a band is computed by adding the squares of the values of the individual pixels.

Example 4.136

The image of Figure 4.81 is 255×255 **in size. Consider its** 256×256 **version and construct its tree wavelet representation for three levels of resolution, using the four-tap wavelet filter of Table 4.6.**

First we pseudo-convolve the image in one direction with low pass filter $(0.4830, 0.8365, 0.2241, -0.1294)$*, and the result with the same filter in the other direction, assigning the output of each convolution step to the first position of the filter each time. (This clearly creates a shift of the whole image, but we do it here just to be consistent with the concepts we developed for 1D signals. In general, we assign the output to one of the two central pixels of the filter.) Then we remove from the output every second line and every second column, to produce the LL band for level* 1*. If we pseudo-convolve with filter* $(0.1294, 0.2241, -0.8365, 0.4830)$ *in the second direction and sub-sample in the same way, we shall produce band LH for level* 1*. To produce band HL we use the two filters in reverse order, and to produce band HH we use the high pass filter* $(0.1294, 0.2241, -0.8365, 0.4830)$ *in both directions.*

We repeat the process using as input image band LL to produce in the same way the corresponding bands for level 2*, and then again for level* 3*. The final result is shown in Figure 4.107.*

(Continued)

Example 4.136 (Continued)

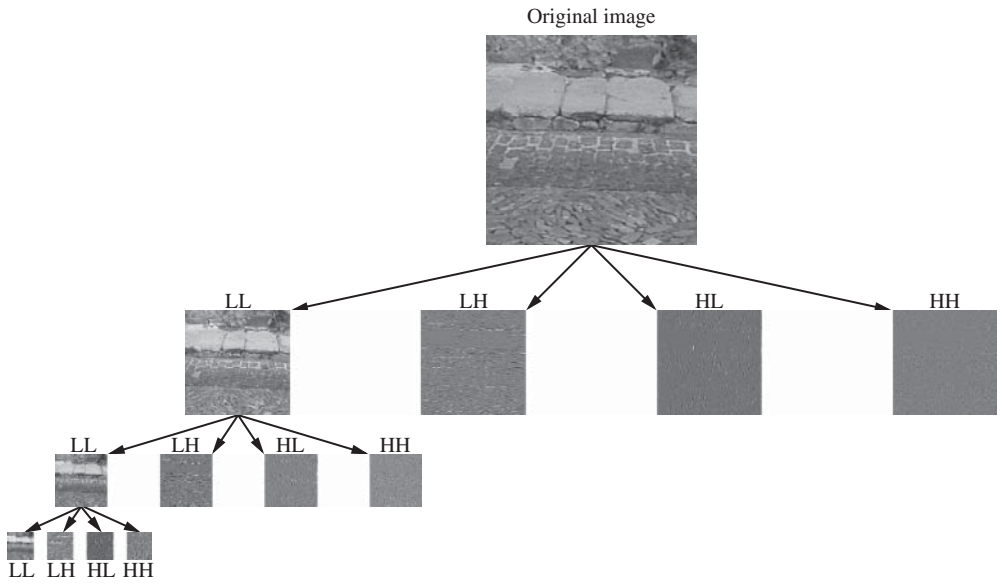


Figure 4.107 Tree wavelet representation of the image shown in Figure 4.81.

Example 4.137

Consider the image of Figure 4.81 and construct its packet wavelet representation for two levels of resolution, using the four-tap wavelet filter of Table 4.6.

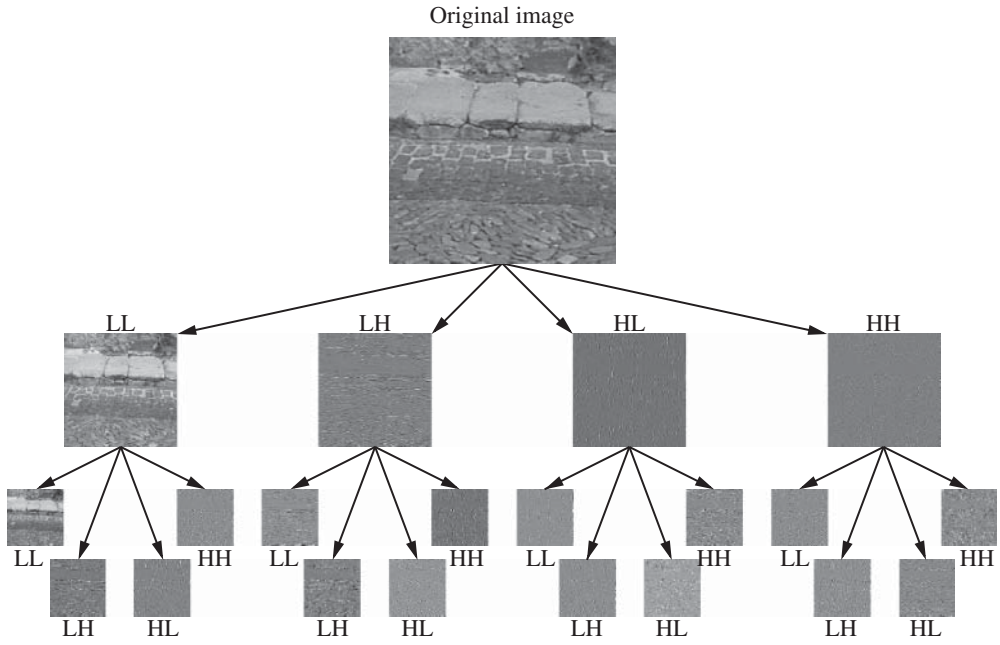


Figure 4.108 Packet wavelet representation of the image shown in Figure 4.81.

The only difference between this case and the previous one is that here we also expand bands LH, HL *and* HH *in addition to expanding band* LL. *The results are shown in Figure 4.108.*

Example 4.138

Construct the structure tree for the image of Figure 3.1f for four levels of analysis and show the corresponding tessellation of the frequency space.

Figure 4.109 shows the structure tree. The number under each band is the corresponding energy of the band.

The tessellation of the frequency space is shown in Figure 4.110.

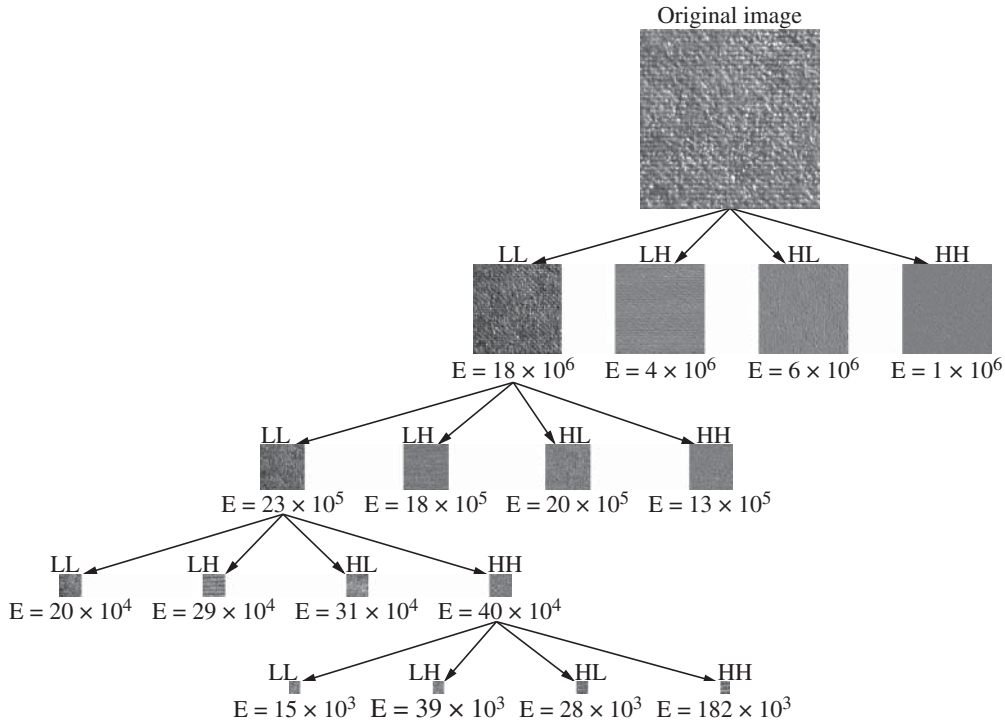
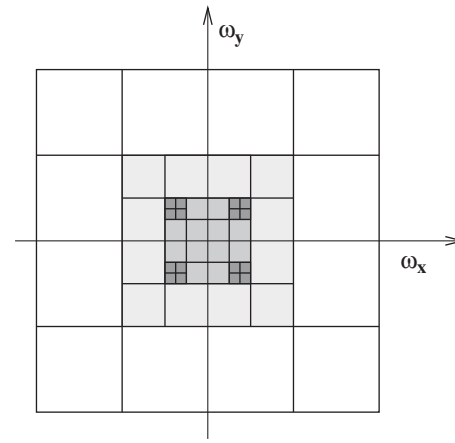


Figure 4.109 The structure tree for the texture image fabric 2.

Figure 4.110 The tessellation of the frequency space that corresponds to the structure tree of Figure 4.109.



How may we use wavelets to construct texture features?

First we use the wavelet and scaling filters to construct the structure tree of the image. So that all bands produced this way have the same size as the original image, we do not perform sub-sampling for the bands we retain, and we perform four-fold sub-sampling and interlacing for the bands we split. Sub-sampling in 2D would have meant that only a quarter of the original pixels are kept, since every second row and every second column of the original grid is omitted. However, this way, there is an asymmetry with which the samples are treated. Why should we keep every first sample and discard every second and not vice versa? Four-fold sub-sampling and interlacing restores parity among the pixels. This is the so-called **maximum overlap** algorithm. So in this approach there is a large amount of redundancy built in. One crucial point is to be able to decide the number of levels of resolution we must use. This is usually data-driven. For example, a threshold might be chosen, and all bands with energy higher than the threshold are split further.

Just as in the case of Gabor coefficients, the wavelet coefficients are not used as texture features because they exhibit great variability within the same texture. Features are extracted from the wavelet coefficients by using the local energy value.

What is the maximum overlap algorithm?

Figure 4.111 shows the steps of this algorithm for up to three levels of analysis. If only one level of analysis is used, the process is trivial: we filter the original image to produce the four bands LL, LH, HL and HH and without sub-sampling them we exit the algorithm.

If two levels of analysis are used, we follow these steps.

Step 1: Filter the original image to produce bands LL, LH, HL and HH.

Step 2: Choose the band with the maximum energy for further analysis and output the remaining three bands. Let us say that the chosen band is band XX.

Step 3: Sub-sample XX in four different ways to produce XX1, XX2, XX3 and XX4, each a quarter the size of the original image.

Step 4: Process each of these four bands with the two filters, so from each one of them you produce four bands. Call them bands XX1LL, XX1LH, XX1HL, XX1HH, XX2LL, XX2LH, XX2HL, XX2HH, XX3LL, XX3LH, etc.

Step 5: Interlace bands XX1LL, XX2LL, XX3LL and XX4LL to produce band XXLL which has the same size as the original image. Also, interlace bands XX1LH, XX2LH, XX3LH and XX4LH to produce band XXLH, and so on, to produce bands XXHL and XXHH.

This algorithm is schematically shown in Figure 4.112.

For three levels of analysis, one has to perform all 16 combinations of two-stage sub-sampling and interlacing. However, in practice one may implement this algorithm in a much more efficient way. See Examples 4.139–4.145 for that.

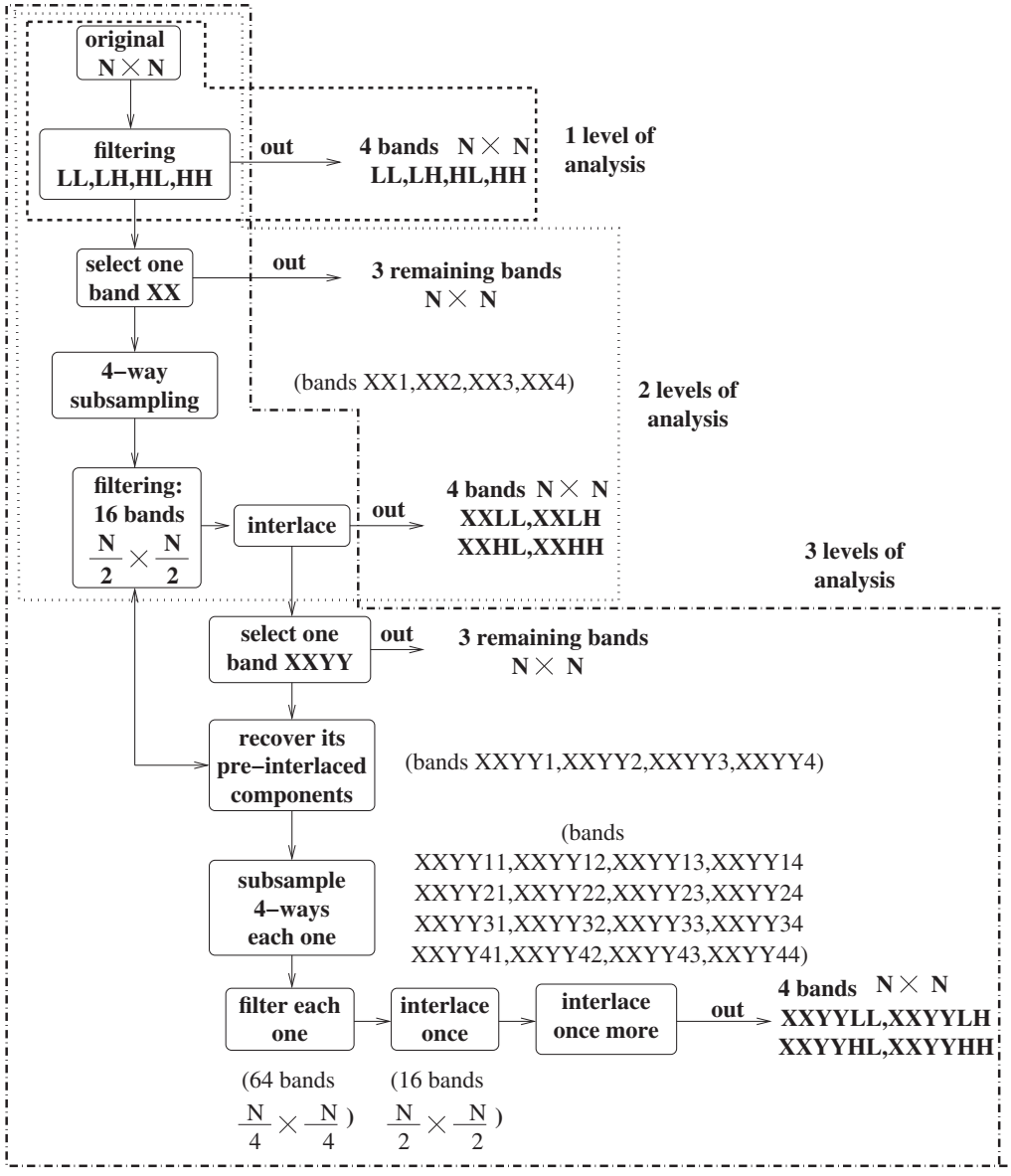


Figure 4.111 Three levels of analysis with the maximum overlap algorithm. The analyses up to one, two or three levels of resolution are identified within frames drawn with different styles of line.

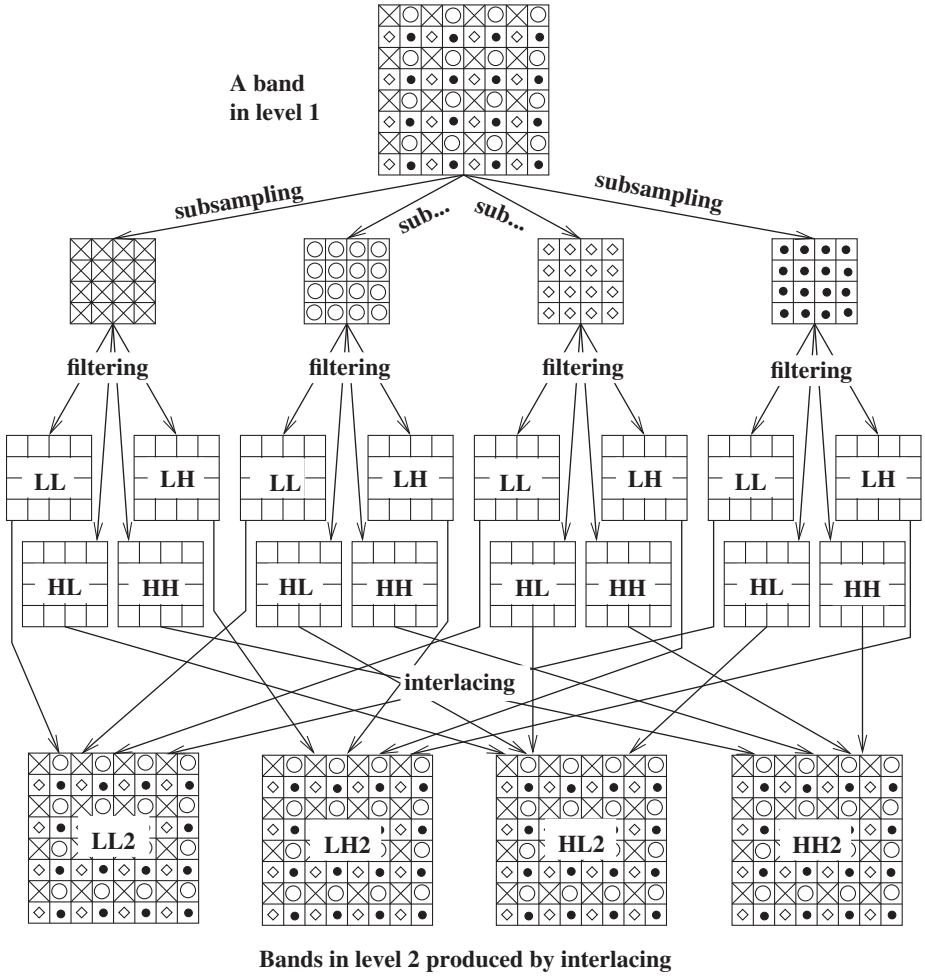


Figure 4.112 At the top, a band that has to be analysed. It is sub-sampled in four different ways, with each sub-sampled band made up from the pixels marked with the same symbol in the top band. Each sub-sampled band is processed by the low and high pass wavelet filters. The corresponding bands produced by the four sub-sampled bands are interleaved to produce the bands for the next level of analysis, which have the same size as the original band.

Example 4.139

Use the maximum overlap algorithm to produce features for the image of Figure 4.81. *As we did in all previous examples of texture analysis, the mean value of the image is removed before the analysis starts, i.e. the image has* 0 *value for* 0 *frequency. We use the Daubechies* 4 *and Daubechies* 20 *scaling filters of Table 4.6. First, we construct the maximum overlap structure tree of this image which for filter Daubechies 4 is shown in Figure 4.113. In this figure the values of each channel were scaled independently to the range* [0,255] *for better visualisation. The energy of each band is given under it. We can see that for this image the band with the maximum energy for both levels of resolution happens to be the* LL *band. This indicates that the image has strong low-frequency components.*

Next, we compute the local energy for each pixel in each band. To do this we may either use a square flat window and find the straight average of the squared values of the wavelet coefficients inside the window, or use a Gaussian window. When we use a Gaussian window care should be taken so that its weights sum up to 1*. The standard deviation of the Gaussian is chosen so that its weights at the edges of the window are reasonably close to* 0*. Then we use those weights to compute the weighted average of the squared values of the wavelet coefficients inside the window.*

In either case, the value we compute is assigned to the central pixel of the window and it serves as an estimate of the local energy of the image at the particular band, in the vicinity of that pixel. The values of these local energies of the same pixel across the different bands constitute the feature vector of the pixel.

By considering the various filtered outputs we created and various ways of estimating the local energy, we may produce various trees of features for this image.

An example of such a tree of features is shown in Figure 4.114 for filter Daubechies 20 *and a Gaussian averaging window of size* 25×25*. Here each band was scaled independently to the range* [0,255] *for better visualisation.*

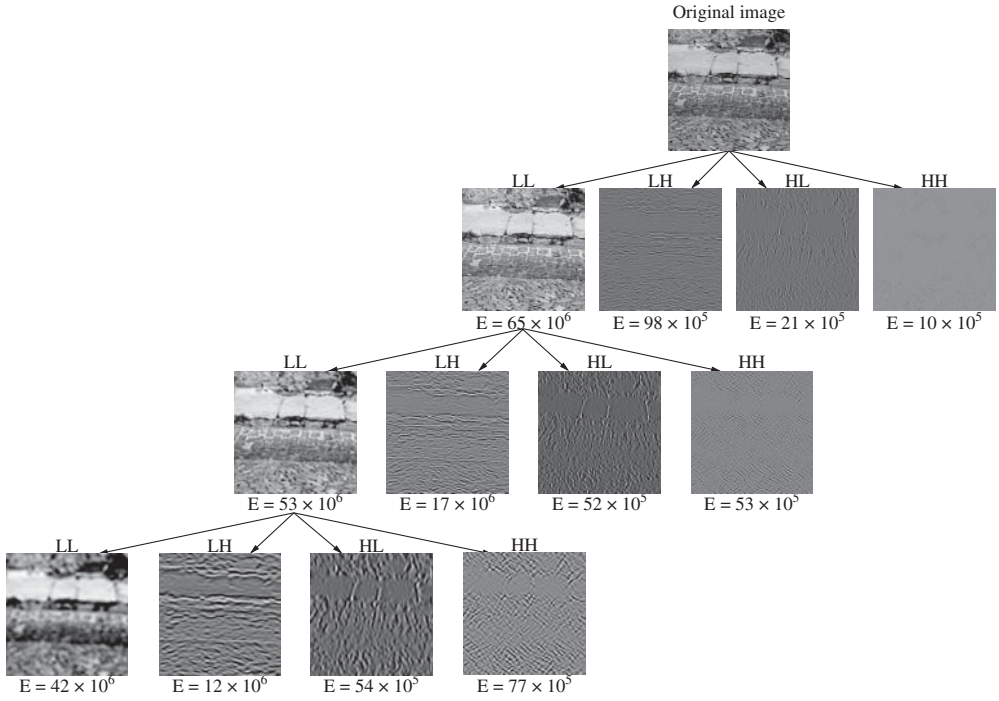


Figure 4.113 Maximum overlap structure tree of the image shown in Figure 4.81, constructed using filter Daubechies 4.

(Continued)

Example 4.139 (Continued)

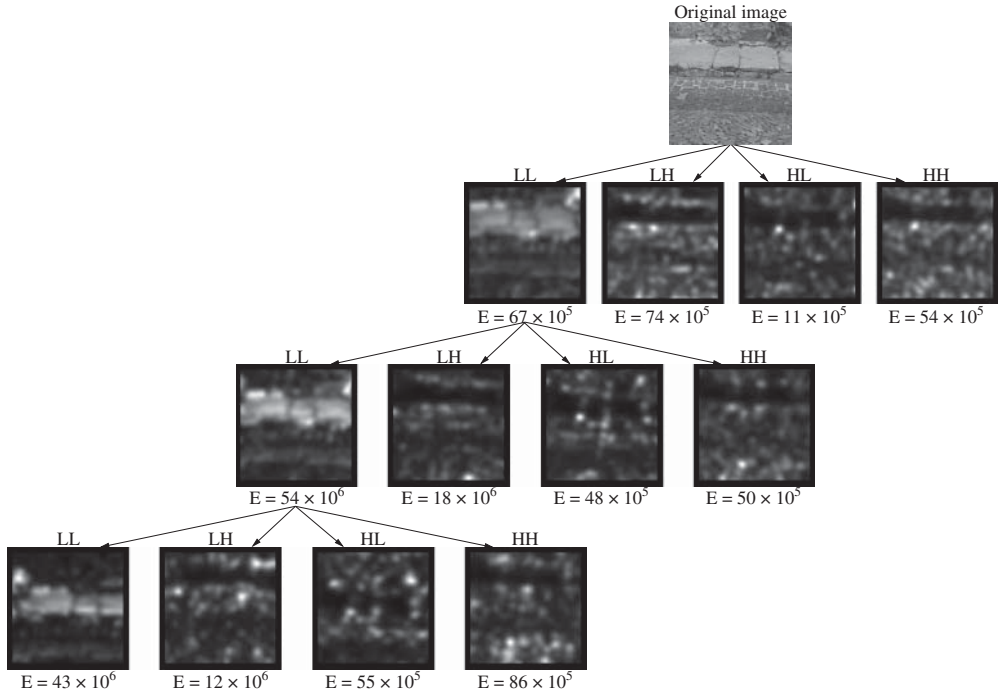


Figure 4.114 Averaged energies of the coefficients of the maximum overlap analysis with filter Daubechies 20 of the image in Figure 4.81a.

Example B4.140

Can you perform the process of sub-sampling by a factor of 2, by multiplying an image with a matrix?

Let us say that an image B is of size $N \times N$. Its sub-sampled version by a factor of 2 will be an image of size $N/2 \times N/2$. It is not possible to perform sub-sampling by simply multiplying image B with a matrix, because such a matrix should be of size $N/2 \times N$ in order to be appropriate for multiplication with B, and it will produce a result also of size $N/2 \times N$, not the desirable $N/2 \times N/2$. To reduce both dimensions of matrix B, we must multiply it from both sides, left and right with matrices of size $N/2 \times N$ and $N \times N/2$, respectively.

Example B4.141

For an 8×8 image construct appropriate matrices which, when used to multiply the original image from left and right, will sub-sample it by a factor of 2 in four different ways.

Let us say that the original image is:

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} & b_{17} & b_{18} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} & b_{27} & b_{28} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} & b_{37} & b_{38} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} & b_{47} & b_{48} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & b_{56} & b_{57} & b_{58} \\ b_{61} & b_{62} & b_{63} & b_{64} & b_{65} & b_{66} & b_{67} & b_{68} \\ b_{71} & b_{72} & b_{73} & b_{74} & b_{75} & b_{76} & b_{77} & b_{78} \\ b_{81} & b_{82} & b_{83} & b_{84} & b_{85} & b_{86} & b_{87} & b_{88} \end{bmatrix}. \tag{4.583}$$

The four sub-sampled versions of this image are:

$$B1 = \begin{bmatrix} b_{11} & b_{13} & b_{15} & b_{17} \\ b_{31} & b_{33} & b_{35} & b_{37} \\ b_{51} & b_{53} & b_{55} & b_{57} \\ b_{71} & b_{73} & b_{75} & b_{77} \end{bmatrix} \quad B2 = \begin{bmatrix} b_{12} & b_{14} & b_{16} & b_{18} \\ b_{32} & b_{34} & b_{36} & b_{38} \\ b_{52} & b_{54} & b_{56} & b_{58} \\ b_{72} & b_{74} & b_{76} & b_{78} \end{bmatrix}$$

$$B3 = \begin{bmatrix} b_{21} & b_{23} & b_{25} & b_{27} \\ b_{41} & b_{43} & b_{45} & b_{47} \\ b_{61} & b_{63} & b_{65} & b_{67} \\ b_{81} & b_{83} & b_{85} & b_{87} \end{bmatrix} \quad B4 = \begin{bmatrix} b_{22} & b_{24} & b_{26} & b_{28} \\ b_{42} & b_{44} & b_{46} & b_{48} \\ b_{62} & b_{64} & b_{66} & b_{68} \\ b_{82} & b_{84} & b_{86} & b_{88} \end{bmatrix}. \tag{4.584}$$

Let us define two matrices S_a and S_b as follows:

$$S_a \equiv \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad S_b \equiv \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.585}$$

We may easily verify then that:

$$B1 = S_a B S_a^T \qquad B2 = S_a B S_b^T \qquad B3 = S_b B S_a^T \qquad B4 = S_b B S_b^T. \tag{4.586}$$

Example B4.142

Write down a formula that will allow you to reconstruct image B of Example 4.141 from its sub-sampled versions, with the help of matrices (4.585).

By simple inspection, we can verify that the upsampling formula is:

$$B = S_a^T B1 S_a + S_a^T B2 S_b + S_b^T B3 S_a + S_b^T B4 S_b. \tag{4.587}$$

Example B4.143

Write down two matrices that can be used to filter a 4×4 image with a low pass filter (g_1, g_2) and a high pass filter (h_1, h_2) and produce the LL, LH, HL and HH versions of it.

(Continued)

Example B4.143 (Continued)

We create the matrices we need by writing the scaling and wavelet vectors of size 1×4 *one under the other:*

$$G = \begin{bmatrix} g_1 & g_2 & 0 & 0 \\ 0 & g_1 & g_2 & 0 \\ 0 & 0 & g_1 & g_2 \\ g_2 & 0 & 0 & g_1 \end{bmatrix} \qquad H = \begin{bmatrix} h_1 & h_2 & 0 & 0 \\ 0 & h_1 & h_2 & 0 \\ 0 & 0 & h_1 & h_2 \\ h_2 & 0 & 0 & h_1 \end{bmatrix}. \tag{4.588}$$

We may easily then verify that the LL, LH, HL *and* HH *bands of a* 4×4 *image* A *may be produced by operating upon* A *from left and right with combinations of matrices* G *and* H *and their transpositions:*

$$A_{\rm LL} = GAG^T \qquad A_{\rm LH} = GAH^T \qquad A_{\rm HL} = HAG^T \qquad A_{\rm HH} = HAH^T. \tag{4.589}$$

Example B4.144

Using the results of Examples 4.141–4.143, write down matrix formulae that express the sequence of four-fold sub-sampling of an 8×8 **matrix** B**, its four-band filtering and the subsequent interlacing to produce matrices in bands LL, LH, HL and HH of the same size as the original image** B**.**

We have first to sub-sample matrix B *to produce matrices* $B1$, $B2$, $B3$ *and* $B4$ *as shown in Example 4.141. Each one of these matrices is of size* 4×4*, and it may be filtered with matrices* G *and* H *of Example 4.143 to produce the following matrices:*

$$\begin{aligned} B1_{\rm LL} &= GB1G^T = GS_aBS_a^TG^T \\ B1_{\rm LH} &= GB1H^T = GS_aBS_a^TH^T \\ B1_{\rm HL} &= HB1G^T = HS_aBS_a^TG^T \\ B1_{\rm HH} &= HB1H^T = HS_aBS_a^TH^T \end{aligned} \tag{4.590}$$

$$\begin{aligned} B2_{\rm LL} &= GB2G^T = GS_aBS_b^TG^T \\ B2_{\rm LH} &= GB2H^T = GS_aBS_b^TH^T \\ B2_{\rm HL} &= HB2G^T = HS_aBS_b^TG^T \\ B2_{\rm HH} &= HB2H^T = HS_aBS_b^TH^T \end{aligned} \tag{4.591}$$

$$\begin{aligned} B3_{\rm LL} &= GB3G^T = GS_bBS_a^TG^T \\ B3_{\rm LH} &= GB3H^T = GS_bBS_a^TH^T \\ B3_{\rm HL} &= HB3G^T = HS_bBS_a^TG^T \\ B3_{\rm HH} &= HB3H^T = HS_bBS_a^TH^T \end{aligned} \tag{4.592}$$

$$\begin{aligned} B4_{\rm LL} &= GB4G^T = GS_bBS_b^TG^T \\ B4_{\rm LH} &= GB4H^T = GS_bBS_b^TH^T \end{aligned}$$

$$B4_{\rm HL} = HB4G^T = HS_bBS_b^TG^T$$
$$B4_{\rm HH} = HB4H^T = HS_bBS_b^TH^T. \tag{4.593}$$

The next step is to interlace the corresponding matrices to create matrices of size 8×8*. We do that by using the formula of Example 4.142:*

$$\begin{aligned}
B_{\rm LL} &= S_a^TB1_{LL}S_a + S_a^TB2_{LL}S_b + S_b^TB3_{LL}S_a + S_b^TB4_{LL}S_b\\
&= S_a^TGS_aBS_a^TG^TS_a + S_a^TGS_aBS_b^TG^TS_b +\\
&\quad S_b^TGS_bBS_a^TG^TS_a + S_b^TGS_bBS_b^TG^TS_b\\
&= (S_a^TGS_a)B(S_a^TGS_a)^T + (S_a^TGS_a)B(S_b^TGS_b)^T +\\
&\quad (S_b^TGS_b)B(S_a^TGS_a)^T + (S_b^TGS_b)B(S_b^TGS_b)^T\\
B_{\rm LH} &= S_a^TB1_{LH}S_a + S_a^TB2_{LH}S_b + S_b^TB3_{LH}S_a + S_b^TB4_{LH}S_b\\
&= S_a^TGS_aBS_a^TH^TS_a + S_a^TGS_aBS_b^TH^TS_b +\\
&\quad S_b^TGS_bBS_a^TH^TS_a + S_b^TGS_bBS_b^TH^TS_b\\
&= (S_a^TGS_a)B(S_a^THS_a)^T + (S_a^TGS_a)B(S_b^THS_b)^T +\\
&\quad (S_b^TGS_b)B(S_a^THS_a)^T + (S_b^TGS_b)B(S_b^TGS_b)^T\\
B_{\rm HL} &= S_a^TB1_{HL}S_a + S_a^TB2_{HL}S_b + S_b^TB3_{HL}S_a + S_b^TB4_{HL}S_b\\
&= S_a^THS_aBS_a^TG^TS_a + S_a^THS_aBS_b^TG^TS_b +\\
&\quad S_b^THS_bBS_a^TG^TS_a + S_b^THS_bBS_b^TG^TS_b\\
&= (S_a^THS_a)B(S_a^TGS_a)^T + (S_a^THS_a)B(S_b^TGS_b)^T +\\
&\quad (S_b^THS_b)B(S_a^TGS_a)^T + (S_b^THS_b)B(S_b^TGS_b)^T\\
B_{\rm HH} &= S_a^TB1_{HH}S_a + S_a^TB2_{HH}S_b + S_b^TB3_{HH}S_a + S_b^TB4_{HH}S_b\\
&= S_a^THS_aBS_a^TH^TS_a + S_a^THS_aBS_b^TH^TS_b +\\
&\quad S_b^THS_bBS_a^TH^TS_a + S_b^THS_bBS_b^TH^TS_b\\
&= (S_a^THS_a)B(S_a^THS_a)^T + (S_a^THS_a)B(S_b^THS_b)^T +\\
&\quad (S_b^THS_b)B(S_a^THS_a)^T + (S_b^THS_b)B(S_b^THS_b)^T.
\end{aligned} \tag{4.594}$$

We see that in order to perform sub-sampling, filtering and interlacing, we must operate on the original matrix B from left and right with combinations of only four distinct matrices:

$$\tilde{G}_1 \equiv S_a^TGS_a \qquad \tilde{G}_2 \equiv S_b^TGS_b$$
$$\tilde{H}_1 \equiv S_a^THS_a \qquad \tilde{H}_2 \equiv S_b^THS_b. \tag{4.595}$$

By direct substitution from the definitions of matrices G, H, S_a *and* S_b*, we can easily work out matrices* $\tilde{G}_1$*,* $\tilde{G}_2$*,* $\tilde{H}_1$ *and* $\tilde{H}_2$*. We simply derive here the result for matrix* $\tilde{G}_1$*:*

$$\tilde{G}_1 = \begin{bmatrix} 1&0&0&0\\ 0&0&0&0\\ 0&1&0&0\\ 0&0&0&0\\ 0&0&1&0\\ 0&0&0&0\\ 0&0&0&1\\ 0&0&0&0 \end{bmatrix} \begin{bmatrix} g_1&g_2&0&0\\ 0&g_1&g_2&0\\ 0&0&g_1&g_2\\ g_2&0&0&g_1 \end{bmatrix} \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0 \end{bmatrix}$$

(Continued)

Example B4.144 (Continued)

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} g_1 & 0 & g_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_1 & 0 & g_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_1 & 0 & g_2 & 0 \\ g_2 & 0 & 0 & 0 & 0 & 0 & g_1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} g_1 & 0 & g_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_1 & 0 & g_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_1 & 0 & g_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_2 & 0 & 0 & 0 & 0 & 0 & g_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.596}$$

Example B4.145

Use the results of Example 4.144 to show how the maximum overlap algorithm may be applied in practice.

We can see from the formula of matrix $\tilde{G}_1$ and the corresponding formulae for $\tilde{G}_2$, $\tilde{H}_1$ and $\tilde{H}_2$ (which are not shown in Example 4.144), that the steps of sub-sampling and filtering may be performed by convolving the original image with the scaling and wavelet filters we are using, padded with 0*s so they pick up the pixels that are at the desired positions according to the type of desired sub-sampling each time. The results of the four-fold sub-samplings will simply then have to be added to produce the bands we require. So, although the maximum overlap algorithm appears at first sight to require a lot of storage space to maintain all possible combinations of sub-samplings at all levels, in practice it can be performed efficiently by using sparse convolution filters. Such filters may be thought of as having "holes", so such algorithms are called* **à trous** *algorithms, which in French means "with holes".*

Example 4.146

Construct the maximum overlap structure tree for the image of Figure 3.1c for four levels of analysis.

This maximum overlap structure tree is shown in Figure 4.115.

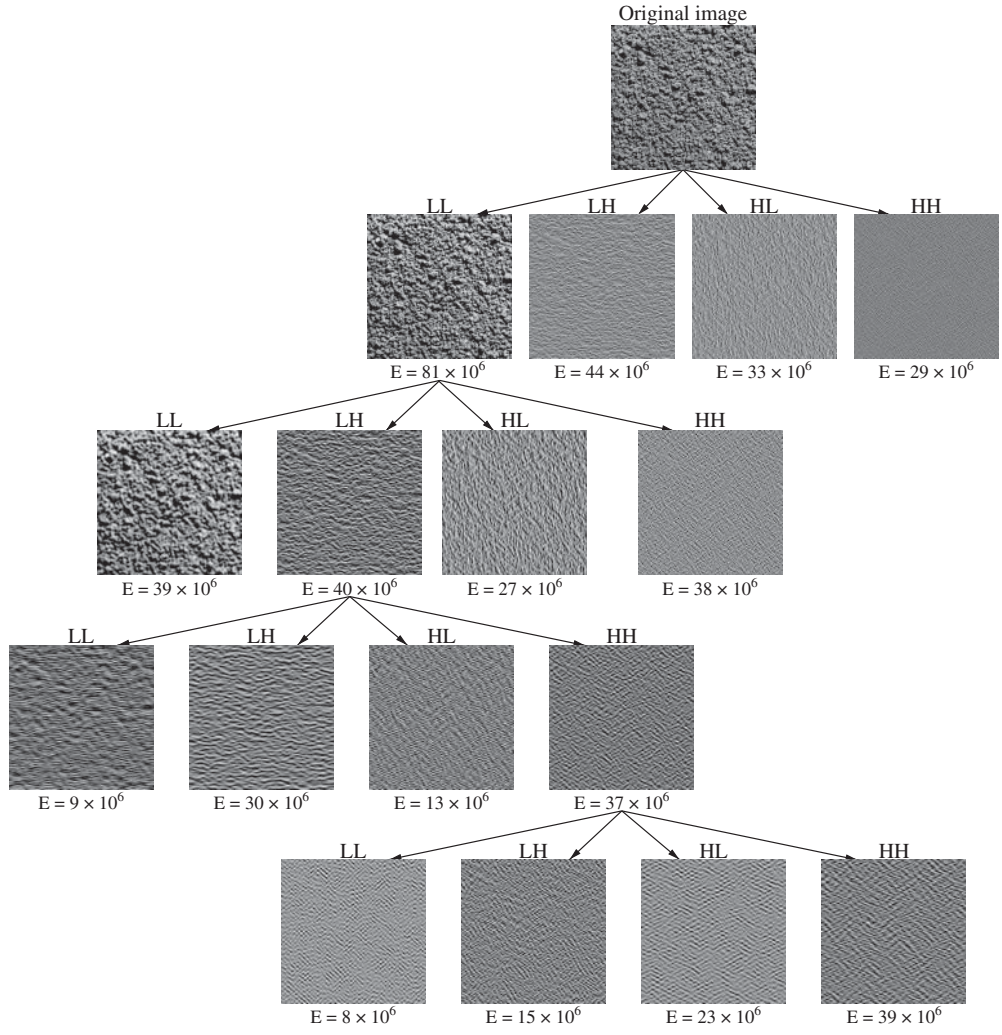


Figure 4.115 The maximum overlap structure tree for the texture image food. The number under each band is the corresponding energy of the band.

Example 4.147

Use the structure tree constructed in Example 4.146 to construct the tessellation of the frequency domain of Figure 3.1c and in it identify the band with the maximum energy.

Figure 4.116 shows the tessellation of the frequency domain and the grey squares identify the band with the maximum energy.

(Continued)

Example 4.147 (Continued)

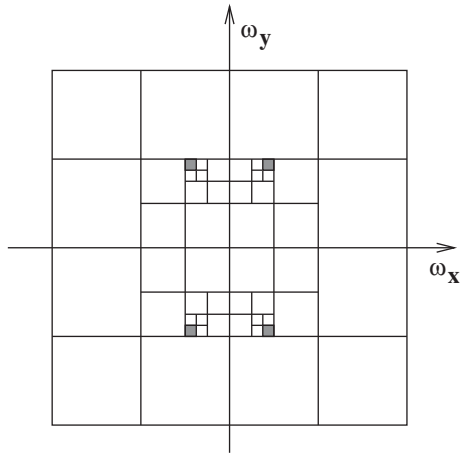


Figure 4.116 The tessellation of the frequency space that corresponds to the structure tree of Figure 4.115. The leftmost leaf of the bottom level of the tree has the most energy at that level and it corresponds to the shaded bands identified in this diagram.

What is the relationship between Gabor functions and wavelets?

Both these approaches try to answer the question "what is where" in an image. Each of them considers the whole answer space (which is the spatio-frequency four-dimensional domain with two spatial axes and two frequency axes) and creates a basis in terms of which any answer to the above question may be expressed as a linear combination of them.

The Gabor basis functions are generated by **modulation** and **translation** of the Gaussian function. The spatial dispersion and bandwidth of these functions are constant and independent of the modulation and translation parameters. This is shown schematically in Figure 4.117 for 1D signals (for which the answer space to the question "what is where" is 2D only). Figure 4.118 shows some of these functions for an 1D signal.

The wavelet basis is created by **scaling** and **translating** the mother wavelet and the scaling function. The time dispersion and bandwidth of the wavelet functions are not constant, but one may

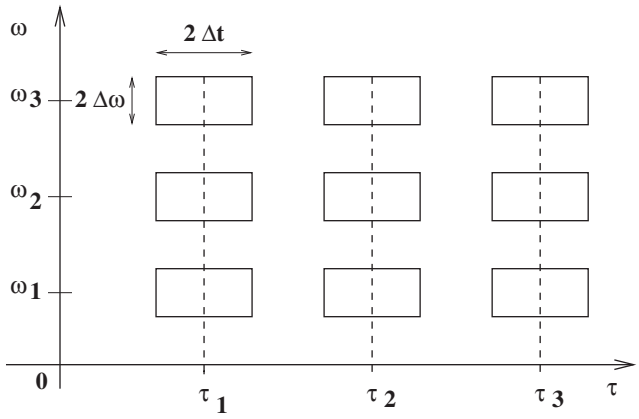


Figure 4.117 Time–frequency space showing schematically the resolution cells for Gabor functions. Each resolution cell is an elementary answer to the fundamental question "what is where" in the signal. Any answer concerning any signal of the same size may be expressed as a linear superposition of these elementary answers.

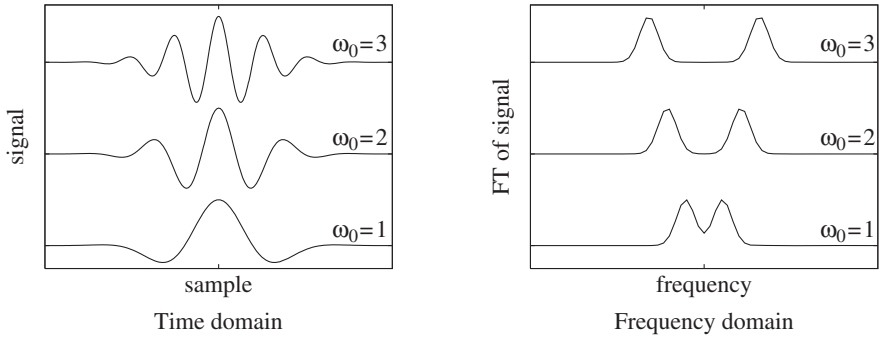
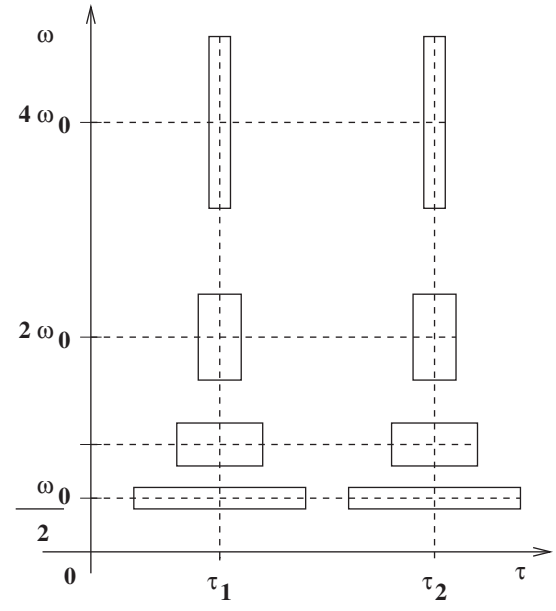


Figure 4.118 Gabor basis functions in the time and frequency domains for $\omega_0 = 1$, $\omega_0 = 2$, and $\omega_0 = 3$. The different modulations of the Gaussian function shift its frequency bands, but they do not change their bandwidths.

be increased at the expense of the other. Figure 4.119 shows schematically the resolution cells in terms of which a wavelet basis expands the answer to the question "what is where" for an 1D signal. Some elementary wavelet functions are shown in Figure 4.120. Figure 4.119 should be compared with Figure 4.117 and Figure 4.120 should be compared with Figure 4.118.

The basis created by the Gabor functions is not orthogonal, i.e. each of the elementary answers to the question "what is where" is not orthogonal to all other possible answers, and in a sense it contains part of the other answers too. If we choose the widths of the bands we use to tessellate the frequency space to be equal to the bandwidths of the Gaussian filters we use, there is significant leakage between the different bands. In practice, however, we tessellate first the frequency space in the bands we want, and then we choose the standard deviations of the Gaussian windows so that the value of a Gaussian tail that enters in the neighbouring band is about half the peak value of the Gaussian. This still causes significant leakage. On the contrary, the basis in terms of which wavelets express the answer to the question "what is where" is designed to be orthogonal.

Figure 4.119 Time–frequency space showing schematically the resolution cells for a wavelet function for an 1D signal. Each resolution cell is an elementary answer to the fundamental question "what is where" in the signal. Any answer concerning any signal of the same size can be expressed as a linear superposition of these elementary answers. In reality the cells are chosen so that they tile the whole space without gaps. Here they are shown sparsely placed for clarity.



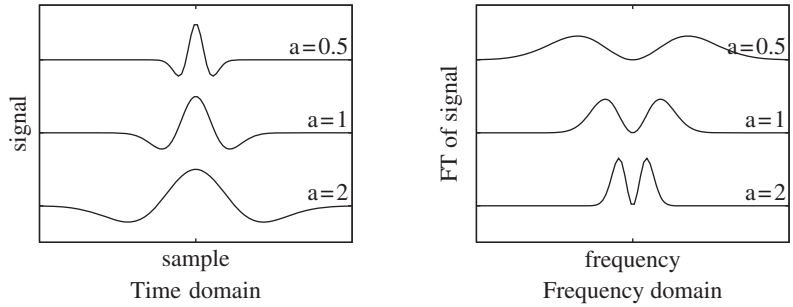


Figure 4.120 Some wavelet basis functions in the time and frequency domains. The different scalings of the original function shift its frequency bands and also change their bandwidths.

4.6 The Dual Tree Complex Wavelet Transform

What is the dual tree complex wavelet transform?

The dual tree complex wavelet transform is the expansion of a signal using two different wavelet bases, one of which is approximately the Hilbert pair of the other (see the section What is a Hilbert transform pair?).

Why was the dual tree complex wavelet transform developed?

It was developed in order to solve two drawbacks of the wavelet expansions:

(i) The wavelet coefficients are not shift invariant (see Examples 4.148 and 4.149).
(ii) In 2D the wavelets are not very good in orientation selection. For example, in Figure 4.110 we can see that working with real wavelets we picked up features in the image that manifest themselves in the highlighted bands. However, we cannot tell whether these features are orientated along the 45°–225° axis or along the 135°–315° axis, since the four highlighted bands express such features jointly.

The latter problem can be overcome if we use prolate spheroidal sequence expansions that allow the use of complex filters, which respond only to positive or only to negative frequencies, and thus have good orientation selectivity (see the section Is it possible to have a window with sharp edges in one domain which has minimal side ripples in the other domain? and Figures 4.79 and 4.80). However, prolate spheroidal sequence expansions do not have the property of perfect reconstruction required in some signal processing applications. If we are only interested in extracting features useful for texture segmentation, we are not interested in that property and we might as well stick to the use of prolate spheroidal sequence expansions.

Example 4.148

Consider the pulse signals:

$\mathbf{s_1} = (0, 0, 1, 1, 1, 0, 0, 0)$
$\mathbf{s_2} = (0, 0, 0, 0, 1, 1, 1, 0)$.

Consider also the Haar wavelet basis formed by keeping only the first, third, fifth and seventh wavelet functions shown in (4.488). Project the two signals on this basis and compare the two outputs. Are the computed wavelet coefficients the same irrespective of where the pulse of the signal is? Consider also the wavelet basis formed by keeping the second, fourth and sixth of the wavelet vectors shown in (4.488), with the addition of vector:

$\left(\frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0, 0, -\frac{1}{\sqrt{2}}\right)$**.**

Does the result change?

Projecting $\mathbf{s_1}$ *on the first wavelet basis we obtain:* $\left(0, 0, -\frac{1}{\sqrt{2}}, 0\right)$.
Projecting $\mathbf{s_2}$ *on the first wavelet basis we obtain:* $\left(0, 0, 0, -\frac{1}{\sqrt{2}}\right)$.
Projecting $\mathbf{s_1}$ *on the second wavelet basis we obtain:* $\left(\frac{1}{\sqrt{2}}, 0, 0, 0\right)$.
Projecting $\mathbf{s_2}$ *on the second wavelet basis we obtain:* $\left(0, \frac{1}{\sqrt{2}}, 0, 0\right)$.

We observe that once the wavelet basis is fixed, the wavelet coefficients have the same value at the locality of the signal pulse, even if the pulse is shifted. When we change basis the result changes because the non-overlapping segments with which we cover the time axis of the signal are different, and so the pulse is marked in shifted localities by the two bases. Nevertheless, the value with which the presence of the pulse is indicated is the same irrespective of the location of the pulse. We may say, therefore, that in this case, the wavelet we used exhibits the property of shift-invariance.

Example 4.149

Consider the following signals which contain the same pulse, but in slightly shifted positions:

$\mathbf{s_1} = (0, 0, 0, 1, 1, 0, 0, 0)$
$\mathbf{s_2} = (0, 0, 0, 0, 0, 1, 1, 0)$
$\mathbf{s_3} = (0, 0, 0, 0, 1, 1, 0, 0)$**.**

Compute the first level wavelet coefficients for each signal using the Daubechies 4 wavelet filter, listed in Table 4.6. Are the values of the wavelet coefficients in response to the presence of the pulse the same, irrespective of where the pulse is?

The filter given in Table 4.6 is the scaling filter: $(0.483, 0.8365, 0.2241, -0.1294)$*. From (4.544) we know that we can construct the wavelet filter from the scaling filter by reading the elements back to front and changing the sign of every first element. So, the Daubechies 4 wavelet filter is* $(0.1294, 0.2241, -0.8365, 0.483)$*. By assuming that each signal repeats itself along both directions and performing pseudo-convolution of the signals with this filter, while assigning the result to the first sample covered by the filter each time, we obtain:*

$\mathbf{o_1} = (0.483, -0.3535, -0.6124, 0.3535, 0.1294, 0, 0, 0)$
$\mathbf{o_2} = (0, 0, 0.483, -0.3535, -0.6124, 0.3535, 0.1294, 0)$
$\mathbf{o_3} = (0, 0.483, -0.3535, -0.6124, 0.3535, 0.1294, 0, 0)$.

(Continued)

Example 4.149 (Continued)

After decimation, we obtain the wavelet coefficients for each signal:

$w_1 = (0.483, -0.6124, 0.1294, 0)$
$w_2 = (0, 0.483, -0.6124, 0.1294)$
$w_3 = (0, -0.3535, 0.3535, 0)$.

In Figure 4.121 we plot two of the signals and the corresponding outputs of the convolution, as well as the wavelet coefficients.

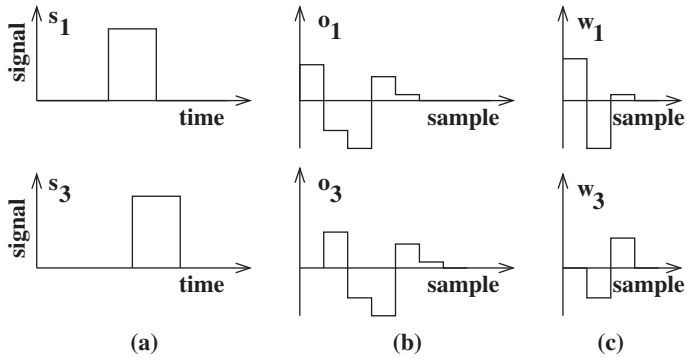


Figure 4.121 (a) A signal containing a pulse and its shifted version by one sample. (b) The outputs of the pseudo-convolutions of the signal with the Daubechies 4 wavelet filter. Note that the shape of the non-zero part of each output is the same, irrespective of the shifting of the pulse. So, we have shift invariance of the computed projection values. (c) The wavelet coefficients after removing every second sample of the output of the pseudo-convolution. The shift invariance has been lost.

We note that, depending on the amount by which the pulse is shifted, we may get the same or different wavelet coefficients for the same pulse. So, in this case the wavelet coefficients we get are not shift-invariant. On the contrary, if we were not decimating the pseudo-convolution outputs, we would have identical values irrespective of the shifting of the pulse. Not decimating the outputs is equivalent to projecting the signal on the over-complete basis made up by the following vectors:

$(0.1294, 0.2241, -0.8365, 0.483, 0, 0, 0, 0)$
$(0, 0.1294, 0.2241, -0.8365, 0.483, 0, 0, 0)$
$(0, 0, 0.1294, 0.2241, -0.8365, 0.483, 0, 0)$
$(0, 0, 0, 0.1294, 0.2241, -0.8365, 0.483, 0)$
$(0, 0, 0, 0, 0.1294, 0.2241, -0.8365, 0.483)$
$(0.483, 0, 0, 0, 0, 0.1294, 0.2241, -0.8365)$
$(-0.8365, 0.483, 0, 0, 0, 0, 0.1294, 0.2241)$
$(0.2241, -0.8365, 0.483, 0, 0, 0, 0, 0.1294)$.

How do we select the filters used by the dual tree complex wavelet transform?

The corresponding filters of the two wavelet bases have to constitute Hilbert pairs. This is impossible to achieve, as designing such filters requires the satisfaction of many constraints:

(i) The filters of the same wavelet basis have to be quadrature mirror filters so perfect reconstruction of the original signal, from the knowledge of its wavelet and scaling coefficients, is achievable.
(ii) The wavelet filters selected have to obey the admissibility condition (see Box 4.8).
(iii) The corresponding filters of the two different wavelet bases have to be Hilbert pairs. It turns out that these conditions can only approximately be satisfied simultaneously.

How can we select two wavelet filters that form a Hilbert transform pair?

It can be shown that one filter should be shifted by half a sample with respect to the other (see Box 4.11). This is not possible in the digital domain, so approximations are used in practice.

Example 4.150

You are given the scaling four-tap filter $h_0 = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$, with the index n that identifies the individual elements of the filter taking values $-2, -1, 0, 1$. The shifted by $2k$ versions of this filter are orthogonal to each other:

$$\sum_n h_0(n)h_0(n+2k) = \delta(k). \tag{4.597}$$

Derive the relationships scalars α_i have to obey for this to be correct.
For $k = 0$, we have:

$$\alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \alpha_3^2 = 1. \tag{4.598}$$

For $k = 1$ we have:

$$\begin{aligned} h_0(-2)h_0(0) + h_0(-1)h_0(1) + h_0(0)h_0(2) + h_0(1)h_0(3) = 0 \\ \Rightarrow \alpha_0\alpha_2 + \alpha_1\alpha_3 + \alpha_2\alpha_0 + \alpha_3\alpha_1 = 0 \\ \Rightarrow \alpha_0\alpha_2 + \alpha_1\alpha_3 = 0. \end{aligned} \tag{4.599}$$

Example 4.151

Construct the corresponding wavelet filter h_1 from the scaling filter h_0 of Example 4.150, using Equation (4.541).
The formula is applied using $M = 4$ and $p = 1, \ldots, 4$ to identify the elements of each filter. We work out that: $h_1 = (-\alpha_3, \alpha_2, -\alpha_1, \alpha_0)$.

Example B4.152

The z-transform of a digital signal $h(n)$ is defined as:

$$H(z) \equiv \sum_{n=-\infty}^{\infty} h(n)z^n. \tag{4.600}$$

(Continued)

Example B4.152 (Continued)

Show that the z-transforms of the filters h_1 and h_0 of Examples 4.151 and 4.150, respectively, obey the equations:

$$H_0(z)H_0\left(\frac{1}{z}\right) + H_0(-z)H_0\left(-\frac{1}{z}\right) = 2$$
$$H_1(z) = \frac{1}{z}H_0\left(-\frac{1}{z}\right). \tag{4.601}$$

Index n for these filters ranges from −2 to 1. Then, by applying formula (4.600), we obtain:

$$\begin{aligned} H_0(z) &= \frac{\alpha_0}{z^2} + \frac{\alpha_1}{z} + \alpha_2 + \alpha_3 z \\ H_0\left(-\frac{1}{z}\right) &= z^2\alpha_0 - z\alpha_1 + \alpha_2 - \frac{\alpha_3}{z} \\ H_1(z) &= -\frac{\alpha_3}{z^2} + \frac{\alpha_2}{z} - \alpha_1 + \alpha_0 z. \end{aligned} \tag{4.602}$$

Then, by direct substitution into (4.601), and by making use of (4.598) and (4.599), we can see that Equations (4.601) are valid.

Example B4.153

When we know the z-transform $H(z)$ of a digital signal $h(n)$, its frequency response $\hat{H}(\omega)$ is given by $H(e^{j\omega})$. Starting from the second of equations (4.601), show that:

$$\hat{H}_1(\omega) = e^{-j\omega}\hat{H}_0^*(\omega - \pi). \tag{4.603}$$

If we substitute $z = e^{j\omega}$ in the second of Equations (4.601), we obtain:

$$\begin{aligned} H_1(e^{j\omega}) &= e^{-j\omega}H_0(-e^{-j\omega}) \\ \Rightarrow \hat{H}_1(\omega) &= e^{-j\omega}H_0(e^{j\pi}e^{-j\omega}) \\ &= e^{-j\omega}H_0(e^{-j(\omega-\pi)}) \\ &= e^{-j\omega}H_0^*(e^{j(\omega-\pi)}) \\ &= e^{-j\omega}\hat{H}_0^*(\omega - \pi). \end{aligned} \tag{4.604}$$

Example B4.154

For the filters h_1 and h_0 of Examples 4.151 and 4.150, respectively, verify Equation (4.603).

By substituting $z = e^{j\omega}$ into equations (4.602), we obtain:

$$\begin{aligned} \hat{H}_0(\omega) &= \alpha_0 e^{-j2\omega} + \alpha_1 e^{-j\omega} + \alpha_2 + \alpha_3 e^{j\omega} \\ \hat{H}_1(\omega) &= -\alpha_3 e^{-j2\omega} + \alpha_2 e^{-j\omega} - \alpha_1 + \alpha_0 e^{j\omega}. \end{aligned} \tag{4.605}$$

Then:

$$\hat{H}_0(\omega-\pi) = \alpha_0 \mathrm{e}^{-\mathrm{j}2\omega}\underbrace{\mathrm{e}^{\mathrm{j}2\pi}}_{=1} + \alpha_1 \mathrm{e}^{-\mathrm{j}\omega}\underbrace{\mathrm{e}^{\mathrm{j}\pi}}_{=-1} + \alpha_2 + \alpha_3 \mathrm{e}^{\mathrm{j}\omega}\underbrace{\mathrm{e}^{-\mathrm{j}\pi}}_{=-1}$$

$$\Rightarrow \hat{H}_0^*(\omega-\pi) = \alpha_0 \mathrm{e}^{\mathrm{j}2\omega} + \alpha_1 \mathrm{e}^{\mathrm{j}\omega} + \alpha_2 + \alpha_3 \mathrm{e}^{-\mathrm{j}\omega}. \tag{4.606}$$

By substituting then from (4.606) and the second of Equations (4.605) into (4.603), we can verify the latter.

Example 4.155

Plot the magnitude frequency responses of the quadrature mirror filters:

$$h_0(n) = (1,2,3,3,2,1) \qquad h_1(n) = (-1,2,-3,3,-2,1) \text{ for } n = 0,\ldots,5.. \tag{4.607}$$

These filters have even length, so the centre of each filter is in between its two central elements, which are shifted by 0.5 away from the nearest integer position. If N is the number of elements, the first $N/2$ elements correspond to negative values of the indices, with the sampling points placed at locations $-0.5, -1.5, \ldots, -N/2+0.5$. The second $N/2$ samples are at locations $0.5, 1.5, \ldots, N/2-0.5$. The DFT of such a filter is given by:

$$H(k) = \sum_{n=0}^{N-1} h(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}\left(n-\frac{N-1}{2}\right)k}. \tag{4.608}$$

In this example $N = 6$. The DFTs of the two filters are:

$$\begin{aligned} H_0(k) &= \sum_{n=0}^{5} h_0(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}(n-2.5)k} \\ &= 1\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}(-2.5)k} + 2\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}(-1.5)k} + 3\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}(-0.5)k} \\ &\quad +3\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}0.5k} + 2\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}1.5k} + 1\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}2.5k} \\ &= \mathrm{e}^{\mathrm{j}\frac{2\pi}{6}2.5k} + \mathrm{e}^{-\frac{2\pi}{6}2.5k} + 2\left(\mathrm{e}^{\mathrm{j}\frac{2\pi}{6}1.5k} + \mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}1.5k}\right) + 3\left(\mathrm{e}^{\mathrm{j}\frac{2\pi}{6}0.5k} + \mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}0.5k}\right) \\ &= 2\cos\frac{5\pi k}{6} + 4\cos\frac{3\pi k}{6} + 6\cos\frac{\pi k}{6} \end{aligned} \tag{4.609}$$

$$\begin{aligned} H_1(k) &= \sum_{n=0}^{5} h_1(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}(n-2.5)k} \\ &= 1\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}(-2.5)k} - 2\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}(-1.5)k} + 3\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}(-0.5)k} \\ &\quad -3\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}0.5k} + 2\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}1.5k} - 1\times\mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}2.5k} \\ &= \mathrm{e}^{\mathrm{j}\frac{2\pi}{6}2.5k} - \mathrm{e}^{-\frac{2\pi}{6}2.5k} - 2\left(\mathrm{e}^{\mathrm{j}\frac{2\pi}{6}1.5k} - \mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}1.5k}\right) + 3\left(\mathrm{e}^{\mathrm{j}\frac{2\pi}{6}0.5k} - \mathrm{e}^{-\mathrm{j}\frac{2\pi}{6}0.5k}\right) \\ &= 2\mathrm{j}\sin\frac{5\pi k}{6} - 4\mathrm{j}\sin\frac{3\pi k}{6} + 6\mathrm{j}\sin\frac{\pi k}{6}. \end{aligned} \tag{4.610}$$

Figure 4.122 shows the two filters and the magnitudes of their Fourier transforms.

(Continued)

Example 4.155 (Continued)

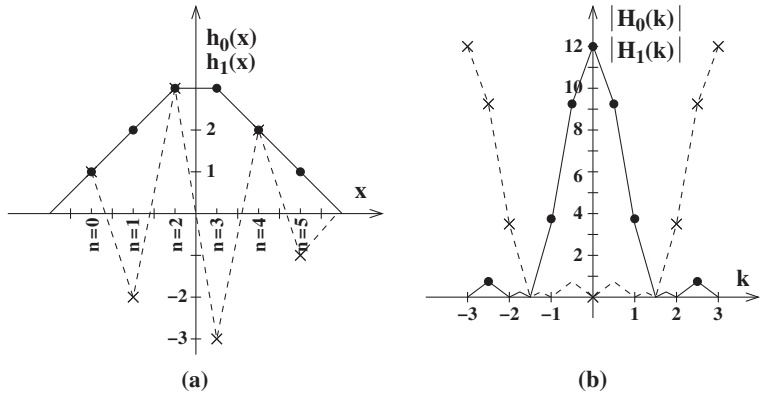


Figure 4.122 In both panels, the black dots and the solid line corresponds to the original low pass filter, while the crosses and the dashed line correspond to its quadrature mirror filter. (a) The two filters. (b) The magnitudes of their Fourier components.

Example B4.156

Compute the Fourier transforms of filters (4.607) using the z-transform and compare your results with those of Example 4.155.

The filters are

$$h_0(n) = (1, 2, 3, 3, 2, 1)$$
$$h_1(n) = (-1, 2, -3, 3, -2, 1). \tag{4.611}$$

We compute their z-transforms assuming that index n takes values $n = -3, -2, -1, 0, 1, 2$:

$$H_0(z) = z^{-3} + 2z^{-2} + 3z^{-1} + 3 + 2z + z^2$$
$$H_1(z) = -z^{-3} + 2z^{-2} - 3z^{-1} + 3 - 2z + z^2. \tag{4.612}$$

The Fourier transforms of these filters then are:

$$\hat{H}_0(\omega) = H_0(e^{j\omega}) = e^{-3j\omega} + 2e^{-2j\omega} + 3e^{-j\omega} + 3 + 2e^{j\omega} + e^{2j\omega}$$
$$\hat{H}_1(\omega) = H_1(e^{j\omega}) = -e^{-3j\omega} + 2e^{-2j\omega} - 3e^{-j\omega} + 3 - 2e^{j\omega} + e^{2j\omega}. \tag{4.613}$$

Figure 4.123 shows the magnitude plots of these Fourier transforms for $\omega \in [-\pi, \pi]$.

We observe that the results are in agreement with those of Example 4.155. The difference is that here we allowed the frequency variable to take continuous values, while in Example 4.155 it was allowed to take only discrete values. If we make the connection $2\pi k/6 = \omega$, the discrete and continuous spectra of each filter are identical.

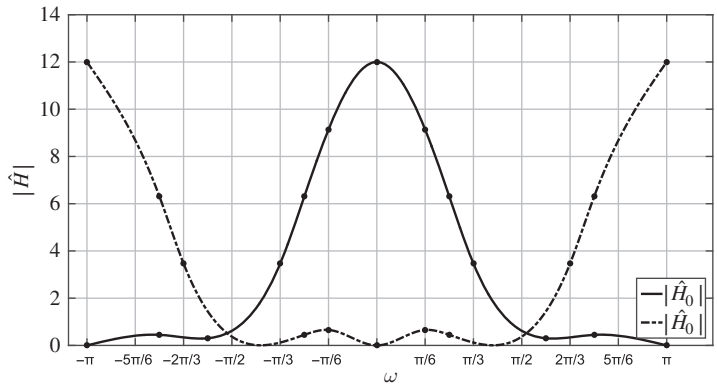


Figure 4.123 The magnitude plots of these Fourier transforms.

Example B4.157

Consider the filters of Example 4.156. Construct from it the following two filters:

$$h_{\mathrm{n}}(n) = \mathrm{j}^n h_0(n) \qquad h_{\mathrm{p}}(n) = \mathrm{j}^n h_1(n). \tag{4.614}$$

Show that $h_{\mathrm{p}}(n)$ has most of its energy in the positive frequencies, while $h_{\mathrm{n}}(n)$ has most of its energy in the negative frequencies.

We observe that $j^n = \left(\mathrm{e}^{\mathrm{j}\frac{\pi}{2}}\right)^n = \mathrm{e}^{\mathrm{j}\frac{\pi n}{2}}$. *Equations (4.613) then take the form:*

$$\begin{aligned}
\hat{H}_{\mathrm{n}}(\omega) &= H_n\left(\mathrm{e}^{\mathrm{j}\omega}\right) = \mathrm{e}^{-3\mathrm{j}\left(\omega+\frac{\pi}{2}\right)} + 2\mathrm{e}^{-2\mathrm{j}\left(\omega+\frac{\pi}{2}\right)} + 3\mathrm{e}^{-\mathrm{j}\left(\omega+\frac{\pi}{2}\right)} + 3 + 2\mathrm{e}^{\mathrm{j}\left(\omega+\frac{\pi}{2}\right)} + \mathrm{e}^{2\mathrm{j}\left(\omega+\frac{\pi}{2}\right)} \\
\hat{H}_{\mathrm{p}}(\omega) &= H_p\left(\mathrm{e}^{\mathrm{j}\omega}\right) = -\mathrm{e}^{-3\mathrm{j}\left(\omega+\frac{\pi}{2}\right)} + 2\mathrm{e}^{-2\mathrm{j}\left(\omega+\frac{\pi}{2}\right)} - 3\mathrm{e}^{-\mathrm{j}\left(\omega+\frac{\pi}{2}\right)} + 3 - 2\mathrm{e}^{\mathrm{j}\left(\omega+\frac{\pi}{2}\right)} + \mathrm{e}^{2\mathrm{j}\left(\omega+\frac{\pi}{2}\right)}.
\end{aligned} \tag{4.615}$$

Note that multiplication with j^n *rotates each harmonic either by* $\pi/2$ *or by* $-\pi/2$.

Example B4.158

Starting from the two scale Equations (4.576) show that

$$\Phi(\omega) = \frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2}\right)\Phi\left(\frac{\omega}{2}\right) \tag{4.616}$$

$$\Psi(\omega) = \frac{1}{\sqrt{2}}\hat{H}_1\left(\frac{\omega}{2}\right)\Phi\left(\frac{\omega}{2}\right) \tag{4.617}$$

(Continued)

Example B4.158 (Continued)

where $\Phi(\omega)$, $\Psi(\omega)$, $\hat{H}_0(\omega)$ and $\hat{H}_1(\omega)$ are the Fourier transforms of $\phi(t)$, $\psi(t)$, $h_0(n)$ and $h_1(n)$, respectively, with $h_0(n)$ being the low pass filter and $h_1(n)$ its corresponding quadrature mirror wavelet filter.

For these filters, the two scales equations are written as:

$$\begin{aligned}\phi(t) &= \sqrt{2}\sum_n h_0(n)\phi(2t-n)\\ \psi(t) &= \sqrt{2}\sum_n h_1(n)\phi(2t-n).\end{aligned} \tag{4.618}$$

If $\Phi(\omega)$ is the Fourier transform of $\phi(t)$, then the Fourier transform of $\phi(2t)$ is $\Phi(\omega/2)/2$ (scaling property of the Fourier transform). Equations (4.618) are convolutions between functions $\phi(2t)$ and $h_0(n)$ or $h_1(n)$. However, convolutions in the time domain correspond to point by point multiplications of the corresponding Fourier transforms in the frequency domain. The Fourier transforms of filters $h_0(n)$ and $h_1(n)$ are $\hat{H}_0(\omega)$ and $\hat{H}_1(\omega)$, respectively, and these have to be computed at the frequencies of function $\phi(2t)$, i.e. at $\omega/2$. Then (4.616) and (4.617) follow.

Example B4.159

Starting from Equations (4.616) and (4.617) prove that

$$\Phi(\omega) = \Phi(0)\prod_{k=1}^{\infty}\left[\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2^k}\right)\right] \tag{4.619}$$

$$\Psi(\omega) = \Phi(0)\prod_{k=1}^{\infty}\left[\frac{1}{\sqrt{2}}\hat{H}_1\left(\frac{\omega}{2^k}\right)\right] \tag{4.620}$$

where $\Phi(\omega)$, $\Psi(\omega)$, $\hat{H}_0(\omega)$ and $\hat{H}_1(\omega)$ are the Fourier transforms of $\phi(t)$, $\psi(t)$, $h_0(n)$ and $h_1(n)$, respectively. Such equations are known as the infinite product formula of the Fourier transform obeyed by a refinable function.

Let us prove Equation (4.619). The proof of (4.620) is similar. Let us apply Equation (4.616) N times, recursively:

$$\begin{aligned}\Phi(\omega) &= \frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2}\right)\Phi\left(\frac{\omega}{2}\right)\\ &= \frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2}\right)\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{4}\right)\Phi\left(\frac{\omega}{4}\right)\\ &= \frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2}\right)\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{4}\right)\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{8}\right)\Phi\left(\frac{\omega}{8}\right)\\ &\cdots\quad\cdots\\ &= \frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2}\right)\cdots\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2^N}\right)\Phi\left(\frac{\omega}{2^N}\right).\end{aligned} \tag{4.621}$$

Then letting $N \to \infty$, Equation (4.619) follows.

Example B4.160

Two scaling filters g_0 and h_0 have Fourier transforms $\hat{G}_0(\omega)$ and $\hat{H}_0(\omega)$, respectively, related by

$$\hat{G}_0(\omega) = \hat{H}_0(\omega)\mathrm{e}^{\mathrm{j}\theta(\omega)} \tag{4.622}$$

where $\theta(\omega)$ is a 2π periodic function. Show that the Fourier transforms of the corresponding scaling functions are related by:

$$\Phi_g(\omega) = \frac{\Phi_g(0)}{\Phi_h(0)}\Phi_h(\omega)\mathrm{e}^{\mathrm{j}\sum_{k=1}^{\infty}\theta\left(\frac{\omega}{2^k}\right)}. \tag{4.623}$$

We start by applying Equation (4.619) for scaling function Φ_g and its corresponding filter $\hat{G}_0$, and then substitute filter $\hat{G}_0$ from (4.622):

$$\begin{aligned}
\Phi_g(\omega) &= \Phi_g(0)\prod_{k=1}^{\infty}\left[\frac{1}{\sqrt{2}}\hat{G}_0\left(\frac{\omega}{2^k}\right)\right] \\
&= \Phi_g(0)\prod_{k=1}^{\infty}\left[\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2^k}\right)\mathrm{e}^{\mathrm{j}\theta\left(\frac{\omega}{2^k}\right)}\right] \\
&= \Phi_g(0)\prod_{k=1}^{\infty}\left[\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2^k}\right)\right]\prod_{k=1}^{\infty}\mathrm{e}^{\mathrm{j}\theta\left(\frac{\omega}{2^k}\right)} \\
&= \Phi_g(0)\prod_{k=1}^{\infty}\left[\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2^k}\right)\right]\mathrm{e}^{\mathrm{j}\sum_{k=1}^{\infty}\theta\left(\frac{\omega}{2^k}\right)}.
\end{aligned} \tag{4.624}$$

Then we apply Equation (4.619) for scaling function Φ_h and its corresponding filter $\hat{H}_0$:

$$\begin{aligned}
\Phi_h(\omega) &= \Phi_h(0)\prod_{k=1}^{\infty}\left[\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2^k}\right)\right] \\
\Rightarrow \prod_{k=1}^{\infty}\left[\frac{1}{\sqrt{2}}\hat{H}_0\left(\frac{\omega}{2^k}\right)\right] &= \frac{\Phi_h(\omega)}{\Phi_h(0)}.
\end{aligned} \tag{4.625}$$

Substituting then from (4.625) into (4.624), we obtain (4.623).

Example B4.161

Two scaling filters g_0 and h_0 have Fourier transforms $\hat{G}_0(\omega)$ and $\hat{H}_0(\omega)$, respectively, related by (4.622). Show that the Fourier transforms of the corresponding wavelet filters are related by:

$$\hat{G}_1(\omega) = \hat{H}_1(\omega)\mathrm{e}^{-\mathrm{j}\theta(\omega-\pi)}. \tag{4.626}$$

First we apply (4.603) for filter $\hat{G}_1$ and substitute $\hat{G}_0$ from (4.622):

$$\begin{aligned}
\hat{G}_1(\omega) &= \mathrm{e}^{-\mathrm{j}\omega}\hat{G}_0^*(\omega-\pi) \\
&= \mathrm{e}^{-\mathrm{j}\omega}\hat{H}_0^*(\omega-\pi)\mathrm{e}^{-\mathrm{j}\theta(\omega-\pi)}.
\end{aligned} \tag{4.627}$$

(Continued)

Example B4.161 (Continued)

Then we apply (4.603) for filter $\hat{H}_1$

$$\hat{H}_1(\omega) = \mathrm{e}^{-\mathrm{j}\omega}\hat{H}_0^*(\omega - \pi) \Rightarrow \hat{H}_0^*(\omega - \pi) = \mathrm{e}^{\mathrm{j}\omega}\hat{H}_1(\omega) \tag{4.628}$$

and substitute into (4.627) to obtain (4.626).

Example B4.162

Two scaling filters g_0 **and** h_0 **have Fourier transforms** $\hat{G}_0(\omega)$ **and** $\hat{H}_0(\omega)$**, respectively, related by (4.622). Show that the Fourier transforms of the corresponding wavelet functions are related by:**

$$\Psi_g(\omega) = \frac{\Phi_g(0)}{\Phi_h(0)}\Psi_h(\omega)\mathrm{e}^{-\mathrm{j}\left[\theta\left(\frac{\omega}{2}-\pi\right)-\sum_{k=2}^{\infty}\theta\left(\frac{\omega}{2^k}\right)\right]}. \tag{4.629}$$

We start by applying Equation (4.617) for the two wavelet functions and their corresponding filters $\hat{H}_1$ *and* $\hat{G}_1$*:*

$$\Psi_h(\omega) = \frac{1}{\sqrt{2}}\hat{H}_1\left(\frac{\omega}{2}\right)\Phi_h\left(\frac{\omega}{2}\right) \tag{4.630}$$

$$\Psi_g(\omega) = \frac{1}{\sqrt{2}}\hat{G}_1\left(\frac{\omega}{2}\right)\Phi_g\left(\frac{\omega}{2}\right). \tag{4.631}$$

In (4.631) we substitute $\hat{G}_1$ *from (4.626) and* Φ_g *from (4.623):*

$$\begin{aligned}\Psi_g(\omega) &= \frac{1}{\sqrt{2}}\hat{H}_1\left(\frac{\omega}{2}\right)\mathrm{e}^{-\mathrm{j}\theta\left(\frac{\omega}{2}-\pi\right)}\Phi_g\left(\frac{\omega}{2}\right)\\ &= \frac{1}{\sqrt{2}}\hat{H}_1\left(\frac{\omega}{2}\right)\mathrm{e}^{-\mathrm{j}\theta\left(\frac{\omega}{2}-\pi\right)}\frac{\Phi_g(0)}{\Phi_h(0)}\Phi_h\left(\frac{\omega}{2}\right)\mathrm{e}^{\mathrm{j}\sum_{k=1}^{\infty}\theta\left(\frac{\omega}{2^{k+1}}\right)}\\ &= \frac{\Phi_g(0)}{\Phi_h(0)}\underbrace{\frac{1}{\sqrt{2}}\hat{H}_1\left(\frac{\omega}{2}\right)\Phi_h\left(\frac{\omega}{2}\right)}\mathrm{e}^{-\mathrm{j}\left[\theta\left(\frac{\omega}{2}-\pi\right)-\sum_{k=1}^{\infty}\theta\left(\frac{\omega}{2^{k+1}}\right)\right]}.\end{aligned} \tag{4.632}$$

This, upon changing the summation variable k in the exponent and substituting the underbraced expression from (4.630), yields (4.629).

Example B4.163

If $\theta(\omega) = \omega/2$**, for** $|\omega| < \pi$**, and** $\theta(\omega)$ **is periodic with period** 2π**, show that**

$$\theta\left(\frac{\omega}{2} - \pi\right) = \begin{cases} -\frac{\pi}{2} + \frac{\omega}{4} & \text{for } 0 < \omega < 4\pi \\ \\ \frac{\pi}{2} + \frac{\omega}{4} & \text{for } -4\pi < \omega < 0 \end{cases}. \tag{4.633}$$

Function $\theta(\omega/2-\pi)$ is equal to $\omega/4-\pi/2$ for:

$$\begin{aligned}&\left|\tfrac{\omega}{2}-\pi\right|<\pi\\ \Leftrightarrow\ &-\pi<\tfrac{\omega}{2}-\pi<\pi\\ \Leftrightarrow\ &0<\tfrac{\omega}{2}<2\pi\\ \Leftrightarrow\ &0<\omega<4\pi.\end{aligned}\tag{4.634}$$

At the same time, $\theta(\omega)$ is 2π periodic, i.e. $\theta(\omega+2\pi)=\theta(\omega)$. So, $\theta(\omega/2-\pi)=\theta(\omega/2-\pi+2\pi)=\theta(\omega/2+\pi)$. So, we must also have:

$$\begin{aligned}&\left|\tfrac{\omega}{2}+\pi\right|<\pi\\ \Leftrightarrow\ &-\pi<\tfrac{\omega}{2}+\pi<\pi\\ \Leftrightarrow\ &-2\pi<\tfrac{\omega}{2}<0\\ \Leftrightarrow\ &-4\pi<\omega<0.\end{aligned}\tag{4.635}$$

In this case, $\theta(\omega/2-\pi)=\theta(\omega/2+\pi)=\omega/4+\pi/2$. This completes the proof of (4.633).

Example B4.164

If $\theta(\omega)=\omega/2$, for $|\omega|<\pi$, and $\theta(\omega)$ is periodic with period 2π, and $\beta(\omega)$ is defined as

$$\beta(\omega)\equiv-\sum_{k=2}^{\infty}\theta\left(\frac{\omega}{2^k}\right)\tag{4.636}$$

show that:

$$\beta(\omega)=\begin{cases}-\frac{\omega}{4} & \text{for } |\omega|<4\pi\\ \beta(\omega-4\pi) & \text{for } 4\pi<\omega\\ \beta(\omega+4\pi) & \text{for } \omega<-4\pi\end{cases}.\tag{4.637}$$

By definition:

$$\begin{aligned}\beta(\omega)&=-\sum_{k=2}^{\infty}\frac{\omega}{2^{k+1}}=-\frac{\omega}{4}\sum_{k=2}^{\infty}\frac{1}{2^{k-1}}\\ &=-\frac{\omega}{4}\left[\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots\right]\\ &=-\frac{\omega}{4}.\end{aligned}\tag{4.638}$$

The last result was obtained by applying the geometric series formula

$$\sum_{k=0}^{\infty}ar^k=a+ar+ar^2+\cdots=\frac{a}{1-r}\tag{4.639}$$

to the sum inside the square brackets, for $a=r=1/2$.

We note that to obtain this result, we substituted the expression of $\theta\left(\frac{\omega}{2^k}\right)$ which is only valid for the argument of the function being in the range $[-\pi,\pi]$. So, we must check the range of values

(Continued)

Example B4.164 (Continued)

of ω for which the above result is valid. For every value of k, we must have:

$$-\pi < \frac{\omega}{2^k} < \pi \Leftrightarrow -\pi 2^k < \omega < \pi 2^k. \tag{4.640}$$

We note that:

$$\begin{array}{lll} For & k=2 & -4\pi < \omega < 4\pi \\ For & k=3 & -8\pi < \omega < 8\pi \\ \ldots & \ldots & \ldots \\ For & k & -2^k\pi < \omega < 2^k\pi. \end{array}$$

We note that if the first of these conditions is satisfied, all of them will be satisfied too. So, the first branch of Equation (4.637) has been proven.

In order to prove the other branches of the equation, let us start by writing the definition of $\theta(\omega)$ explicitly for $-3\pi < \omega < 3\pi$:

$$\theta(\omega) = \begin{cases} \frac{\omega}{2} & -\pi < \omega < \pi \\ \frac{\omega-2\pi}{2} & \pi < \omega < 3\pi \\ \frac{\omega+2\pi}{2} & -3\pi < \omega < -\pi \end{cases}. \tag{4.641}$$

Then, let us consider each term of the sum that appears in the definition of $\beta(\omega)$, in turn, starting by setting $k = 2, 3, \ldots$. In each case we use (4.641):

$$\theta\left(\frac{\omega}{4}\right) = \begin{cases} \frac{\omega}{8} & -4\pi < \omega < 4\pi \\ \frac{\omega/4-2\pi}{2} & 4\pi < \omega < 12\pi \\ \frac{\omega/4+2\pi}{2} & -12\pi < \omega < -4\pi \end{cases}$$

$$\theta\left(\frac{\omega}{8}\right) = \begin{cases} \frac{\omega}{16} & -8\pi < \omega < 8\pi \\ \frac{\omega/8-2\pi}{2} & 8\pi < \omega < 24\pi \\ \frac{\omega/8+2\pi}{2} & -24\pi < \omega < -8\pi \end{cases}$$

$$\ldots \quad \ldots$$

$$\theta\left(\frac{\omega}{2^k}\right) = \begin{cases} \frac{\omega}{2^{k+1}} & -2^k\pi < \omega < 2^k\pi \\ \frac{\omega/2^k-2\pi}{2} & 2^k\pi < \omega < 3\times 2^k\pi \\ \frac{\omega/2^k+2\pi}{2} & -3\times 2^k\pi < \omega < -2^k\pi \end{cases}. \tag{4.642}$$

Now, let us try to form the sum that appears in the definition of $\beta(\omega)$. To sum functions that are only piecewise continuous, we must consider the union of the points of discontinuity of all terms and sum freely only in between successive discontinuity points. We note that as the value of k increases, the branches over which the function is continuous are broader. So, the subdivisions we need are dictated by the first terms of the sum and actually they go in steps of 4π. For clarity, we redefine here only the first two terms in the form we need them:

$$\theta\left(\frac{\omega}{4}\right) = \begin{cases} \frac{\omega}{8} & -4\pi < \omega < 4\pi \\ \frac{\omega/4-2\pi}{2} & 4\pi < \omega < 8\pi \\ \frac{\omega/4-2\pi}{2} & 8\pi < \omega < 12\pi \\ \frac{\omega/4+2\pi}{2} & -8\pi < \omega < -4\pi \\ \frac{\omega/4+2\pi}{2} & -12\pi < \omega < -8\pi \end{cases}$$

$$\theta\left(\frac{\omega}{8}\right)=\begin{cases}\frac{\omega}{16} & -4\pi<\omega<4\pi\\ \frac{\omega}{16} & 4\pi<\omega<8\pi\\ \frac{\omega/8-2\pi}{2} & 8\pi<\omega<12\pi\\ \frac{\omega/8-2\pi}{2} & 12\pi<\omega<16\pi\\ \frac{\omega/8-2\pi}{2} & 16\pi<\omega<20\pi\\ \frac{\omega/8-2\pi}{2} & 20\pi<\omega<24\pi\\ \frac{\omega}{16} & -8\pi<\omega<-4\pi\\ \frac{\omega/8+2\pi}{2} & -12\pi<\omega<-8\pi\\ \frac{\omega/8+2\pi}{2} & -16\pi<\omega<-12\pi\\ \frac{\omega/8+2\pi}{2} & -20\pi<\omega<-16\pi\\ \frac{\omega/8+2\pi}{2} & -24\pi<\omega<-20\pi\end{cases}. \tag{4.643}$$

Now let us sum these two functions for the first few ranges of continuity:

$$\begin{aligned}\theta\left(\frac{\omega}{4}\right)+\theta\left(\frac{\omega}{8}\right)&=\begin{cases}\frac{\omega}{8}+\frac{\omega}{16} & -4\pi<\omega<4\pi\\ \frac{\omega/4-2\pi}{2}+\frac{\omega}{16} & 4\pi<\omega<8\pi\\ \frac{\omega/4-2\pi}{2}+\frac{\omega/8-2\pi}{2} & 8\pi<\omega<12\pi\\ \frac{\omega/4+2\pi}{2}+\frac{\omega}{16} & -8\pi<\omega<-4\pi\\ \frac{\omega/4+2\pi}{2}+\frac{\omega/8+2\pi}{2} & -12\pi<\omega<-8\pi\end{cases}\\ &=\begin{cases}\frac{\omega}{8}+\frac{\omega}{16} & -4\pi<\omega<4\pi\\ \frac{\omega}{8}+\frac{\omega}{16}-\pi & 4\pi<\omega<8\pi\\ \frac{\omega}{8}+\frac{\omega}{16}-2\pi & 8\pi<\omega<12\pi\\ \frac{\omega}{8}+\frac{\omega}{16}+\pi & -8\pi<\omega<-4\pi\\ \frac{\omega}{8}+\frac{\omega}{16}+2\pi & -12\pi<\omega<-8\pi\end{cases}.\end{aligned} \tag{4.644}$$

We know that if we consider all terms of the sum, the sum in the first branch is equal to $\omega/4$. So, we realise that, in the full sum, we shall have $\omega/4$ instead of $\omega/8+\omega/16$ we have in (4.644). Then, we may write:

$$\sum_{k=2}^{\infty}\theta\left(\frac{\omega}{2^k}\right)=\begin{cases}\frac{\omega}{4} & -4\pi<\omega<4\pi\\ \frac{\omega}{4}-\pi & 4\pi<\omega<8\pi\\ \frac{\omega}{4}-2\pi & 8\pi<\omega<12\pi\\ \frac{\omega}{4}+\pi & -8\pi<\omega<-4\pi\\ \frac{\omega}{4}+2\pi & -12\pi<\omega<-8\pi\end{cases}=\begin{cases}\frac{\omega}{4} & -4\pi<\omega<4\pi\\ \frac{\omega-4\pi}{4} & 4\pi<\omega<8\pi\\ \frac{\omega-8\pi}{4} & 8\pi<\omega<12\pi\\ \frac{\omega+4\pi}{4} & -8\pi<\omega<-4\pi\\ \frac{\omega+8\pi}{4} & -12\pi<\omega<-8\pi\end{cases}. \tag{4.645}$$

We can see then how the recursive relation that appears in (4.637) emerges: the value of the sum in the second branch of (4.645) can be computed from the first by replacing ω with $\omega-4\pi$. The third branch from the second, again by replacing ω with $\omega-4\pi$, and so on. Similarly, the value in the range $(-8\pi,-4\pi)$ can be calculated from the value in the range $(-4\pi,4\pi)$ by replacing ω with $\omega+4\pi$, while the value in the range $(-12\pi,-8\pi)$ can be calculated from the value in the range $(-8\pi,-4\pi)$ by again replacing ω with $\omega+4\pi$.

Box 4.11 How should two scaling filters be chosen so that the corresponding wavelets they generate form a Hilbert transform pair?

Let us call the two scaling filters g_0 and h_0, corresponding to scaling functions ϕ_g and ϕ_h, respectively. The corresponding wavelet filters are g_1 and h_1, corresponding to mother wavelets ψ_g and ψ_h, respectively. The Fourier transforms of functions ϕ_g, ϕ_h, ψ_g and ψ_h, are Φ_g, Φ_h, Ψ_g and Ψ_h, respectively. The Fourier transforms of filters g_0, h_0, g_1 and h_1 are $\hat{G}_0(\omega)$, $\hat{H}_0(\omega)$, $\hat{G}_1(\omega)$ and $\hat{H}_1(\omega)$, respectively.

Let us assume that the two scaling filters are related by

$$\hat{G}_0(\omega) = \hat{H}_0(\omega)\mathrm{e}^{-\mathrm{j}\theta(\omega)} \tag{4.646}$$

where $\theta(\omega)$ is a 2π periodic function.

We want to choose function $\theta(\omega)$ so the Fourier transforms of the corresponding mother wavelets, Ψ_g and Ψ_h, constitute a Hilbert transform pair.

It can be shown (see Examples 4.160–4.162) that Ψ_g and Ψ_h are related by:

$$\Psi_g(\omega) = \frac{\Phi_g(0)}{\Phi_h(0)}\Psi_h(\omega)\mathrm{e}^{-\mathrm{j}\left[\theta\left(\frac{\omega}{2}-\pi\right)-\sum_{k=2}^{\infty}\theta\left(\frac{\omega}{2^k}\right)\right]}. \tag{4.647}$$

Assuming that $\Phi_g(0) = \Phi_h(0)$, we realise that Ψ_g and Ψ_h will constitute a Hilbert pair if the quantity in the square bracket of the exponent in (4.647) has the right behaviour (see the section What is a Hilbert transform pair?):

$$\theta\left(\frac{\omega}{2}-\pi\right) - \sum_{k=2}^{\infty}\theta\left(\frac{\omega}{2^k}\right) = \begin{cases} -\frac{\pi}{2} & \omega > 0 \\ \frac{\pi}{2} & \omega < 0 \end{cases}. \tag{4.648}$$

If we now assume that $\theta(\omega)$ is defined so that $\theta(\omega) = -\omega/2$ for $|\omega| < \pi$, and make use of the results of Examples 4.163 and 4.164, we can see that (4.648) is valid. In other words, if the two scaling filters are defined to have the same magnitude but a phase difference of half a sample, then the corresponding wavelets constitute a Hilbert pair. In the digital domain half a sample phase different is not possible, and so we cannot really construct two wavelet bases that are Hilbert transform pairs exactly.

How do we construct the filters we use in the dual tree complex wavelet transform?

As these filters can only approximately fulfil the necessary conditions, there have been several ways by which such filters have been constructed, each time sacrificing one of more of the exact properties they have to have. Some of the first attempts were involving the construction of pairs of filters with one having most of its energy in the positive and the other in the negative frequencies (see Examples 4.607–4.157). Such filters had significant defects and they are not used in the dual tree complex wavelet transform. The most popular filters have been constructed with the so-called **q-shift** method, where the letter q stands for "quarter" (see Box 4.12). Once we construct a filter with its centre of symmetry 0.25 sampling distance away from the centre of the axis, if we read its elements backwards, we shall have a filter with its centre of symmetry 0.25 sampling distance away from the centre of the axis but in the other direction. So, the centres of symmetries of the two filters will be 0.5 sampling distance away from each other. As the original filter was symmetric, the fact that we read its elements backwards does not change it much. This way we shall have two filters that are almost identical but shifted away from each other by 0.5, as required. Filter $h_0(n)$ in

Table 4.7 Two sets of wavelet filters that can be used in the dual tree complex wavelet transform, after the first level of analysis

$h_0(n)$	$g_0(n)$	$h_1(n)$	$g_1(n)$
0.03516384	0	0	−0.03516384
0	0	0	0
−0.08832942	−0.11430184	0.11430184	0.08832942
0.23389032	0	0	0.23389032
0.76027237	0.58751830	−0.58751830	−0.76027237
0.58751830	0.76027237	0.76027237	0.58751830
0	0.23389032	−0.23389032	0
−0.11430184	−0.08832942	0.08832942	0.11430184
0	0	0	0
0	0.03516384	−0.03516384	0

Table 4.7 was constructed by the q-shift method. Filter $g_0(n)$ in the same table is the same filter with its elements read backwards. From these two scaling filters we can then construct the corresponding wavelet filters by reading their elements backwards and changing the sign of every first one. That is how filters $h_1(n)$ and $g_1(n)$ were constructed from filters $h_0(n)$ and $g_0(n)$, respectively. This set of filters may be used for the dual tree complex wavelet transform, after the first level of analysis. At the first level of analysis it was found that using an ordinary set of wavelet filters, followed by decimation done in two ways (once decimating every second sample and once decimating every first sample) was adequate for shift invariance (see Example protohronia201).

Box 4.12 The quarter shift idea

Consider a scaling filter $h_0(n)$ defined for $n = 0, 1, \ldots, N-1$. Define another scaling filter $g_0(n)$ as:

$$g_0(n) = h_0(N-1-n). \tag{4.649}$$

If we take the Fourier transform of both sides of this equation, we shall have (see Example 4.165)

$$\hat{G}_0(\omega) = \hat{H}_0^*(\omega)\mathrm{e}^{-\mathrm{j}(N-1)\omega} \tag{4.650}$$

where the complex conjugation comes from the reversal of the sign of n and the exponential factor from the shifting property of the Fourier transform. According to this equation:

$$\text{Phase}[\hat{G}_0(\omega)] = -\text{Phase}[\hat{H}_0(\omega)] - (N-1)\omega. \tag{4.651}$$

If we want, therefore, $g_0(n)$ and $h_0(n)$ to differ in phase by $\omega/2$ so they create wavelets that are Hilbert pairs, we must have:

$$\begin{aligned}\text{Phase}[\hat{H}_0(\omega)] - \frac{\omega}{2} &= -\text{Phase}[\hat{H}_0(\omega)] - (N-1)\omega \\ \Rightarrow \text{Phase}[\hat{H}_0(\omega)] &= -\frac{N-1}{2}\omega + \frac{\omega}{4}.\end{aligned} \tag{4.652}$$

(Continued)

Box 4.12 (Continued)

The term $\omega/4$ in the phase of this filter is the one that gives the name to this method. This relationship says that filter $h_0(n)$ should be symmetric about point $(N-1)/2 - 1/4$ (see Example 4.168). So, the problem now becomes one of designing a low pass filter $h_0(n)$ that is approximately symmetric about this point and approximately satisfies all other constraints a scaling filter should satisfy. One way of doing this is to construct an even filter of size $2N$ and then sub-sample it by keeping every first element (see Example 4.170). Figure 4.124 shows a filter constructed this way. It is filter $h_0(n)$ of Table 4.7. We can see that it is approximately symmetric about a point to the right of the vertical axis.

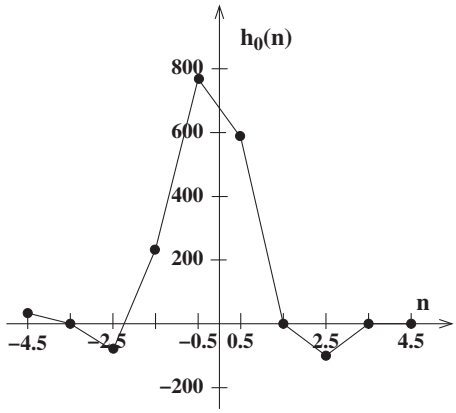


Figure 4.124 Filter $h_0(n) \times 1000$ plotted against the real distance of its elements from the centre of the axis. The values of this filter are given in the first column of Table 4.7.

Example B4.165

You are given a real filter $h_0(n) = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$. Construct filter $g_0(n)$ using Equation (4.649). Then demonstrate that Equation (4.650) holds.

Here $N = 4$. Then by applying (4.649) for $n = 0, 1, 2, 3$, we obtain:

$$g_0(n) = (\alpha_4, \alpha_3, \alpha_2, \alpha_1). \tag{4.653}$$

The Fourier transforms of $h_0(n)$ and $g_0(n)$ are:

$$\hat{H}_0(\omega) = \alpha_1 + \alpha_2 e^{-j\omega} + \alpha_3 e^{-j2\omega} + \alpha_4 e^{-j3\omega} \tag{4.654}$$

$$\hat{G}_0(\omega) = \alpha_4 + \alpha_3 e^{-j\omega} + \alpha_2 e^{-j2\omega} + \alpha_1^{-j3\omega}. \tag{4.655}$$

We calculate next the right-hand side of (4.650):

$$\begin{aligned} \hat{H}_0^*(\omega) e^{-j3\omega} &= [\alpha_1 + \alpha_2 e^{j\omega} + \alpha_3 e^{j2\omega} + \alpha_4 e^{j3\omega}] e^{-j3\omega} \\ &= \alpha_1 e^{-j3\omega} + \alpha_2 e^{-j2\omega} + \alpha_3 e^{-j\omega} + \alpha_4. \end{aligned} \tag{4.656}$$

By direct comparison with Equation (4.655) we verify (4.650).

Example B4.166

For the filters of Example 4.165, show that $|\hat{H}_0(\omega)| = |\hat{G}_0(\omega)|$.
From Equations (4.656) we can compute the magnitude for each harmonic component of these filters (i.e. assuming that ω is fixed):

$$\begin{aligned}
|\hat{H}_0(\omega)|^2 &= \hat{H}_0(\omega)\hat{H}_0^*(\omega)\\
&= [\alpha_1 + \alpha_2 e^{-j\omega} + \alpha_3 e^{-j2\omega} + \alpha_4 e^{-j3\omega}]\\
&\quad [\alpha_1 + \alpha_2 e^{j\omega} + \alpha_3 e^{j2\omega} + \alpha_4 e^{j3\omega}]\\
&= \alpha_1^2 + \alpha_1\alpha_2 e^{j\omega} + \alpha_1\alpha_3 e^{j2\omega} + \alpha_1\alpha_4 e^{j3\omega}\\
&\quad +\alpha_2\alpha_1 e^{-j\omega} + \alpha_2^2 + \alpha_2\alpha_3 e^{j\omega} + \alpha_2\alpha_4 e^{j2\omega}\\
&\quad +\alpha_3\alpha_1 e^{-j2\omega} + \alpha_3\alpha_2 e^{-j\omega} + \alpha_3^2 + \alpha_3\alpha_4 e^{j\omega}\\
&\quad +\alpha_4\alpha_1 e^{-j3\omega} + \alpha_4\alpha_2 e^{-j2\omega} + \alpha_4\alpha_3 e^{-j\omega} + \alpha_4^2\\
&= \alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2\\
&\quad +2\alpha_1\alpha_2\cos\omega + 2\alpha_1\alpha_3\cos(2\omega) + 2\alpha_1\alpha_4\cos(3\omega)\\
&\quad +2\alpha_2\alpha_3\cos\omega + 2\alpha_2\alpha_4\cos(2\omega) + 2\alpha_3\alpha_4\cos\omega
\end{aligned} \tag{4.657}$$

$$\begin{aligned}
|\hat{G}_0(\omega)|^2 &= \hat{G}_0(\omega)\hat{G}_0^*(\omega)\\
&= [\alpha_4 + \alpha_3 e^{-j\omega} + \alpha_2 e^{-j2\omega} + \alpha_1 e^{-j3\omega}]\\
&\quad [\alpha_4 + \alpha_3 e^{j\omega} + \alpha_2 e^{j2\omega} + \alpha_1 e^{j3\omega}]\\
&= \alpha_4^2 + \alpha_4\alpha_3 e^{j\omega} + \alpha_4\alpha_2 e^{j2\omega} + \alpha_4\alpha_1 e^{j3\omega}\\
&\quad +\alpha_3\alpha_4 e^{-j\omega} + \alpha_3^2 + \alpha_3\alpha_2 e^{j\omega} + \alpha_3\alpha_1 e^{j2\omega}\\
&\quad +\alpha_2\alpha_4 e^{-j2\omega} + \alpha_2\alpha_3 e^{-j\omega} + \alpha_2^2 + \alpha_2\alpha_1 e^{j\omega}\\
&\quad +\alpha_1\alpha_4 e^{-j3\omega} + \alpha_1\alpha_3 e^{-j2\omega} + \alpha_1\alpha_2 e^{-j\omega} + \alpha_1^2\\
&= \alpha_4^2 + \alpha_3^2 + \alpha_2^2 + \alpha_1^2\\
&\quad +2\alpha_4\alpha_3\cos\omega + 2\alpha_4\alpha_2\cos(2\omega) + 2\alpha_4\alpha_1\cos(3\omega)\\
&\quad +2\alpha_3\alpha_2\cos\omega + 2\alpha_3\alpha_1\cos(2\omega) + 2\alpha_2\alpha_1\cos\omega.
\end{aligned} \tag{4.658}$$

By direct comparison between the two results, we confirm the relationship.

Example B4.167

Starting from Equation (4.654) show that, if filter $h_0(n)$ is symmetric, its natural point of symmetry is point $(N-1)/2$.
For $N = 4$, we must show that the natural point of symmetry is point $3/2$. Symmetry means that if we shift the origin of the time axis at this point, the Fourier transform of the signal will become real. According to the shifting property of the Fourier transform, such a shifting will result in the Fourier transform of the unshifted signal to be multiplied with $e^{j3\omega/2}$. We shall multiply now both

(Continued)

Example B4.167 (Continued)

sides of Equation (4.654) with this factor, knowing that what we shall have on the left will be the Fourier transform of the shifted function, and what we shall get on the right must be real if the filter is symmetric:

$$\hat{H}_0(\omega)e^{j\frac{3\omega}{2}} = \alpha_1 e^{j\frac{3\omega}{2}} + \alpha_2 e^{j\frac{\omega}{2}} + \alpha_3 e^{-j\frac{\omega}{2}} + \alpha_4 e^{-j\frac{3\omega}{2}}. \tag{4.659}$$

We note that if the filter is symmetric, $\alpha_1 = \alpha_4$ and $\alpha_2 = \alpha_3$ and all the imaginary parts on the right-hand side of (4.659) will cancel.

Example B4.168

Show that Equation (4.652) implies that filter $h_0(n)$ is symmetric about point $(N-1)/2 - 1/4$.

Equation (4.652) implies that the left-hand side of Equation (4.654), for example, has the form $|\hat{H}_0(\omega)|e^{j\left(-\frac{N-1}{2}+\frac{1}{4}\right)\omega}$. This means that if we multiply both sides of the equation with $e^{-j\left(-\frac{N-1}{2}+\frac{1}{4}\right)\omega}$, the left-hand side will be real and so the right-hand side will be real too. Such a multiplication is equivalent to shifting the centre of the function to point $\frac{N-1}{2} - \frac{1}{4}$. So, when we shift the centre of the function at this point, the Fourier transform of the function becomes real. This means that the function is symmetric about its new centre. So, this point is the point of its symmetry.

Example B4.169

You are given an eight-tap symmetric filter $\tilde{h}(n)$: $(\alpha_1, \alpha_4, \alpha_2, \alpha_3, \alpha_3, \alpha_2, \alpha_4, \alpha_1)$. Show that its z-transform $\tilde{H}(z)$ may be written as

$$\tilde{H}(z) = H(z^2) + z^{-1}H(z^{-2}). \tag{4.660}$$

By definition:

$$\begin{aligned}
\tilde{H}(z) &= \alpha_1 z^{-4} + \alpha_4 z^{-3} + \alpha_2 z^{-2} + \alpha_3 z^{-1} + \alpha_3 + \alpha_2 z + \alpha_4 z^2 + \alpha_1 z^3 \\
&= \alpha_1 z^{-4} + \alpha_2 z^{-2} + \alpha_3 + \alpha_4 z^2 + \alpha_4 z^{-3} + \alpha_3 z^{-1} + \alpha_2 z + \alpha_1 z^3 \\
&= \alpha_1 (z^2)^{-2} + \alpha_2 (z^2)^{-1} + \alpha_3 + \alpha_4 (z^2)^1 \\
&\quad + z^{-1}(\alpha_4 z^{-2} + \alpha_3 + \alpha_2 z^2 + \alpha_1 z^4) \\
&= H(z^2) + z^{-1}[\alpha_4 (z^{-2})^1 + \alpha_3 + \alpha_2 (z^{-2})^{-1} + \alpha_1 (z^{-2})^{-2}] \\
&= H(z^2) + z^{-1}H(z^{-2})
\end{aligned} \tag{4.661}$$

where we defined:

$$H(z) \equiv \alpha_1 z^{-2} + \alpha_2 z^{-1} + \alpha_3 + \alpha_4 z. \tag{4.662}$$

Example B4.170

Assume that we wish to construct a four-tap filter that is low pass and approximately symmetric about a point 0.25 distance from the centre of the axis. Demonstrate that if you construct an eight-tap filter $\tilde{h}(n)$ that is low pass and symmetric, and sub-sample it, you will have a filter that is low pass and approximately symmetric about a point 0.25 distance from the centre of the axis.

Consider the filter of Example 4.169. We shall show that filter $h(n) = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ with z-transform $H(z)$ given by (4.662) is the filter we need.

Let us assume now that we start by constructing filter $\tilde{h}(n)$, so it is symmetric, low pass, of even size, with bandwidth half of that we want. When we sub-sample it by a factor of 2, the bandwidth will be doubled, so it will become what we want. As the filter is low pass and symmetric, it will be somehow bell-shaped. This is schematically shown in Figure 4.125. As long as the filter is relatively smooth, sub-sampling will not change its smooth shape much. These approximations are better as the filter is larger, and that is one of the reasons the filters we use in the dual tree complex wavelet transform tend to be large. Note that because of the symmetry and the even size, the centre of symmetry of the filter is between the fourth and the fifth elements. When we sub-sample, we keep the third and the fifth elements, the distance of which is two sampling spaces. The sub-sampled filter, however, halves all distances in terms of the continuous variable, so the distance of these points will be treated as 1, which means that the distance of the fifth element (which in the new filter is its third element) from the centre of symmetry will also be halved, i.e. it will be 0.25. This is the way the filter shown in Figure 4.124 and listed in the first column of Table 4.7 was constructed.

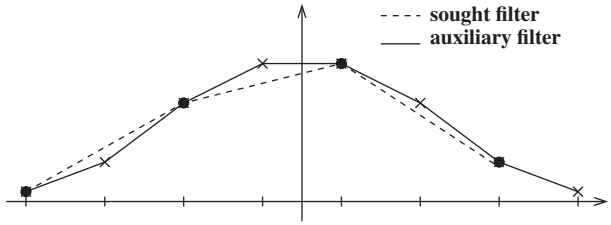


Figure 4.125 An eight-tap symmetric low pass filter, when sub-sampled by keeping the elements marked with the black dots creates a four-tap low pass filter, approximately symmetric about a point 0.25 away from the original point of symmetry. This filter then can be used to construct a second filter that will have approximately 0.5ω phase difference from it, so the two filters can be used as the scaling filters of two wavelet bases that are approximately the Hilbert transform pair of each other.

How do we apply the dual tree complex wavelet transform to analyse 1D signals?

We apply the following algorithm.

Step 1: Select an ordinary wavelet basis you wish to use. For example, any of the scaling filters of Table 4.6 will do.
Step 2: Convolve the input signal with the selected scaling and wavelet filters.
Step 3: Decimate every second sample, to produce bands L_2 and H_2.
Step 4: Decimate every first sample, to produce bands L_1 and H_1.

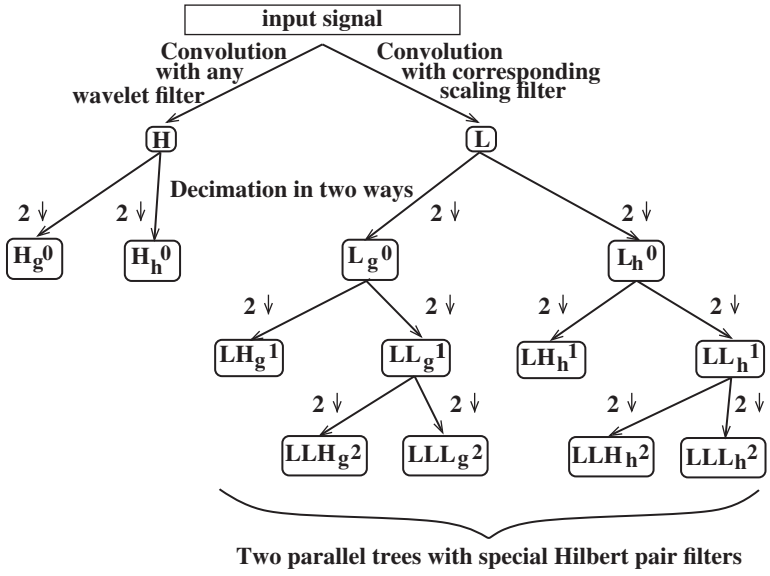


Figure 4.126 The signal may be fully reconstructed from the scaling and wavelet coefficients of each tree separately. One tree consists from bands $H_g^0, LH_g^1, LLH_g^2, LLL_g^2$ and the other from bands $H_h^0, LH_h^1, LLH_h^2, LLL_h^2$. The two reconstructions may be averaged to yield the recovered signal. Bands in level 0 have half the samples of the original signal, bands in level 1 have half the samples of the bands in level 0 and the bands in level 2 have half the samples of those in level 1. If the original signal consisted of N samples, this representation consists of $2N$ samples. Note that the two way decimation happens only at the first stage. The rest are two ordinary wavelet trees.

Step 5: Consider band L_1. Treat it as the input signal for tree wavelet expansion using the pair of scaling/wavelet filters g_0 and g_1 of the selected set of dual tree filters.

Step 6: Consider band L_2. Treat it as the input signal for tree wavelet expansion using the pair of scaling/wavelet filters h_0 and h_1 of the selected set of dual tree filters.

It is important to note that the first level of analysis achieves shift invariance by simply performing two different decimations. This is in accordance to our understanding that shift invariance could be achieved if we did not use decimation (see Example 4.149). So, by doing two complementary decimations, we effectively keep all samples. The need of special filters that satisfy the Hilbert pair conditions is necessary from level 2 onwards.

This analysis is schematically shown in Figure 4.126.

How do we use the dual tree complex wavelet transform to analyse images?

Figure 4.127 shows the frequency domain of a band and how one step of using two different wavelet filters may analyse it. Let us examine which bands are selected by the use of which filter. We shall use suffixes g and h to separate the outputs obtained with the g and h sets of filters.

Band E_h is extracted by using along both the x and the y axes the low pass filter h_0.
Band E_g is extracted by using along both the x and the y axes the low pass filter g_0.
Bands B_h and H_h are extracted by using along the x axis filter h_0 and along the y axis filter h_1.
Bands B_g and H_g are extracted by using along the x axis filter g_0 and along the y axis filter g_1.
Bands D_h and F_h are extracted by using along the x axis filter h_1 and along the y axis filter h_0.

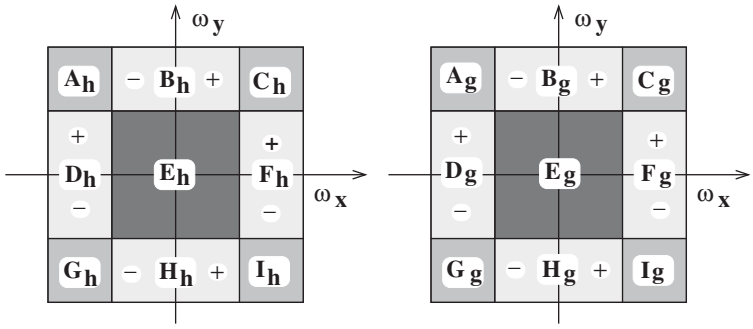


Figure 4.127 The different grey tones identify the bands to which the 2D frequency domain of an image is split in conventional wavelet analysis, by applying combinations of low and high pass filters along the *x* and *y* axes. When we use a dual tree wavelet transform, we use two sets of filters, indicated here by indices *h* and *g*. The + sign in bands *B* and *H* refers to the sub-bands with the positive frequencies along the *x* axis and the − sign to the sub-bands with the negative frequencies along the same axis. The + sign in bands *D* and *F* refers to the sub-bands with the positive frequencies along the *y* axis and the − sign to the sub-bands with the negative frequencies along the same axis. The issue here is how to combine the bands of the two different wavelet expansions so we extract useful information about the structure of the image.

Bands D_g and F_g are extracted by using along the x axis filter g_1 and along the y axis filter g_0.
Bands A_h, C_h, G_h and I_h are extracted by using along both axes filter h_1.
Bands A_g, C_g, G_g and I_g are extracted by using along both axes filter g_1.

In the dual tree complex wavelet transform, the corresponding filters are chosen to be approximate Hilbert transform pairs. This means that:

$h_0 + \mathrm{j}g_0$ picks up only positive frequencies.
$h_0 - \mathrm{j}g_0$ picks up only negative frequencies.
$h_1 + \mathrm{j}g_1$ picks up only positive frequencies.
$h_1 - \mathrm{j}g_1$ picks up only negative frequencies.

In the following, we shall indicate the positive frequencies of a band with a + next to the symbol of the band and the negative frequencies with a −. So:

If we apply filter $h_0 + \mathrm{j}g_0$ along the x axis and filter $h_1 + \mathrm{j}g_1$ along the y axis, we shall pick band $B+$ only.
If we apply filter $h_0 - \mathrm{j}g_0$ along the x axis and filter $h_1 + \mathrm{j}g_1$ along the y axis, we shall pick band $B-$ only.
If we apply filter $h_0 + \mathrm{j}g_0$ along the x axis and filter $h_1 - \mathrm{j}g_1$ along the y axis, we shall pick band $H+$ only.
If we apply filter $h_0 - \mathrm{j}g_0$ along the x axis and filter $h_1 - \mathrm{j}g_1$ along the y axis, we shall pick band $H-$ only.
If we apply filter $h_1 + \mathrm{j}g_1$ along the x axis and filter $h_0 + \mathrm{j}g_0$ along the y axis, we shall pick band $F+$ only.
If we apply filter $h_1 + \mathrm{j}g_1$ along the x axis and filter $h_0 - \mathrm{j}g_0$ along the y axis, we shall pick band $F-$ only.
If we apply filter $h_1 - \mathrm{j}g_1$ along the x axis and filter $h_0 + \mathrm{j}g_0$ along the y axis, we shall pick band $D+$ only.
If we apply filter $h_1 - \mathrm{j}g_1$ along the x axis and filter $h_0 - \mathrm{j}g_0$ along the y axis, we shall pick band $D-$ only.

If we apply filter $h_1 + \mathrm{j}g_1$ along both axes, we shall pick band C only.
If we apply filter $h_1 + \mathrm{j}g_1$ along the x axis and filter $h_1 - \mathrm{j}g_1$ along the y axis, we shall pick band I only.
If we apply filter $h_1 - \mathrm{j}g_1$ along the x axis and filter $h_1 + \mathrm{j}g_1$ along the y axis, we shall pick band A only.
If we apply filter $h_1 - \mathrm{j}g_1$ along both axes, we shall pick band G only.

Let us consider each of the above cases in turn, so we separate the real and the imaginary parts of the outputs. In the following, we shall use suffixes R and I to indicate the real and the imaginary parts of the output for each band. An extra bonus of the dual tree complex wavelet transform will be to be able to work out the symmetry of the local feature, from the ratio of the real and the imaginary parts of the two filter outputs.

Band $B+$:

$(h_0 + \mathrm{j}g_0)(h_1 + \mathrm{j}g_1) = (h_0h_1 - g_0g_1) + \mathrm{j}(h_0g_1 + g_0h_1)$.
Apply h_0 along the x axis and h_1 along the y axis. Output: LH_{hh}.
Apply g_0 along the x axis and g_1 along the y axis. Output: LH_{gg}.
Apply h_0 along the x axis and g_1 along the y axis. Output: LH_{hg}.
Apply g_0 along the x axis and h_1 along the y axis. Output: LH_{gh}.

Combine outputs:

$LH_{hh} - LH_{gg} \equiv B+_{\mathrm{R}}$ and $LH_{hg} + LH_{gh} \equiv B+_{\mathrm{I}}$.

Band $B-$:

$(h_0 - \mathrm{j}g_0)(h_1 + \mathrm{j}g_1) = (h_0h_1 + g_0g_1) + \mathrm{j}(h_0g_1 - g_0h_1)$.

Combine outputs:

$LH_{hh} + LH_{gg} \equiv B-_{\mathrm{R}}$ and $LH_{hg} - LH_{gh} \equiv B-_{\mathrm{I}}$

Band $H+$:

$(h_0 + \mathrm{j}g_0)(h_1 - \mathrm{j}g_1) = (h_0h_1 + g_0g_1) + \mathrm{j}(-h_0g_1 + g_0h_1)$.

Combine outputs:

$LH_{hh} + LH_{gg} \equiv H+_{\mathrm{R}} = B-_{\mathrm{R}}$ and $-LH_{hg} + LH_{gh} \equiv H+_{\mathrm{I}} = -B-_{\mathrm{I}}$

Band $H-$:

$(h_0 - \mathrm{j}g_0)(h_1 - \mathrm{j}g_1) = (h_0h_1 - g_0g_1) + \mathrm{j}(-h_0g_1 - g_0h_1)$.

Combine outputs:

$LH_{hh} - LH_{gg} \equiv H-_{\mathrm{R}} = B+_{\mathrm{R}}$ and $-LH_{hg} - LH_{gh} \equiv H-_{\mathrm{I}} = -B+_{\mathrm{I}}$

Band $F+$:

$(h_1 + \mathrm{j}g_1)(h_0 + \mathrm{j}g_0) = (h_1h_0 - g_1g_0) + \mathrm{j}(h_1g_0 + g_1h_0)$.
Apply h_1 along the x axis and h_0 along the y axis. Output: HL_{hh}.
Apply g_1 along the x axis and g_0 along the y axis. Output: HL_{gg}.
Apply h_1 along the x axis and g_0 along the y axis. Output: HL_{hg}.
Apply g_1 along the x axis and h_0 along the y axis. Output: HL_{gh}.

Combine outputs:

$HL_{hh} - HL_{gg} \equiv F+_{\mathrm{R}}$ and $HL_{hg} + HL_{gh} \equiv F+_{\mathrm{I}}$.

Band $F-$:

$(h_1 + \mathrm{j}g_1)(h_0 - \mathrm{j}g_0) = (h_1h_0 + g_1g_0) + \mathrm{j}(-h_1g_0 + g_1h_0)$.

Combine outputs:

$HL_{hh} + HL_{gg} \equiv F-_{\mathrm{R}}$ and $-HL_{hg} + HL_{gh} \equiv F-_{\mathrm{I}}$.

Band $D+$:

$(h_1 - \mathrm{j}g_1)(h_0 + \mathrm{j}g_0) = (h_1h_0 + g_1g_0) + \mathrm{j}(h_1g_0 - g_1h_0)$.

Combine outputs:

$HL_{hh} + HL_{gg} \equiv D+_{\mathrm{R}} = F-_{\mathrm{R}}$ and $HL_{hg} - HL_{gh} \equiv D+_{\mathrm{I}} = -F-_{\mathrm{I}}$.

Band $D-$:

$(h_1 - \mathrm{j}g_1)(h_0 - \mathrm{j}g_0) = (h_1h_0 - g_1g_0) + \mathrm{j}(-h_1g_0 - g_1h_0)$.

Combine outputs:

$HL_{hh} - HL_{gg} \equiv D-_{\mathrm{R}} = F+_{\mathrm{R}}$ and $-HL_{hg} - HL_{gh} \equiv D-_{\mathrm{I}} = -F+_{\mathrm{I}}$.

Band C:

$(h_1 + \mathrm{j}g_1)(h_1 + \mathrm{j}g_1) = (h_1h_1 - g_1g_1) + \mathrm{j}(h_1g_1 + g_1h_1)$.
Apply h_1 along the x axis and h_1 along the y axis. Output: HH_{hh}.
Apply g_1 along the x axis and g_1 along the y axis. Output: HH_{gg}.
Apply h_1 along the x axis and g_1 along the y axis. Output: HH_{hg}.
Apply g_1 along the x axis and h_1 along the y axis. Output: HH_{gh}.

Combine outputs:

$HH_{hh} - HH_{gg} \equiv C_{\mathrm{R}}$ and $HH_{hg} + HH_{gh} \equiv C_{\mathrm{I}}$.

Band I:

$(h_1 + \mathrm{j}g_1)(h_1 - \mathrm{j}g_1) = (h_1h_1 + g_1g_1) + \mathrm{j}(-h_1g_1 + g_1h_1)$.

Combine outputs:

$HH_{hh} + HH_{gg} \equiv I_{\mathrm{R}}$ and $-HH_{hg} + HH_{gh} \equiv I_{\mathrm{I}}$.

Band A:

$(h_1 - \mathrm{j}g_1)(h_1 + \mathrm{j}g_1) = (h_1h_1 + g_1g_1) + \mathrm{j}(h_1g_1 - g_1h_1)$.

Combine outputs:

$HH_{hh} + HH_{gg} \equiv A_{\mathrm{R}} = I_{\mathrm{R}}$ and $HH_{hg} - HH_{gh} \equiv AI = -I_{\mathrm{I}}$.

Band G:

$(h_1 - \mathrm{j}g_1)(h_1 - \mathrm{j}g_1) = (h_1h_1 - g_1g_1) + \mathrm{j}(-h_1g_1 - g_1h_1)$.

Combine outputs:

$HH_{hh} - HH_{gg} \equiv G_{\mathrm{R}} = C_{\mathrm{R}}$ and $-HH_{hg} - HH_{gh} \equiv GI = -C_{\mathrm{I}}$.

This process is schematically shown in Figure 4.128.

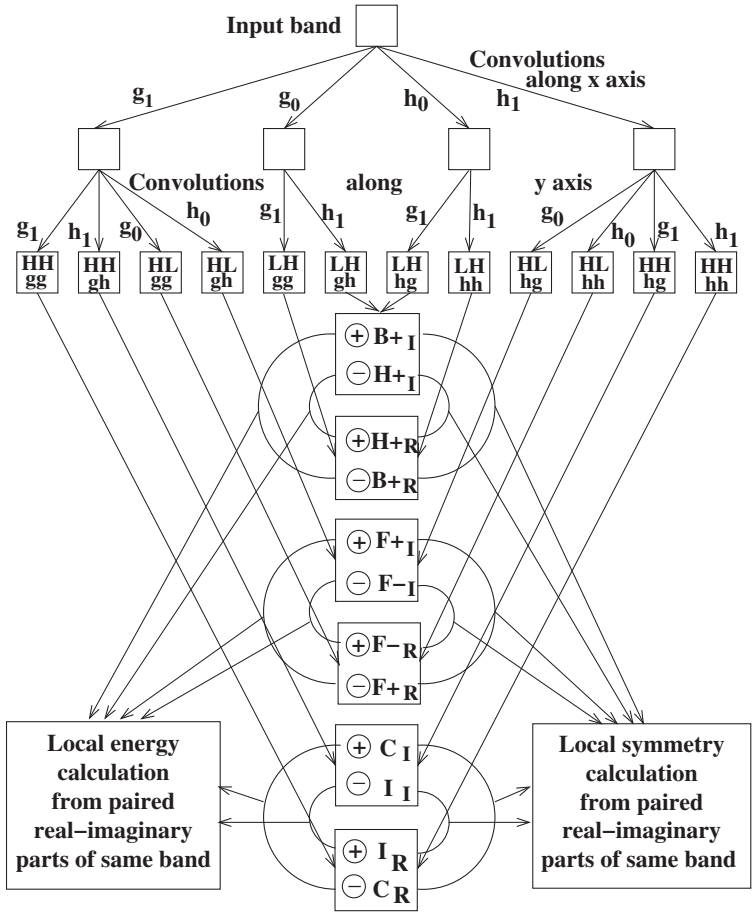


Figure 4.128 The application of the dual tree complex wavelet transform to an image. For simplicity operations like downsampling are not shown explicitly. Results for other bands for which the difference is in the sign of the shown outputs are not shown either. Symbol ⊕ implies addition of the input bands, while ⊖ implies subtraction. Next to each such symbol is the symbol of the produced band. Suffixes R and I indicate real and imaginary parts. Combining the real and the imaginary part of the same band allows the calculation of local energy and local symmetry.

How do we apply the dual tree complex wavelet transform to images in practice?

The answer depends on whether we wish to analyse the image and then re-synthesise it without certain bands, or we simply want to extract image features. If we simply want to extract image features, we may proceed as follows.

Step 0: Select the filters you will use for the two wavelet trees you will construct. Let us call them filters h_0 and h_1, and g_0 and g_1. These filters could be those listed in Table 4.7.

Step 1: Convolve the input image with the following filter combinations of these four filters, one applied along the x axis and the other along the y axis. The code we use to represent these outputs is as follows: L means that a filter with suffix 0 was used. H means a filter with suffix 1 was used. The distinction of the filter used is by suffixes h and g. In all cases, the first letter refers to convolution along the x axis, while the second letter refers to a subsequent convolution along

the y axis. So, we produce the following outputs: $LL_{gg}, LL_{hh}, LH_{gg}, LH_{hh}, LH_{gh}, LH_{hg}, HL_{gg}, HL_{hh}, HL_{gh}, HL_{hg}, HH_{gg}, HH_{hh}, HH_{gh}, HH_{hg}$.

Step 2: Combine these outputs as follows:

$BR \equiv LH_{hh} - LH_{gg}$, $BI \equiv LH_{gh} + LH_{hg}$. It corresponds to band $B+$ in Figure 4.127.
$HR \equiv LH_{hh} + LH_{gg}$, $HI \equiv LH_{gh} - LH_{hg}$. It corresponds to band $H+$ in Figure 4.127.
$FR \equiv HL_{hh} - HL_{gg}$, $FI \equiv HL_{gh} + HL_{hg}$. It corresponds to band $F+$ in Figure 4.127.
$DR \equiv HL_{hh} + HL_{gg}$, $DI \equiv HL_{gh} - HL_{hg}$. It corresponds to band $F-$ in Figure 4.127.
$CR \equiv HH_{hh} - HH_{gg}$, $CI \equiv HH_{gh} + HH_{hg}$. It corresponds to band C in Figure 4.127.
$IR \equiv HH_{hh} + HH_{gg}$, $II \equiv HH_{gh} - HH_{hg}$. It corresponds to band I in Figure 4.127.

Step 3: Compute the following features. Local energies per pixel (i,j)

$$E_X(i,j) \equiv XR(i,j)^2 + XI(i,j)^2 \tag{4.663}$$

where X stands for one of the letters $\{B, H, F, D, C, I\}$. Local symmetry features per pixel:

$$\Phi_X(i,j) \equiv \tan^{-1}\frac{|XR(i,j)|}{|XI(i,j)|}. \tag{4.664}$$

If the feature is purely symmetric, the numerator of this fraction is maximal and the denominator is minimal. So the calculated angle will tend to be close to 90°. If the feature is mostly antisymmetric, the numerator approaches 0 while the denominator is large, and $\Phi_X(i,j)$ is close to 0. So, $\Phi_X(i,j)$ is a measure of local symmetry of features in band X, taking values in the range $[0, \pi/2]$, with higher values indicating higher symmetry. Note that the value of $\Phi_X(i,j)$ is really meaningful only for pixels with energy $E_X(i,j)$ above a certain threshold. In other words, if both the numerator and the denominator in (4.664) are small, (i,j) is probably in a region that does not have any features in the particular band and what we calculate is dominated by noise.

If now we are interested in reconstructing the image from its wavelet components, then for the analysis we must follow the next algorithm.

Step 0: Select the ordinary wavelet and scaling filter you will use for the first level. For example, you may select a filter from Table 4.6. Call them **H** and **G**, respectively. Select also the dual tree complex wavelet transform filters you will select for the subsequent levels. For example, you may use the filters listed in Table 4.7. Call h_0 and g_0, and h_1 and g_1, the scaling and wavelet filters, respectively.

Step 1: Using cascaded convolutions along the x and y axes with filters **H** and **G**, produce bands LL, LH, HL, HH.

Step 2: Decimate each of these bands in four different ways, as shown in Figure 4.112, to produce bands $LL1, LH1, HL1, HH1, LL2, LH2, HL2, HH2, LL3, LH3, HL3, HH3, LL4, LH4, HL4$ and $HH4$.

Step 3: Treat each of the bands $LL1, LL2, LL3$ and $LL4$ as follows.

Convolve the band with the following pairs of filters h_0 and h_1, and g_0 and g_1, one applied along the x axis and the other along the y axis. The code we use to represent these outputs is as follows: L means that a filter with suffix 0 was used. H means a filter with suffix 1 was used. The distinction of the filter used is by suffixes h and g. In all cases, the first letter refers to convolution along the x axis, while the second letter refers to a subsequent convolution along the y axis. So for each one of the four input bands, we produce the following outputs:

$LL_{hh}, LH_{hh}, HL_{hh}, HH_{hh}$
$LL_{gg}, LH_{gg}, HL_{gg}, HH_{gg}$.

There will be in total 32 such output bands, eight for each input band.

Step 4: Decimate the bands you produced in step 2. This decimation now is ordinary decimation where we omit every second row and every second column.

Step 5: Each band you produce in step 3 may further be analysed by applying step 2. Stop when you have applied all levels of analysis you wanted.

If you want now to re-synthesise the original image, you have to use the following algorithm. Note that in the synthesis, you may omit any of the bands you like by simply setting all its coefficients to 0. This way you may clean the image from interference of certain frequencies and/or reduce noise. The algorithm that follows assumes that only one level of dual tree analysis was applied.

Step 0: Consider a band XY_{nn} you produced in step 2 of the previous algorithm. Here X and Y take values from the set $\{L, H\}$, while n stands either for h or for g. We remember that letter L indicates the use of the filter with suffix 0 (low pass scaling filter), while letter H indicates the use of the filter with suffix 1 (high pass, wavelet filter).

Step 1: Augment the band by adding rows of 0s along the y axis, after every existing row of pixels.

Step 2: Convolve the augmented band with filter n along the y axis.

Step 3: Augment the output by adding columns of 0s along the x axis, after every existing column of pixels.

Step 4: Convolve the augmented band with filter n along the x axis. Call the output you produce this way $\tilde{XY}_{nn}$.

Step 5: Add all bands that refer to the same filter n and come from the same original input band:

$A \equiv LL_{hh} + LH_{hh} + HL_{hh} + HH_{hh}$

$B \equiv LL_{gg} + LH_{gg} + HL_{gg} + HH_{gg}$.

Bands A and B are the two independent reconstructions of the corresponding band of the first level of analysis, recovered by using the two independent wavelet trees.

Step 6: Average bands A and B to recover the corresponding band of the first level of analysis, denoted as $LL1$, $LL2$, $LL3$ or $LL4$. Note that we do not use any suffixes here because these bands were the output of the first level, eg the output of using the ordinary wavelets. Having recovered these bands we are now out of the dual tree analysis. Let us call these recovered bands $L\tilde{L}1$, $L\tilde{L}2$, $L\tilde{L}3$ and $L\tilde{L}4$, to distinguish them from those we created during the analysis stage.

Step 7: Interlace bands $L\tilde{L}1$, $L\tilde{L}2$, $L\tilde{L}3$ and $L\tilde{L}4$, to form band LL the same size as the input image.

Step 8: Interlace bands $LH1$, $LH2$, $LH3$ and $LH4$, to form band LH the same size as the input image.

Step 9: Interlace bands $HL1$, $HL2$, $HL3$ and $HL4$, to form band HL the same size as the input image.

Step 10: Interlace bands $HH1$, $HH2$, $HH3$ and $HH4$, to form band HH the same size as the input image.

Step 11: Sum up the bands you produced in steps 7–10 to recover the original image.

It may appear futile sub-sampling and then interlacing bands that are not processed at all. This indeed is the case. However, the same way as we treat the sub-sampled versions of band LL of the first level of analysis, we may treat any other of the sub-sampled bands with the help of the dual tree complex wavelet transform. In the reconstruction, we may also omit any bands from the frequency contents of which we wish to clean the image.

Example 4.171

Use the dual tree complex wavelet transform to construct feature maps for the image of Figure 4.71.

Figure 4.129 shows the results of feature maps for the image 4.71 using the dual tree complex wavelet transform.

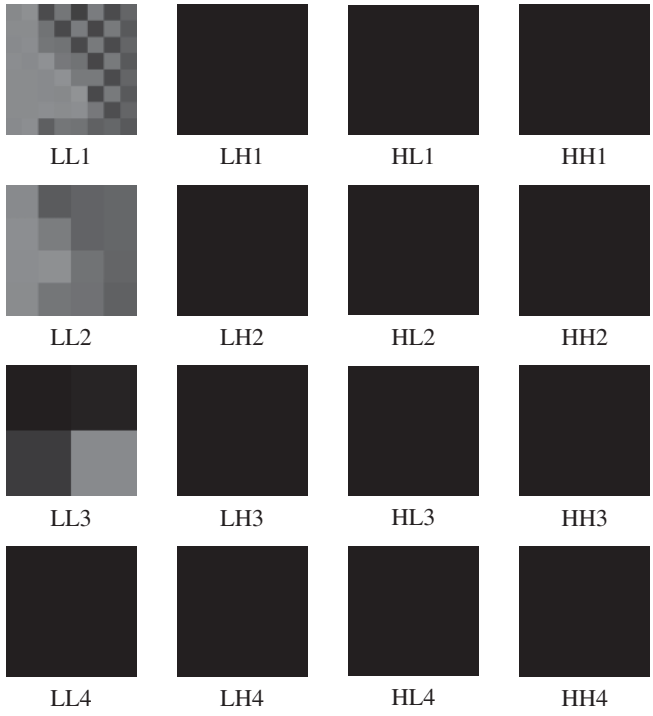


Figure 4.129 The feature maps for the image 4.71 using the dual tree complex wavelet transform. Source: Maria Petrou.

What are the drawbacks of wavelets in image processing?

Wavelets have been developed to catch optimally point singularities, e.g. spikes that might be present in a signal. In images, the type of high frequency structural characteristics we have are line-like, e.g. edges that have some elongation and specific orientation. That is the reason various derivatives of the wavelet transform have been developed, appropriate for catching the type of singularities relevant to images. One such development is the ridgelet transform, which makes use of the wavelet transform.

4.7 Ridgelets and Curvelets

What is a ridgelet?

A ridgelet is a function that is constant along straight lines and has a wave-like cross-section. For example, the following function is a ridgelet:

$$r(x_1, x_2) = \mathrm{e}^{-\frac{(x_1\cos\theta + x_2\sin\theta - \tau)^2}{2\sigma^2}} \cos(x_1\cos\theta + x_2\sin\theta - \tau). \tag{4.665}$$

Here τ, θ and σ are some constants.

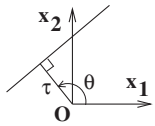


Figure 4.130 The length τ of the normal from the centre of the axes to the line and the angle θ this normal forms with the horizontal axis uniquely define the line. (θ, τ) are the Radon parameters of the line.

What is the continuous ridgelet transform?

The continuous ridgelet transform $\text{RID}_f(a, b, \theta)$ of a function $f(x_1, x_2)$ is defined as

$$\text{RID}_f(a, b, \theta) \equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \psi_{ab\theta}(x_1, x_2) f(x_1, x_2) \mathrm{d}x_1 \mathrm{d}x_2 \tag{4.666}$$

where

$$\psi_{ab\theta}(x_1, x_2) \equiv \frac{1}{\sqrt{a}} \psi \left(\frac{x_1 \cos\theta + x_2 \sin\theta - b}{a} \right) \tag{4.667}$$

with a being a scaling constant, b a shifting parameter and $\psi(x)$ a mother wavelet function, like for example:

$$\psi(x) = \mathrm{e}^{-x^2} \cos x. \tag{4.668}$$

How do we calculate the continuous ridgelet transform?

We calculate it in two stages:
(i) First we calculate the continuous Radon transform of the function
(ii) Then we apply the 1D wavelet transform to the Radon transform, with respect to its displacement parameter (see Example 4.172).

What is the Radon transform of a function?

The **Radon transform** of a function $f(x_1, x_2)$ is defined as a function of the parameters that characterise the lines that trace the domain of the function and returns the integral of the function along each such line:

$$\text{RAD}_f(\theta, \tau) \equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x_1, x_2) \delta(x_1 \cos\theta + x_2 \sin\theta - \tau) \mathrm{d}x_1 \mathrm{d}x_2 \tag{4.669}$$

Figure 4.130 shows the definition of parameters (θ, τ) that uniquely characterises each line that traces the function domain. Parameter θ defines the orientation of the line, while τ is the displacement of the line away from the origin of the axes.

Example B4.172

Show that if you apply the 1D wavelet transform to the Radon transform, with respect to parameter τ, you will obtain the ridgelet transform of function $f(x_1, x_2)$.
The wavelet transform of a function is defined by (4.387).

Let us use this definition in conjunction with (4.383) to work out the wavelet transform of $\text{RAD}_f(\theta, \tau)$*:*

$$w_{\text{RAD}_f}(a, b) = \int_{-\infty}^{\infty} \text{RAD}_f(\theta, \tau) \psi_{ab}(\tau) \mathrm{d}\tau$$

$$
\begin{aligned}
&= \iiint_{-\infty}^{\infty} f(x_1, x_2)\delta(x_1\cos\theta + x_2\sin\theta - \tau)\frac{1}{\sqrt{a}}\psi\left(\frac{\tau - b}{a}\right)\mathrm{d}x_1\mathrm{d}x_2\mathrm{d}\tau \\
&= \frac{1}{\sqrt{a}}\iint f(x_1, x_2)\left\{\int \psi\left(\frac{\tau - b}{a}\right)\delta(x_1\cos\theta + x_2\sin\theta - \tau)\mathrm{d}\tau\right\}\mathrm{d}x_1\mathrm{d}x_2 \\
&= \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x_1, x_2)\psi\left(\frac{x_1\cos\theta + x_2\sin\theta - b}{a}\right)\mathrm{d}x_1\mathrm{d}x_2 \\
&= \mathrm{RID}_f(a, b, \theta). \qquad (4.670)
\end{aligned}
$$

Here we made use of (4.669) and (4.666).

What exactly does the ridgelet transform do to the function?

To answer this question we have to understand what each of the two stages of the calculation of the ridgelet transform achieves.
(i) In the first stage, the Radon transform uses batches of parallel lines (fixed θ) along which it

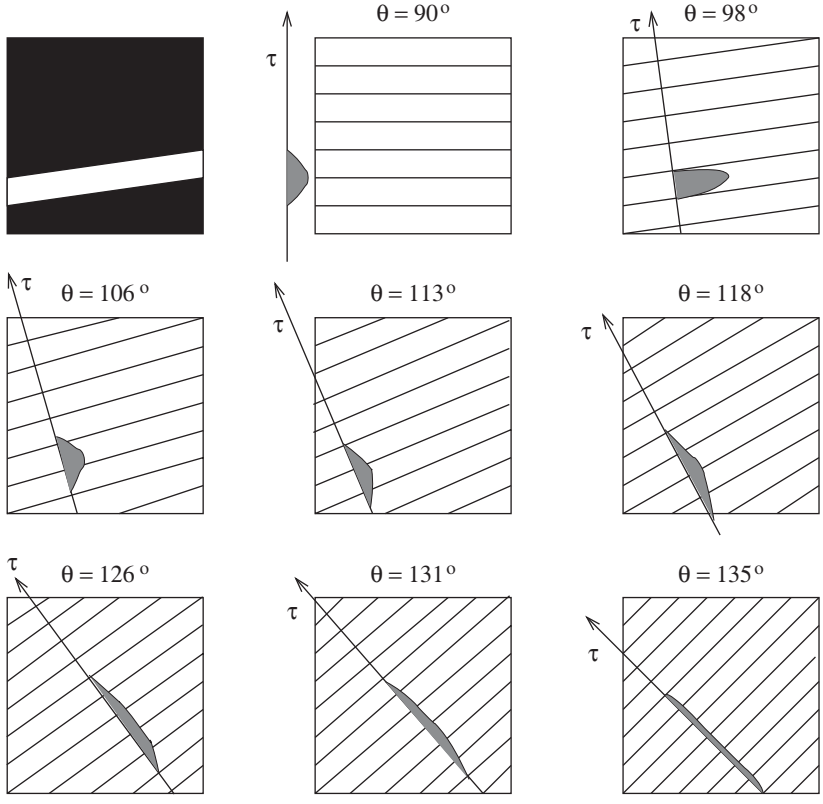


Figure 4.131 At the top left, an image containing a line-like structure. The Radon transform uses batches of parallel lines (a few of which are shown here) to project the image along an axis orthogonal to them, parametrised by variable τ. If an image has a line-like structure along a certain direction, this structure will create a sharp peak along the τ axis when the orientation of the lines coincides with the orientation of the structure. Thus, a line-like discontinuity in the image is converted into a point discontinuity, to be detected at a further stage, by processing the signals we create along the τ axes.

projects the function. By allowing θ to take all possible values, we effectively consider all possible lines that criss-cross the domain of the function. Let us consider one such batch of parallel lines (fixed θ). Let us also assume that the function happens to have a structure, parallel to these lines. As the function is integrated along these lines, there will be a line that will coincide with this structure and it will create a sharp peak in the signal along the τ axis. The structure of the image will also affect the projection of the image along many other orientations. However, in all other orientations, the structure will create a more or less flat bump rather than a sharp peak. This is schematically shown in Figure 4.131.

(ii) When we take the wavelet transform with respect to parameter τ in the second stage of the calculation, it is the sharp peak in the preferred orientation we expect to detect. We use the wavelet transform, since it is ideal for identifying pulse-like discontinuities in signals.

Thus, the ridgelet transform converts the line-like discontinuities of a 2D function into point discontinuities and subsequently detects them using wavelets.

Can the second stage of the ridgelet transform be performed with the help of a transform other than the wavelet transform?

Possibly, as long as the transform is appropriate for the detection of point discontinuities. However, the wavelet transform remains ideal for the job. For example, we might use Gabor functions instead of wavelets, as they also extract local information. We could not use the Fourier transform as that is a global transform inappropriate for the job. If the Fourier transform is used, all we do is taking a slice through the 2D Fourier transform of the original image, something that is not particularly helpful in the representation of 2D linear structures of the image (see Example 4.173).

Example B4.173

Show that if you apply the 1D Fourier transform to the Radon transform $\mathrm{RAD}_f(\theta,\tau)$ of a function $f(x_1,x_2)$, with respect to parameter τ, the result is a cross-section of the 2D Fourier transform of the function, along the line that is at an angle θ with respect to the frequency axis that corresponds to coordinate x_1. This is known as the *projection slice theorem*.

The Fourier transform of $\mathrm{RAD}_f(\theta,\tau)$*, given by Equation (4.669), is*

$$\begin{aligned}
\int_{-\infty}^{\infty}\mathrm{RAD}_f(\theta,\tau)\mathrm{e}^{-\mathrm{j}\omega\tau}\mathrm{d}\tau &= \iiint_{-\infty}^{\infty} f(x_1,x_2)\delta(x_1\cos\theta+x_2\sin\theta-\tau)\mathrm{e}^{-\mathrm{j}\omega\tau}\mathrm{d}x_1\mathrm{d}x_2\mathrm{d}\tau\\
&= \iint f(x_1,x_2)\left\{\int\delta(x_1\cos\theta+x_2\sin\theta-\tau)\mathrm{e}^{-\mathrm{j}\omega\tau}\mathrm{d}\tau\right\}\mathrm{d}x_1\mathrm{d}x_2\\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x_1,x_2)\mathrm{e}^{-\mathrm{j}\omega(x_1\cos\theta+x_2\sin\theta)}\mathrm{d}x_1\mathrm{d}x_2\\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x_1,x_2)\mathrm{e}^{-\mathrm{j}(\omega\cos\theta)x_1-\mathrm{j}(\omega\sin\theta)x_2}\mathrm{d}x_1\mathrm{d}x_2\\
&= F(\omega\cos\theta,\omega\sin\theta) \qquad (4.671)
\end{aligned}$$

where $F(\omega_1,\omega_2)$ is the 2D Fourier transform of $f(x_1,x_2)$.

Figure 4.132 shows schematically the relationship between the Radon and the Fourier transforms.

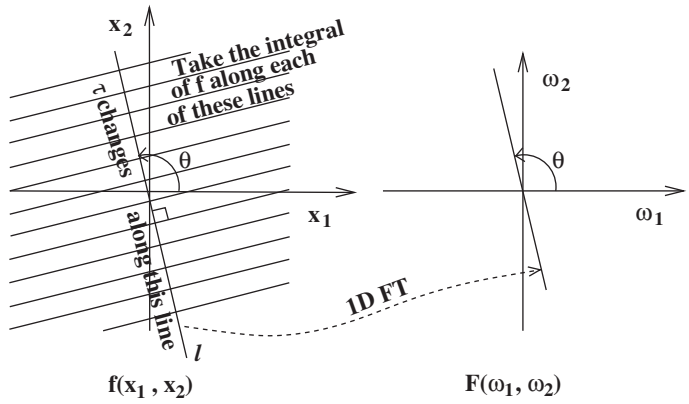


Figure 4.132 For fixed θ, Radon transform considers a batch of parallel lines tracing the domain of the 2D function. The function is integrated along each one of these lines. This is equivalent to collapsing the function on a line l, orthogonal to these lines, to create a 1D signal, consisting of the values of the integral of the function along each tracing line. Parameter τ varies along line l. Taking the 1D Fourier transform of the 1D signal we created that way, as a function of τ, gives us the cross-section of the Fourier transform of the function along a line that forms angle θ with respect to frequency axis ω_1.

How can we apply the ridgelet transform to digital images?

This is done with the help of the **finite Radon transform**. In particular, we obtain the so called **finite ridgelet transform** by applying the 1D wavelet transform to the finite Radon transform, along the axis of the displacement parameter.

What is the finite Radon transform?

The finite Radon transform considers sets of pixels that form straight lines in an image. It then sums up the values of the pixels that make up each line. If the indices of a pixel are (i,j), we may say that for the pixels that make up a line, $j = ki + l$, where k is the slope of the line and l is its intersection with the j axis. Let us denote such a line with L_{kl}. The finite Radon transform then is given by

$$F_{\text{RAD}}(k,l) \equiv \frac{1}{\sqrt{N}} \sum_{(i,j)\in L_{kl}} I(i,j) \tag{4.672}$$

where $N \times N$ is the size of image $I(i,j)$. In particular, the lines we use are defined as the following sets of pixels:

$$L_{kl} \equiv \{(i,j) : j = ki + l \text{ modulo } N,\ i = 0,1,\dots,N-1\} \tag{4.673}$$

$$L_{Nl} \equiv \{(l,j) : j = 0,1,\dots,N-1\}. \tag{4.674}$$

Equation (4.674) defines the set of vertical lines, i.e. those that have $k = N$, which corresponds to infinite slope.

How do we form the lines that we use in the finite Radon transform?

Only lines with slops from a discrete set of slopes are considered, as the requirement is to form lines made up from pixels with their centres exactly on the line. The bottom right panel of Figure 4.133 shows all possible lines that can be formed when starting from the bottom left pixel of a 7×7 image and joining its centre with the centres of the pixels in the column next to it and the pixel directly above it. The other panels of the same figure show all straight lines that can be formed for each of the permissible slopes. The pixels that make up the same line are marked with the same symbol. Figure 4.134 shows how the lines of a particular slope are formed. The image is considered to be repeated ad infinitum in all directions, so the lines are wrapped back to the same original image. Note that although the pixels are read in a certain order to form the line, at the end of the day, as we simply sum the values of these pixels, the order by which the pixels are read does not matter.

It can be shown that for N being a **prime** number, the lines of each slope exactly fill the image, i.e. all image pixels belong to a line of that slope, and that each image pixel belongs to exactly one line for each different slope.

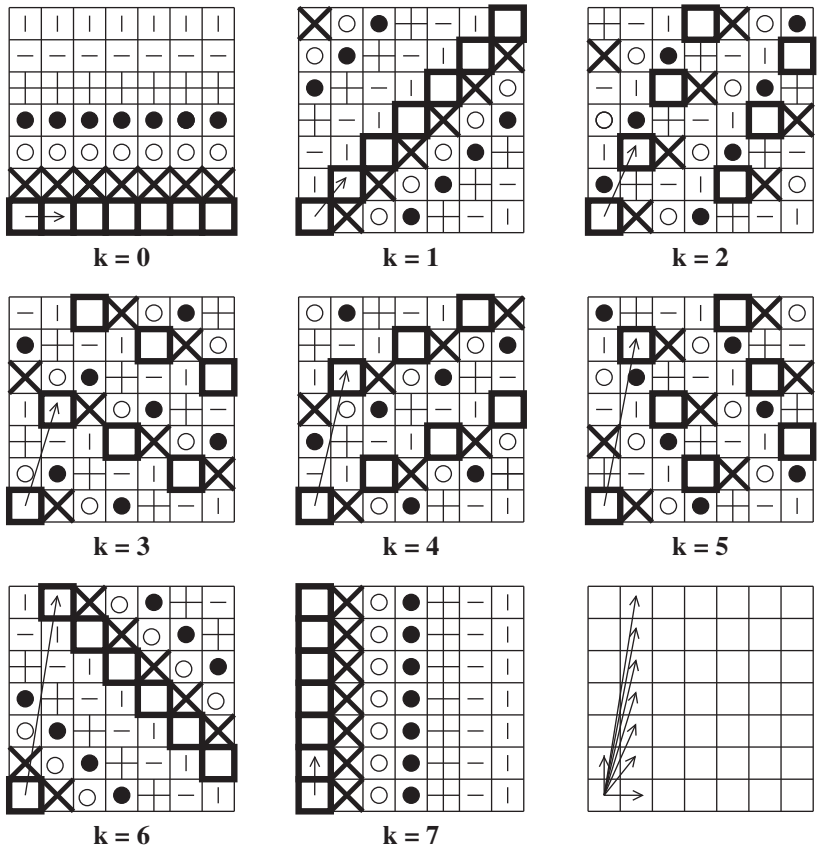


Figure 4.133 All lines that can be formed in a 7×7 image. Each panel shows the lines of a different slope. In each panel, pixels marked with the same symbol belong to the same line. In each panel, an arrow shows the direction of the line by joining its first two pixels. See Figure 4.134 to see how the lines of a certain slope are created. The bottom right panel shows all line directions considered.

Why *N* has to be prime for the lines of a certain slope to fill the image?

Figure 4.135 shows an attempt to construct the lines of a certain slope for an 8×8 image. Compare this with Figure 4.134. To make up such a line, we connect a pixel with the pixel that is 2 places up in the next column. Number 2 divides exactly number 8, the size of the image, so when we wrap the line back in the original image, the pixel goes again at the bottom row of the image (8 *modulo* $2 = 0$) and so there is no way to cover the pixels of the second row from the bottom of the image by shifting a line of this particular slope. So, N has to be prime. Only in that case number N *modulo* x, where x is any number between 2 and $N - 1$, is not 0 and so each time wrapping round happens, the pixels of a row, other than the original row, are considered.

How many lines made up from pixels can we have in an $N \times N$ image?

We can have $N(N + 1)$ different lines, forming $N + 1$ batches of parallel lines consisting of N lines each. In Figures 4.133 and 4.134, in order to construct the lines, we started from the bottom left

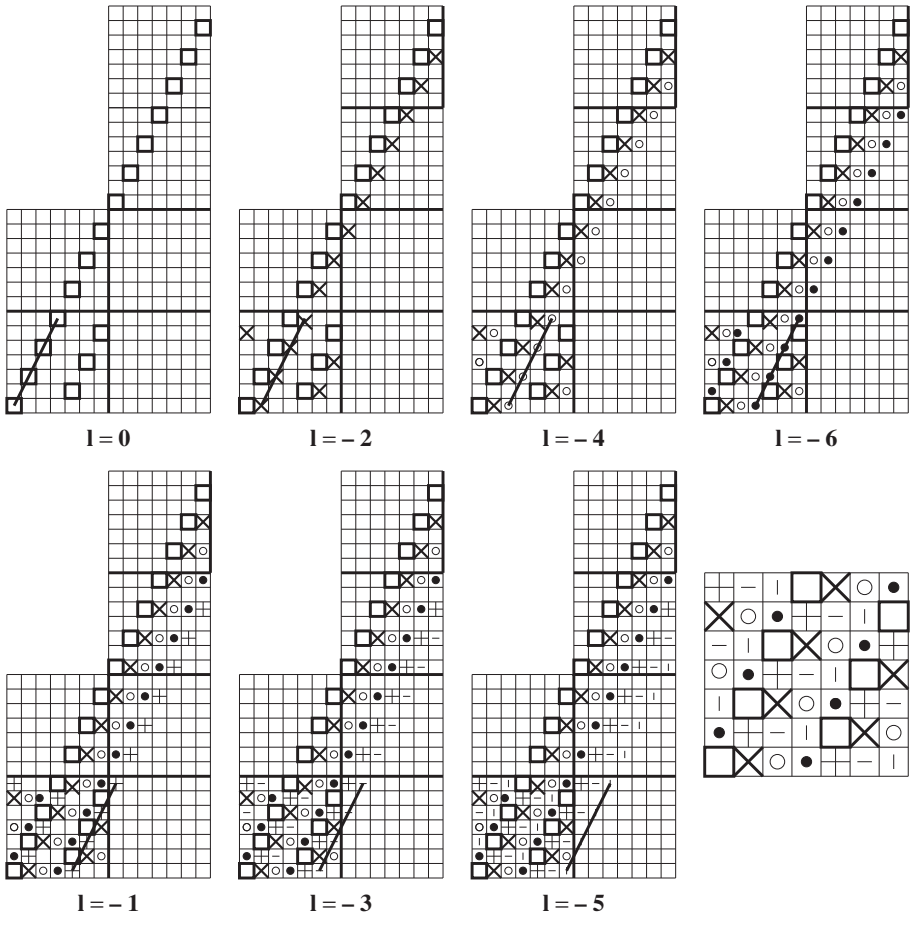


Figure 4.134 All lines of a certain fixed slope, made up from pixels, that can be formed by shifting their abscissae in a 7×7 image. The pixels that make up each line are marked with the same symbol. The image is assumed to be repeated in all directions. The thick horizontal and vertical lines separate these repetitions. The bottom right panel shows all lines of this particular slope which have been constructed.

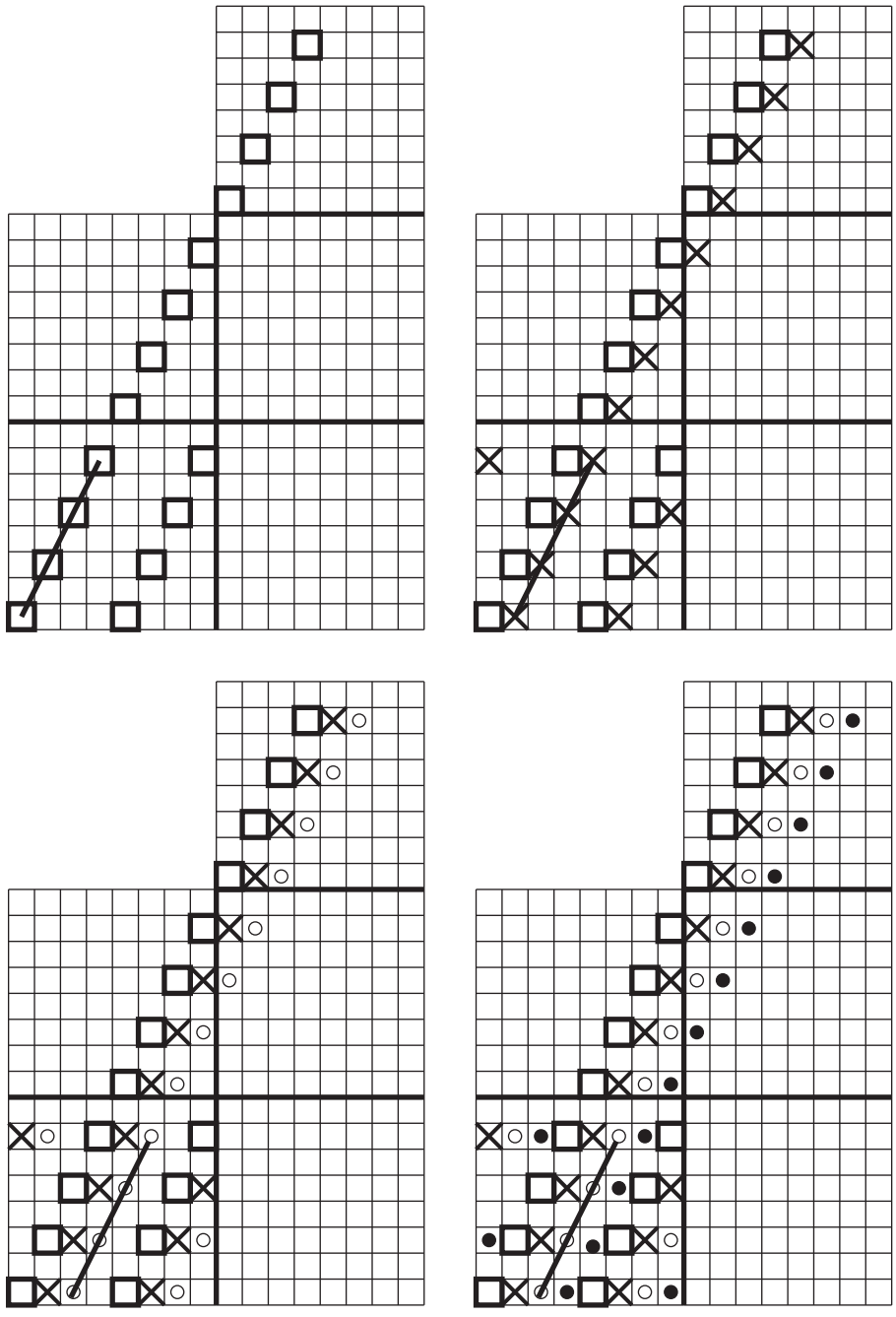


Figure 4.135 Lines of a certain slope created in an image with size not a prime number fail to pass through all the pixels of the image, because when the vertical shift from the first pixel to the next divides exactly the height of the image, the line wraps round to the pixels of the same row. In this example, the height of the image is 8 and the vertical shift of the lines we are constructing is 2, so it is impossible for lines of this slope to pass through pixels of the second, fourth, sixth and eighth row of the image.

pixel and we joined it with the pixel above it and with all pixels in the column on its right to form lines of different slopes. This gave us $N+1$ lines, all of different slopes. Then we shifted the origin of each one of these lines to all N pixels of the bottom row of the image, so we constructed N parallel lines for each slope.

How many pixels does each line in the finite Radon transform consist of?

For an $N \times N$ image, with N being a prime number, each line is made up from N pixels. We can see that from Equation (4.673) where i takes up only N distinct values and to each i corresponds only one j.

If the lines we use are made up from pixels, how sure are we that all pixels are used in an unbiased way?

It can be shown that each pixel belongs exactly to one line only from each batch of parallel lines (see Example 4.175). Also, that between any two pixels only one line of the $N(N+1)$ lines we construct passes (see Example 4.174). So, there are no pixels left out and each pixel belongs to $N+1$ different lines, one from each batch of parallel lines.

Example 4.174

Show that with the method of constructing lines demonstrated in Figures 4.133 and 4.134, each pair of points belongs only to a single line.

First of all, we can consider any pair of pixels in Figure 4.133 and observe that only in a single panel these two pixels have the same symbol. So, it is obvious that only a line of a particular slope is defined from these two pixels.

To show this with the help of equations, let us consider two pixels (i_A, j_A) *and* (i_B, j_B)*, with* $0 \leq i_A, i_B, j_A, j_B < N$*. We must remember that any set of such coordinates represents an infinite number of points, as we can add as many Ns as we like to each one of them, since the image is assumed to be repeated ad infinitum in all directions. We can then assume that if necessary, we add N to* j_A *and/or to* i_A*, so that* $j_A \geq j_B$ *and* $i_A \geq i_B$*. Let us assume that the line that passes through these two points has slope k and intercept l. Then obviously:*

$$\left.\begin{array}{rcl} j_A &=& ki_A + l \\ j_B &=& ki_B + l \end{array}\right| \Rightarrow j_A - j_B = k(i_A - i_B) \Rightarrow k = \frac{j_A - j_B}{i_A - i_B}. \tag{4.675}$$

Note that if $j_A - j_B < i_A - i_B$*, we can always add another N to* $j_A - j_B$ *(it is as if we added 2 Ns to* j_A *to begin with) so that* $j_A - j_B \geq i_A - i_B$*. Once we know k, we can substitute in one of the original equations to work out l:*

$$l = j_A - ki_A \tag{4.676}$$

If this number is negative, we may add N to make it positive, so we may place it along the vertical axis in the panels of Figure 4.133, with $l = 0$ *being at the bottom left corner.*

Consider, for example, pixels $(2, 3)$ *and* $(1, 4)$ *in the* 7×7 *grid of Figure 4.133. We have* $i_A = 2$*,* $j_A = 3$*,* $i_B = 1$ *and* $j_B = 4$*. We add 7 to* j_A*, so* $j_A = 10 > j_B$*. Then* $k = (10-4)/(4-2) = 6$*. To work out l we use (4.676):* $l = 10 - 6 \times 2 = -2$*. We add 7, so* $l = 5$*. We can check that* $j_A = 6i_A + 5$ *modulo 7 and* $j_B = 6i_B + 5$ *modulo 7 hold, by direct substitution of the values* $i_A = 2$*,* $j_A = 3$*,*

(Continued)

Example 4.174 (Continued)

$i_B = 1$ and $j_B = 4$. In Figure 4.133 lines with slope $k = 6$ are those shown in the bottom left panel. The line with ordinate 5 is marked with symbol −. We note that pixels $(2, 3)$ and $(1, 4)$ indeed belong to this line.

Example 4.175

Show that every pixel in the image belongs to a line from each batch of slopes, i.e. show that each pixel is crossed by lines of all possible slopes.

First, we can see that from Figure 4.133, where if we focus on a specific pixel, we notice that in each panel it belongs to exactly one of the lines.

Consider a pixel (i_0, j_0). Let us consider all possible line slopes we use, in turn. For line slope k, we can solve the equation that defines the pixel coordinates that make up the line as follows: $l = j_0 - ki_0$. To prove that the pixel is crossed by one line from each considered slope, we must show that for each legitimate k, the value of l is also a legitimate ordinate, i.e. it is in the range $[-N + 1, 0]$. This range may be mapped to the range $[1, N]$, by adding N to l. Note that the range $[1, N]$ is the same as the range $[0, N - 1]$, since $N_{modulo\ N} = 0$, and so numbers $[1, 2, \dots, N]_{modulo\ N}$ become numbers $[0, 1, \dots, N - 1]$. So, we shall show that l takes values in the range $[0, N - 1]$. Note also that, as j_0, k and i_0 take values in the range $[0, N - 1]$, the maximum value l can take is $N - 1$. So, as long as l turns out to be positive, it is unique and acceptable. The minimum value l can take is $-(N - 1)^2$. However, when l is negative, we can add as many Ns as required to make it positive. So, if $l = -(N - 1)^2$, we can add $(N - 1)N$ to have $l = -(N - 1)^2 + (N - 1)N = N - 1 \in [0, N - 1]$.

In general, since $-(N - 1)^2 \leq l \leq N - 1$, when l turns out to be negative and different from $-(N - 1)^2$, it will be equal to $-(N - 1)^2 + A = -N^2 - 1 + 2N + A$, where A is a positive integer ($0 < A < (N - 1)^2$). Given that l is computed modulo N, we can omit from this sum terms N^2 and $2N$ since they are multiples of N. Therefore, in all cases $l = A - 1 \geq 0$, which can be made to be modulo N, so l will be unique and in the range $[0, N - 1]$. Therefore, for every k there will be a unique value of ordinate l which will make the corresponding line pass through pixel (i_0, j_0).

How do we compute the finite Radon transform?

We apply the following algorithm.

Step 0: Make sure the image is of size $N \times N$, with N being a prime number. Remove the mean of the image.

Step 1: Create an array F_{RAD} of size $(N + 1) \times N$, and set all its elements to 0.

Step 2: Consider all combinations (k, l), where $k, l = 0, 1, \dots, N - 1$. For each combination (k, l), and for each $i = 0, 1, \dots, N - 1$, work out $j = ki + l$ modulo N and set $F_{\text{RAD}}(k, l) = F_{\text{RAD}}(k, l) + I(i, j)$.

Step 3: Set $F_{\text{RAD}}(N, l) = \sum_{j=0}^{N-1} I(N, j)$.

Step 4: Divide all elements of F_{RAD} by $\sqrt{N}$.

Example 4.176

Demonstrate the finite Radon transform using the following image:

$$g = \begin{pmatrix} 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 100 & 100 & 100 & 255 & 255 \\ 255 & 100 & 150 & 150 & 150 & 100 & 255 \\ 255 & 100 & 150 & 200 & 150 & 100 & 255 \\ 255 & 100 & 150 & 150 & 150 & 100 & 255 \\ 255 & 255 & 100 & 100 & 100 & 255 & 255 \\ 255 & 255 & 255 & 50 & 255 & 255 & 255 \end{pmatrix}. \tag{4.677}$$

This is a 7×7 image. So, we can use the coding of the pixels in Figure 4.133 to work out the values of RAD_{kl}, for each slope k and each ordinate l. First of all, we have to remove the mean of the image. The mean is 194.59. There are 49 values we have to work out, but we give here the working out of just a few of them. In particular, we sum the values of the pixels marked with an "X" in the top right panel of Figure 4.133 and form line $L_{2,-2}$, and the pixels marked with a black square in the central panel, which form line L_{40}:

$$\begin{aligned} F_{\mathrm{RAD}}(4,0) &= \frac{3 \times 255 + 3 \times 100 + 150 - 7 \times 194.59}{7} = -21.02 \\ F_{\mathrm{RAD}}(2,-2) &= \frac{5 \times 255 + 2 \times 150 - 7 \times 194.59}{7} = 1.12. \end{aligned} \tag{4.678}$$

The full F_{RAD} of this image is:

$$F_{\mathrm{RAD}} = \begin{pmatrix} 60.41 & -6.02 & -28.88 & -51.02 & -28.88 & -6.02 & 60.41 \\ -13.88 & 1.12 & 16.12 & -13.16 & 1.12 & -13.88 & 22.55 \\ 1.12 & 30.41 & 8.26 & 1.12 & 1.12 & -21.02 & -21.02 \\ -21.02 & 30.41 & 30.41 & -50.31 & 1.12 & 8.26 & 1.12 \\ 1.12 & 8.26 & 1.12 & -50.31 & 30.41 & 30.41 & -21.02 \\ -21.02 & -21.02 & 1.12 & 1.12 & 8.26 & 30.41 & 1.12 \\ 22.55 & -13.88 & 1.12 & -13.16 & 16.12 & 1.12 & -13.88 \\ 60.41 & -6.02 & -28.88 & -21.73 & -28.88 & -6.02 & 31.12 \end{pmatrix}. \tag{4.679}$$

For the Radon transform, k takes values from 0 to 7 and l from -6 to 0.

Is the finite Radon transform invertible?

Yes, as long as the image has zero mean value (see Examples 4.177 and 4.178).

Example 4.177

Show that if the image has 0 mean, when we sum up all the values of the finite Radon transform for a fixed slope, we get 0.

The proof is obvious since lines of a certain orientation cover the image fully, with each pixel belonging to only one of these lines, as shown in Figure 4.133. Then summing up the sums of the individual lines, we effectively sum up the values of all pixels, which yields 0 since the image is assumed to be zero-mean.

Example 4.178

Show that the image can be fully reconstructed from the values of its $F_{\mathbf{RAD}}$, given by (4.672) provided it is zero-mean.

Let us sum up the values of all lines that pass through pixel (i,j). First of all, we know from Example 4.174 that this pixel forms only one line with every other pixel in the image. So, every pixel in the image belongs to exactly one line passing through pixel (i,j), except pixel (i,j) itself that belongs to $N+1$ lines, i.e. to all lines passing through it. Let us call P_{ij} the set of lines that pass through pixel (i,j). Let us then sum all values of F_{RAD} along these lines, and divide with $\sqrt{N}$:

$$\frac{1}{\sqrt{N}}\sum_{(k,l)\in P_{ij}} F_{\mathrm{RAD}}(k,l) = \frac{1}{N}\sum_{(k,l)\in P_{ij}}\sum_{(i',j')\in L_{kl}} I(i',j')$$
$$= \frac{1}{N}\left(\sum_{(i',j')} I(i',j') + NI(i,j)\right). \tag{4.680}$$

This is because summing over all lines that pass through pixel (i,j) will sum over all pixels of the image and it will include pixel (i,j) $N+1$ times. Since the image has zero mean, the first term inside the brackets of (4.680) is zero, so:

$$\frac{1}{\sqrt{N}}\sum_{(k,l)\in P_{ij}} F_{\mathrm{RAD}}(k,l) = I(i,j). \tag{4.681}$$

*This is the **inverse finite Radon transform**, also known as the **back projection operator**, as it allows one to reconstruct the image from its projections.*

How do we compute the inverse finite Radon transform?

We apply the back projection operator as follows.

Step 1: Create an array J, $N\times N$, the same size as the original image, and set all its elements equal to 0.

Step 2: For each pixel (i,j), and for each $k=0,1,\ldots,N-1$, work out $l=j-ki$ modulo N. Set $J(i,j)=J(i,j)+F_{\mathrm{RAD}}(k,l)+F_{\mathrm{RAD}}(N,i)$.

Step 3: Divide all elements of J with $\sqrt{N}$.

Step 4: Add the mean of the original image to all elements.

Can we perform the finite Radon transform using matrix multiplications?

Yes (see Examples 4.179–4.183). The matrix we need is very large, but it is sparse with a very well defined partition structure, that allows it to be used easily, both for the direct and the inverse transform.

Example 4.179

For an image of size 7×7 work out a matrix which, when used to multiply the image from the left, written as a column vector, it will produce its finite Radon transform.

Let us construct a vector $\mathbf{\Phi_{kl}}$ so that its elements have value 1 if the pixels they multiply belong to line L_{kl} and they have value 0 if they do not belong. Let us do that for $k = 4$ and $l = 0$, using Figure 4.133. Pixels that make up line L_{40} are marked with the black squares in the central panel of this figure, and have coordinates:

$$(0,0), \quad (1,4), \quad (2,1), \quad (3,5), \quad (4,2), \quad (5,6), \quad (6,3). \tag{4.682}$$

When the image is written in vector form, its columns are stuck one under the other. We remember that in conventional image representations, the coordinates change from top left to bottom right, while in Figure 4.133 we put the $(0,0)$ element at the bottom left. So, when we stuck the image, we must stuck the second column above the first, the third above the second, and so on, and then turn this column upside down to form vector:

$$\begin{aligned}\mathbf{g} = \big(&g_{00}, g_{01}, g_{02}, g_{03}, g_{04}, g_{05}, g_{06}, g_{10}, g_{11}, g_{12},\\ &g_{13}, g_{14}, g_{15}, g_{16}, g_{20}, g_{21}, g_{22}, g_{23}, g_{24}, g_{25},\\ &g_{26}, g_{30}, g_{31}, g_{32}, g_{33}, g_{34}, g_{35}, g_{36}, g_{40}, g_{41},\\ &g_{42}, g_{43}, g_{44}, g_{45}, g_{46}, g_{50}, g_{51}, g_{52}, g_{53}, g_{54},\\ &g_{55}, g_{56}, g_{60}, g_{61}, g_{62}, g_{63}, g_{64}, g_{65}, g_{66}\big)^T.\end{aligned} \tag{4.683}$$

So,

$$\begin{aligned}\mathbf{\Phi_{40}} = (&1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,\\ &0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0).\end{aligned} \tag{4.684}$$

Similarly, we can work out $\mathbf{\Phi_{41}}$:

$$\begin{aligned}\mathbf{\Phi_{41}} = (&0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,\\ &1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0).\end{aligned} \tag{4.685}$$

Now, let us consider that we do this for all values of k and all values of l, and write these row vectors one under the other to form matrix Φ: first all vectors for $k = 0$, then all vectors for $k = 1$, etc., finishing with all vectors for $k = 7$. We shall get the following matrix:

(Continued)

Example 4.179 (Continued)

$$\Phi = \begin{bmatrix}
1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0\\
0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0\\
1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&1&1&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

$$
\begin{bmatrix}
1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0\\
0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1
\end{bmatrix}.
$$

(Continued)

Example 4.179 (Continued)

We note that Φ may be partitioned into 49 matrices of size 7×7 each, as follows.

$$\Phi = \begin{bmatrix} I & I & I & I & I & I & I \\ I & A & B & C & D & E & F \\ I & B & D & F & A & C & E \\ I & C & F & B & E & A & D \\ I & D & A & E & B & F & C \\ I & E & C & A & F & D & B \\ I & F & E & D & C & B & A \\ I_1 & I_2 & I_3 & I_4 & I_5 & I_6 & I_7 \end{bmatrix} \tag{4.686}$$

where I is the unit 7×7 matrix, I_j is a matrix with all its elements 0 except those in row j which are all 1, and

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad F = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.687}$$

Example 4.180

You are given the following matrix A

$$A = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 \\ -- & -- & -- & -- \\ \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \delta_1 & \delta_2 & \delta_3 & \delta_4 \end{bmatrix} \tag{4.688}$$

where the vertical and horizontal lines indicate partitions of the matrix into four submatrices defined as:

$$A_1 = \begin{bmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{bmatrix} \quad A_2 = \begin{bmatrix} \alpha_3 & \alpha_4 \\ \beta_3 & \beta_4 \end{bmatrix} \quad A_3 = \begin{bmatrix} \gamma_1 & \gamma_2 \\ \delta_1 & \delta_2 \end{bmatrix} \quad A_4 = \begin{bmatrix} \gamma_3 & \gamma_4 \\ \delta_3 & \delta_4 \end{bmatrix}. \tag{4.689}$$

Show that

$$AA^T = \begin{bmatrix} A_1A_1^T + A_2A_2^T & A_1A_3^T + A_2A_4^T \\ A_3A_1^T + A_3A_2^T & A_3A_3^T + A_4A_4^T \end{bmatrix}. \tag{4.690}$$

First, we calculate AA^T by direct substitution:

$$\begin{aligned} AA^T &= \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 \\ \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \delta_1 & \delta_2 & \delta_3 & \delta_4 \end{bmatrix} \begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 & \delta_1 \\ \alpha_2 & \beta_2 & \gamma_2 & \delta_2 \\ \alpha_3 & \beta_3 & \gamma_3 & \delta_3 \\ \alpha_4 & \beta_4 & \gamma_4 & \delta_4 \end{bmatrix} \\ &= \begin{bmatrix} \alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 & \alpha_1\beta_1 + \alpha_2\beta_2 + \alpha_3\beta_3 + \alpha_4\beta_4 \\ \alpha_1\beta_1 + \alpha_2\beta_2 + \alpha_3\beta_3 + \alpha_4\beta_4 & \beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2 \\ \alpha_1\gamma_1 + \alpha_2\gamma_2 + \alpha_3\gamma_3 + \alpha_4\gamma_4 & \beta_1\gamma_1 + \beta_2\gamma_2 + \beta_3\gamma_3 + \beta_4\gamma_4 \\ \alpha_1\delta_1 + \alpha_2\delta_2 + \alpha_3\delta_3 + \alpha_4\delta_4 & \beta_1\delta_1 + \beta_2\delta_2 + \beta_3\delta_3 + \beta_4\delta_4 \end{bmatrix} \\ &\quad \begin{bmatrix} \alpha_1\gamma_1 + \alpha_2\gamma_2 + \alpha_3\gamma_3 + \alpha_4\gamma_4 & \alpha_1\delta_1 + \alpha_2\delta_2 + \alpha_3\delta_3 + \alpha_4\delta_4 \\ \beta_1\gamma_1 + \beta_2\gamma_2 + \beta_3\gamma_3 + \beta_4\gamma_4 & \beta_1\delta_1 + \beta_2\delta_2 + \beta_3\delta_3 + \beta_4\delta_4 \\ \gamma_1^2 + \gamma_2^2 + \gamma_3^2 + \gamma_4^2 & \gamma_1\delta_1 + \gamma_2\delta_2 + \gamma_3\delta_3 + \gamma_4\delta_4 \\ \gamma_1\delta_1 + \gamma_2\delta_2 + \gamma_3\delta_3 + \gamma_4\delta_4 & \delta_1^2 + \delta_2^2 + \delta_3^2 + \delta_4^2 \end{bmatrix}. \end{aligned} \tag{4.691}$$

Next, let us calculate product AA^T by considering the partitions of A and treating them as if they were the elements of the matrix:

$$\begin{aligned} AA^T &= \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} A_1^T & A_3^T \\ A_2^T & A_4^T \end{bmatrix} = \begin{bmatrix} A_1A_1^T + A_2A_2^T & A_1A_3^T + A_2A_4^T \\ A_3A_1^T + A_3A_2^T & A_3A_3^T + A_4A_4^T \end{bmatrix} \\ &= \begin{bmatrix} \begin{pmatrix} \alpha_1^2 + \alpha_2^2 & \alpha_1\beta_1 + \alpha_2\beta_2 \\ \alpha_1\beta_1 + \alpha_2\beta_2 & \beta_1^2 + \beta_2^2 \end{pmatrix} + \begin{pmatrix} \alpha_3^2 + \alpha_4^2 & \alpha_3\beta_3 + \alpha_4\beta_4 \\ \alpha_3\beta_3 + \alpha_4\beta_4 & \beta_3^2 + \beta_4^2 \end{pmatrix} \\ \begin{pmatrix} \alpha_1\gamma_1 + \alpha_2\gamma_2 & \beta_1\gamma_1 + \beta_2\gamma_2 \\ \alpha_1\delta_1 + \alpha_2\delta_2 & \beta_1\delta_1 + \beta_2\delta_2 \end{pmatrix} + \begin{pmatrix} \gamma_1\alpha_3 + \gamma_2\alpha_4 & \gamma_1\beta_3 + \gamma_2\beta_4 \\ \alpha_3\delta_1 + \alpha_4\delta_2 & \beta_3\delta_1 + \beta_4\delta_2 \end{pmatrix} \\ \begin{pmatrix} \alpha_1\gamma_1 + \alpha_2\gamma_2 & \alpha_1\delta_1 + \alpha_2\delta_2 \\ \beta_1\gamma_1 + \beta_2\gamma_2 & \beta_1\delta_1 + \beta_2\delta_2 \end{pmatrix} + \begin{pmatrix} \alpha_3\gamma_3 + \alpha_4\gamma_4 & \alpha_3\delta_3 + \alpha_4\delta_4 \\ \beta_3\gamma_3 + \beta_4\gamma_4 & \beta_3\delta_3 + \beta_4\delta_4 \end{pmatrix} \\ \begin{pmatrix} \gamma_1^2 + \gamma_2^2 & \gamma_1\delta_1 + \gamma_2\delta_2 \\ \delta_1\gamma_1 + \delta_2\gamma_2 & \delta_1^2 + \delta_2^2 \end{pmatrix} + \begin{pmatrix} \gamma_3^2 + \gamma_4^2 & \gamma_3\delta_3 + \gamma_4\delta_4 \\ \gamma_3\delta_3 + \gamma_4\delta_4 & \delta_3^2 + \delta_4^2 \end{pmatrix} \end{bmatrix}. \end{aligned} \tag{4.692}$$

By comparing the right-hand sides of (4.691) and (4.692) we confirm (4.690).

Example 4.181

By using the method demonstrated in Example 4.180, work out the product $\Phi^T\Phi$, where matrix Φ is given by (4.686).

(Continued)

Example 4.181 (Continued)

$$\Phi^T\Phi = \begin{bmatrix} I^T & I^T & I^T & I^T & I^T & I^T & I^T & I_1^T \\ I^T & A^T & B^T & C^T & D^T & E^T & F^T & I_2^T \\ I^T & B^T & D^T & F^T & A^T & C^T & E^T & I_3^T \\ I^T & C^T & F^T & B^T & E^T & A^T & D^T & I_4^T \\ I^T & D^T & A^T & E^T & B^T & F^T & C^T & I_5^T \\ I^T & E^T & C^T & A^T & F^T & D^T & B^T & I_6^T \\ I^T & F^T & E^T & D^T & C^T & B^T & A^T & I_7^T \end{bmatrix} \begin{bmatrix} I & I & I & I & I & I & I \\ I & A & B & C & D & E & F \\ I & B & D & F & A & C & E \\ I & C & F & B & E & A & D \\ I & D & A & E & B & F & C \\ I & E & C & A & F & D & B \\ I & F & E & D & C & B & A \\ I_1 & I_2 & I_3 & I_4 & I_5 & I_6 & I_7 \end{bmatrix}. \quad (4.693)$$

Obviously $I^T = I$. Also, from the definitions of matrices A–F given by Equations (4.687) we observe that:

$$D = C^T \qquad E = B^T \qquad F = A^T. \quad (4.694)$$

Then (4.693) takes the form:

$$\Phi^T\Phi = \begin{bmatrix} I & I & I & I & I & I & I & I_1^T \\ I & A^T & B^T & C^T & C & B & A & I_2^T \\ I & B^T & C & A & A^T & C^T & B & I_3^T \\ I & C^T & A & B^T & B & A^T & C & I_4^T \\ I & C & A^T & B & B^T & A & C^T & I_5^T \\ I & B & C^T & A^T & A & C & B^T & I_6^T \\ I & A & B & C & C^T & B^T & A^T & I_7^T \end{bmatrix} \begin{bmatrix} I & I & I & I & I & I & I \\ I & A & B & C & C^T & B^T & A^T \\ I & B & C^T & A^T & A & C & B^T \\ I & C & A^T & B & B^T & A & C^T \\ I & C^T & A & B^T & B & A^T & C \\ I & B^T & C & A & A^T & C^T & B \\ I & A^T & B^T & C^T & C & B & A \\ I_1 & I_2 & I_3 & I_4 & I_5 & I_6 & I_7 \end{bmatrix} = \quad (4.695)$$

$$\begin{bmatrix} 7I + I_1^T I_1 \\ I + A^T + B^T + C^T + C + B + A + I_2^T I_1 \\ I + B^T + C + A + A^T + C^T + B + I_3^T I_1 \\ I + C^T + A + B^T + B + A^T + C + I_4^T I_1 \\ I + C + A^T + B + B^T + A + C^T + I_5^T I_1 \\ I + B + C^T + A^T + A + C + B^T + I_6^T I_1 \\ I + A + B + C + C^T + B^T + A^T + I_7^T I_1 \end{bmatrix}$$

$$\begin{matrix} I + A + B + C + C^T + B^T + A^T + I_1^T I_2 \\ I + A^T A + B^T B + C^T C + CC^T + BB^T + AA^T + I_2^T I_2 \\ I + B^T A + CB + AC + A^T C^T + C^T B^T + BA^T + I_3^T I_2 \\ I + C^T A + AB + B^T C + BC^T + A^T B^T + CA^T + I_4^T I_2 \\ I + CA + A^T B + BC + B^T C^T + AB^T + C^T A^T + I_5^T I_2 \\ I + BA + C^T B + A^T C + AC^T + CB^T + B^T A^T + I_6^T I_2 \\ I + AA + BB + CC + C^T C^T + B^T B^T + A^T A^T + I_7^T I_2 \end{matrix}$$

$$\begin{matrix} I + B + C^T + A^T + A + C + B^T + I_1 I_3 \\ I + A^T B + B^T C^T + C^T A^T + CA + BC + AB^T + I_2^T I_3 \\ I + B^T B + CC^T + AA^T + A^T A + C^T C + BB^T + I_3^T I_3 \\ I + C^T B + AC^T + B^T A^T + BA + A^T C + CB^T + I_4^T I_3 \\ I + CB + A^T C^T + BA^T + B^T A + AC + C^T B^T + I_5^T I_3 \\ I + BB + C^T C^T + A^T A^T + AA + CC + B^T B^T + I_6^T I_3 \\ I + AB + BC^T + CA^T + C^T A + B^T C + A^T B^T + I_7^T I_3 \end{matrix} \quad (4.696)$$

$$
\begin{bmatrix}
I + C + A^T + B + B^T + A + C^T + I_1^T I_4 \\
I + A^T C + B^T A^T + C^T B + CB^T + BA + AC^T + I_2^T I_4 \\
I + B^T C + CA^T + AB + A^T B^T + C^T A + BC^T + I_3^T I_4 \\
I + C^T C + AA^T + B^T B + BB^T + A^T A + CC^T + I_4^T I_4 \\
I + CC + A^T A^T + BB + B^T B^T + AA + C^T C^T + I_5^T I_4 \\
I + BC + C^T A^T + A^T B + AB^T + CA + B^T C^T + I_6^T I_4 \\
I + AC + BA^T + CB + C^T B^T + B^T A + A^T C^T + I_7^T I_4 \\
\\
I + B^T + C + A + A^T + C^T + B + I_1^T I_5 \\
I + A^T C^T + B^T A + C^T B^T + CB + BA^T + AC + I_2^T I_5 \\
I + B^T C^T + CA + AB^T + A^T B + C^T A^T + BC + I_3^T I_5 \\
I + C^T C^T + AA + B^T B^T + BB + A^T A^T + CC + I_4^T I_5 \\
I + CC^T + A^T A + BB^T + B^T B + AA^T + C^T C + I_5^T I_5 \\
I + BC^T + C^T A + A^T B^T + AB + CA^T + B^T C + I_6^T I_5 \\
I + AC^T + BA + CB^T + C^T B + B^T A^T + A^T C + I_7^T I_5 \\
\\
I + B^T + C + A + A^T + C^T + B + I_1^T I_6 \\
I + A^T B^T + B^T C + C^T A + CA^T + BC^T + AB + I_2^T I_6 \\
I + B^T B^T + CC + AA + A^T A^T + C^T C^T + BB + I_3^T I_6 \\
I + C^T B^T + AC + B^T A + BA^T + A^T C^T + CB + I_4^T I_6 \\
I + CB^T + A^T C + BA + B^T A^T + AC^T + C^T B + I_5^T I_6 \\
I + CB^T + C^T C + A^T A + AA^T + CC^T + B^T B + I_6^T I_6 \\
I + AB^T + BC + CA + C^T A^T + B^T C^T + A^T B + I_7^T I_6 \\
\\
\left.\begin{array}{l}
I + A^T + B^T + C^T + C + B + A + I_1^T I_7 \\
I + A^T A^T + B^T B^T + C^T C^T + CC + BB + AA + I_2^T I_7 \\
I + B^T A^T + CB^T + AC^T + A^T C + C^T B + BA + I_3^T I_7 \\
I + C^T A^T + AB^T + B^T C^T + BC + A^T B + CA + I_4^T I_7 \\
I + CA^T + A^T B^T + BC^T + B^T C + AB + C^T A + I_5^T I_7 \\
I + BA^T + C^T B^T + A^T C^T + AC + CB + B^T A + I_6^T I_7 \\
I + AA^T + BB^T + CC^T + C^T C + B^T B + A^T A + I_7^T I_7
\end{array}\right]. \qquad (4.697)
$$

Example 4.182

From the definitions of matrices A, B and C, we can easily work out the following relationships:

$$
\begin{array}{lll}
AA = B & BA = C & CA = C^T \\
AB = C & BB = C^T & CB = B^T \\
AC = C^T & BC = B^T & CC = A^T \\
AC^T = B^T & BC^T = A^T & CC^T = I \\
AB^T = A^T & BB^T = I & CB^T = A \\
AA^T = I & BA^T = A & CA^T = B
\end{array}
$$

(Continued)

Example 4.182 (Continued)

$$\begin{array}{lll} A^TA = I & B^TA = A^T & C^TA = B^T \\ A^TB = A & B^TB = I & C^TB = A^T \\ A^TC = B & B^TC = A & C^TC = I \\ A^TC^T = C & B^TC^T = B & C^TC^T = A \\ A^TB^T = C^T & B^TB^T = C & C^TB^T = B \\ A^TA^T = B^T & B^TA^T = C^T & C^TA^T = C. \end{array} \tag{4.698}$$

We also observe that

$$I + A + B + C + A^T + B^T + C^T = W \tag{4.699}$$

where

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.700}$$

Finally, we observe that

$$I_1^TI_1 = I_2^TI_2 = I_3^TI_3 = I_4^TI_4 = I_5^TI_5 = I_6^TI_6 = I_7^TI_7 = W \tag{4.701}$$

and that

$$I_i^TI_j = \mathbf{0}_{7\times7} \text{ for } i \neq j \tag{4.702}$$

where $\mathbf{0}_{7\times7}$ is the zero matrix of size 7×7. Taking all these equations into consideration, we finally obtain

$$\Phi^T\Phi = W_{49\times49} + 7I_{49\times49} \tag{4.703}$$

where $W_{49\times49}$ is the flat matrix of size 49×49 and $I_{49\times49}$ is the unit matrix of the same size.

Example 4.183

Show that

$$\frac{1}{N}\Phi^TF_{\mathbf{RAD}} = \text{Image} \tag{4.704}$$

where Φ is as defined by (4.686) and $F_{\mathbf{RAD}}$ is defined by (4.672). Note that *Image* here means the input image G, of size $N \times N$, with its dc component removed, written as a column vector, as described in Example 4.179.

Matrix Φ has been constructed in Example 4.179, so that

$$\Phi\mathbf{g} = F_{\text{RAD}} \tag{4.705}$$

where $\mathbf{g}$ *is the vectorised image given by (4.683). We multiply both sides of this equation with* Φ^T*:*

$$\Phi^T\Phi\mathbf{g} = \Phi^T F_{\text{RAD}}. \tag{4.706}$$

Let us substitute $\Phi^T\Phi$ *from Equation (4.703), dropping the subscripts that indicate the size of each matrix, for the sake of simplicity:*

$$(W + 7I)\mathbf{g} = \Phi^T F_{\text{RAD}} \Leftrightarrow W\mathbf{g} + 7I\mathbf{g} = \Phi^T F_{\text{RAD}}. \tag{4.707}$$

Note that matrix W *is a flat matrix. Then, when one of its rows multiplies column vector* $\mathbf{g}$*, it simply sums up all the elements of* $\mathbf{g}$*, i.e. all pixel values of the input image, which we know sum up to 0. So, we obtain:*

$$7I\mathbf{g} = \Phi^T F_{\text{RAD}} \Leftrightarrow \mathbf{g} = \frac{1}{7}\Phi^T F_{\text{RAD}}. \tag{4.708}$$

Example B4.184

Work out the eigenvalues of matrix $\Phi^T\Phi$**.**

This is a circulant matrix (see Box 4.13) with the elements of a column produced from the elements of the previous column shifted down by one position and the element that sticks out at the bottom placed at the top. It can be shown (see Book I [75]) that the eigenvalues of such a matrix are given by Equation (4.713). Applying this formula for $M = 49$*,* $d(0) = 8$ *and all other elements equal to 1, we obtain:*

$$\lambda(k) = 8 + \sum_{m=0}^{48} e^{\frac{2\pi j}{49}mk}. \tag{4.709}$$

We may use the following formula (see Book I [75]) to work out the sum in (4.709):

$$\sum_{m=0}^{S-1} e^{\frac{2\pi j}{S}mt} = \begin{cases} S & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases}. \tag{4.710}$$

Applying this formula for $S = 49$*, we can easily work out that*

$$\lambda(0) = 56 \qquad \lambda(1) = \lambda(2) = \ldots = \lambda(48) = 7. \tag{4.711}$$

Box 4.13 Circulant matrices

An $M \times M$ matrix D is called **circulant** if it has the form:

$$D = \begin{pmatrix} d(0) & d(M-1) & d(M-2) & \ldots & d(1) \\ d(1) & d(0) & d(M-1) & \ldots & d(2) \\ d(2) & d(1) & d(0) & \ldots & d(3) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d(M-1) & d(M-2) & d(M-3) & \ldots & d(0) \end{pmatrix}. \tag{4.712}$$

(Continued)

Box 4.13 (Continued)

The eigenvalues of such a matrix are given by

$$\lambda(k) = d(0) + d(M-1)e^{\frac{2\pi j}{M}k} + d(M-2)e^{\frac{2\pi j}{M}2k} + \cdots + d(1)e^{\frac{2\pi j}{M}(M-1)k} \tag{4.713}$$

for $k = 0, 1, 2, \ldots, M-1$, and the corresponding eigenvectors are:

$$\mathbf{w}(k) = \frac{1}{\sqrt{M}}\left(1, e^{\frac{2\pi j}{M}k}, e^{\frac{2\pi j}{M}2k}, \ldots, e^{\frac{2\pi j}{M}(M-1)k}\right)^T. \tag{4.714}$$

Example B4.185

Let

$$\mathbf{a} \equiv F\mathbf{b} \tag{4.715}$$

where a is an M-dimensional vector, b is an N-dimensional vector and F is an $M \times N$ matrix. Show that the square magnitude (norm) of vector a is given by:

$$||\mathbf{a}||^2 \equiv \sum_{m=1}^{M} a_m^2 = \mathbf{b}^T F^T F \mathbf{b}. \tag{4.716}$$

Obviously:

$$\sum_{m=1}^{M} a_m^2 = \mathbf{a}^T\mathbf{a} = (F\mathbf{b})^T F\mathbf{b} = \mathbf{b}^T F^T F\mathbf{b}. \tag{4.717}$$

Example B4.186

If x is an eigenvector of matrix F with eigenvalue λ, show that:

$$\mathbf{x}^T F^T F\mathbf{x} = \lambda^2 \mathbf{x}^T\mathbf{x}. \tag{4.718}$$

By definition:

$$F\mathbf{x} = \lambda\mathbf{x} \Rightarrow \mathbf{x}^T F^T = \lambda\mathbf{x}^T. \tag{4.719}$$

Multiplying these two equations by parts, and remembering that λ is a scalar and so it may change position in a matrix/vector equation, we obtain:

$$\mathbf{x}^T F^T F\mathbf{x} = \mathbf{x}^T\lambda^2\mathbf{x}. \tag{4.720}$$

Example B4.187

If

$$\mathbf{x}^T F\mathbf{x} = \mathbf{x}^T\lambda\mathbf{x} \tag{4.721}$$

show that λ is an eigenvalue of matrix F.

Multiply both sides of Equation (4.721) from the left with $(\mathbf{x}^T)^{-1}$ to show that $F\mathbf{x} = \lambda\mathbf{x}$, i.e. that λ is an eigenvalue of F and $\mathbf{x}$ the corresponding eigenvector.

Box 4.14 Elements of frame theory

Consider a set of M vectors $\boldsymbol{\phi}_\mathbf{m}$, of size $N \times 1$. We say that these vectors constitute a **frame** if for any vector $\mathbf{x}$ of the same size, we may write

$$A||\mathbf{x}||^2 \leq \sum_{m=1}^{M} |\mathbf{x} \cdot \boldsymbol{\phi}_\mathbf{m}|^2 \leq B||\mathbf{x}||^2 \tag{4.722}$$

where A and B are some positive constants with B finite, known as **frame bounds**, and $||\mathbf{x}||$ is the magnitude of vector $\mathbf{x}$. If $A = B$ the frame is said to be **tight**.

Imagine now that we write vectors $\boldsymbol{\phi}_\mathbf{m}$ one under the other, as row vectors, to form a matrix F of size $M \times N$. Equation (4.722) may then be written as:

$$\mathbf{x}^T A\mathbf{x} \leq \mathbf{x}^T F^T F\mathbf{x} \leq \mathbf{x}^T B\mathbf{x}. \tag{4.723}$$

From Example 4.187, it is obvious that the eigenvalues of $F^T F$ are between A and B. So, the tightest possible frame bounds A and B we may have are the minimum and maximum eigenvalues of $F^T F$. Obviously, the tightest frame we may have then is when all eigenvalues of $F^T F$ are equal to $A = B$. This will happen when $F^T F = AI$, where I is the unit matrix. In turn, this means that matrix F is a unitary matrix, i.e. its transpose is its inverse within a scaling factor. If F is a real matrix, the term we use is that it is an orthogonal matrix in this case. An orthogonal matrix is made up from rows that are vectors orthogonal to each other. Such matrices are discussed extensively in Book I [75]. We may say, therefore, that the expansion of a vector in terms of a set of orthogonal basis vectors is a special case of the expansion of the vector in terms of a set of vectors that constitute a frame.

Example B4.188

Show that if a matrix has all its eigenvalues identical, it may be written as the unit matrix times a constant.

First of all, it is obvious that matrix AI where A is a scalar, has eigenvalues equal to A, since for any vector $\mathbf{x}$, we have $AI\mathbf{x} = A\mathbf{x}$.

We know from linear algebra that if P is the matrix made up from the eigenvectors of F, written next to each other as its columns, and Λ is a diagonal matrix made up from the eigenvalues written along the diagonal in the same column as the corresponding eigenvectors, we may write: $F = P^T \Lambda P$ (see Book I [75]). If all eigenvalues of F are equal to A, matrix Λ is the unit matrix times A. Since the unit matrix plays no role in matrix multiplication, and since a scalar may also change place in matrix multiplication, we may write: $F = AP^T P$. Now, P being made from columns that are orthogonal to each other and each one being an eigenvector of magnitude 1, implies that $P^T P = I$. This proves the assertion.

Example B4.189

Work out the tightest frame bounds for matrix $\Phi^T\Phi$.
The eigenvalues of this matrix have been worked out in Example 4.184 to be 7 and 56. So, the tightest frame bounds are $A = 7$ and $B = 56$. This means that the sum of the squares of the projections of any 49-dimensional vector on the columns of this matrix will be between 7 and 56 times the square magnitude of the vector.

Example B4.190

Calculate the eigenvector of matrix $\Phi^T\Phi$ that corresponds to eigenvalue $\lambda(0) = 56$.
This is given by (4.714) for $k = 0$. We observe that this leads to $\mathbf{w}(0) = (1, 1, 1, \dots, 1)^T$, which, if treated like a wrapped image, is a flat image.

Example B4.191

Show that if you omit the flat images, the $F_{\mathbf{RAD}}$ transform constitutes a tight frame.
The space spanned by the F_{RAD} transform is the space spanned by the eigenvectors of matrix $\Phi^T\Phi$. We note that this matrix has only one eigenvalue equal to 56 and it has 48 eigenvalues equal to 1. We also note that eigenvalue 56 corresponds to the sub-space of the flat image (see Example 4.190). If we omit that sub-space, the remaining space is spanned by eigenvectors, all of which have the same eigenvalue, equal to 1. So, the lower and upper limit of the sum of squares of the projections of any image on the columns of matrix Φ will be equal to each other and equal to the sum of the squares of the pixel values of the image, i.e. $A = B = 1$ in (4.723) and the frame will be tight.

This shows that the finite Radon transform is an orthogonal transform, easily invertible, as long as the image has zero mean, i.e. the image can be represented without the flat basis image.

Does the order in which we consider the tracing lines in the finite Radon transform matter?

For the Radon transform itself it does not matter. However, if the Radon transform is the first stage of some further processing, as is the case of the ridgelet transform, then the order in which we consider the lines matters.

What determines the order in which we consider the lines in the finite Radon transform?

This is largely determined by the way we parametrise the lines.

How may a line be parametrised?

The **normal** form of the equation of a line is $i\cos\phi + j\sin\phi - t = 0$, where $(\cos\phi, \sin\phi)$ is the normal vector to the line, having unit length. The **standard** form of the equation of a line is $ai + bj - t = 0$, where (a, b) is a vector normal to the line.

Example 4.192

Work out the relationship between parameters (k, l) of the line as defined by (4.673) and parameters (a, b, t), of the standard form of the equation of a line.

$$\begin{aligned} & ai + bj - t = 0 \\ \Rightarrow \quad & j = -\frac{a}{b}i + \frac{t}{b} \\ \Rightarrow \ & \frac{t}{b} \leftrightarrow l \quad \text{and} - \frac{a}{b} \leftrightarrow k. \end{aligned} \tag{4.724}$$

Example 4.193

The coordinates of the foot of the normal from the centre of the coordinate axes to a line are (a, b). Work out the equation of the line.

Figure 4.136 A line and the normal to the line from the centre of the axes.

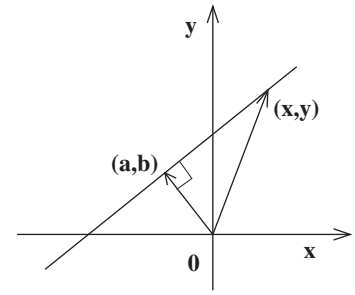


Figure 4.136 shows the line perpendicular to position vector $(a, b)^T$ at point (a, b). Any point (x, y) of the line has position vector $(x, y)^T$. Then the position vector of the same point with respect to the foot of the normal is $(x - a, y - b)$. Vectors $(a, b)^T$ and $(x - a, y - b)^T$ are orthogonal to each other, so their dot product is 0:

$$(x - a)a + (y - b)b = 0 \Rightarrow ax + by - (a^2 + b^2) = 0. \tag{4.725}$$

This is the equation of the line in its standard form.

Example 4.194

By looking at the bottom right panel of Figure 4.133 work out the directional vectors of each of the constructed lines and from them the normal vector of each line.
We note that the directional vectors are:

$$(1, 0), \quad (1, 1), \quad (1, 2), \quad (1, 3), \quad (1, 4), \quad (1, 5), \quad (1, 6), \quad (0, 1). \tag{4.726}$$

These vectors correspond to vectors $(x - a, y - b)$ of Example 4.193. The normal vector of a line should be such that its dot product with the directional vector of the line is 0 and it has unit length. So, from the above directional vectors we work out that the corresponding orthogonal vectors to

(Continued)

Example 4.194 (Continued)

these lines are:

$$(0,1),\quad (-1,1),\quad (-2,1),\quad (-3,1),\quad (-4,1),\quad (-5,1),\quad (-6,1),\quad (1,0). \tag{4.727}$$

These vectors may be normalised to have unit length, so they become the normal vectors of the lines (as opposed to being vectors normal to the lines):

$$(0,1),\quad \left(-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right),\quad \left(-\frac{2}{\sqrt{5}},\frac{1}{\sqrt{5}}\right),\quad \left(-\frac{3}{\sqrt{10}},\frac{1}{\sqrt{10}}\right),\quad \left(-\frac{4}{\sqrt{17}},\frac{1}{\sqrt{17}}\right),$$
$$\left(-\frac{5}{\sqrt{26}},\frac{1}{\sqrt{26}}\right),\quad \left(-\frac{6}{\sqrt{37}},\frac{1}{\sqrt{37}}\right),\quad (-1,0). \tag{4.728}$$

Example 4.195

Using Equation (4.725) work out vector (a,b) **for each line represented by the equation** $j = ki + l$ **for a** 7×7 **image.**

Equation (4.725) in (i,j) *coordinates may be written as*

$$\frac{a}{b}i + j - \frac{a^2+b^2}{b} = 0 \Rightarrow j = -\frac{a}{b}i + \frac{a^2+b^2}{b}. \tag{4.729}$$

We observe that

$$k = -\frac{a}{b} \qquad l = \frac{a^2+b^2}{b}. \tag{4.730}$$

Solving the first equation for a and substituting in the second, we obtain:

$$a = -kb \qquad l = \frac{k^2b^2+b^2}{b} \Rightarrow l = (k+1)b. \tag{4.731}$$

So,

$$a = -\frac{kl}{k+1} \qquad b = \frac{l}{k+1}. \tag{4.732}$$

Let us apply this for the line with $k = 4$ *and* $l = 1$*. We obtain* $a = -0.8$ *and* $b = 0.2$*. If we consider all values of k, we obtain:*

$$\begin{aligned}
&l=1 \quad k=0 \quad (0,1)\\
&l=1 \quad k=1 \quad \left(-\frac{1}{2},\frac{1}{2}\right)\\
&l=1 \quad k=2 \quad \left(-\frac{2}{3},\frac{1}{3}\right)\\
&l=1 \quad k=3 \quad \left(-\frac{3}{4},\frac{1}{4}\right)\\
&l=1 \quad k=4 \quad \left(-\frac{4}{5},\frac{1}{5}\right)\\
&l=1 \quad k=5 \quad \left(-\frac{5}{6},\frac{1}{6}\right)\\
&l=1 \quad k=6 \quad \left(-\frac{6}{7},\frac{1}{7}\right).
\end{aligned} \tag{4.733}$$

Note that if we normalise these vectors to have unit length, we recover the normal vectors we worked out in Example 4.194.

What is the best way to parametrise the Radon lines for use in the ridgelet transform?

A line used by the finite Radon transform corresponds to a line in the frequency domain of the image (see Examples 4.196–4.202). As most of the information in images is contained in the low frequency components, ideally we select the ordering of the lines so that they capture as best as possible the low frequency information.

Example 4.196

Using the standard form of the equation for a line, the finite Radon transform of an $N \times N$ image is defined as

$$F_{\text{RAD};a,b}(t) \equiv \frac{1}{\sqrt{N}} \sum_{(i,j)\in L_{a,b,t}} I(i,j) \tag{4.734}$$

where

$$L_{a,b,t} = \{(i,j) \in \text{Image} : \ ai + bj - t = 0 \text{ modulo } N\}. \tag{4.735}$$

Show that the 1D Fourier transform of $F_{\text{RAD};a,b}(t)$ with respect to t is identical to the 2D DFT $G(q_1, q_2)$ of $I(i,j)$, evaluated along a discrete slice through the origin and in direction (a,b). This is the *Discrete projection-slice theorem*.

The DFT of $F_{\text{RAD};a,b}(t)$ is

$$\begin{aligned}
\hat{F}_{\text{RAD};a,b}(\omega) &= \frac{1}{N}\sum_t F_{\text{RAD};a,b}(t) \mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}\omega t} \\
&= \frac{1}{N\sqrt{N}} \underbrace{\sum_t \sum_{(i,j)\in L_{a,b,t}}}_{\text{all points in the image}} I(i,j)\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}\omega t} \\
&= \frac{1}{N\sqrt{N}} \underbrace{\sum_{(i,j)\in \text{ Image}}}_{\text{all points in the image}} I(i,j)\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}\omega(ai+bj)}
\end{aligned} \tag{4.736}$$

where we made use of the equation of a line $ai + bj - t = 0 \Rightarrow t = ai + bj$.

The DFT of $I(i,j)$ is:

$$G(q_1, q_2) = \frac{1}{N^2} \sum_{(i,j)\in \text{ Image}} I(i,j)\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}(q_1 i + q_2 j)}. \tag{4.737}$$

By comparing with (4.736) we can see that $\hat{F}_{\text{RAD};a,b}(\omega) = \sqrt{N}G(a\omega, b\omega)$.

Example B4.197

For a 7×7 image, compute $F_{\text{RAD}}(4,l)$ for $l = 0, 1, \ldots, 6$ and then take the 1D Fourier transform of it.

This is the case of lines with slope $k = 4$, depicted in the central panel of Figure 4.133. Let us assume that the i index of the elements of the image changes from left to right, and the j index changes from bottom to top, both taking values from 0 to 6. Let us also call the elements of the image g_{ij}. Looking at the lines marked in Figure 4.133, we can easily work out the following values

(Continued)

Example B4.197 (Continued)

for $F_{\rm RAD}(4,l)$. Note that l takes values along the vertical axis, with 0 being at the bottom left.

$$\begin{aligned}
F_{\rm RAD}(4,0) &= \frac{1}{\sqrt{7}}(g_{00}+g_{14}+g_{21}+g_{35}+g_{42}+g_{56}+g_{63})\\
F_{\rm RAD}(4,1) &= \frac{1}{\sqrt{7}}(g_{50}+g_{01}+g_{22}+g_{43}+g_{64}+g_{15}+g_{36})\\
F_{\rm RAD}(4,2) &= \frac{1}{\sqrt{7}}(g_{30}+g_{51}+g_{02}+g_{23}+g_{44}+g_{65}+g_{16})\\
F_{\rm RAD}(4,3) &= \frac{1}{\sqrt{7}}(g_{10}+g_{31}+g_{52}+g_{03}+g_{24}+g_{45}+g_{66})\\
F_{\rm RAD}(4,4) &= \frac{1}{\sqrt{7}}(g_{60}+g_{11}+g_{53}+g_{32}+g_{04}+g_{25}+g_{46})\\
F_{\rm RAD}(4,5) &= \frac{1}{\sqrt{7}}(g_{40}+g_{61}+g_{12}+g_{33}+g_{54}+g_{05}+g_{26})\\
F_{\rm RAD}(4,6) &= \frac{1}{\sqrt{7}}(g_{20}+g_{41}+g_{62}+g_{13}+g_{34}+g_{55}+g_{06}).
\end{aligned} \tag{4.738}$$

The Fourier transform $\hat{F}_{\rm RAD}(4;q)$ of this string of numbers with respect to its second argument is:

$$\begin{aligned}
\hat{F}_{\rm RAD}(4;q) = \Big[&F_{\rm RAD}(4,0)+F_{\rm RAD}(4,1)e^{-\frac{2\pi j}{7}q}+F_{\rm RAD}(4,2)e^{-\frac{2\pi j}{7}2q}+F_{\rm RAD}(4,3)e^{-\frac{2\pi j}{7}3q}\\
&+F_{\rm RAD}(4,4)e^{-\frac{2\pi j}{7}4q}+F_{\rm RAD}(4,5)e^{-\frac{2\pi j}{7}5q}+F_{\rm RAD}(4,6)e^{-\frac{2\pi j}{7}6q}\Big]\frac{1}{7}.
\end{aligned} \tag{4.739}$$

Here q is the frequency index, taking values from 0 to 6.

We may then substitute from (4.738) into (4.739) to work out $\hat{F}_{\rm RAD}(4;q)$ as a function of the pixel values:

$$\begin{aligned}
\hat{F}_{\rm RAD}(4;q) = \frac{1}{7\sqrt{7}}\Big[&g_{00}+g_{14}+g_{21}+g_{35}+g_{42}+g_{56}+g_{63}\\
&+(g_{50}+g_{01}+g_{22}+g_{43}+g_{64}+g_{15}+g_{36})e^{-\frac{2\pi j}{7}q}\\
&+(g_{30}+g_{51}+g_{02}+g_{23}+g_{44}+g_{65}+g_{16})e^{-\frac{2\pi j}{7}2q}\\
&+(g_{10}+g_{31}+g_{52}+g_{03}+g_{24}+g_{45}+g_{66})e^{-\frac{2\pi j}{7}3q}\\
&+(g_{60}+g_{11}+g_{53}+g_{32}+g_{04}+g_{25}+g_{46})e^{-\frac{2\pi j}{7}4q}\\
&+(g_{40}+g_{61}+g_{12}+g_{33}+g_{54}+g_{05}+g_{26})e^{-\frac{2\pi j}{7}5q}\\
&+(g_{20}+g_{41}+g_{62}+g_{13}+g_{34}+g_{55}+g_{06})e^{-\frac{2\pi j}{7}6q}\Big].
\end{aligned} \tag{4.740}$$

Applying this for the different values of q, we obtain, for example:

$$\begin{aligned}
\hat{F}_{\rm RAD}(4;0) = \frac{1}{7\sqrt{7}}\Big[&g_{00}+g_{14}+g_{21}+g_{35}+g_{42}+g_{56}+g_{63}\\
&+g_{50}+g_{01}+g_{22}+g_{43}+g_{64}+g_{15}+g_{36}\\
&+g_{30}+g_{51}+g_{02}+g_{23}+g_{44}+g_{65}+g_{16}\\
&+g_{10}+g_{31}+g_{52}+g_{03}+g_{24}+g_{45}+g_{66}\\
&+g_{60}+g_{11}+g_{53}+g_{32}+g_{04}+g_{25}+g_{46}\\
&+g_{40}+g_{61}+g_{12}+g_{33}+g_{54}+g_{05}+g_{26}\\
&+g_{20}+g_{41}+g_{62}+g_{13}+g_{34}+g_{55}+g_{06}\Big]
\end{aligned} \tag{4.741}$$

$$\hat{F}_{\text{RAD}}(4;1) = \frac{1}{7\sqrt{7}}\left[g_{00}+g_{14}+g_{21}+g_{35}+g_{42}+g_{56}+g_{63}\right.$$
$$+(g_{50}+g_{01}+g_{22}+g_{43}+g_{64}+g_{15}+g_{36})e^{-\frac{2\pi j}{7}}$$
$$+(g_{30}+g_{51}+g_{02}+g_{23}+g_{44}+g_{65}+g_{16})e^{-\frac{2\pi j}{7}2}$$
$$+(g_{10}+g_{31}+g_{52}+g_{03}+g_{24}+g_{45}+g_{66})e^{-\frac{2\pi j}{7}3}$$
$$+(g_{60}+g_{11}+g_{53}+g_{32}+g_{04}+g_{25}+g_{46})e^{-\frac{2\pi j}{7}4}$$
$$+(g_{40}+g_{61}+g_{12}+g_{33}+g_{54}+g_{05}+g_{26})e^{-\frac{2\pi j}{7}5}$$
$$\left.+(g_{20}+g_{41}+g_{62}+g_{13}+g_{34}+g_{55}+g_{06})e^{-\frac{2\pi j}{7}6}\right]. \quad (4.742)$$

In the same way we may compute $\hat{F}_{\text{RAD}}(4;2),\ldots,\hat{F}_{\text{RAD}}(4;6)$. *We must remember that:*

$$e^{-\frac{2\pi j}{N}m} = e^{-\frac{2\pi j}{N}(m \text{ modulo } N)}. \quad (4.743)$$

This means that every time the number that multiplies $2\pi j/7$ *in these formulae becomes larger than or equal to 7, it may be replaced with the residual it leaves when it is divided by 7. So:*

$$e^{-\frac{2\pi j}{7}8} = e^{-\frac{2\pi j}{7}} \qquad e^{-\frac{2\pi j}{7}12} = e^{-\frac{2\pi j}{7}5} \qquad e^{-\frac{2\pi j}{7}36} = e^{-\frac{2\pi j}{7}}, \quad \textit{etc.} \quad (4.744)$$

To save space, we show in Figure 4.137 the factors that multiply $2\pi j/7$ *in the exponent of the exponential thata multiplies the corresponding pixel value of the image. We omit the case of* $\hat{F}_{\text{RAD}}(4;0)$, *where this factor is trivially 0 for all pixels.*

$\hat{\mathbf{F}}_{\mathbf{RAD}}(4,1)$

6	2	5	1	4	0	3
5	1	4	0	3	6	2
4	0	3	6	2	5	1
3	6	2	5	1	4	0
2	5	1	4	0	3	6
1	4	0	3	6	2	5
0	3	6	2	5	1	4

$\hat{\mathbf{F}}_{\mathbf{RAD}}(4,2)$

5	4	3	2	1	0	6
3	2	1	0	6	5	4
1	0	6	5	4	3	2
6	5	4	3	2	1	0
4	3	2	1	0	6	5
2	1	0	6	5	4	3
0	6	5	4	3	2	1

$\hat{\mathbf{F}}_{\mathbf{RAD}}(4,3)$

4	6	1	3	5	0	2
1	3	5	0	2	4	6
5	0	2	4	6	1	3
2	4	6	1	3	5	0
6	1	3	5	0	2	4
3	5	0	2	4	6	1
0	2	4	6	1	3	5

$\hat{\mathbf{F}}_{\mathbf{RAD}}(4,4)$

3	1	6	4	2	0	5
6	4	2	0	5	3	1
2	0	5	3	1	6	4
5	3	1	6	4	2	0
1	6	4	2	0	5	3
4	2	0	5	3	1	6
0	5	3	1	6	4	2

$\hat{\mathbf{F}}_{\mathbf{RAD}}(4,5)$

2	3	4	5	6	0	1
4	5	6	0	1	2	3
6	0	1	2	3	4	5
1	2	3	4	5	6	0
3	4	5	6	0	1	2
5	6	0	1	2	3	4
0	1	2	3	4	5	6

$\hat{\mathbf{F}}_{\mathbf{RAD}}(4,6)$

1	5	2	6	3	0	4
2	6	3	0	4	1	5
3	0	4	1	5	2	6
4	1	5	2	6	3	0
5	2	6	3	0	4	1
6	3	0	4	1	5	2
0	4	1	5	2	6	3

Figure 4.137 The numbers that have to multiply exponent $-2\pi j/7$ to produce the complex exponential with which the corresponding pixel of the image is multiplied to produce $\hat{F}_{\text{RAD}}(4;q)$ for $q = 1, 2, \ldots, 6$.

Example B4.198

For the image of Example 4.197, calculate explicitly the 49 components of its 2D Fourier transform.

(Continued)

Example B4.198 (Continued)

The 2D Fourier transform of the image is:

$$G(q_1, q_2) \equiv \frac{1}{49} \sum_{i=0}^{6} \sum_{j=0}^{6} g_{ij} e^{-\frac{2\pi j}{7}(q_1 i + q_2 j)}. \tag{4.745}$$

Indices (q_1, q_2) are the frequency domain indices and they take values from 0 to 6.

If we apply this for all combinations of values of (q_1, q_2), we shall produce 49 outputs. Instead of writing them explicitly, we decide to give only the numbers that have to multiply $-2\pi j/7$ to produce the complex exponential with which the corresponding pixel of the image is multiplied to produce $G(q_1, q_2)$. To produce these numbers we make use of (4.743) and (4.744).

We note that for $q_1 = q_2 = 0$,

$$G(0,0) = \frac{1}{\sqrt{7}} \hat{F}_{\text{RAD}}(4;0). \tag{4.746}$$

G(0,1)

6	6	6	6	6	6	6
5	5	5	5	5	5	5
4	4	4	4	4	4	4
3	3	3	3	3	3	3
2	2	2	2	2	2	2
1	1	1	1	1	1	1
0	0	0	0	0	0	0

G(0,2)

5	5	5	5	5	5	5
3	3	3	3	3	3	3
1	1	1	1	1	1	1
6	6	6	6	6	6	6
4	4	4	4	4	4	4
2	2	2	2	2	2	2
0	0	0	0	0	0	0

G(0,3)

4	4	4	4	4	4	4
1	1	1	1	1	1	1
5	5	5	5	5	5	5
2	2	2	2	2	2	2
6	6	6	6	6	6	6
3	3	3	3	3	3	3
0	0	0	0	0	0	0

G(0,4)

3	3	3	3	3	3	3
6	6	6	6	6	6	6
2	2	2	2	2	2	2
5	5	5	5	5	5	5
1	1	1	1	1	1	1
4	4	4	4	4	4	4
0	0	0	0	0	0	0

G(0,5)

2	2	2	2	2	2	2
4	4	4	4	4	4	4
6	6	6	6	6	6	6
1	1	1	1	1	1	1
3	3	3	3	3	3	3
5	5	5	5	5	5	5
0	0	0	0	0	0	0

G(0,6)

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
0	0	0	0	0	0	0

G(1,0)

0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6

G(1,1)

6	0	1	2	3	4	5
5	6	0	1	2	3	4
4	5	6	0	1	2	3
3	4	5	6	0	1	2
2	3	4	5	6	0	1
1	2	3	4	5	6	0
0	1	2	3	4	5	6

G(1,2)

5	6	0	1	2	3	4
3	4	5	6	0	1	2
1	2	3	4	5	6	0
6	0	1	2	3	4	5
4	5	6	0	1	2	3
2	3	4	5	6	0	1
0	1	2	3	4	5	6

G(1,3)

4	5	6	0	3	2	3
1	2	3	4	6	6	0
5	6	0	1	2	3	4
2	3	4	5	5	0	1
6	0	1	2	1	4	5
3	4	5	6	4	1	2
0	1	2	3	0	5	6

G(1,4)

3	4	5	6	0	1	2
6	0	1	2	3	4	5
2	3	4	5	6	0	1
5	6	0	1	2	3	4
1	2	3	4	5	6	0
4	5	6	0	1	2	3
0	1	2	3	4	5	6

G(1,5)

2	3	4	5	6	0	1
4	5	6	0	1	2	3
6	0	1	2	3	4	5
1	2	3	4	5	6	0
3	4	5	6	0	1	2
5	6	0	1	2	3	4
0	1	2	3	4	5	6

Figure 4.138 These numbers have to multiply $-2\pi j/7$ to form the exponent of the factor with which the corresponding pixel has to be multiplied to contribute to the creation of the indicated Fourier component of the image. The first 12 components of the Fourier transform of the image.

G(1,6)

1	2	3	4	5	6	0
2	3	4	5	6	0	1
3	4	5	6	0	1	2
4	5	6	0	1	2	3
5	6	0	1	2	3	4
6	0	1	2	3	4	5
0	1	2	3	4	5	6

G(2,0)

0	2	4	6	1	3	5
0	2	4	6	1	3	5
0	2	4	6	1	3	5
0	2	4	6	1	3	5
0	2	4	6	1	3	5
0	2	4	6	1	3	5
0	2	4	6	1	3	5

G(2,1)

6	1	3	5	0	2	4
5	0	2	4	6	1	3
4	6	1	3	5	0	2
3	5	0	2	4	6	1
2	4	6	1	3	5	0
1	3	5	0	2	4	6
0	2	4	6	1	3	5

G(2,2)

5	0	2	4	6	1	3
3	5	0	2	4	6	1
1	3	5	0	2	4	6
6	1	3	5	0	2	4
4	6	1	3	5	0	2
2	4	6	1	3	5	0
0	2	4	6	1	3	5

G(2,3)

4	6	1	3	5	0	2
1	3	5	0	2	4	6
5	0	2	4	6	1	3
2	4	6	1	3	5	0
6	1	3	5	0	2	4
3	5	0	2	4	6	1
0	2	4	6	1	3	5

G(2,4)

3	5	0	2	4	6	1
6	1	3	5	0	2	4
2	4	6	1	3	5	0
5	0	2	4	6	1	3
1	3	5	0	2	4	6
4	6	1	3	5	0	2
0	2	4	6	1	3	5

G(2,5)

2	4	6	1	3	5	0
4	6	1	3	5	0	2
6	1	3	5	0	2	4
1	3	5	0	2	4	6
3	5	0	2	4	6	1
5	0	2	4	6	1	3
0	2	4	6	1	3	5

G(2,6)

1	3	5	0	2	4	6
2	4	6	1	3	5	0
3	5	0	2	4	6	1
4	6	1	3	5	0	2
5	0	2	4	6	1	3
6	1	3	5	0	2	4
0	2	4	6	1	3	5

G(3,0)

0	3	6	2	5	1	4
0	3	6	2	5	1	4
0	3	6	2	5	1	4
0	3	6	2	5	1	4
0	3	6	2	5	1	4
0	3	6	2	5	1	4
0	3	6	2	5	1	4

G(3,1)

6	2	5	1	4	0	3
5	1	4	0	3	6	2
4	0	3	6	2	5	1
3	6	2	5	1	4	0
2	5	1	4	0	3	6
1	4	0	3	6	2	5
0	3	6	2	5	1	4

G(3,2)

5	1	4	0	3	6	2
3	6	2	5	1	4	0
1	4	0	3	6	2	5
6	2	5	1	4	0	3
4	0	3	6	2	5	1
2	5	1	4	0	3	6
0	3	6	2	5	1	4

G(3,3)

4	0	3	6	2	5	1
1	4	0	3	6	2	5
5	1	4	0	3	6	2
2	5	1	4	0	3	6
6	2	5	1	4	0	3
3	6	2	5	1	4	0
0	3	6	2	5	1	4

Figure 4.139 As Figure 4.138, but for components 13–24 of the Fourier transform of the image.

(Continued)

Example B4.198 (Continued)

G(3,4)

3	6	2	5	1	4	0
6	2	5	1	4	0	3
2	5	1	4	0	3	6
5	1	4	0	3	6	2
1	4	0	3	6	2	5
4	0	3	6	2	5	1
0	3	6	2	5	1	4

G(3,5)

2	5	1	4	0	3	6
4	0	3	6	2	5	1
6	2	5	1	4	0	3
1	4	0	3	6	2	5
3	6	2	5	1	4	0
5	1	4	0	3	6	2
0	3	6	2	5	1	4

G(3,6)

1	4	0	3	6	2	5
2	5	1	4	0	3	6
3	6	2	5	1	4	0
4	0	3	6	2	5	1
5	1	4	0	3	6	2
6	2	5	1	4	0	3
0	3	6	2	5	1	4

G(4,0)

0	4	1	5	2	6	3
0	4	1	5	2	6	3
0	4	1	5	2	6	3
0	4	1	5	2	6	3
0	4	1	5	2	6	3
0	4	1	5	2	6	3
0	4	1	5	2	6	3

G(4,1)

6	3	0	4	1	5	2
5	2	6	3	0	4	1
4	1	5	2	6	3	0
3	0	4	1	5	2	6
2	6	3	0	4	1	5
1	5	2	6	3	0	4
0	4	1	5	2	6	3

G(4,2)

5	2	6	3	0	4	1
3	0	4	1	5	2	6
1	5	2	6	3	0	4
6	3	0	4	1	5	2
4	1	5	2	6	3	0
2	6	3	0	4	1	5
0	4	1	5	2	6	3

G(4,3)

4	1	5	2	6	3	0
1	5	2	6	3	0	4
5	2	6	3	0	4	1
2	6	3	0	4	1	5
6	3	0	4	1	5	2
3	0	4	1	5	2	6
0	4	1	5	2	6	3

G(4,4)

3	0	4	1	5	2	6
6	3	0	4	1	5	2
2	6	3	0	4	1	5
5	2	6	3	0	4	1
1	5	2	6	3	0	4
4	1	5	2	6	3	0
0	4	1	5	2	6	3

G(4,5)

2	6	3	0	4	1	5
4	1	5	2	6	3	0
6	3	0	4	1	5	2
1	5	2	6	3	0	4
3	0	4	1	5	2	6
5	2	6	3	0	4	1
0	4	1	5	2	6	3

G(4,6)

1	5	2	6	3	0	4
2	6	3	0	4	1	5
3	0	4	1	5	2	6
4	1	5	2	6	3	0
5	2	6	3	0	4	1
6	3	0	4	1	5	2
0	4	1	5	2	6	3

G(5,0)

0	5	3	1	6	4	2
0	5	3	1	6	4	2
0	5	3	1	6	4	2
0	5	3	1	6	4	2
0	5	3	1	6	4	2
0	5	3	1	6	4	2
0	5	3	1	6	4	2

G(5,1)

6	4	2	0	5	3	1
5	3	1	6	4	2	0
4	2	0	5	3	1	6
3	1	6	4	2	0	5
2	0	5	3	1	6	4
1	6	4	2	0	5	3
0	5	3	1	6	4	2

Figure 4.140 As Figure 4.138, but for components 25–36 of the Fourier transform of the image.

G(5,2)

5	3	1	6	4	2	0
3	1	6	4	2	0	5
1	6	4	2	0	5	3
6	4	2	0	5	3	1
4	2	0	5	3	1	6
2	0	5	3	1	6	4
0	5	3	1	6	4	2

G(5,3)

4	2	0	5	3	1	6
1	6	4	2	0	5	3
5	3	1	6	4	2	0
2	0	5	3	1	6	4
6	4	2	0	5	3	1
3	1	6	4	2	0	5
0	5	3	1	6	4	2

G(5,4)

3	1	6	4	2	0	5
6	4	2	0	5	3	1
2	0	5	3	1	6	4
5	3	1	6	4	2	0
1	6	4	2	0	5	3
4	2	0	5	3	1	6
0	5	3	1	6	4	2

G(5,5)

2	0	5	3	1	6	4
4	2	0	5	3	1	6
6	4	2	0	5	3	1
1	6	4	2	0	5	3
3	1	6	4	2	0	5
5	3	1	6	4	2	0
0	5	3	1	6	4	2

G(5,6)

1	6	4	2	0	5	3
2	0	5	3	1	6	4
3	1	6	4	2	0	5
4	2	0	5	3	1	6
5	3	1	6	4	2	0
6	4	2	0	5	3	1
0	5	3	1	6	4	2

G(6,0)

0	6	5	4	3	2	1
0	6	5	4	3	2	1
0	6	5	4	3	2	1
0	6	5	4	3	2	1
0	6	5	4	3	2	1
0	6	5	4	3	2	1
0	6	5	4	3	2	1

G(6,1)

6	5	4	3	2	1	0
5	4	3	2	1	0	6
4	3	2	1	0	6	5
3	2	1	0	6	5	4
2	1	0	6	5	4	3
1	0	6	5	4	3	2
0	6	5	4	3	2	1

G(6,2)

5	4	3	2	1	0	6
3	2	1	0	6	5	4
1	0	6	5	4	3	2
6	5	4	3	2	1	0
4	3	2	1	0	6	5
2	1	0	6	5	4	3
0	6	5	4	3	2	1

G(6,3)

4	3	2	1	0	6	5
1	0	6	5	4	3	2
5	4	3	2	1	0	6
2	1	0	6	5	4	3
6	5	4	3	2	1	0
3	2	1	0	6	5	4
0	6	5	4	3	2	1

G(6,4)

3	2	1	0	6	5	4
6	5	4	3	2	1	0
2	1	0	6	5	4	3
5	4	3	2	1	0	6
1	0	6	5	4	3	2
4	3	2	1	0	6	5
0	6	5	4	3	2	1

G(6,5)

2	1	0	6	5	4	3
4	3	2	1	0	6	5
6	5	4	3	2	1	0
1	0	6	5	4	3	2
3	2	1	0	6	5	4
5	4	3	2	1	0	6
0	6	5	4	3	2	1

G(6,6)

1	0	6	5	4	3	2
2	1	0	6	5	4	3
3	2	1	0	6	5	4
4	3	2	1	0	6	5
5	4	3	2	1	0	6
6	5	4	3	2	1	0
0	6	5	4	3	2	1

Figure 4.141 As Figure 4.138, but for components 37–49 of the Fourier transform of the image.

Example B4.199

Identify the line that the Fourier transform of $F_{\text{RAD}}(4, l)$ for $l = 0, 1, \ldots, 6$ of Example 4.197 defines in the Fourier transform of the full image worked out in Example 4.198.

Using the results of Examples 4.197 and 4.198, we can see that:

$$\hat{F}_{\text{RAD}}(4; 0) = \sqrt{7}G(0, 0)$$
$$\hat{F}_{\text{RAD}}(4; 1) = \sqrt{7}G(3, 1)$$
$$\hat{F}_{\text{RAD}}(4; 2) = \sqrt{7}G(6, 2)$$
$$\hat{F}_{\text{RAD}}(4; 3) = \sqrt{7}G(2, 3)$$

(Continued)

Example B4.199 (Continued)

$$\hat{F}_{\text{RAD}}(4;4) = \sqrt{7}G(5,4)$$
$$\hat{F}_{\text{RAD}}(4;5) = \sqrt{7}G(1,5)$$
$$\hat{F}_{\text{RAD}}(4;6) = \sqrt{7}G(4,6). \tag{4.747}$$

We can see that, if we ignore factor $\sqrt{7}$, the FT of the Radon lines with slope $k = 4$, and all possible intercepts l along the vertical axis, cut a slice through the 2D Fourier transform of the image, identified by the black cells in Figure 4.142.

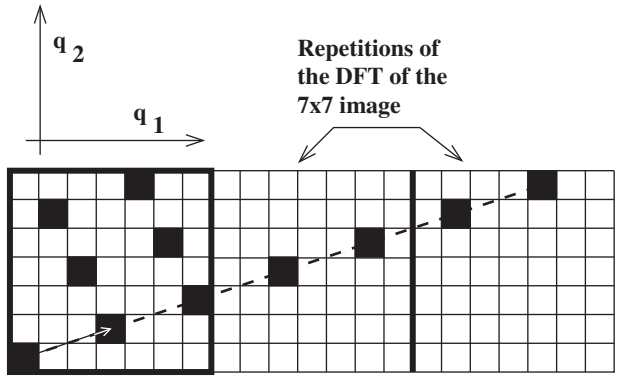


Figure 4.142 The thick black frame identifies the 2D DFT of the image. The dc component is the bottom left cell. The black pixels within this frame identify the coordinates of $G(q_1, q_2)$ listed in (4.747). As this frame is repeated ad infinitum in all directions, we may say that these pixels form the line indicated by the dashed line. The arrow shows the position of the first point of this digital line.

Example B4.200

Repeat the process of Example 4.197 to work out the Fourier transforms of the lines with slope $k = 0, 1, 2, 3, 5, 6, 7$ and in each case identify the pixels in the 2D Fourier space that make up the Fourier transform of each batch of lines with the same slope k.

For each value of k we work out $F_{\text{RAD}}(k;l)$ and then take its Fourier transform with respect to l. We may represent these Fourier transform like matrices indicating the factor in the exponent for each pixel, like we did in Figure 4.137. By comparing then these panels with the panels in Figures 4.138–4.141, we can identify which $G(q_1, q_2)$ corresponds to which $F_{\text{RAD}}(k;l)$. The result is shown in Figure 4.143, where the pixels that make up each line are marked with the same number, namely the slope k of the line.

Figure 4.143 The numbers below the grid are the values of frequency index q_1. The numbers on the left are the values of frequency index q_2. The bottom left pixel belongs to all slices (digital lines) that have been formed in the 2D Fourier domain. Each such line is identified with the value of k, i.e. the value of the slope of the batch of Radon lines from which it was created. The first point of each line is identified with a black frame.

6	0	1	2	3	4	5	6
5	0	4	1	5	2	6	3
4	0	5	3	1	6	4	2
3	0	2	4	6	1	3	5
2	0	3	6	2	5	1	4
1	0	6	5	4	3	2	1
0	■	7	7	7	7	7	7
	0	1	2	3	4	5	6

Example B4.201

The Fourier slices (lines) in the 2D Fourier transform of a 7×7 image are shown in Figure 4.143, where the dc component of the image is at the bottom left. Show how the same lines will appear if we use the Fourier space with the dc component in the middle of the grid.

The Fourier transform of the image is repeated ad infinitum in all directions. To construct the Fourier grid with the dc component in the middle, we repeat the existing grid four times and then we use a frame of size 7×7, centred on the dc component to extract the Fourier space we desire. This is shown in Figure 4.144

0	1	2	3	4	5	6	0	1	2	3	4	5	6
0	4	1	5	2	6	3	0	4	1	5	2	6	3
0	5	3	1	6	4	2	0	5	3	1	6	4	2
0	2	4	6	1	3	5	0	2	4	6	1	3	5
0	3	6	2	5	1	4	0	3	6	2	5	1	4
0	6	5	4	3	2	1	0	6	5	4	3	2	1
■	7	7	7	7	7	7	■	7	7	7	7	7	7
0	1	2	3	4	5	6	0	1	2	3	4	5	6
0	4	1	5	2	6	3	0	4	1	5	2	6	3
0	5	3	1	6	4	2	0	5	3	1	6	4	2
0	2	4	6	1	3	5	0	2	4	6	1	3	5
0	3	6	2	5	1	4	0	3	6	2	5	1	4
0	6	5	4	3	2	1	0	6	5	4	3	2	1
■	7	7	7	7	7	7	■	7	7	7	7	7	7

⇨

1	3	5	0	2	4	6
5	1	4	0	3	6	2
3	2	1	0	6	5	4
7	7	7	■	7	7	7
4	5	6	0	1	2	3
2	6	3	0	4	1	5
6	4	2	0	5	3	1

Figure 4.144 The original Fourier frame of Figure 4.143 is repeated in all directions. We identify the Fourier domain with the dc component in the middle by cutting out a frame of size 7×7 centred at the dc point.

Example B4.202

Working like in Example 4.173 and using the (k,l) parametrisation of the discrete lines, prove the projection slice theorem in the discrete domain. Verify it using the results of Example 4.199.

Let us take the DFT of $F_{\rm RAD}$ as defined by equations (4.672) and (4.673):

$$\begin{aligned}\hat{F}_{\rm RAD}(k;q) &= \frac{1}{N}\sum_{l=0}^{N-1} F_{\rm RAD}(k,l)\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}ql}\\ &= \frac{1}{N\sqrt{N}}\sum_{l=0}^{N-1}\sum_{(i,j)\in L_{kl}} I(i,j)\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}ql}\\ &= \frac{1}{N\sqrt{N}}\sum_{l=0}^{N-1}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\delta(j-(ki+l)_{\text{modulo }N})I(i,j)\\ &= \frac{1}{N\sqrt{N}}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(\sum_{l=0}^{N-1}\delta(j-(ki+l)_{\text{modulo }N})\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}ql}\right)I(i,j). \end{aligned} \tag{4.748}$$

To perform the summation inside the large parenthesis, we must solve

$$j-(ki+l)_{modulo\ N}=0 \tag{4.749}$$

for l and substitute in the exponent of the exponential factor. We note that this equation may be written as

$$ki+l=nN+j \Rightarrow l=nN+j-ki \tag{4.750}$$

where n is some integer number. If we substitute from (4.750) into (4.748) we obtain:

$$\begin{aligned}\hat{F}_{\rm RAD}(k;q) &= \frac{1}{N\sqrt{N}}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} I(i,j)\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}q(nN+j-ki)}\\ &= \frac{1}{N\sqrt{N}}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} I(i,j)\underbrace{\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}qnN}}_{=1}\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}q(j-ki)}\\ &= \sqrt{N}G(-kq,q). \end{aligned} \tag{4.751}$$

Applying this for $N=7$ and $k=4$, we obtain: $\hat{F}_{\rm RAD}(4;q)=\sqrt{7}G(-4q,q)$. For $q=1$, we obtain that $\hat{F}_{\rm RAD}(4;1)\leftrightarrow G(-4,1)$. Due to the periodic repetition of the image and its DFT, we may add 7 to any of the arguments of G without making a mistake. So, $G(-4,1)=G(3,1)$, which is confirmed by the empirical result (4.747).

Let us consider next $F_{\rm RAD}$ as defined by Equations (4.672) and (4.674) and let us take its DFT:

$$\begin{aligned}\hat{F}_{\rm RAD}(N;q) &= \frac{1}{N\sqrt{N}}\sum_{l=0}^{N-1}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\delta(i-l)\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}ql}I(i,j)\\ &= \frac{1}{N\sqrt{N}}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(\sum_{l=0}^{N-1}\delta(i-l)\mathrm{e}^{-\frac{2\pi \mathrm{j}}{N}ql}\right)I(i,j)\\ &= \sqrt{N}G(q,0). \end{aligned} \tag{4.752}$$

So, the projection slice theorem is valid in all cases.

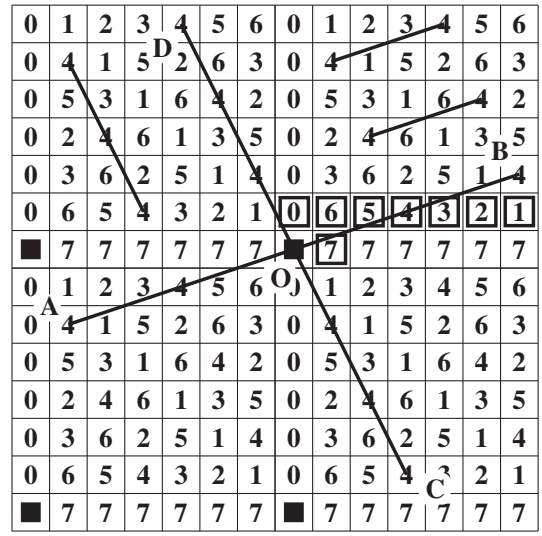


Figure 4.145 The two slices one may consider as being cut in the 2D Fourier domain by the 1D DFT of the batch of parallel lines that have slope $k = 4$. The numbers identify the slices cut by other values of slope k. The dark square marks the zero frequency point. The isolated line segments represent wrapped back parts of the corresponding lines.

How can we select the tracing lines so low frequency information is captured best?

The order of the lines is also dictated by the order of the pixels that make up the line. Let us consider for example the lines with $k = 5$ in Figure 4.133 and in particular the first line, formed by the black squares. We constructed it as a line with direction from bottom left to top right (as the arrow that connects its first two pixels shows). Nothing stops us, however, from considering it as a line that goes from top left to bottom right. In fact, this way seems more natural as the pixels that make it up are closer to each other. The order of the pixels makes no difference for the Radon transform, as all we do is to sum their values. If, however, we consider the line to be from top left to bottom right, the order of the lines of the same slope changes. When we align the pixels according to their shorter distance, we also capture better low frequency information, since pixels closer to each other have more chance to be similar. Indeed, one way of selecting the best order of the lines is to look at the frequency domain as each batch of lines creates a line in the frequency domain passing through the $(0, 0)$ frequency point. Let us consider the right panel of Figure 4.144 where each line created by a batch of parallel lines is marked with the value of the corresponding k parameter (the slope parameter of the lines). Figure 4.145 shows the two lines that may be constructed in the Fourier space from the samples that make up the Fourier transform of the batch of lines with slope $k = 4$. The black rectangle indicates the first frequency point that has to be joined with the dc component to form the slice through the Fourier space when the 1D DFT is taken according to displacement parameter l. This is line AB. Line CD is formed if we connect the dc component with the sample that represents the smallest frequency from among all samples marked with 4 (the slope of the batch of lines we are currently considering). Each of the two lines is made up from a sequence of frequencies. Line CD captures lower frequencies than line AB.

As shown in Figure 4.146, we can see that if we consider as lines those formed by connecting the central sample (the dc sample) with the nearest sample of each line, the lines we shall create in the frequency domain will have nicely spread round orientations and they will be capturing low frequency information better, as they will be made up from samples closer to each other (see Examples 4.203 and 4.204).

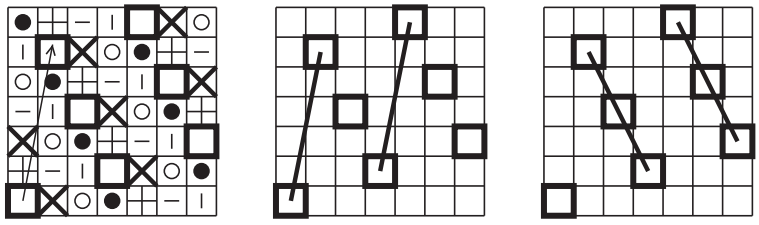


Figure 4.146 (a) All lines with the same slope ($k = 5$), as in Example 4.203, for a 7×7 image. (b) If we follow the original method of constructing the lines using parameters (k, l), the orientation of the line is marked here by the thick lines that join two successive pixels of it. (c) If we join the pixels according to their proximity, the line we form has the direction shown here by the thick black lines.

Example 4.203

From Figure 4.144 work out the normal vectors of the lines and compare them with the normal vectors of the same lines worked out in Example 4.194. From all vectors (a, b) normal to a line, we select the one that is closest to the dc component of the image and, to avoid ambiguity, we select the one that has $b > 0$.

The selected (a, b) vectors of the lines are those with the minimum value of $\sqrt{a^2 + b^2}$, since the dc component coincides with the origin of the axes. Using also the convention $b > 0$, we conclude that the list of vectors (a, b), as shown in Figure 4.147, and their corresponding normalised versions that are the normal vectors to the line, are:

$$\begin{aligned}
&k = 0 \quad (0,1) \quad (0,1)\\
&k = 1 \quad (-1,1) \quad \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)\\
&k = 2 \quad (-2,1) \quad \left(-\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}}\right)\\
&k = 3 \quad (1,2) \quad \left(\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}\right)\\
&k = 4 \quad (-1,2) \quad \left(-\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}\right)\\
&k = 5 \quad (2,1) \quad \left(\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}}\right)\\
&k = 6 \quad (1,1) \quad \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)\\
&k = 7 \quad (1,0) \quad (1,0).
\end{aligned} \tag{4.753}$$

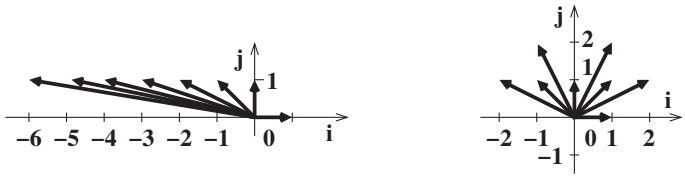


Figure 4.147 The (a, b) vectors of the digital Radon lines as originally constructed on the left (see (4.727)) and as defined from the Fourier domain, on the right.

Example 4.204

Work out the order of the lines for the case $k = 4$ in Figure 4.133 when the parametrisation of the lines is in terms of (a, b, t) instead of (k, l), with vector (a, b) defined from the Fourier domain as in Example 4.203, and the standard form of the equation of the line is used.

The standard form of the equation of the line is

$$ai + bj - t = 0 \tag{4.754}$$

where t is the displacement parameter, taking values $0, 1, \ldots, 6$. *For slope* $k = 4$, *vector* (a, b) *has been worked out to be* $(-1, 2)$ *in Example 4.203. So, we may write:*

$$-i + 2j = t \Rightarrow j = \frac{t + i}{2} \qquad \text{for} t, i = 0, 1, \ldots, 6. \tag{4.755}$$

The order of the lines will be determined by the values of t. So, for every value of i, we must work out the corresponding value of j in order to see which pixels make up each line. We must remember that each line has to be made from 7 pixels, the value of j has to be integer, and so the values we select for i have to lead to integer values of j, and that all coordinates are modulo 7.

For $t = 0$
For $i = 0 \Rightarrow j = 0$. *Pixel* $(0, 0)$
For $i = 2 \Rightarrow j = 1$. *Pixel* $(2, 1)$
For $i = 4 \Rightarrow j = 2$. *Pixel* $(4, 2)$
For $i = 6 \Rightarrow j = 3$. *Pixel* $(6, 3)$
For $i = 8 \Rightarrow j = 4$. *Pixel* $(1, 4)$
For $i = 10 \Rightarrow j = 5$. *Pixel* $(3, 5)$
For $i = 12 \Rightarrow j = 6$. *Pixel* $(5, 6)$.

For $t = 1$
For $i = 1 \Rightarrow j = 1$. *Pixel* $(1, 1)$
For $i = 3 \Rightarrow j = 2$. *Pixel* $(3, 2)$
For $i = 5 \Rightarrow j = 3$. *Pixel* $(5, 3)$
For $i = 7 \Rightarrow j = 4$. *Pixel* $(0, 4)$
For $i = 9 \Rightarrow j = 5$. *Pixel* $(2, 5)$
For $i = 11 \Rightarrow j = 6$. *Pixel* $(4, 6)$
For $i = 13 \Rightarrow j = 7$. *Pixel* $(6, 0)$.

(Continued)

Example 4.204 (Continued)

For $t = 2$
For $i = 0 \Rightarrow j = 1$. *Pixel* $(0, 1)$
For $i = 2 \Rightarrow j = 2$. *Pixel* $(2, 2)$
For $i = 4 \Rightarrow j = 3$. *Pixel* $(4, 3)$
For $i = 6 \Rightarrow j = 4$. *Pixel* $(6, 4)$
For $i = 8 \Rightarrow j = 5$. *Pixel* $(1, 5)$
For $i = 10 \Rightarrow j = 6$. *Pixel* $(3, 6)$
For $i = 12 \Rightarrow j = 7$. *Pixel* $(5, 0)$.

For $t = 3$
For $i = 1 \Rightarrow j = 2$. *Pixel* $(1, 2)$
For $i = 3 \Rightarrow j = 3$. *Pixel* $(3, 3)$
For $i = 5 \Rightarrow j = 4$. *Pixel* $(5, 4)$
For $i = 7 \Rightarrow j = 5$. *Pixel* $(0, 5)$
For $i = 9 \Rightarrow j = 6$. *Pixel* $(2, 6)$
For $i = 11 \Rightarrow j = 7$. *Pixel* $(4, 0)$
For $i = 13 \Rightarrow j = 8$. *Pixel* $(6, 1)$

For $t = 4$
For $i = 0 \Rightarrow j = 2$. *Pixel* $(0, 2)$
For $i = 2 \Rightarrow j = 3$. *Pixel* $(2, 3)$
For $i = 4 \Rightarrow j = 4$. *Pixel* $(4, 4)$
For $i = 6 \Rightarrow j = 5$. *Pixel* $(6, 5)$
For $i = 8 \Rightarrow j = 6$. *Pixel* $(1, 6)$
For $i = 10 \Rightarrow j = 7$. *Pixel* $(3, 0)$
For $i = 12 \Rightarrow j = 8$. *Pixel* $(5, 1)$.

For $t = 5$
For $i = 1 \Rightarrow j = 3$. *Pixel* $(1, 3)$
For $i = 3 \Rightarrow j = 4$. *Pixel* $(3, 4)$
For $i = 5 \Rightarrow j = 5$. *Pixel* $(5, 5)$
For $i = 7 \Rightarrow j = 6$. *Pixel* $(0, 6)$
For $i = 9 \Rightarrow j = 7$. *Pixel* $(2, 0)$
For $i = 11 \Rightarrow j = 8$. *Pixel* $(4, 1)$
For $i = 13 \Rightarrow j = 9$. *Pixel* $(6, 2)$.

For $t = 6$
For $i = 0 \Rightarrow j = 3$. *Pixel* $(0, 3)$
For $i = 2 \Rightarrow j = 4$. *Pixel* $(2, 4)$
For $i = 4 \Rightarrow j = 5$. *Pixel* $(4, 5)$
For $i = 6 \Rightarrow j = 6$. *Pixel* $(6, 6)$
For $i = 8 \Rightarrow j = 7$. *Pixel* $(1, 0)$
For $i = 10 \Rightarrow j = 8$. *Pixel* $(3, 1)$
For $i = 12 \Rightarrow j = 9$. *Pixel* $(5, 2)$.

Figure 4.148 shows the lines as marked in the central panel of Figure 4.133 and the sequence of lines created with parametrisation (k, l) *and* (a, b, t)*, with each line identified by a different symbol.*

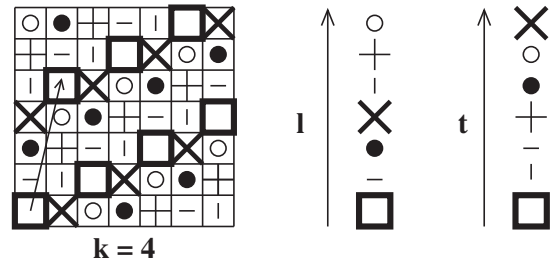


Figure 4.148 The lines with slope $k = 4$, all marked with different symbols. Sorting them according to parameter l and sorting them according to parameter t creates two different line sequences.

How do we use the finite Radon transform to calculate the finite ridgelet transform?

The **finite ridgelet transform**, F_{RID}, is obtained from the finite Radon transform F_{RAD}, by taking the 1D discrete wavelet transform of each column of F_{RAD}, characterised by a particular line orientation. So the wavelet transform is with respect to displacement parameter t, which identifies the different versions of the line with a particular slope scanning the image. As there are as many values of t as pixels in an image row, the total number of different samples with respect to which we take the wavelet transform is a prime number.

How do we create a wavelet basis for a signal that is not a power of 2 in length?

In general, if N is a prime number and we wish to take the wavelet transform of a N-sample long signal, we can do that as follows. First, we find the largest power of 2, say 2^m which is lower than N. We set $n \equiv N - 2^m$. Then we use the wavelet filters for a 2^m-sample long signal and augment them by adding n 0s at the end. This way we shall have $2^n - 1$ basis vectors for the N-sample long signal. We add to these vectors the following ones:

$$\begin{aligned}
\mathbf{e_o} &= (\underbrace{1,1,\ldots,1}_{N\ 1s})/s_0\\
\mathbf{e_1} &= (\underbrace{1,1,\ldots,1}_{N-1\ 1s},-N+1)/s_1\\
\mathbf{e_2} &= (\underbrace{1,1,\ldots,1}_{N-2\ 1s},-N+2,0)/s_2\\
&\ldots\\
\mathbf{e_n} &= (\underbrace{1,1,\ldots,1}_{N-n\ 1s},-N+n,\underbrace{0,0,\ldots,0}_{n\ 0s})/s_n
\end{aligned} \tag{4.756}$$

where $s_0, s_1, \ldots, s_n$ are normalising constants to ensure that the magnitude of each vector is 1.

Example 4.205

Show how you may take the 1D wavelet transform of an 11-sample long signal, using the Haar wavelets.

(Continued)

Example 4.205 (Continued)

For an 11-sample long signal, the nearest power of 2 that is lower than 11 is $2^3 = 8$. For an 8-sample long signal, the basis vectors are:

$$\tilde{\mathbf{e}}_{\mathbf{o}} = \left(\frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}\right)$$

$$\tilde{\mathbf{w}}_{\mathbf{1}} = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0, 0\right)$$

$$\tilde{\mathbf{w}}_{\mathbf{2}} = \left(0, 0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0\right)$$

$$\tilde{\mathbf{w}}_{\mathbf{3}} = \left(0, 0, 0, 0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0\right)$$

$$\tilde{\mathbf{w}}_{\mathbf{4}} = \left(0, 0, 0, 0, 0, 0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$$

$$\tilde{\mathbf{w}}_{\mathbf{5}} = \left(-\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0\right)$$

$$\tilde{\mathbf{w}}_{\mathbf{6}} = \left(0, 0, 0, 0, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$$

$$\tilde{\mathbf{w}}_{\mathbf{7}} = \left(-\frac{1}{2\sqrt{2}}, -\frac{1}{2\sqrt{2}}, -\frac{1}{2\sqrt{2}}, -\frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}\right). \tag{4.757}$$

From this we construct a wavelet basis for the 11-sample long signal as:

$$\mathbf{e}_{\mathbf{o}} = \left(\frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}, \frac{1}{\sqrt{11}}\right)$$

$$\mathbf{e}_{\mathbf{1}} = \left(\frac{1}{\sqrt{110}}, \frac{1}{\sqrt{110}}, \frac{1}{\sqrt{110}}, \frac{1}{\sqrt{110}}, \frac{1}{\sqrt{110}}, \frac{1}{\sqrt{110}}, \frac{1}{\sqrt{110}}, \frac{1}{\sqrt{110}}, \frac{1}{\sqrt{110}}, \frac{1}{\sqrt{110}}, -\frac{10}{\sqrt{110}}\right)$$

$$\mathbf{e}_{\mathbf{2}} = \left(\frac{1}{\sqrt{90}}, \frac{1}{\sqrt{90}}, \frac{1}{\sqrt{90}}, \frac{1}{\sqrt{90}}, \frac{1}{\sqrt{90}}, \frac{1}{\sqrt{90}}, \frac{1}{\sqrt{90}}, \frac{1}{\sqrt{90}}, \frac{1}{\sqrt{90}}, -\frac{9}{\sqrt{90}}, 0\right)$$

$$\mathbf{e}_{\mathbf{3}} = \left(\frac{1}{\sqrt{72}}, \frac{1}{\sqrt{72}}, \frac{1}{\sqrt{72}}, \frac{1}{\sqrt{72}}, \frac{1}{\sqrt{72}}, \frac{1}{\sqrt{72}}, \frac{1}{\sqrt{72}}, \frac{1}{\sqrt{72}}, -\frac{8}{\sqrt{72}}, 0, 0\right)$$

$$\mathbf{w}_{\mathbf{1}} = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0, 0, 0, 0, 0\right)$$

$$\mathbf{w}_{\mathbf{2}} = \left(0, 0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0, 0, 0\right)$$

$$\mathbf{w}_{\mathbf{3}} = \left(0, 0, 0, 0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0, 0, 0\right)$$

$$\mathbf{w}_{\mathbf{4}} = \left(0, 0, 0, 0, 0, 0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0\right)$$

$$\mathbf{w}_{\mathbf{5}} = \left(-\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0, 0, 0, 0\right)$$

$$\mathbf{w_6} = \left(0, 0, 0, 0, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$$

$$\mathbf{w_7} = \left(-\frac{1}{2\sqrt{2}}, -\frac{1}{2\sqrt{2}}, -\frac{1}{2\sqrt{2}}, -\frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, 0, 0, 0\right). \tag{4.758}$$

What are the basis images in terms of which the finite Radon transform expands an image?

They are the images formed by wrapping into images the rows of matrix Φ as defined in Example 4.179 for the case of a 7×7 image (see Example 4.206). However, the rows of this matrix simply identify the pixels that make up the lines used by the Radon transform. So, the basis images are binary images, one basis image for each line used. In each such basis image only the pixels that make up the corresponding line are white, and all others are black. For a 7×7 image, and as there are 56 lines used, we expect to have 56 basis images.

Example 4.206

You are given a 3×3 image

$$\tilde{g} = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 3 & 1 \\ 2 & 2 & 2 \end{pmatrix}. \tag{4.759}$$

Work out matrix Φ you need in order to compute its finite Radon transform, and use it to compute the expansion of the image in terms of the basis images of the finite Radon transform.

First, we must remove the mean of the image. The mean is 2. So, we construct image:

$$g = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}. \tag{4.760}$$

Figure 4.149 shows the lines that have to be used by the Radon transform, and the notation we use for the image elements.

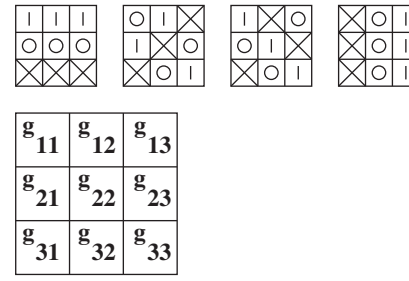


Figure 4.149 At the top, the digital lines we shall use for the Radon transform. At the bottom, the convention of pixel notation.

(Continued)

Example 4.206 (Continued)

By mere inspection, we see that matrix Φ is:

$$\Phi = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}. \tag{4.761}$$

Let us compute the finite Radon transform of this image by simply summing up the values identified by the same symbol in the lines shown at the top of Figure 4.149:

$$\begin{aligned} \mathbf{F}_{\text{RAD}} &= (0,0,0,0,0,0,2,-1,-1,1,1,-2)^T \\ &\equiv (f_1,f_2,f_3,f_4,f_5,f_6,f_7,f_8,f_9,f_{10},f_{11},f_{12})^T. \end{aligned} \tag{4.762}$$

According to Equation (4.704) we have

$$\frac{1}{3}\Phi^T\mathbf{F}_{\text{RAD}} = (g_{11},g_{21},g_{31},g_{12},g_{22},g_{32},g_{13},g_{23},g_{33})^T \tag{4.763}$$

where we have written the finite Radon transform in bold to stress that in this equation it has to be written as a column vector, following the same convention as the convention we used for writing the input image as a vector, on the right-hand side of the equation. Equation (4.763) may be written as:

$$3\begin{pmatrix} g_{11} \\ g_{21} \\ g_{31} \\ g_{12} \\ g_{22} \\ g_{32} \\ g_{13} \\ g_{23} \\ g_{33} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \\ f_{11} \\ f_{12} \end{pmatrix}$$

$$= f_1\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + f_2\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + f_3\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} + f_4\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + f_5\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + f_6\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} +$$

$$f_7\begin{bmatrix}1\\0\\0\\0\\1\\0\\0\\0\\1\end{bmatrix}+f_8\begin{bmatrix}0\\1\\0\\0\\0\\1\\1\\0\\0\end{bmatrix}+f_9\begin{bmatrix}0\\0\\1\\1\\0\\0\\0\\1\\0\end{bmatrix}+f_{10}\begin{bmatrix}1\\1\\1\\0\\0\\0\\0\\0\\0\end{bmatrix}+f_{11}\begin{bmatrix}0\\0\\0\\1\\1\\1\\0\\0\\0\end{bmatrix}+f_{12}\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\1\\1\end{bmatrix}. \tag{4.764}$$

We can now write the column representations of the various images in the ordinary 2D image format:

$$\begin{aligned}3\begin{bmatrix}g_{11}&g_{12}&g_{13}\\g_{21}&g_{22}&g_{23}\\g_{31}&g_{32}&g_{33}\end{bmatrix} &= f_1\begin{bmatrix}1&1&1\\0&0&0\\0&0&0\end{bmatrix}+f_2\begin{bmatrix}0&0&0\\1&1&1\\0&0&0\end{bmatrix}+f_3\begin{bmatrix}0&0&0\\0&0&0\\1&1&1\end{bmatrix}\\ &+f_4\begin{bmatrix}1&0&0\\0&0&1\\0&1&0\end{bmatrix}+f_5\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix}+f_6\begin{bmatrix}0&0&1\\0&1&0\\1&0&0\end{bmatrix}\\ &+f_7\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}+f_8\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}+f_9\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix}\\ &+f_{10}\begin{bmatrix}1&0&0\\1&0&0\\1&0&0\end{bmatrix}+f_{11}\begin{bmatrix}0&1&0\\0&1&0\\0&1&0\end{bmatrix}+f_{12}\begin{bmatrix}0&0&1\\0&0&1\\0&0&1\end{bmatrix}.\end{aligned} \tag{4.765}$$

We can now substitute the values we have, to verify the result:

$$\begin{aligned}3\begin{bmatrix}1&0&-1\\0&1&-1\\0&0&0\end{bmatrix} &= 2\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}-\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}-\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix}\\ &+\begin{bmatrix}1&0&0\\1&0&0\\1&0&0\end{bmatrix}+\begin{bmatrix}0&1&0\\0&1&0\\0&1&0\end{bmatrix}-2\begin{bmatrix}0&0&1\\0&0&1\\0&0&1\end{bmatrix}.\end{aligned} \tag{4.766}$$

This equation is exactly right.

Is the finite ridgelet transform invertible?

Yes, since both its stages are invertible. The finite Radon transform is invertible. The 1D wavelet transform we apply to the columns of the Radon transform is also invertible, since we make sure we use a complete and orthonormal basis for it.

What are the basis images in terms of which the finite ridgelet transform expands an image?

Let us start from Equation (4.704), which may be written as

$$\frac{1}{N}\Phi^T\mathbf{F}_{\text{RAD}}=\mathbf{g} \tag{4.767}$$

where $\mathbf{g}$ is the input image, written in a column vector form, and $\mathbf{F}_{\text{RAD}}$ is its finite Radon transform written in the same way. Note that matrix Φ is $N(N+1) \times N^2$, and so matrix Φ^T is $N^2 \times N(N+1)$. Vector $\mathbf{F}_{\text{RAD}}$ is $N(N+1)$-samples long, made up from the sums of the grey values of the image along all N lines of the same slope, then the N lines of the second slope, and so on, until all $N+1$ slopes have been considered. When we take the wavelet transform, we take it with respect to the displacement parameter of all lines of the same slope. So, what we actually do is to project the first N elements of $\mathbf{F}_{\text{RAD}}$ on the wavelet vector basis, then the next N elements, then the next, and so on. In other words, we need the matrix that is appropriate for the wavelet analysis of an N-sample long signal. Let us call this matrix W_N. The matrix that has to operate on the full column vector $\mathbf{F}_{\text{RAD}}$ to produce the ridgelet transform coefficients of the full $N \times N$ image then must have the form

$$W = \begin{pmatrix} W_N & 0_N & \dots & 0_N \\ 0_N & W_N & \dots & 0_N \\ \dots & \dots & \dots & \dots \\ 0_N & 0_N & \dots & W_N \end{pmatrix} \tag{4.768}$$

where 0_N is an $N \times N$ matrix made up from zeros.

We note that matrix W_N is orthogonal by construction (see the section How do we create a wavelet basis for a signal that is not a power of 2 in length? and Example 4.205). Then, we know that the inverse of the matrix is its transpose, and so, every successive N elements of $\mathbf{F}_{\text{RAD}}$ may be recovered by multiplying the vector of the coefficients of this expansion with matrix W^T. In other words, we may write

$$\mathbf{F}_{\text{RAD}} = \begin{pmatrix} W_N^T & 0_N & \dots & 0_N \\ 0_N & W_N^T & \dots & 0_N \\ \dots & \dots & \dots & \dots \\ 0_N & 0_N & \dots & W_N^T \end{pmatrix} \mathbf{F}_{\text{RID}} = W^T \mathbf{F}_{\text{RID}} \tag{4.769}$$

where $\mathbf{F}_{\text{RID}}$ are the coefficients of the finite ridgelet transform of the image written as a column vector. We may now substitute from (4.769) into (4.767):

$$\frac{1}{N}\Phi^T W^T \mathbf{F}_{\text{RID}} = \mathbf{g} \Rightarrow \mathbf{g} = \frac{1}{N}(W\Phi)^T \mathbf{F}_{\text{RID}}. \tag{4.770}$$

This equation expresses the expansion of image $\mathbf{g}$ in terms of the basis images of the ridgelet transform. The basis images are formed by the columns of matrix $(W\Phi)^T$, i.e. the rows of matrix $W\Phi$, wrapped round to form images. The coefficients of the expansion are the elements of vector $\mathbf{F}_{\text{RID}}$.

Example 4.207

Work out the expansion of image (4.759) in terms of the basis functions of the ridgelet transform. For the wavelet basis use the Haar wavelet.

The image without its mean value is given by (4.760). This is the image we shall be using in the expansion. We have first to calculate the wavelet basis for a three-sample long signal, as here $N = 3$. *The power of 2 that is smaller than 3 is 2. The Haar wavelet for such a signal is* $(-1/\sqrt{2}, 1/\sqrt{2})$. *Following the process described in the section How do we create a wavelet basis for a signal that is not a power of 2 in length?, we construct the basis we need as follows:*

$$\mathbf{e_0} = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$$

$$\mathbf{e_1} = \left(\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}, -\frac{2}{\sqrt{6}}\right)$$

$$\mathbf{w} = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0 \right). \tag{4.771}$$

Matrix W_3 is:

$$W_3 = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}. \tag{4.772}$$

Matrix W then is:

$$W = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}. \tag{4.773}$$

We may use now this matrix to operate on $\mathbf{F}_{\text{RAD}}$, given by (4.762) to work out $\mathbf{F}_{\text{RID}}$:

$$\mathbf{F}_{\text{RID}} = W\mathbf{F}_{\text{RAD}} =$$

$$\begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ -1 \\ -1 \\ 1 \\ 1 \\ -2 \end{pmatrix}$$

(Continued)

Example 4.207 (Continued)

$$= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{3}{\sqrt{6}} \\ -\frac{3}{\sqrt{2}} \\ 0 \\ \frac{6}{\sqrt{6}} \\ 0 \end{pmatrix}. \tag{4.774}$$

Next, we work out matrix $W\Phi$, with Φ given by (4.761):

$$\begin{pmatrix}
\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0
\end{pmatrix} \times$$

$$\begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{pmatrix} =$$

$$\begin{pmatrix}
\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\
\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\
-\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0
\end{pmatrix}. \tag{4.775}$$

The basis images are the columns of the transpose of this matrix, wrapped around to form images, by writing their first three elements as the first column, the next three elements as the second column, and so on, in a way analogous to the one we used to extract the basis images from Equation (4.764). So, the elementary images this transform uses are:

$$\begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix}$$

$$\begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}. \tag{4.776}$$

We note that these basis images are really ridgelets, i.e. ridge-like ripples along the diagonal and the main directions of the image. There is clearly redundancy, as the flat image appears four times. This is expected as we have 9 pixel values and we use 12 expansion coefficients.

We can verify now the expansion of the original image in terms of them by observing that $\mathbf{F}_{\text{RID}}$ *has only three non-zero values that correspond to the second, fourth and fifth from the end of these basis images. We may then write:*

$$3\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} = \frac{3}{\sqrt{6}}\begin{bmatrix} \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix} - \frac{3}{\sqrt{2}}\begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$
$$+\frac{6}{\sqrt{6}}\begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} \end{bmatrix}. \tag{4.777}$$

This equation may easily be verified.

Does the finite ridgelet transform extract local image information?

No. The Radon transform we apply first is a global transform. It is clear then that this transform in the form discussed so far cannot be used to characterise an image locally. We may, however, apply the transform locally, using non-overlapping windows of prime size. For example, if an image is of size 35×35, we may consider it that it consists of 49 windows of size 5×5 or of 25 windows of size 7×7. Each of these windows may be expanded in terms of the basis images of size 5×5 and 7×7, respectively, with the coefficients of the expansion forming the finite ridgelet transform of the window and playing the role of local image features. This is the **curvelet transform**, as curves may locally be approximated by straight line segments. So, the ridgelet transform applied locally is the known as the curvelet transform.

Example 4.208

Apply the curvelet transform to the image of Figure 4.71 using the finite ridgelet transform with the Haar wavelet.

This image is of size 15×15. *So, we can divide it into 9 windows of size* 5×5 *and into 25 windows of size* 3×3.

The basis vectors for the Haar wavelet transform of a five-sample long signal are:

$$\begin{aligned}
\mathbf{e_0} &= \left(\frac{1}{\sqrt{5}}, \frac{1}{\sqrt{5}}, \frac{1}{\sqrt{5}}, \frac{1}{\sqrt{5}}, \frac{1}{\sqrt{5}}\right)\\
\mathbf{e_1} &= \left(\frac{1}{\sqrt{20}}, \frac{1}{\sqrt{20}}, \frac{1}{\sqrt{20}}, \frac{1}{\sqrt{20}}, -\frac{4}{\sqrt{20}}\right)\\
\mathbf{w_1} &= \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, 0\right)\\
\mathbf{w_2} &= \left(0, 0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)\\
\mathbf{w_3} &= \left(-\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0\right).
\end{aligned} \tag{4.778}$$

Figure 4.150 shows the nine finite ridgelet transforms of the corresponding windows, obtained by projecting the columns of each finite Radon transform on vectors (4.778). This can be obtained by multiplying from the left each column with matrix:

$$W_5 = \begin{pmatrix}
\frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}}\\
\frac{1}{\sqrt{20}} & \frac{1}{\sqrt{20}} & \frac{1}{\sqrt{20}} & \frac{1}{\sqrt{20}} & -\frac{4}{\sqrt{20}}\\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0\\
0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0\\
-\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0
\end{pmatrix}. \tag{4.779}$$

223.6	335.4	223.6	335.4	223.6
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

214.7	214.7	232.6	232.6	214.7
17.9	17.9	−17.9	−17.9	17.9
0	28.3	0	−28.3	0
0	−28.3	0	28.3	0
40	0	−40	0	40

214.7	232.6	232.6	214.7	214.7
17.9	−17.9	−17.9	17.9	17.9
28.3	0	−28.3	0	28.3
−28.3	0	28.3	0	−28.3
0	−40	0	40	0

223.6	335.4	223.6	335.4	223.6
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

335.4	232.6	290.7	214.7	245.9
0	45	−22.3	−4.5	−44.7
0	−14.1	−28.3	0	28.3
0	0	0	−14.1	−28.3
0	−10	50	30	0

232.6	232.6	214.7	214.7	232.6
−17.9	−17.9	17.9	17.9	−17.9
0	−28.3	0	28.3	0
0	28.3	0	−28.3	0
−40	0	40	0	−40

223.6	335.4	223.6	335.4	223.6
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

335.4	223.6	335.4	223.6	335.4
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

232.6	214.7	214.7	232.6	232.6
−17.9	17.9	17.9	−17.9	−17.9
−28.3	0	28.3	0	−28.3
28.3	0	−28.3	0	28.3
0	40	0	−40	0

Figure 4.150 The nine finite ridgelet transforms of the corresponding 5 × 5 windows.

How do we apply the curvelet transform in practice?

First we must create the masks that will identify the pixels that make up each line we shall use. We can do that by using the following algorithm.

Step 1: Select the size of the window you will use, $N \times N$, where N must be a prime number.

Step 2: For $k = 0, 1, \ldots, N$, consider the set of locations $(-kq, q)$, allowing q to take values in the range $\left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$. These locations are modulo N. For example, if $-kq < -\frac{N-1}{2}$, you add N until it is in the range $\left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$. If $-kq > \frac{N-1}{2}$, you subtract N until it is in the range $\left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$. Make a list of these locations. Call it T_k.

Step 3: If (m, n) is an element of T_k, compute $d_{mn} \equiv \sqrt{m^2 + n^2}$. Identify the pairs (m, n) for which this value is the smallest, for fixed k, without considering the $(m, n) = (0, 0)$ value. If there are several such pairs, select the pair that has $n > 0$. Call the selected pair (a_k, b_k). You will have one such pair for each k.

Step 4: For each pair (a_k, b_k), construct an array $N \times N$. Call it M_k.

If $b_k \neq 0$, consider $\tilde{j} = \frac{t - a_k \tilde{i}}{b_k}$. For each $t = 0, 1, \ldots, N-1$, allow $\tilde{i}$ to take values $0, 1, 2, \ldots$, checking every time whether $\frac{t - a_k \tilde{i}}{b_k}$ is exactly an integer, until you find N such integer values of $\tilde{j}$. Form pairs $(\tilde{i}, \tilde{j})$. Reduce each such pair to (i, j), where $i = \tilde{i}_{\text{modulo } N}$ and $j = \tilde{j}_{\text{modulo } N}$, so that indices i and j take values in the range $0, 1, \ldots, N-1$. Set $M_k(i, j) = t$. If $b_k = 0$, it will be $a_k = 1$. Consider $i = t$. For each $t = 0, 1, \ldots, N-1$, set $M_k(i, j) = t$, for $j = 0, 1, \ldots, N-1$.

Step 5: For each (a_k, b_k), consider the angle

$$\phi_k = \tan^{-1} \frac{b_k}{a_k} \tag{4.780}$$

allowing it to take values in the range $[0, 180°)$.

0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6

(a,b) = (1,0) k = 7

6	1	3	5	0	2	4
5	0	2	4	6	1	3
4	6	1	3	5	0	2
3	5	0	2	4	6	1
2	4	6	1	3	5	0
1	3	5	0	2	4	6
0	2	4	6	1	3	5

(a,b) = (2,1) k = 5

6	0	1	2	3	4	5
5	6	0	1	2	3	4
4	5	6	0	1	2	3
3	4	5	6	0	1	2
2	3	4	5	6	0	1
1	2	3	4	5	6	0
0	1	2	3	4	5	6

(a,b) = (1,1) k = 6

5	6	0	1	2	3	4
3	4	5	6	0	1	2
1	2	3	4	5	6	0
6	0	1	2	3	4	5
4	5	6	0	1	2	3
2	3	4	5	6	0	1
0	1	2	3	4	5	6

(a,b) = (1,2) k = 3

6	6	6	6	6	6	6
5	5	5	5	5	5	5
4	4	4	4	4	4	4
3	3	3	3	3	3	3
2	2	2	2	2	2	2
1	1	1	1	1	1	1
0	0	0	0	0	0	0

(a,b) = (0,1) k = 0

5	4	3	2	1	0	6
3	2	1	0	6	5	4
1	0	6	5	4	3	2
6	5	4	3	2	1	0
4	3	2	1	0	6	5
2	1	0	6	5	4	3
0	6	5	4	3	2	1

(a,b) = (–1,2) k = 4

6	5	4	3	2	1	0
5	4	3	2	1	0	6
4	3	2	1	0	6	5
3	2	1	0	6	5	4
2	1	0	6	5	4	3
1	0	6	5	4	3	2
0	6	5	4	3	2	1

(a,b) = (–1,1) k = 1

6	4	2	0	5	3	1
5	3	1	6	4	2	0
4	2	0	5	3	1	6
3	1	6	4	2	0	5
2	0	5	3	1	6	4
1	6	4	2	0	5	3
0	5	3	1	6	4	2

(a,b) = (–2,1) k = 2

Figure 4.151 The numbers in each panel identify the pixels that make up the same line. The indices are the values of parameter t with respect to which the values of the Radon transform have to be written within each column, i.e. the sum of pixels marked with 0 will be written at the top, next the sum of the pixels marked with 1 and so on. The columns should be arranged next to each other, starting from the top left and working our way towards the bottom right in lexicographic order.

Step 6: Rank angles ϕ_k, in increasing order. Let us call $\tilde{k}(k)$ the rank of angle ϕ_k.

Step 7: Rename arrays M_k, so that $M_k = M_{\tilde{k}(k)}$. These are the masks that identify the pixels that make up each line, in the desired order, i.e. so that the digital lines are arranged so they capture low frequency information best, and the order of the batches of parallel lines is such that the angles are well spread.

Figure 4.151 shows the lines for windows of size 7×7.

These masks can be used for any image, so they are constructed just once. Having the masks then we proceed to take the curvelet transform of an image as follows.

Step 1: Select the size of window you are going to use. Call it N. It must be a prime number. For example, you may consider windows of size 5×5, 7×7, 11×11 or 13×13. You either trim or augment your image so it can be tiled exactly with non-overlapping windows of the selected size.

Step 2: Use the masks of the selected size to compute the finite Radon transform of each tiling window. The values of the Radon transform should be arranged so that each column corresponds to a different batch of parallel lines (different slope) and within each column, the sums of the pixels with identical label in the mask are given one under the other, with increasing value of label.

Step 3: Construct the wavelet basis for signals of length N, following the procedure described in the section How do we create a wavelet basis for a signal that is not a power of 2 in length?. By writing the basis vectors one under the other, form matrix W_N.

Step 4: Multiply from the left with matrix W_N each local Radon transform, to produce the ridgelet transform of each window.

Step 5: Tile all ridgelet transforms together to form the curvelet transform of the full image.

4.8 Where Image Processing and Pattern Recognition Meet

Why in wavelet analysis do we always split the band with the maximum energy?

If an image does not contain components within a certain frequency band, there is no point in splitting it any further. A band with low energy most likely contains noise only. However, concentrating on the band with the most energy does not necessarily mean that we concentrate on the band that will allow us to differentiate better the different textures present in the image. Ideally we want to identify separate sub-bands inside which the different textures manifest themselves. A band may have high energy but it may correspond to a single texture. It is therefore useful, when we decide which band (or bands) to split further, not only to identify a band with high energy, but also to examine whether the coefficient values in that band indicate the presence of more than one textures. This may be done by examining the histogram of the values of the band. If it is smooth and monomodal or it has an inflexion point hinting at the presence of a secondary peak, and at the same time it is of high energy, it may be worth splitting the band further. If the histogram of the band is bimodal, the band may contain more than one class and if it is replaced by its own sub-bands the clusters of the different textures may be fragmented unnecessarily. We may earmark such a band as one that might yield a good feature.

This is a special case of a process called **feature selection**. In this case, we are dealing with only one feature and we wish to assess its potential to discriminate the different textures present in the image. In general, most texture analysis methods tend to produce a large number of features, not all of which are expected to be useful for texture segmentation. The issue then is to select a subset of them that is most useful for the task at hand.

What is feature selection?

Feature selection is the process by which we identify the best subset of features for the discrimination of the classes that are present in an image. Feature selection is the subject of pattern recognition. In order to be able to identify the best subset of features, we must have a way of assessing the quality of a subset of features in terms of their discriminating power. We shall discuss here only one method of assessing the quality of a feature set, namely the one based on histogramming. We must emphasise that feature selection is not the same as **feature reduction**, where we simply wish to reduce the number of features we are dealing with. An example of feature reduction is what we are doing in wavelet analysis when we select the band with the maximum energy for further splitting, without worrying whether it is the band with the maximum potential for texture discrimination. When we also examine whether the band contains distinct clusters or not, then we introduce the concept of feature selection. Later on (Example 4.230), we shall see another example of feature reduction, where we select features that are linear combinations of the original features but have greater spread in their values than the original features. Such a selection is performed by principal component analysis (PCA). However, larger spread in values, just like higher energy, does not necessarily imply better discriminating ability. That is why principal component analysis is a method of feature reduction and not feature selection.

Example 4.209

Compute the histogram of the values of the wavelet coefficients for each band you created in the first level of analysis in Example 4.139 for the filter Daubechies 20.

(Continued)

Example 4.209 (Continued)

Comment on the smoothness of these histograms and on whether you could use them in an automated way to decide which band should be split further.

Since the features we construct from each band are energy features, we must compute the histogram of the absolute values of the wavelet coefficients of each band. Figure 4.152 shows the histograms constructed by using 30 bins for each band. We see that the histogram of the LL band hints at the presence of a second peak. This band should be split further because there is no clear separation between the two peaks. These histograms are quite smooth so we can envisage a fully automated process that could decide which band to split further by examining the number of local maxima of each histogram and the depth of the valleys that separate them.

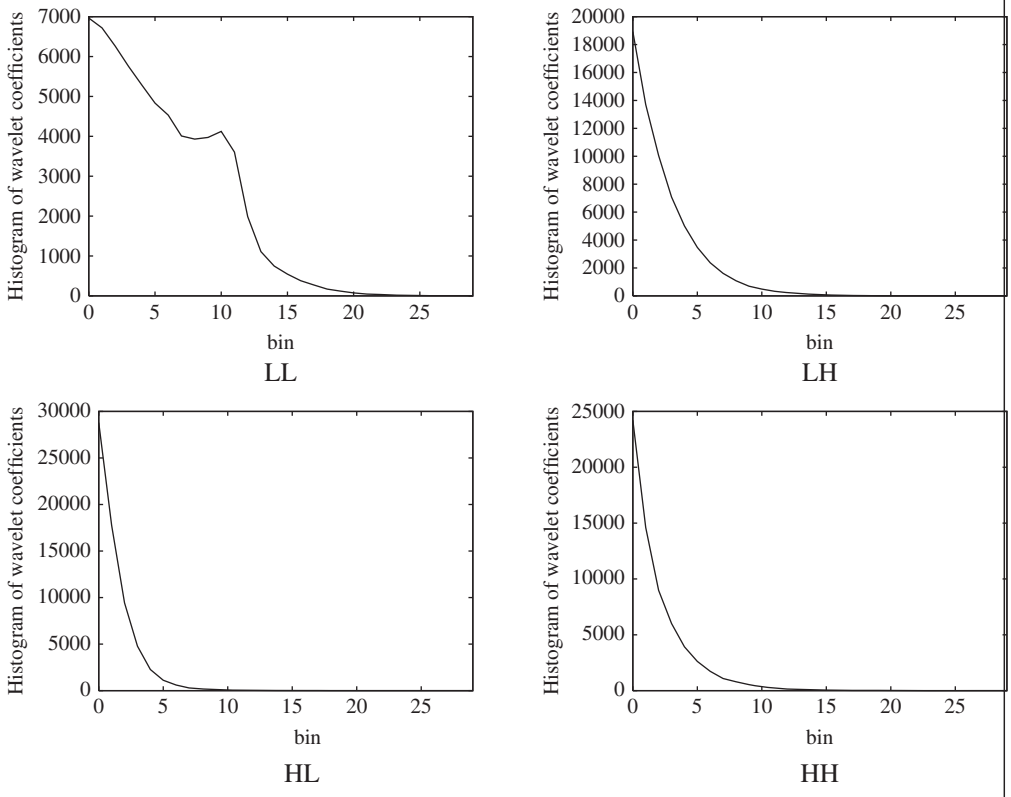


Figure 4.152 Histograms of the absolute values of the wavelet coefficients for the four bands of the first level of analysis in Example 4.139 for filter Daubechies 20. The range of real values of the coefficients in each band was divided into 30 bins.

How can we visualise the histogram of more than one feature in order to decide whether they constitute a good feature set?

One may use the **histogram of distances** of the data points in the feature space in order to assess whether a certain set of features may be used to identify the separate textures of an image.

What is the feature space?

Once each pixel has been assigned a set of feature values, we may consider that the image has been transformed into the feature space. The dimensionality of the feature space is equal to the number of features. The value of each feature may be measured along a separate axis. Thus, each pixel becomes a point in the feature space, with its coordinate position determined by its feature values. Points that belong to the same type of texture are expected to be plotted near each other and form a cluster. The idea here is to assess whether clusters exist or not in the feature space without having to identify them first.

What is the histogram of distances in a feature space?

Suppose that we consider pairs of points in the feature space, picked at random. The "distance" of these points is measured using an appropriate metric, say the Euclidean metric. Then we may plot the histogram of the distances we found. Let us first consider the case of a single feature (i.e. 1D feature space). If the histogram of the feature values were bimodal, we should see two peaks in this histogram of distances, one created by picking both members of the pair from the same population of the data, and one created by picking one member of the pair from one population and the other from the other. This is better demonstrated in Figure 4.153. Obviously, for 1D it is futile to construct the histogram of distances, because we may use the histogram of the feature values directly. The histogram of distances becomes useful for cases when more than one or two features are to be used together.

This is demonstrated in Figure 4.154. Let us say that we use two features, i.e. the feature space is 2D. If the points (pixels) in this feature space are uniformly distributed, the feature space will look like Figure 4.154a. If there are clusters present in it, the feature space may look like Figure 4.154b. Suppose now that we choose pairs of pixels at random, and we create the histogram of the number of pairs of pixels versus their distance. Note: the pairs of pixels may be chosen directly from the image, but the word "distance" here refers to the feature space, i.e. it means the difference in their feature values. The histogram that will be created from Figure 4.154a will look like 4.154c. If there are two clusters in the feature space, this histogram is expected to be bimodal: the first peak will be created when we pick pairs of pixels, both of which belong to the same cluster (small relative distances in feature space); the second peak will be created from pairs of pixels that belong to two different clusters (large relative distances in feature space). The histograms are constructed from many pairs of pixels, but not **all** pairs because that would make the process computationally too

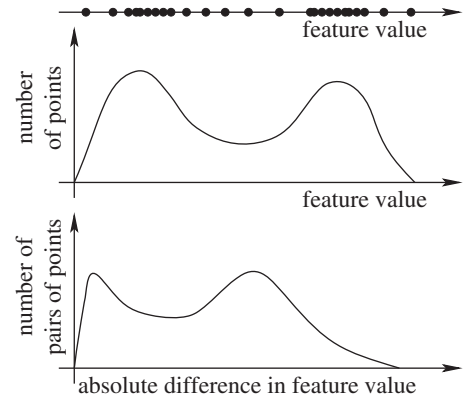


Figure 4.153 At the top, data points along the axis of a feature. In the middle, the histogram of the values of these points. It reflects the fact that the points are clustered around two distinct values. At the bottom, the distance histogram of pairs of points. It reflects the fact that we have a large number of pairs created by points that are tightly together, forming a single cluster, and also points that are far apart because they belong to two different clusters. The location of the first peak indicates the intra-cluster mean distance, while the location of the second peak indicates the inter-cluster mean distance.

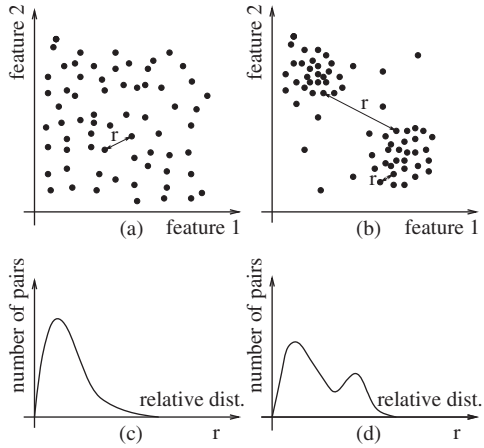


Figure 4.154 If the features we are using are not good for class discrimination, even if the data belong to two different classes, they will be uniformly distributed in the feature space. Such is the case shown in (a). If the features we are using are good for discriminating the two classes, the data will form distinct clusters in the feature space. This is the case shown in (b). We may assess whether the feature space contains clumps of points or not, without viewing it, by using the histogram of the number of pairs of points we find at a certain distance r from each other. In the first case, this histogram will look like (c) because all relative distances may arise. In the second case, this histogram will look like (d) because it will be dominated by pairs of points that belong to the same cluster, and therefore they are at small distances from each other, and by pairs of points which belong to two separate clusters, and therefore are at large distances from each other.

intensive. Due to the large number of possible pairs one may choose, such histograms tend to be smoother than histograms constructed from the values of individual features. So, it is relatively easy to automate the process of assessing whether they are monomodal or bimodal.

Is it possible that the histogram of distances does not pick up the presence of clusters, even though clusters are present?

Yes. For example if one population is much larger than the other, the histogram of distances will fail. Also, if the clusters are not very tight, the histogram of distances may appear monomodal.

Example 4.210

Compute the histograms of relative distances for the bands you created in the first level of analysis in Example 4.139 for filter Daubechies 20. Compare them with the histograms you constructed in Example 4.209.

The histograms of the relative distances of the absolute values of the wavelet coefficients of the bands of the first level of analysis of Example 4.139 for filter Daubechies 20 are shown in Figure 4.155.

We note that although the LL band according to the histogram in Example 4.209 hints at the presence of more than one cluster, this does not appear to be the case according to the distance histogram presented here. This indicates that the distance histogram is less sensitive to the presence of clusters than the histogram of the original values.

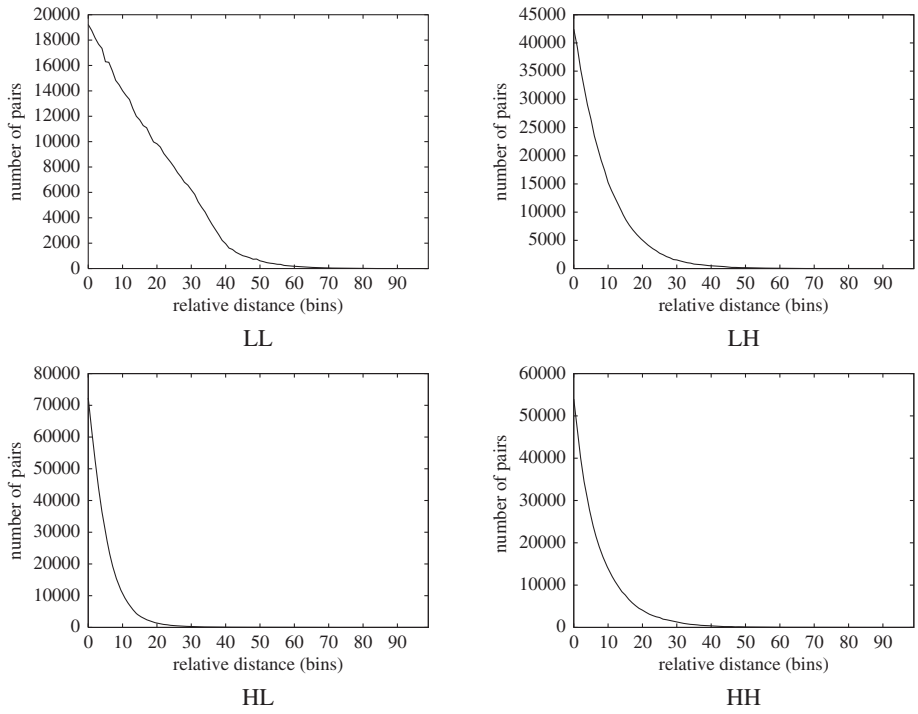


Figure 4.155 Histograms of the relative distances between the absolute values of the wavelet coefficients for the four bands in the first level of analysis of Example 4.139 for filter Daubechies 20. The range of real values for each band was divided into 100 bins. In total 429 496 pairs of pixels were used, corresponding to 10^{-4} of the total number of pairs one may construct from a 256×256 image.

Example 4.211

Assess the quality of one of the feature spaces produced in Example 4.139 with the help of the histogram of distances.

We select the feature space produced with the Haar wavelet and with the 25×25 *Gaussian window for calculating the local energy features. The feature space is 10D, as 10 energy features were computed for each pixel. These came from channels HH1, HH2, HH3, HL1, HL2, HL3, LH1, LH2, LH3 and LL3. The range of values of these features was* [0,104], [0,386], [0,518], [0,313], [0,593], [0,576], [0,701], [0, 1207], [0, 1051] *and* [0, 3526], *respectively.*

In order to create the histogram of distances we must decide upon the number of bins we shall use. The length of the main diagonal of the hyper-parallelepiped defined by the above features in the 10D feature space is 4086. *If we decide to use* 100 *bins, the width of each bin should be* 40.86. *The image is* 256×256 *in size, so it contains* 655 36 *pixels. There are* $655\,36^2 = 4\,294\,967\,000$ *possible pairs of pixels. We decide to pick at random only a fraction of* 0.0001 *of them, i.e. the histogram is constructed using* 429 496 *pairs of pixels picked at random. This histogram of distances is shown in Figure 4.156a.*

(Continued)

Example 4.211 (Continued)

When we use the Euclidean metric, we must bear in mind that all features are treated as equivalent. So, if a feature takes values on a much larger scale than the other features, it tends to dominate the calculation. In Figure 4.156b we show the histogram of relative feature distances computed after the features were scaled to take values in roughly the same range. The scaling factors used were 1, 3, 5, 3, 5, 5, 7, 10, 10 *and* 30*, for bands HH1, HH2, HH3, HL1, HL2, HL3, LH1, LH2, LH3 and LL3, respectively. These scaling factors were used to divide all feature values computed from the corresponding band before the Euclidean metric was used. Such a process is sometimes referred to a* **whitening** *of the data. It means that we make all bands of roughly equal energy, just like the white light theoretically has equal energy in all frequencies. We note that while histogram 4.156a clearly indicates the presence of clusters, histogram 4.156b does not. This shows that accentuating bands with low energies may destroy the structure of the feature space by drowning it in the enhanced noise. The process of whitening is only a good idea if we have reasons to believe that the structure we are trying to identify is hidden in the low energy bands.*

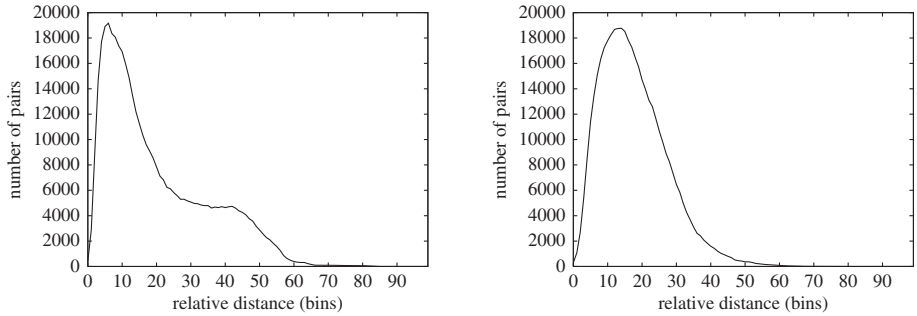


Figure 4.156 Histogram of the relative distances in the 10D feature space of 429 496 pairs of pixels picked at random from the image in Figure 4.81. (a) Unscaled feature values. The width of each bin is 40.86. (b) Scaled feature values. The width of each bin is 3.55.

How do we segment the image once we have produced a set of features for each pixel?

Once the pixels have been assigned feature values, the problem of image segmentation is transformed into a problem of clustering in the feature space. There are many clustering techniques, usually presented in pattern recognition books. We shall present here a commonly used method, the ***K*-means algorithm** and a more sophisticated version of it, called **deterministic annealing**.

What is the *K*-means algorithm?

Let us assume that we have K clusters, i.e. we expect to find in the image K different types of texture. Each pixel (i, j) is represented by an L-dimensional vector $\mathbf{x_{ij}}$, consisting of the values of the L features we computed for it. Let us also say that $\mathbf{y_k}$ is the position vector of the centre of cluster k, in the L-dimensional feature space. Let us call $d(\mathbf{x_{ij}}, \mathbf{y_k})$ the distance of point $\mathbf{x_{ij}}$ from the centre of cluster k. The algorithm consists of the following steps.

Step 0: Choose the number of clusters K.
Step 1: Choose at random the centres of the K clusters.
Step 2: Compute the distance of each pixel from each cluster centre and assign the pixel to the cluster from which its distance is minimal. Exit if no pixel changes cluster.
Step 3: Compute each cluster centre as the average of the pixels assigned to the cluster, and go to step 2.

This algorithm is very popular because of its simplicity, but it has many drawbacks: the distance of a pixel from the cluster centre is computed using the Euclidean metric. This means that all features are treated as equivalent. As we saw in Example 4.211, this is fine as long as the features are energy features in frequency bands. If, however, the features are of diverse nature and they genuinely are expected to take values in different ranges, one should make sure that all features are scaled so that they take values roughly in the same range. This way the distances will not be dominated by the features that take large numbers as values. One way to do that is to scale the value of every feature by dividing it with the standard deviation of the values this feature takes over the whole image.

The second major drawback of this algorithm is that it often gets stuck in sub-optimal solutions because it is a greedy algorithm. One way to overcome this problem is to run it with several different initialisations and use as final values of the cluster centres some consensus of the various answers received.

What is deterministic annealing?

Deterministic annealing is an algorithm that assigns every pixel to every cluster with a certain degree of confidence. The fact that it keeps all options open until the final step goes some way to overcome the drawbacks of the K-means algorithm.

We assume again that we have K clusters. Each point $\mathbf{x_{ij}}$ may be associated with each cluster k, with various degrees of confidence. Let us say that point $\mathbf{x_{ij}}$ belongs to cluster k with probability $p_k(\mathbf{x_{ij}})$, given by

$$p_k(\mathbf{x_{ij}}) = \frac{1}{Z_{ij}} \mathrm{e}^{-\frac{1}{T} d(\mathbf{x_{ij}}, \mathbf{y_k})} \tag{4.781}$$

where T is a temperature parameter, and Z_{ij} is a normalising constant computed so that $p_k(\mathbf{x_{ij}})$ is a probability, i.e.:

$$Z_{ij} \equiv \sum_{k=1}^{K} \mathrm{e}^{-\frac{1}{T} d(\mathbf{x_{ij}}, \mathbf{y_k})}. \tag{4.782}$$

The centre of cluster k is defined by:

$$\mathbf{y_k} = \frac{\sum_i \sum_j \mathbf{x_{ij}} p_k(\mathbf{x_{ij}})}{\sum_i \sum_j p_k(\mathbf{x_{ij}})}. \tag{4.783}$$

The algorithm then is as follows.

Step 0: Choose parameter values α, $T_{\max}$ and $T_{\min}$. Set $T = T_{\max}$.
Step 1: Choose at random the centres of the K clusters.
Step 2: Use Equation (4.781) to compute the initial values of $p_k(\mathbf{x_{ij}})$.
Step 3: Use Equation (4.783) to compute the centre of each cluster.

Step 4: Use Equation (4.781) to update the values of $p_k(\mathbf{x_{ij}})$ for all pixels.
Step 5: Set $T = \alpha T$, and if $T > T_{\min}$, go to step 3. Else exit.

At the end, assign each pixel to the cluster with which it is associated with maximum probability. Typical values for the cooling parameter α are 0.99 or 0.999 and typical values of $T_{\max}$ are of order 10, while typical values of $T_{\min}$ are of order 1. We may also use an alternative criterion of termination: the algorithm stops when no pixel changes its previous assignment and the centres of the clusters do not move from their positions in the previous iteration. The algorithm is similar to the simulated annealing algorithm we used in Chapter 3, but it does not have any stochastic step that allows one to worsen the solution in order to escape from local optima. So, it is very easy for this algorithm to lead to bad solutions, by collapsing, for example, two clusters into a single one when this is not appropriate. To avoid such situations, we may add an extra stochastic step to the algorithm, between steps 3 and 4:

Step 3.5: If the centres of two clusters come closer to each other than a certain threshold distance, one of the two is moved at random in another part of the feature space.

This way the algorithm outputs exactly the number of regions the user originally specified, and it produces very satisfactory results.

It can be shown that the algorithm of deterministic annealing chooses probabilities $p_k(\mathbf{x_{ij}})$ so that it maximises an entropy criterion (see Box 4.15).

Box 4.15 Maximum entropy clustering

A pixel (i,j) is associated with cluster k with probability $p_k(\mathbf{x_{ij}})$. We require the sum of all these probabilities to be 1:

$$\sum_{k=1}^{K} p_k(\mathbf{x_{ij}}) = 1 \qquad \text{for all pixels } (i,j). \tag{4.784}$$

If we were to replace the feature vector of pixel (i,j) with the feature vector that represents the centre of cluster k, we would have committed an error measured by the distance of points $\mathbf{x_{ij}}$ and $\mathbf{y_k}$, i.e. $d(\mathbf{x_{ij}}, \mathbf{y_k})$. However, if pixel (i,j) belongs to cluster k only with probability $p_k(\mathbf{x_{ij}})$, we may say that the error we commit is only $p_k(\mathbf{x_{ij}})d(\mathbf{x_{ij}}, \mathbf{y_k})$. Then the total error we commit is given by:

$$E = \sum_i \sum_j \sum_k p_k(\mathbf{x_{ij}}) d(\mathbf{x_{ij}}, \mathbf{y_k}). \tag{4.785}$$

There is a minimum value of this error, intrinsic to the data. No matter how we choose probabilities $p_k(\mathbf{x_{ij}})$, we cannot go below that minimum value. Let us call it E_0.

The entropy of the probability density function $p_k(\mathbf{x_{ij}})$ is given by

$$H = -\sum_i \sum_j \sum_k p_k(\mathbf{x_{ij}}) \log p_k(\mathbf{x_{ij}}). \tag{4.786}$$

The idea is to choose values of $p_k(\mathbf{x_{ij}})$, for all pixels and for all clusters, so that the entropy H is maximised, the energy E is minimised, and condition (4.784) is obeyed. We can express

these requirements by defining a composite entropy function

$$H_c \equiv H + \lambda_1 \left(\sum_{k=1}^{K} p_k(\mathbf{x_{ij}}) - 1 \right) + \lambda_2 \left(E_0 - \sum_i \sum_j \sum_k p_k(\mathbf{x_{ij}}) d(\mathbf{x_{ij}}, \mathbf{y_k}) \right) \tag{4.787}$$

where λ_1 and λ_2 are some positive constants. H_c will have to be maximised to yield the optimal assignment of $p_k(\mathbf{x_{ij}})$ values.

Let us compute the first derivative of H_c with respect to $p_k(\mathbf{x_{ij}})$:

$$\begin{aligned} \frac{\partial H_c}{\partial p_k(\mathbf{x_{ij}})} &= -\log p_k(\mathbf{x_{ij}}) - \log p_k(\mathbf{x_{ij}}) \frac{1}{\log p_k(\mathbf{x_{ij}})} + \lambda_1 - \lambda_2 d(\mathbf{x_{ij}}, \mathbf{y_k}) \\ &= -\log p_k(\mathbf{x_{ij}}) - 1 + \lambda_1 - \lambda_2 d(\mathbf{x_{ij}}, \mathbf{y_k}). \end{aligned} \tag{4.788}$$

We set the last expression to 0 to compute the value of $p_k(\mathbf{x_{ij}})$ that corresponds to the extremum of H_c:

$$\begin{aligned} -\log p_k(\mathbf{x_{ij}}) - 1 + \lambda_1 - \lambda_2 d(\mathbf{x_{ij}}, \mathbf{y_k}) = 0 \Rightarrow \\ \log p_k(\mathbf{x_{ij}}) = -1 + \lambda_1 - \lambda_2 d(\mathbf{x_{ij}}, \mathbf{y_k}) \Rightarrow \\ p_k(\mathbf{x_{ij}}) = \mathrm{e}^{-(1-\lambda_1+\lambda_2 d(\mathbf{x_{ij}}, \mathbf{y_k}))}. \end{aligned} \tag{4.789}$$

We know that these probabilities should sum up to 1 for each pixel. Then:

$$\sum_{k=1}^{K} \mathrm{e}^{-(1-\lambda_1+\lambda_2 d(\mathbf{x_{ij}}, \mathbf{y_k}))} = 1 \Rightarrow \mathrm{e}^{-(1-\lambda_1)} = \frac{1}{\sum_{k=1}^{K} \mathrm{e}^{-\lambda_2 d(\mathbf{x_{ij}}, \mathbf{y_k})}}. \tag{4.790}$$

We may substitute $\mathrm{e}^{-(1-\lambda_1)}$ from (4.790) into (4.789) to obtain

$$p_k(\mathbf{x_{ij}}) = \frac{\mathrm{e}^{-\lambda_2 d(\mathbf{x_{ij}}, \mathbf{y_k})}}{\sum_{k=1}^{K} \mathrm{e}^{-\lambda_2 d(\mathbf{x_{ij}}, \mathbf{y_k})}} \tag{4.791}$$

which upon the substitution $1/\lambda_2 \equiv T$ yields Equation (4.781) with definition (4.782).

So, Equation (4.781) gives the probability $p_k(\mathbf{x_{ij}})$ that maximises the composite entropy function H_c. The deterministic annealing algorithm performs iterations with ever-decreasing values of T. As T decreases, λ_2 increases. As λ_2 increases, the last term in the composite entropy expression (4.787) is given more importance than the other two terms of the expression. Since this term is non-positive, as T decreases, its value should go closer to 0 in order to maximise H_c, i.e. the total error E should go closer to its minimum value E_0.

Example 4.212

Use the features you produced in Example 4.139 to segment the image of Figure 4.81. *Each pixel of the original image is represented by a 10D feature vector, the elements of which are read from the 10 leaf bands of the wavelet expansion performed in Example 4.139. We shall use the deterministic annealing algorithm to segment the image. The results are shown in Figure 4.157, where we used a Gaussian window to compute the local energy features.*

(Continued)

Example 4.212 (Continued)

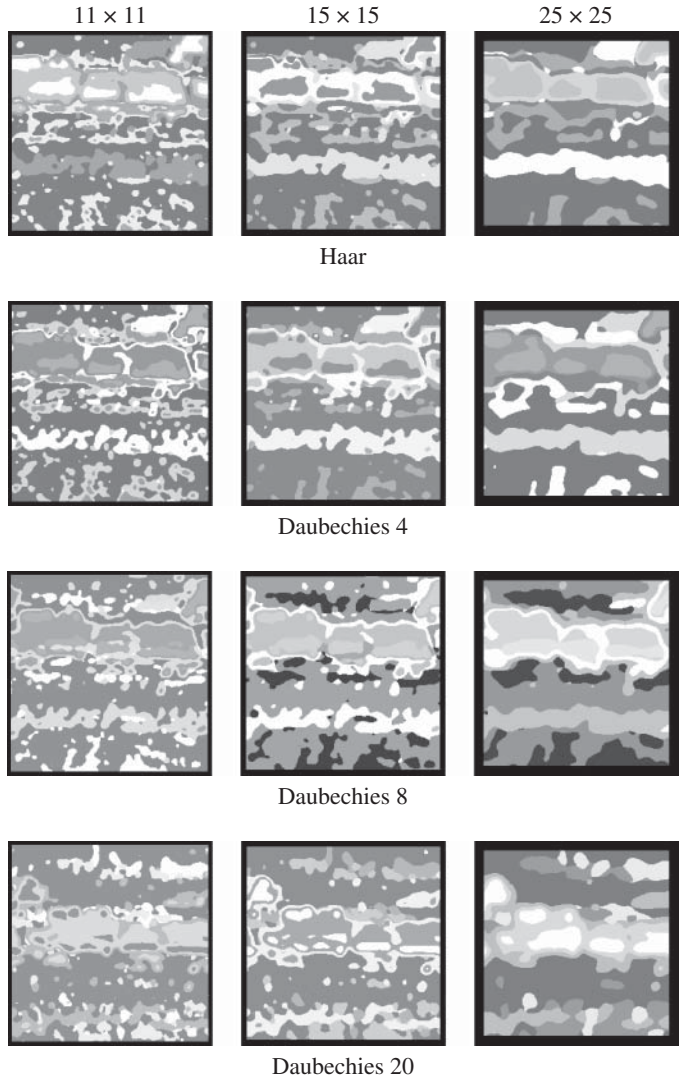


Figure 4.157 Segmentation results for $K = 6$ using a Gaussian window to compute the local energies. The features were constructed using the Haar, Daubechies 4, Daubechies 8 and Daubechies 20 filters, from top to bottom, respectively. The averaging window was of size 11×11, 15×15 and 25×25, from left to right, respectively. Source: Maria Petrou.

How may we assess the quality of a segmentation?

This is a difficult question. The standard way to proceed is to compare the produced segmentation with one produced manually. Ideally one would like to have objective measures of the quality of results, so full automation may be achieved. There are some functions which may be used to assess the quality of the clusters identified in the feature space **after** the clusters have been produced.

Note that the distance histogram we discussed at the beginning of this section was for assessing the quality of a set of features **before** the clusters had been identified. A commonly used measure of cluster separability is the **Bhattacharyya distance**. The Bhattacharyya distance may be used to measure the distance between two clusters only. So, if more than two clusters have been identified, it has to be applied for all pairs of identified clusters. Its smallest value computed over all pairs of clusters may be used to characterise the whole segmentation result.

How is the Bhattacharyya distance defined?

Let us say that we have two clusters k_1 and k_2, with means $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$, and covariance matrices Σ_1 and Σ_2, respectively. Their Bhattacharyya distance is given by

$$d(k_1,k_2) \equiv \frac{1}{8}(\boldsymbol{\mu_1}-\boldsymbol{\mu_2})^T\left(\frac{\Sigma_1+\Sigma_2}{2}\right)^{-1}(\boldsymbol{\mu_1}-\boldsymbol{\mu_2})+\frac{1}{2}\ln\frac{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \tag{4.792}$$

where $|\Sigma_i|$ means the determinant of matrix Σ_i.

How can we compute the Bhattacharyya distance in practice?

Let us say that each pixel is characterised by L features. The value of feature l for pixel (i,j) is given by $f_l(i,j)$. Let us also say that cluster k_1 (i.e. segmented region with label k_1) consists of N_1 pixels, and segmented region with label k_2 consists of N_2 pixels. We first compute the mean feature for each region. The lth component of each of the mean vectors $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ is given by:

$$\mu_{1l}=\frac{1}{N_1}\sum_{\text{all\_pixels\_labelled\_}k_1} f_l(i,j) \qquad \mu_{2l}=\frac{1}{N_2}\sum_{\text{all\_pixels\_labelled\_}k_2} f_l(i,j). \tag{4.793}$$

The (m,n) element of each of the covariance matrices we need is given by

$$\begin{aligned}\Sigma_1(m,n)&=\frac{1}{N_1}\sum_{\text{all\_pixels\_labelled\_}k_1}(f_m(i,j)-\mu_{1m})(f_n(i,j)-\mu_{1n})\\ \Sigma_2(m,n)&=\frac{1}{N_2}\sum_{\text{all\_pixels\_labelled\_}k_2}(f_m(i,j)-\mu_{2m})(f_n(i,j)-\mu_{2n}).\end{aligned} \tag{4.794}$$

After the inverse of matrix $\Sigma_1+\Sigma_2$ is calculated and the determinants of matrices Σ_1 and Σ_2 computed, formula (4.792) may be readily applied.

Example 4.213

Apply the formula of the Bhattacharyya distance for a 1D feature space and discuss the meaning of its two terms.

When we have only one feature, $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ are not vectors but scalars. They are the averages of the feature values of the regions labelled k_1 and k_2, respectively. Further, Σ_1 and Σ_2 are not matrices but the variances σ_1^2 and σ_2^2 of the same regions. Then formula (4.792) takes the form:

$$d(k_1,k_2)=\frac{1}{8}\frac{2(\mu_1-\mu_2)^2}{\sigma_1^2+\sigma_2^2}+\frac{1}{2}\ln\frac{\sigma_1^2+\sigma_2^2}{2\sigma_1\sigma_2}. \tag{4.795}$$

(Continued)

Example 4.213 (Continued)

The first term in the above expression measures the difference of the means of the two regions (clusters) weighed by the mean variance of the two regions. The tighter the clusters are, i.e. the smaller their variances, the more significant a certain difference in their means becomes. If the clusters are very diffuse and their spreads are extensive, even a large difference in their means may be insignificant if it is comparable with the average spread of the clusters. The second term measures the difference of the two clusters in terms of their spreads. Note that this term becomes 0 *only if* $\sigma_1 = \sigma_2$*. For all other values of* σ_1 *and* σ_2*, it takes values greater than* 0 *because* $(\sigma_1^2 + \sigma_2^2)/(2\sigma_1\sigma_2)$ *is always greater than 1. So, two regions/clusters may be considered as having non-zero distance even when they have identical mean values, but distinct variances.*

How may we assess the quality of a segmentation using a manual segmentation as reference?

The first step is to identify which region in one segmentation corresponds to which region in the other. Let us say that the manual segmentation had labels $\{m_1, m_2, \dots, m_J\}$ and the automatic segmentation produced labels $\{k_1, k_2, \dots, k_I\}$. First we have to establish a correspondence between these labels. We do that following this procedure:

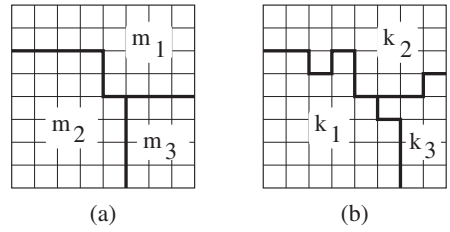
- For a region in the hand-segmented result with label m_j we check the labels of the same pixels in the automatic result.
- We count how many of the pixels of region m_j have label k_1, k_2, etc. in the automatic result.
- The label that is most represented among the pixels of hand-segmented region m_j is considered to be the corresponding label to m_j.

Once we have established such a correspondence, we may proceed to measure the committed errors. There are two ways to go about that: producing a **confusion matrix**, or defining the **over-detection** and **under-detection** errors. The latter are also known as **commission** and **omission** errors, respectively.

What is a confusion matrix?

This method is most appropriate when both segmentations have the same number of regions. The method may be best explained with an example. Let us say that the hand segmentation had three distinct regions with labels m_1, m_2 and m_3, and the automatic segmentation produced three distinct regions with labels k_1, k_2 and k_3. Let us also say that we established that label m_1 corresponds to label k_2, label m_2 corresponds to label k_1 and label m_3 corresponds to label k_3. Figure 4.158 shows two such segmentations.

The confusion matrix is a double-entry matrix with the corresponding labels arranged along the horizontal and the vertical axes as shown in Figure 4.158c. Each element of the matrix shows the number of pixels that according to the hand segmentation belong to class m_j and according to the automatic segmentation belong to class k_l. The larger the numbers along the main diagonal of the matrix in comparison with the off-diagonal entries, the better the result. The off-diagonal elements tell us exactly which class/region has been confused with which and that is why this structure is called confusion matrix.



(a) (b)

HS \ AS	k 2	k 1	k 3
m 1	23	0	1
m 2	1	27	0
m 3	0	3	9

(c)

Figure 4.158 (a) The reference hand segmentation of an image. (b) The result of an automatic segmenter. (c) The confusion matrix of the segmentation achieved. Along the left-most column we put the labels of the reference hand segmentation (HS). Along the top row we put the labels of the automatic segmentation (AS) in order of correspondence to the reference segmentation labels. Each entry of the matrix shows the number of pixels that in the reference segmentation had a particular label m_j and in the automatic segmentation had label k_i. The higher the numbers along the main diagonal with respect to the off-diagonal numbers, the better the segmentation.

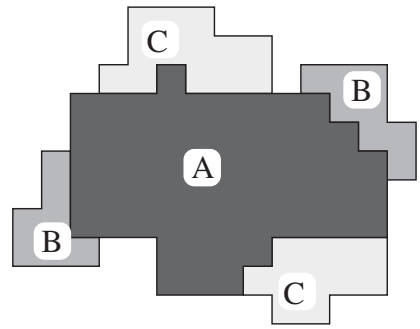


Figure 4.159 A region (A+B) in the reference segmentation partly overlaps with a region (A+C) in the automatic segmentation. Parts B constitute the under-detected parts of the region, while parts C constitute the over-detected parts.

What are the over- and under-detection errors?

This is better explained with the help of Figure 4.159. Let us say that a region in the reference segmentation is made up of parts A and B. The corresponding region in the automatic segmentation is the one with which this region has the maximum overlap, and consists of parts A and C. Let us say that parts A, B and C consist of N_A, N_B and N_C pixels, respectively. Then the over-detection, OD, and under-detection, UD, errors are defined as:

$$\text{OD} \equiv \frac{N_C}{N_A + N_C} \qquad \text{UD} \equiv \frac{N_B}{N_A + N_B}. \tag{4.796}$$

These errors may be computed for all regions of the segmentation and summed up to yield an overall segmentation error.

Example 4.214

Calculate the over-detection and under-detection errors for segmentation 4.158b. What is the average over-detection and under-detection error for this segmentation?

(Continued)

Example 4.214 (Continued)

From the confusion matrix 4.158c we can see that region k_1 overlaps with region m_2 with 27 pixels and it has 3 extra pixels. So, for this region, $N_A = 27$ and $N_C = 3$. At the same time, its corresponding region in the reference segmentation, region m_2, has one extra pixel, so $N_B = 1$. Applying, therefore, formulae (4.796), we obtain for k_1: $\mathrm{OD}_1 = 3/30 = 0.10$ and $\mathrm{UD}_1 = 1/28 = 0.04$. For region k_2 we have $\mathrm{OD}_2 = 1/(23+1) = 0.04$ and $\mathrm{UD} = 1/(23+1) = 0.04$. For region k_3 we have $\mathrm{OD}_3 = 1/(9+1) = 0.10$ and $\mathrm{UD}_3 = 3/(9+3) = 0.25$. So, the average over-detection error is $(\mathrm{OD}_1 + \mathrm{OD}_2 + \mathrm{OD}_3)/3 = 0.08$ and the average under-detection error is $(\mathrm{UD}_1 + \mathrm{UD}_2 + \mathrm{UD}_3)/3 = 0.11$.

How can we search for a pattern in an image?

The obvious way is to scan the image and check where the pattern fits best. So, we need to have a way to measure the similarity of the pattern with the sub-image with which we compare it. There are three such measures we may use. (i) If we believe that the pattern and the image were captured under similar imaging conditions, with possibly the only difference a global change in the level of illumination, we may use the cross-correlation coefficient between the pattern and the sub-image we compare. (ii) If we think that the pattern and the image have been captured by different instruments, or under different imaging conditions, but we expect this difference only to have caused some more or less one-to-one mapping between the grey values in the pattern and the grey values in the image, then we may use either the **mutual information** or the **matching by tone mapping (MTM)** method.

How do we compute the cross-correlation between two image patches?

The cross-correlation between two (sub-)images $I_1(i,j)$ and $I_2(i,j)$ is defined as

$$\rho(I_1, I_2) = \frac{\sum_i \sum_j (I_1(i,j) - \mu_1)(I_2(i,j) - \mu_2)}{NM\sigma_1\sigma_2} \tag{4.797}$$

where $N \times M$ is assumed to be the size of the compared (sub-)images, and μ_1 and μ_2, and σ_1 and σ_2 are the mean and standard deviations of the two (sub-)images, respectively. Note that if one image is α times brighter than the other, this will not affect the result because of the normalisation made by the division with the standard deviations of the two images.

How do we compute the mutual information between two images?

Assume that we place one (sub-)image on top of the other. We construct a 2D histogram, where a cell $P_{g_i g_2}$ contains the number of pixels which have in one image grey value g_i and in the other image grey value g_2. Once we construct this 2D histogram, we may divide its entries with the number of pixels and treat it like the joint probability density function $p_{I_1 I_2}(g_1, g_2)$ of the grey values of the two images. We also construct the histogram of each image separately, and normalise it by dividing by the total number of pixels. Let us call these two normalised histograms $p_{I_1}(g_1)$ and $p_{I_2}(g_2)$. Then the mutual information is computed as

$$M(I_1, I_2) = \sum_{g_1} \sum_{g_2} p_{I_1 I_2}(g_1, g_2) \log \frac{p_{I_1 I_2}(g_1, g_2)}{p_{I_1}(g_1) p_{I_2}(g_2)}. \tag{4.798}$$

In this expression the basis of the logarithm we use does not matter, as long as we are consistent when we make comparisons. Often base 2 is used, inspired by the information theory.

Example 4.215

Show that when the two images we compare are a toned mapped version of each other, the two images have the same entropy and their mutual information is equal to that entropy.

By definition, the entropies of the two images will be:

$$H_1 = -\sum_{g_1} p_{I_1}(g_1)\log p_{I_1}(g_1) \qquad H_2 = -\sum_{g_2} p_{I_2}(g_2)\log p_{I_2}(g_2). \tag{4.799}$$

If the two images differ only by a tone mapping, it means that there is a one-to-one correspondence between grey levels g_1 and g_2. Then for every value of $p_{I_1}(g_1)$ there is one and only one value of $p_{I_2}(g_2)$ with which it is equal. As in the calculation of the entropy the order by which we sum the various terms does not matter, $H_1 = H_2$.

In addition, $p_{I_1I_2}(g_1, g_2) = p_{I_1}(g_1) = p_{I_2}(g_2)$, since the probability of finding a pair (g_1, g_2) will be the same as the probability for finding a grey value g_1 in the first image and a probability of finding a grey value g_2 in the second image due to the one-to-one correspondence. So, we may write:

$$M(I_1, I_2) = \sum_{g_1}\sum_{g_2} p_{I_1}(g_1)\log\frac{p_{I_1}(g_1)}{p^2_{I_1}(g_1)} = -\sum_{g_1}\sum_{g_2} p_{I_1}(g_1)\log p_{I_1}(g_1) = H_1. \tag{4.800}$$

Example 4.216

Show that when the two images we compare are independent, their mutual information is 0.

When the two images are independent, $p_{I_1I_2}(g_1, g_2) = p_{I_1}(g_1)p_{I_2}(g_2)$. Then the fraction on the right-hand side of (4.798) is 1, its logarithm will be 0, and as a result $M(I_1, I_2) = 0$.

How do we perform matching by tone mapping between two images?

In order to match a pattern I_1 to a sub-image I_2 of a larger image, assuming that the pattern and the sub-image differ by a tone mapping, i.e. a non-linear one-to-one transformation between their grey values, we measure the distance $D(I_1, I_2)$ as the minimum distance between all tone mapped transforms of I_1 from I_2

$$D(I_1, I_2) \equiv \min_{\mathbf{h}_1}\frac{||S(I_1)\mathbf{h_1} - \mathbf{I_2}||^2}{NM\sigma^2_{I_2}} \tag{4.801}$$

where $S(I_1)$ is the slice transform of pattern I_1 (see Box 3.23), $\mathbf{h_1}$ is a sequence of the grey values of pattern I_1 written as a vector, $\mathbf{I_2}$ is sub-image I_2 written as a vector, $N \times M$ is the size of the pattern and the sub-image, and σ_{I_2} is the standard deviation of the grey values of sub-image I_2.

It can be shown that this distance may be computed as

$$D(I_1, I_2) = \frac{1}{NM\sigma^2_{I_2}}\left[||\mathbf{I_2}||^2 - \sum_i \frac{\mathbf{p_i}\cdot\mathbf{I_2}}{|\mathbf{p_i}|}\right] \tag{4.802}$$

where $\mathbf{p_i}$ is one of the columns of the slice transform $S(I_1)$ and $|\mathbf{p_i}|$ is the sum of its elements, i.e. the number of pixels in pattern I_1 that have the particular grey value to which this column corresponds. It is obvious from this formula that we omit from the slice transform any columns that correspond to missing grey values, i.e. any columns made up from 0s only, i.e. any columns with $|\mathbf{p_i}| = 0$.

Example B4.217

Show that Equation (4.801) may be written as:

$$D(I_1, I_2) = \frac{||S(I_1)[S(I_1)^T S(I_1)]^{-1} S(I_1)^T \mathbf{I_2} - \mathbf{I_2}||^2}{NM\sigma_{I_2}^2}. \tag{4.803}$$

The numerator of (4.801) is a non-negative number, so its possible minimum value is 0*. If we set* $S(I_1)\mathbf{h_1} - \mathbf{I_2} = 0$*, it is as if we have a system of linear equations with unknowns* $\mathbf{h_1}$ *to solve for* $\mathbf{h_1}$*. Matrix S is not a square matrix, so we can only solve this system by considering the least squares error solution, that will minimise the difference between its left and right hand, i.e. it will minimise the numerator of (4.801). The sequence* $\mathbf{h_1}$ *that will minimise the numerator of (4.801) can be computed by taking the pseudo-inverse of matrix S. In the following we drop the explicit dependence of S on* I_1*, for simplicity:*

$$\begin{aligned} & S\mathbf{h_1} = \mathbf{I_2} \\ \Rightarrow\ & S^T S\mathbf{h_1} = S^T\mathbf{I_2} \\ \Rightarrow\ & \mathbf{h_1} = (S^T S)^{-1} S^T \mathbf{I_2}. \end{aligned} \tag{4.804}$$

So, the sequence of grey values that minimises the numerator of (4.801) is given by (4.804). Note that the denominator is independent of $\mathbf{h_1}$*, so if we substitute in (4.801) we shall obtain (4.803).*

Example B4.218

Show that the numerator of (4.803) may be written as

$$||\mathbf{I_2}||^2 - ||G^{-1/2} S^T \mathbf{I_2}||^2 \tag{4.805}$$

where G **is a diagonal matrix, with values along its diagonal the number of pixels of pattern** I_1 **that have a particular grey value. If** $\mathbf{p_i}$ **is the** i**th column of the slice transform** S **of pattern** I_1**, the** i**th element of matrix** G **along its diagonal will be denoted as** $|\mathbf{p_i}|$ **and be equal to the sum of the elements of vector** $\mathbf{p_i}$**.**

First, we observe that the columns of matrix S are vectors orthogonal to each other (see Box 3.23). We also note that when one of these column vectors is dotted with itself, the result will be the square of the sum of its values, i.e. the square of the number of pixels that have as grey value the value that corresponds to that column vector. So, we may define $S^T S \equiv G$*, where G is a diagonal matrix with elements along the diagonal equal to the square of the number of pixels in* I_1 *that have a particular grey value.*

Let us try to calculate the numerator of (4.803) dropping the explicit dependence of S on I_1*:*

$$\begin{aligned} & ||S(S^T S)^{-1} S^T \mathbf{I_2} - \mathbf{I_2}||^2 \\ &= ||SG^{-1} S^T \mathbf{I_2} - \mathbf{I_2}||^2 \\ &= [SG^{-1} S^T \mathbf{I_2} - \mathbf{I_2}]^T [SG^{-1} S^T \mathbf{I_2} - \mathbf{I_2}] \\ &= [SG^{-1} S^T \mathbf{I_2}]^T [SG^{-1} S^T \mathbf{I_2}] - [SG^{-1} S^T \mathbf{I_2}]^T \mathbf{I_2} - \mathbf{I_2}^T [SG^{-1} S^T \mathbf{I_2}] + \mathbf{I_2}^T \mathbf{I_2} \\ &= \mathbf{I_2}^T SG^{-1} S^T SG^{-1} S^T \mathbf{I_2} - \mathbf{I_2}^T SG^{-1} S^T \mathbf{I_2} - \mathbf{I_2}^T SG^{-1} S^T \mathbf{I_2} + ||\mathbf{I_2}||^2 \\ &= \mathbf{I_2}^T SG^{-1} GG^{-1} S^T \mathbf{I_2} - 2\mathbf{I_2}^T SG^{-1} S^T \mathbf{I_2} + ||\mathbf{I_2}||^2 \end{aligned}$$

$$
\begin{aligned}
&= -\mathbf{I_2}^T S G^{-1} S^T \mathbf{I_2} + ||\mathbf{I_2}||^2 \\
&= -\mathbf{I_2}^T S G^{-1/2} G^{-1/2} S^T \mathbf{I_2} + ||\mathbf{I_2}||^2 \\
&= [G^{-1/2} S^T \mathbf{I_2}]^T [G^{-1/2} S^T \mathbf{I_2}] + ||\mathbf{I_2}||^2 \\
&= ||\mathbf{I_2}||^2 - ||G^{-1/2} S^T \mathbf{I_2}||^2. \qquad (4.806)
\end{aligned}
$$

Example B4.219

Making use of result (4.806), and starting from (4.801), prove Equation (4.802).

We note that result (4.806) concerns the minimisation of the numerator of (4.801), while the denominator does not change. We also note that $S^T\mathbf{I_2}$ is the projection of vector $\mathbf{I_2}$ on the rows of matrix S^T, i.e. the columns of matrix S, the slice transform of pattern I_1. If we denote the ith column of S by $\mathbf{p_i}$, we may write for $S^T\mathbf{I_2} = \sum_i \mathbf{p_i} \cdot \mathbf{I_2}$. Matrix $G^{1/2}$ is a diagonal matrix with elements along the diagonal the number of pixels of I_1 that have a particular grey value, i.e. the ith nelement along the diagonal of $G^{1/2}$ is the sum of elements of the ith column of S, which we call $|\mathbf{p_i}|$. If we substitute then into (4.801), (4.802) follows.

Example 4.220

Identify whether the pattern given on the left of Figure 3.113 is present, and, if yes, where, in image 4.160, using the tone mapping method.

Figure 4.160 An image in which we search to identify the 3 × 3 pattern shown on the left of Figure 3.113.

1	0	3	3	1
1	2	3	2	1
1	2	2	1	0

The slice transform S of the searched pattern is given by the 9×4 *matrix of Equation (3.353). The columns of this matrix are the following vectors:*

$$
\begin{aligned}
&\mathbf{p_1} = (1,1,0,1,0,0,0,0,0)^T \\
&\mathbf{p_2} = (0,0,1,0,1,0,0,0,0)^T \\
&\mathbf{p_3} = (0,0,0,0,0,1,1,1,0)^T \\
&\mathbf{p_4} = (0,0,0,0,0,0,0,0,1)^T. \qquad (4.807)
\end{aligned}
$$

Obviously:

$$|\mathbf{p_1}| = 3 \quad |\mathbf{p_2}| = 2 \quad |\mathbf{p_3}| = 3 \quad |\mathbf{p_4}| = 1. \qquad (4.808)$$

There are three sub-images in the image where the given pattern may match. If we write them as vectors, they are:

(Continued)

Example 4.220 (Continued)

$$\mathbf{w_1} = (1,1,1,0,2,2,3,3,2)^T$$
$$\mathbf{w_2} = (0,2,2,3,3,2,3,2,1)^T$$
$$\mathbf{w_3} = (3,3,2,3,2,1,1,1,0)^T. \tag{4.809}$$

The average of each one of these sub-images is:

$$\mu_1 = \frac{15}{9} \simeq 1.7 \quad \mu_2 = \frac{18}{9} = 2 \quad \mu_3 = \frac{16}{9} \simeq 1.8. \tag{4.810}$$

Using these average values, we compute the variance of each sub-image as:

$$\sigma_1^2 = \frac{8}{9} \quad \sigma_2^2 = \frac{8}{9} \quad \sigma_3^2 = \frac{9.55}{9}. \tag{4.811}$$

We can also work out the magnitude square of each sub-image vector as:

$$||\mathbf{w_1}||^2 = 33 \quad ||\mathbf{w_2}||^2 = 44 \quad ||\mathbf{w_3}||^2 = 38. \tag{4.812}$$

The dot product of each vector sub-image with each one of the columns of matrix S is:

$$\begin{aligned} &\mathbf{p_1}\cdot\mathbf{w_1} = 2 \quad \mathbf{p_2}\cdot\mathbf{w_1} = 3 \quad \mathbf{p_3}\cdot\mathbf{w_1} = 8 \quad \mathbf{p_4}\cdot\mathbf{w_1} = 2\\ &\mathbf{p_1}\cdot\mathbf{w_2} = 5 \quad \mathbf{p_2}\cdot\mathbf{w_2} = 5 \quad \mathbf{p_3}\cdot\mathbf{w_2} = 7 \quad \mathbf{p_4}\cdot\mathbf{w_2} = 1\\ &\mathbf{p_1}\cdot\mathbf{w_3} = 9 \quad \mathbf{p_2}\cdot\mathbf{w_3} = 4 \quad \mathbf{p_3}\cdot\mathbf{w_3} = 3 \quad \mathbf{p_4}\cdot\mathbf{w_3} = 0. \end{aligned} \tag{4.813}$$

Now we are ready to apply formula (4.802) to work out the distance of each sub-image from the reference pattern:

$$\begin{aligned} D(I_1,\mathbf{w_1}) &= \frac{1}{9\times\frac{8}{9}}\left[33 - \left(\frac{4}{3}+\frac{9}{2}+\frac{64}{3}+\frac{4}{1}\right)\right]\\ &= \frac{1}{8}\left[33 - \frac{187}{6}\right] = 0.229\\ D(I_1,\mathbf{w_2}) &= \frac{1}{9\times\frac{8}{9}}\left[44 - \left(\frac{25}{3}+\frac{25}{2}+\frac{49}{3}+\frac{1}{1}\right)\right]\\ &= \frac{1}{8}\left[44 - \frac{229}{6}\right] = 0.729\\ D(I_1,\mathbf{w_3}) &= \frac{1}{9\times\frac{9.55}{9}}\left[38 - \left(\frac{81}{3}+\frac{16}{2}+\frac{9}{3}+\frac{0}{1}\right)\right]\\ &= \frac{1}{9.55}\left[38 - \frac{228}{6}\right] = 0. \end{aligned} \tag{4.814}$$

We can see that the searched for pattern is present in the given image in the last three columns of it, as the distance is minimum between that sub-image and the given pattern. The tone mapping between the pattern and the sub-image was:

$$0 \to 3 \quad 1 \to 2 \quad 2 \to 1 \quad 3 \to 0. \tag{4.815}$$

Example 4.221

Identify whether the pattern given on the left of Figure 3.113 is present, and, if yes, where, in image 4.160, using mutual information.

The normalised histogram of the reference pattern is given by: $p^0 = \frac{1}{9}(3,2,3,1)$. *The histograms of the three possible sub-images where the reference pattern might be present are:* $p^1 = \frac{1}{9}(1,3,3,2)$, $p^2 = \frac{1}{9}(1,1,4,3)$, $p^3 = \frac{1}{9}(1,3,2,3)$. *The joint normalised histograms between the reference pattern and each one of the image windows are:*

$p^{01}(0,0) = 1/9$ $p^{02}(0,0) = 1/9$ $p^{03}(0,0) = 0$
$p^{01}(0,1) = 2/9$ $p^{02}(0,1) = 0$ $p^{03}(0,1) = 0$
$p^{01}(0,2) = 0$ $p^{02}(0,2) = 1/9$ $p^{03}(0,2) = 0$
$p^{01}(0,3) = 0$ $p^{02}(0,3) = 1/9$ $p^{03}(0,3) = 3/9$
$p^{01}(1,0) = 0$ $p^{02}(1,0) = 0$ $p^{03}(1,0) = 0$
$p^{01}(1,1) = 1/9$ $p^{02}(1,1) = 0$ $p^{03}(1,1) = 0$
$p^{01}(1,2) = 1/9$ $p^{02}(1,2) = 1/9$ $p^{03}(1,2) = 2/9$
$p^{01}(1,3) = 0$ $p^{02}(1,3) = 1/9$ $p^{03}(1,3) = 0$
$p^{01}(2,0) = 0$ $p^{02}(2,0) = 0$ $p^{03}(2,0) = 0$
$p^{01}(2,1) = 0$ $p^{02}(2,1) = 0$ $p^{03}(2,1) = 3/9$
$p^{01}(2,2) = 1/9$ $p^{02}(2,2) = 2/9$ $p^{03}(2,2) = 0$
$p^{01}(2,3) = 2/9$ $p^{02}(2,3) = 1/9$ $p^{03}(2,3) = 0$
$p^{01}(3,0) = 0$ $p^{02}(3,0) = 0$ $p^{03}(3,0) = 1/9$
$p^{01}(3,1) = 0$ $p^{02}(3,1) = 1/9$ $p^{03}(3,1) = 0$
$p^{01}(3,2) = 1/9$ $p^{02}(3,2) = 0$ $p^{03}(3,2) = 0$
$p^{01}(3,3) = 0$ $p^{02}(3,3) = 0$ $p^{03}(3,3) = 0$.

Then we can use these values in (4.798) in order to compute the mutual information between the reference pattern and each one of the three sub-images:

$$\begin{aligned}
M^{01} &= p^{01}(0,0)\log\frac{p^{01}(0,0)}{p^0(0)p^1(0)} + p^{01}(0,1)\log\frac{p^{01}(0,1)}{p^0(0)p^1(1)} + p^{01}(1,1)\log\frac{p^{01}(1,1)}{p^0(1)p^1(1)} \\
&\quad + p^{01}(1,2)\log\frac{p^{01}(1,2)}{p^0(1)p^1(2)} + p^{01}(2,2)\log\frac{p^{01}(2,2)}{p^0(2)p^1(2)} + p^{01}(2,3)\log\frac{p^{01}(2,3)}{p^0(2)p^1(3)} \\
&\quad + p^{01}(3,2)\log\frac{p^{01}(3,2)}{p^0(3)p^1(2)} \\
&= \frac{1}{9}\log\frac{\frac{1}{9}}{\frac{3}{9}\times\frac{1}{9}} + \frac{2}{9}\log\frac{\frac{2}{9}}{\frac{3}{9}\times\frac{3}{9}} + \frac{1}{9}\log\frac{\frac{1}{9}}{\frac{2}{9}\times\frac{3}{9}} + \frac{1}{9}\log\frac{\frac{1}{9}}{\frac{2}{9}\times\frac{3}{9}} + \frac{1}{9}\log\frac{\frac{1}{9}}{\frac{3}{9}\times\frac{3}{9}} \\
&\quad + \frac{2}{9}\log\frac{\frac{2}{9}}{\frac{3}{9}\times\frac{2}{9}} + \frac{1}{9}\log\frac{\frac{1}{9}}{\frac{1}{9}\times\frac{3}{9}} = \frac{2}{9}\log 3 = 0.318.
\end{aligned} \tag{4.816}$$

In a similar way, we work out that the mutual information with the second sub-image is $M^{02} = 0.210$ *and with the third* $M^{03} = 0.569$. *So, the reference pattern has most similarity with the third sub-image. However, we cannot tell that this is the maximum possible similarity the pattern would have with a sub-image, unless we calculate its own entropy and check how close the value of 0.569 is to the maximum possible value the mutual information can take. The entropy of the reference image is:*

$$\begin{aligned}
H^0 &= -\frac{3}{9}\log\frac{3}{9} - \frac{2}{9}\log\frac{2}{9} - \frac{3}{9}\log\frac{3}{9} - \frac{1}{9}\log\frac{1}{9} \\
&= 0.569.
\end{aligned} \tag{4.817}$$

(Continued)

Example 4.221 (Continued)

So, the mutual information between the reference pattern and the third sub-image is the maximum possible, indicating that indeed the third sub-image is identical to the reference image apart from a tone transformation.

Example 4.222

Identify whether the pattern given on the left of Figure 3.113 is present, and, if yes, where, in image 4.160, using correlation.

To apply formula (4.797) we have first to remove the mean from each image. The mean of the reference image is 1.2*, while the mean of each of the three sub-images with which we shall compare it is* 1.7, 2 *and* 1.8*, respectively. Figure 4.161 shows the reference pattern and three sub-images after the mean has been removed. Note that the rounding to one decimal point was done so that the sum of the values of each of these images is* 0*. The standard deviation of the reference image is* 1.036*, while that of the three sub-images is* 0.920, 0.943 *and* 1.036*, respectively. Note that these numbers do not exactly agree with the corresponding values we calculated in other examples involving the same sub-images, because here we make use of the approximate values given in Figure 4.161, for consistency when we use formula (4.797).*

−1.2	−1.2	0.8
−1.2	−0.3	0.8
−0.3	0.8	1.8

Reference

−0.6	−1.7	1.3
−0.6	0.3	1.3
−0.6	0.3	0.3

Subimage 1

−2	1	1
0	1	0
0	0	−1

Subimage 2

1.2	1.2	−0.8
1.2	0.3	−0.8
0.3	−0.8	−1.8

Subimage 3

Figure 4.161 The reference image and the three sub-images with which it has to be compared after their means have been removed.

Applying then formula (4.797) for the reference and the first sub-image we obtain:

$$\begin{aligned}
\rho(I_0, I_1) &= \frac{1}{9 \times 1.036 \times 0.920}(1.2 \times 0.6 + 1.2 \times 1.7 + 0.8 \times 1.3 + 1.2 \times 0.6 \\
&\quad -0.3 \times 0.3 + 0.8 \times 1.3 + 0.3 \times 0.6 + 0.8 \times 0.3 + 1.8 \times 0.3) \\
&= \frac{6.43}{8.58} = 0.75 \\
\rho(I_0, I_2) &= \frac{1}{9 \times 1.036 \times 0.943}(1.2 \times 2 - 1.2 \times 1 + 0.8 \times 1 - 0.3 \times 1 - 1.8 \times 1) \\
&= \frac{-0.10}{8.79} = -0.01 \\
\rho(I_0, I_3) &= \frac{1}{9 \times 1.036 \times 1.036}(-1.2 \times 1.2 - 1.2 \times 1.2 - 0.8 \times 0.8 - 1.2 \times 1.2 \\
&\quad -0.3 \times 0.3 - 0.8 \times 0.8 - 0.3 \times 0.3 - 0.8 \times 0.8 - 1.8 \times 1.8) \\
&= \frac{-9.66}{9.66} = -1. \qquad (4.818)
\end{aligned}$$

We note that the largest value of the correlation is between the reference pattern and the first sub-image. This method shows that the reference image and the third sub-image, which is a tone transform of it, are strongly anticorrelated. In general, correlation is only appropriate for searching for patterns that differ by a uniform change in brightness level.

Box 4.16 Fast tone matching algorithm

Step 0: The size of the reference pattern P is $N \times M$, where N and M are preferably odd, and we wish to search for it inside an image I, with size $K \times L$, so that $N << K$ and $M << L$. Set $N_0 = \frac{N-1}{2}$ and $M_0 = \frac{M-1}{2}$.

Step 1: Convolve the input image with a window of size $N \times M$, with all its elements equal to 1. This will sum all pixel values around each pixel, inside a window equal in size with the reference pattern. Call the result W_1

Step 2: Raise to the power of 2 every pixel value of W_1 and divide it by NM. Call the result W_2.

Step 3: Raise to the power of 2 every pixel value of image I. Call the result I^2.

Step 4: Convolve I^2 with a window of size $N \times M$, with all its elements equal to 1. This will sum all square pixel values around each pixel, inside a window equal in size to that of the reference pattern. Call the result W_3

Step 5: Compute $D_2 \equiv W_3 - W_2$. This is the denominator of Equation (4.802).

Step 6: Create the slices of the reference pattern as binary images of size $N \times M$, each one of which identifies with 1 the pixels of the pattern that have a particular grey value and with 0 all other pixels. Call these slices P_i. There will be as many as the number of different grey levels in the pattern.

Step 7: Sum up the elements of each P_i, to compute numbers N_i.

Step 8: Create an array of size $K \times L$, and set all its elements equal to 0. Call it D_1.

Step 9: For all slices P_i, do the following:

Step 9.1: Pseudo-convolve (no filter reversal) image I with P_i. Do not process the boundary pixels, i.e. all pixels in the first and last N_0 columns of the image and all pixels in the first and last M_0 rows of the image. Instead, give to all those pixels a very large value, say 10^6 or the largest number the bits you allocated to the output array of this step allows you, so that these pixels do not interfere with the final result when you are looking for local minima in the output array of the algorithm. Produce result A_i.

Step 9.2: Raise to the power of 2 the elements of A_i, and divide them by N_i. Call the result B_i.

Step 9.3: Add B_i to D_1: $D_1 = D_1 + B_i$.

Step 10: Compute $D_3 = W_3 - D_1$.

Step 11: Divide element by element D_3 by D_2, to produce an output array D, the same size as the input image. The values of this array tell us how different the sub-image around each pixel is from the reference image. The minima in D identify the possible locations of the presence of the pattern.

Box 4.17 From mutual information to a point similarity measure

In the definition of mutual information given by Equation (4.798), we realise that $p_{I_1 I_2}(g_1, g_2)$ is actually the ratio of pairs of pixels with values (g_1, g_2) in the two images that are being compared, over the number of pixels of each image. This ratio may be written as: $N(g_1, g_2)/(NM)$, assuming that the images are of size $N \times M$. Then we realise that the summation over grey values in (4.798) may be replaced by summation over spatial coordinates, with the definition of mutual information now taking the form:

$$M(I_1, I_2) = \frac{1}{NM} \sum_i \sum_j \log \frac{p_{I_1 I_2}(g_1(i,j), g_2(i,j))}{p_{I_1}(g_1(i,j)) p_{I_2}(g_2(i,j))} \tag{4.819}$$

(Continued)

Box 4.17 (Continued)

This allows us to interpret mutual information as the average similarity measure over all pixels positions, by defining the **point similarity** as:

$$S(i,j) \equiv \log \frac{p_{I_1I_2}(g_1(i,j), g_2(i,j))}{p_{I_1}(g_1(i,j))p_{I_2}(g_2(i,j))}. \tag{4.820}$$

We may define the **point similarity function** as:

$$T(g_1, g_2) \equiv \log \frac{p_{I_1I_2}(g_1, g_2)}{p_{I_1}(g_1)p_{I_2}(g_2)}. \tag{4.821}$$

By definition, we set $T(i,j) = 0$ if $p_{I_1I_2}(g_1, g_2) = 0$.

Point similarity measures are useful in:

(i) Registering images.
(ii) Assessing the similarity of small image regions (by averaging them over their extent).
(iii) Identifying anomalies in images at locations where the similarity is unusually high (in absolute terms).
(iv) Identifying changes in images of the same scene captured at different times, e.g. remote sensing images captured before an after a natural disaster.

Example B4.223

Compute the mutual information between the pattern given on the left of Figure 3.113 and the 3×3 left-most sub-image of image 4.160 using formula 4.819.

In Figure 4.162 we repeat the two images for convenience.

0	0	2
0	1	2
1	2	3

1	0	3
1	2	3
1	2	2

Figure 4.162 We must compute the mutual information of these two images by using summation over the spatial coordinates only.

In order to apply formula (4.819) we must make use of p_0, p^1 and p^{01} values listed in Example 4.221. We have:

$$\begin{aligned}
M(I_0, I_1) = \frac{1}{9}\Bigg[&\log\frac{p^{01}(0,1)}{p^0(0)p^1(1)} + \log\frac{p^{01}(0,0)}{p^0(0)p^1(0)} + \log\frac{p^{01}(2,3)}{p^0(2)p^1(3)} \\
&+ \log\frac{p^{01}(0,1)}{p^0(0)p^1(1)} + \log\frac{p^{01}(1,2)}{p^0(1)p^1(2)} + \log\frac{p^{01}(2,3)}{p^0(2)p^1(3)} \\
&+ \log\frac{p^{01}(1,1)}{p^0(1)p^1(1)} + \log\frac{p^{01}(2,2)}{p^0(2)p^1(2)} + \log\frac{p^{01}(3,2)}{p^0(3)p^1(2)}\Bigg] \\
= \frac{1}{9}\Bigg[&\log\frac{\frac{2}{9}}{\frac{3}{9}\times\frac{3}{9}} + \log\frac{\frac{1}{9}}{\frac{3}{9}\times\frac{1}{9}} + \log\frac{\frac{2}{9}}{\frac{3}{9}\times\frac{2}{9}} + \log\frac{\frac{2}{9}}{\frac{3}{9}\times\frac{3}{9}} \\
&+ \log\frac{\frac{1}{9}}{\frac{2}{9}\times\frac{3}{9}} + \log\frac{\frac{2}{9}}{\frac{3}{9}\times\frac{2}{9}} + \log\frac{\frac{1}{9}}{\frac{2}{9}\times\frac{3}{9}} + \log\frac{\frac{1}{9}}{\frac{3}{9}\times\frac{3}{9}} + \log\frac{\frac{1}{9}}{\frac{1}{9}\times\frac{3}{9}}\Bigg] \\
= \frac{2}{3}\log 3 = 0.318.
\end{aligned} \tag{4.822}$$

This is the same value we obtained when we applied the original formula of mutual information in Example 4.221.

Example B4.224

Compute the point similarity function between the images in Figure 4.162. Use it to work out the value of the point similarity measure between the two central pixels of the two images.

We shall make use of formula (4.821) and the values listed in Example 4.221 for p_0, p^1 *and* p^{01}.

$$T(0,0) = \log \frac{\frac{1}{9}}{\frac{3}{9}\times\frac{1}{9}} = \log 3$$

$$T(0,1) = \log \frac{\frac{2}{9}}{\frac{3}{9}\times\frac{2}{9}} = \log 2$$

$$T(1,1) = \log \frac{\frac{1}{9}}{\frac{2}{9}\times\frac{3}{9}} = \log 3 - \log 2$$

$$T(1,2) = \log \frac{\frac{1}{9}}{\frac{2}{9}\times\frac{2}{9}} = \log 3 - \log 2$$

$$T(2,2) = \log \frac{\frac{1}{9}}{\frac{3}{9}\times\frac{3}{9}} = 0$$

$$T(2,3) = \log \frac{\frac{2}{9}}{\frac{3}{9}\times\frac{2}{9}} = \log 3$$

$$T(3,2) = \log \frac{\frac{1}{9}}{\frac{1}{9}\times\frac{3}{9}} = \log 3$$

$$T(0,2) = T(0,3) = T(1,0) = T(1,3) = T(2,0) = T(2,1) = T(3,0) = T(3,1) = T(3,3) = 0.$$

If we place the two images one on the top of the other, the central pixel will have the pair of values $(1,2)$. *According to the table above, The point similarity measure between these two values is* $S(1,2) = \log 3 - \log 2 = 0.176$.

Example B4.225

Explain how you may use the point similarity measure to work out a change in a scene imaged at two different occasions, under similar imaging conditions.

When an anomaly appears in an image, unexpected pairs of grey values appear, having very low probability p^{01}. *Then the value of the point similarity measure at that location becomes very small, signaling the presence of something unexpected at that location.*

How can we compare two histograms?

We may use

(i) the χ^2 test or
(ii) the earth mover's distance.

What is the χ^2 test?

There are various versions of it, appropriate for different situations. Let us say that we have two histograms H_1 and H_2, with N identical bins each. The χ^2 is defined as:

$$\chi^2 \equiv \sum_{i=1}^{N} \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)}. \tag{4.823}$$

Then we select the confidence level α at which we wish to assert that these two histograms represent the same population (usually $\alpha = 0.95$ or $\alpha 0.99$), and look up in tables the threshold for the value of χ^2 that corresponds for the N bins. If χ^2 is greater than the threshold value, we may say that the two histograms are different at the 95% or 99% confidence level, depending on the value of α we chose. If χ^2 is lower than the threshold, we may say that the data are consistent as coming from the same population.

It is important to note that no bin of either of the two histograms should contain less than 5 items. If any bin does that, we combine neighbouring bins to avoid it.

What is the earth mover's distance?

The earth mover's distance is a measure of the distance between two histograms, measuring the minimum amount of mass that has to be moved between the bins of the one histogram in order to make it as similar as the other (see Box 4.18). When the histograms are normalised, this measure is a metric and it is known as the **Mallows distance**. We distinguish two cases:
(i) When the histograms are 1D, we may use the so-called **Hungarian algorithm** to compute their Mallows distance.
(ii) When the histograms are of higher dimensions, other more elaborate algorithms have to be used, which, however, are beyond the scope of this book.

What is the Hungarian algorithm?

The Hungarian algorithm computes the Mallows distance between two sets of feature values. It does not use histograms to bin the data, but rather works with the raw values. Let us call the two sets of data $h_1(i)$ and $h_2(i)$, for $i = 1, \ldots, N$. We sort each one of the sequences of numbers we have and call them: $h_{1(i)}$ and $h_{2(i)}$, respectively. Then the value of Mallows distance between the two sequences is:

$$M_p(h_1, h_2) = \left(\frac{1}{N} \sum_{i=1}^{N} |h_{1(i)} - h_{2(i)}|^p \right)^{1/p} \tag{4.824}$$

where usually $p = 1$ or $p = 2$.

If the two sequences consist of different number of values each, say $h_1(i)$ consists of N numbers and $h_2(i)$ consists of M numbers, then we find the minimum common multiple of N and M, say K, and replicate each sample of $h_{1(i)}$ as many times as needed to make it of length K, and each sample of $h_{2(i)}$ as many times as needed to make it also of length K, before we apply (4.824).

Box 4.18 The formal definition of earth mover's distance

Let us consider two normalised histograms H_1 and H_2 of the same number of bins, with identical bin borders. Let us define $f_{ij;kl}$ to be the amount by which bin (i, j) of H_1 may be reduced and bin (k, l) of H_1 may be increased. If we define such a function between any pair of bins in the histogram, we say we have a **flow** function over the histogram. The idea is that there are many flows one can define between the bins of H_1 that will make it more similar to H_2. We want to select the flow that moves the minimum "material" for the smallest possible distance between the bins of H_1 to achieve this similarity. Then the earth mover's distance between H_1

and H_2 is defined as

$$D_{\text{EM}} \equiv \min_{\text{all flows } f_{ij;kl}} \sum_{(i,j);(k,l)} f_{ij;kl} dij;kl \tag{4.825}$$

where $dij;kl$ is the distance between bins (i,j) and (k,l), measured, for example by the Euclidean metric.
This minimisation has to fulfil certain conditions:
All the flow that leaves bin (i,j), including self-flow, i.e. flow that goes back to the bin itself, must sum up to the original value this bin had in the first histogram:

$$\sum_{(k,l)} f_{ij;kl} = H_1(i,j). \tag{4.826}$$

All the flow that arrives to bin (k,l), including self-flow, i.e. flow that arrived from (k,l) itself, must sum up to the original value this bin had in the second histogram:

$$\sum_{(i,j)} f_{ij;kl} = H_2(k,l). \tag{4.827}$$

There are no negative flows:

$$f_{ij;kl} \geq 0 \qquad \text{for all } (i,j) \quad \text{and} \quad (k,l) \tag{4.828}$$

This minimisation problem can be solved with the help of the simplex algorithm and linear programming. However, for the case the distance used between the bins is

$$dij;kl = |i-k| + |j-l| \tag{4.829}$$

there is an elegant reformulation of the problem that can be solved using trees. It replaces the flow $f_{ij;kl}$ with a sequence of flows only between neighbouring bins. Let us define $g_{ij;kl}$ to be a flow that moves material only from bin (i,j) to bin (k,l), that is one of its four nearest neighbours. Then it can be shown that

$$D_{\text{EM}} = \min_{\textit{all flows } g_{ij;kl}} \sum_{(i,j);(k,l)} g_{ij;kl} \tag{4.830}$$

subject to the conditions:

$$\sum_{(k,l) \text{ nbr of } (i,j)} g_{ij;kl} - g_{kl;ij} - H_1(i,j) - H_2(i,j) \qquad \text{for all } (i,j)$$

$$g_{ij;kl} \geq 0 \text{ for all } (i,j) \text{ and } (k,l). \tag{4.831}$$

4.9 Laws' Masks and the "What Looks Like Where" Space

Is it possible to extract image features without referring to the frequency domain?

Yes, if we abandon the idea of representing a signal in the joint spatio-frequency domain. Wavelet filters were designed to span the joint spatio-frequency domain in an optimal way. They were not designed to extract image information that adheres to any perceptual local signal characteristic. So, if we concentrate on the spatial domain only, we shall have to stop being concerned with identifying "what happens where" in the signal (or image) and rather concentrate on the appearance of the

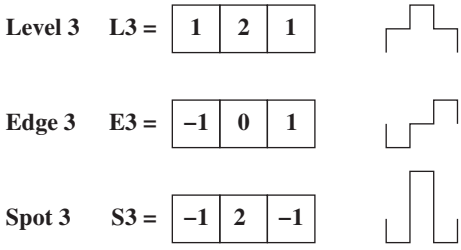


Figure 4.163 Laws' masks of size 3. On the left their names and on the right their shapes, drawn to scale.

image or signal in the real domain alone. We may define then the "what looks like where" space in conjunction with human perception.

This space will have to be spanned by filters that try to capture some local image structure that may be linguistically described. For this purpose people often use filters that have been heuristically defined, but that have been designed to capture local image characteristics such as local edges, corners, bars etc. **Laws' masks** are filters of this type.

How are Laws' masks defined?

Laws' 1D masks and their corresponding names and shapes are shown in Figures 4.163, 4.164 and 4.165 for neighbourhood sizes 3, 5 and 7, respectively. They are used as convolution filters. For a 2D image, and for a fixed size neighbourhood, they may be combined in all possible ways to produce 9,

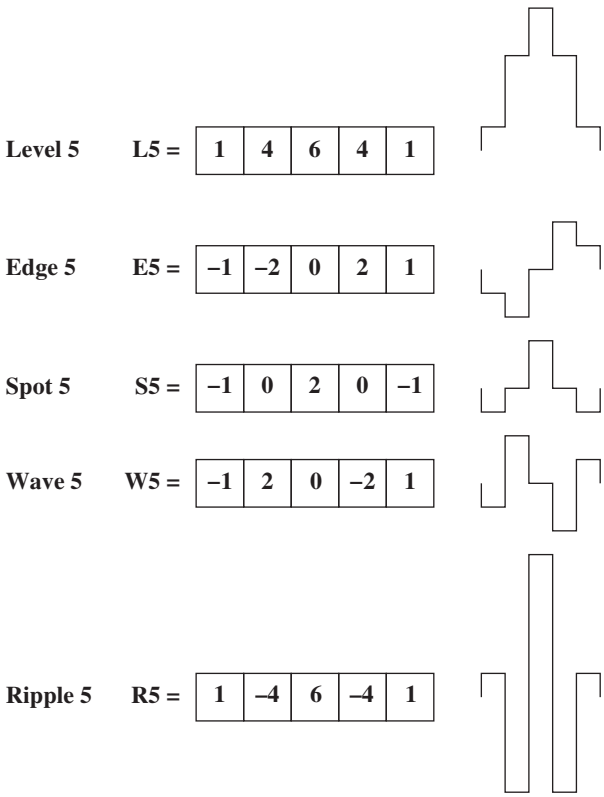


Figure 4.164 Laws' masks of size 5. On the left their names and on the right their shapes, drawn to scale.

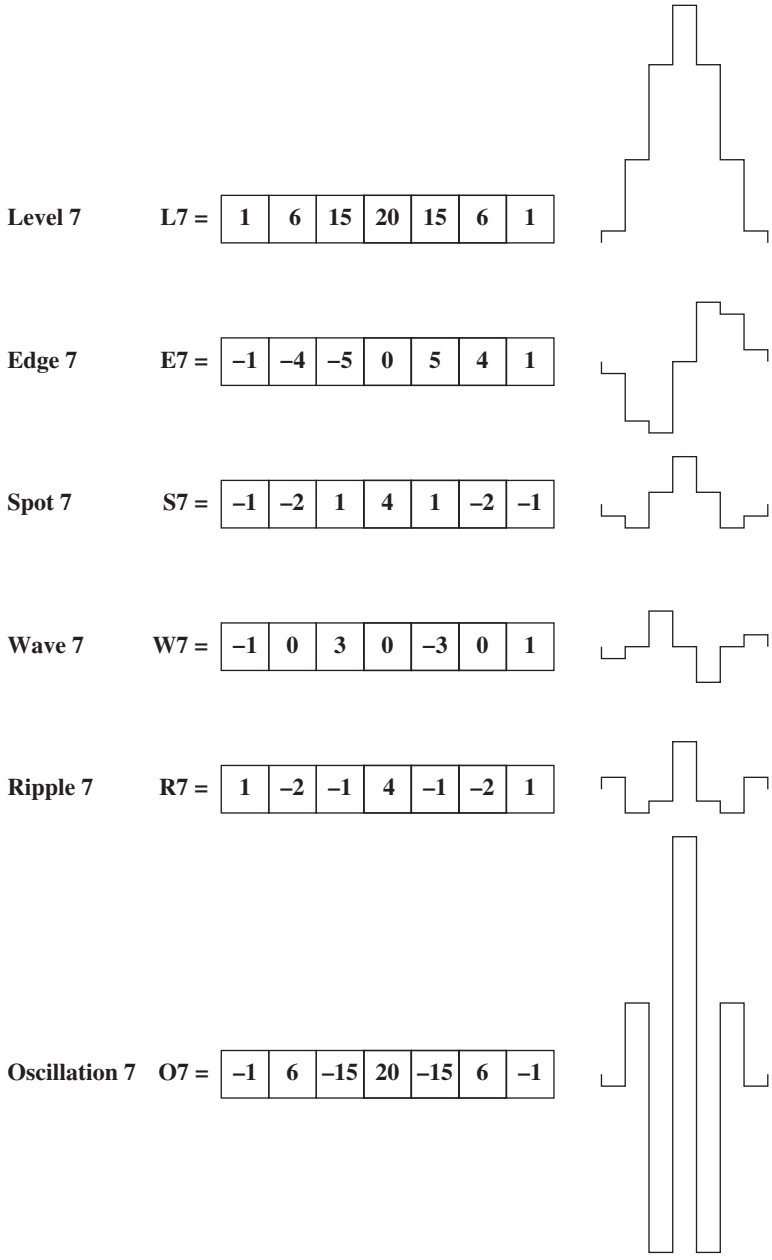


Figure 4.165 Laws' masks of size 7. On the left their names and on the right their shapes, drawn to scale.

25 or 36 local features respectively, by convolving with one filter along one axis and with the other along the other in a cascaded way (i.e. the second filter is applied to the output of the first).

When we perform convolution, it is always advisable to scale the masks so that they leave unaltered a flat input signal. This may be achieved by making sure that the sum of the weights of the low pass filter (the so-called "level" mask) is 1, and the sum of the weights of all other filters is 0. All high pass Laws' masks have weights summing up to 0, but the weights of the low pass masks do

not sum up to 1. So, the outputs of $L3$, $L5$ and $L7$ filters must be divided by 4, 16 and 64 respectively, to ensure that a flat input signal is left unchanged.

Usually we do not use the filter outputs as features, but we may produce from them features by taking the local standard deviation, or local average, or local absolute average within some local window around each pixel, i.e. some sort of local energy estimate. These features may be used for texture segmentation.

Note that it is not necessary to use masks of only a single size. If more than one size masks is used, we have a **multi-resolution** approach. These features may turn out to be too many, as many as $9 + 25 + 36 = 70$. **Feature reduction** may take place by performing principal component analysis (see Book I [75] and Example 4.230) and keeping only the first few principal components as features.

Note also that the three-tap Laws' masks constitute a complete basis for three-sample long signals. The same goes for the five-tap masks. However, the set of seven-tap masks does not constitute a complete basis. Laws' masks were defined guided by human intuition and not in a mathematically consistent way.

Example B4.226

Show that the sets of three- and five-tap Laws' masks constitute complete bases while the seven-tap masks do not.

To show that the three-tap Laws' masks constitute a complete basis we must show that any 3×1 *vector may be expanded in terms of these masks treated as vectors, and that it may be fully recovered from the coefficients of its expansion. Let us consider such a vector* $(x_1, x_2, x_3)^T$ *and let us write it as*

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = a_1 \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} + a_2 \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} + a_3 \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix} \Rightarrow$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 \\ 2 & 0 & 2 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \tag{4.832}$$

where a_1, a_2 *and* a_3 *are the coefficients of the expansion of vector* $(x_1, x_2, x_3)^T$ *in terms of the three basis vectors.*

For the last set of equations to be invertible, the determinant of the 3×3 *matrix should be non-zero, so that the inverse of this matrix exists. If the inverse exists, any vector* $(x_1, x_2, x_3)^T$ *may be represented in a unique way by the coefficients of its expansion in terms of the basis vectors. Indeed, this determinant is* $1 \times (0 - 2) + 1 \times (-2 - 2) - 1 \times (2 - 0) = -8$.

In a similar way we may show that the determinant of the 5×5 *matrix, created by writing the five-tap masks as columns, is non-zero, and actually it is equal to* -1024.

The seven-tap set of masks does not constitute a complete basis because there are only six of them. For a complete basis there should have been seven of them and the determinant of the matrix created by writing them next to each other, as columns, should have been non-zero.

Example 4.227

Use the 1×3 Laws' masks of Figure 4.163 to filter the image of Figure 4.81. Then compute features from the filtered outputs by estimating in each output the local energy of each pixel using a Gaussian window of size 29×29. Segment the image using the deterministic annealing algorithm for number of clusters $K = 6$ and $K = 8$.

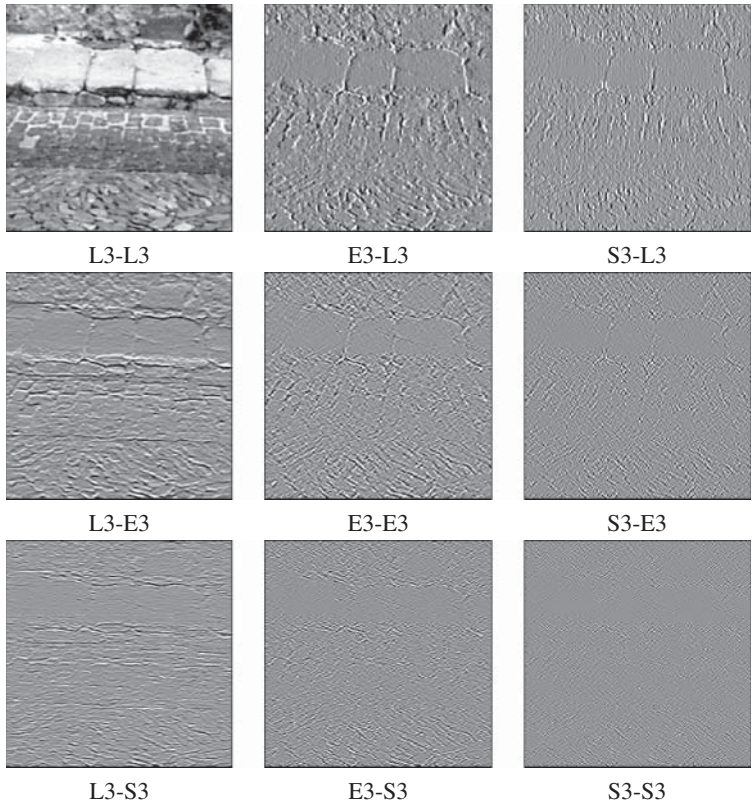
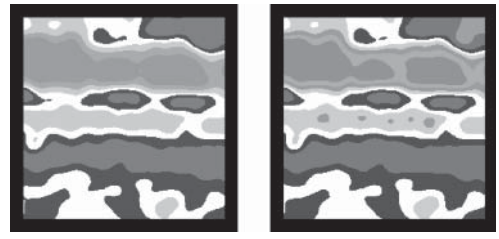


Figure 4.166 Outputs of the Laws' masks of size 3. The 2% extreme values of each panel were saturated and the rest were scaled to the range [0,255] for visualisation. Each panel $X-Y$ represents the output obtained when filter X is applied to the rows of the original image and filter Y is applied to the columns of the output. Source: Maria Petrou.

Figure 4.167 Segmentation results. Source: Maria Petrou.



Example 4.228

Use the Laws' masks of Figure 4.164 of size 5×5 to compute features for the image of Figure 4.81 and segment it using the deterministic annealing algorithm.

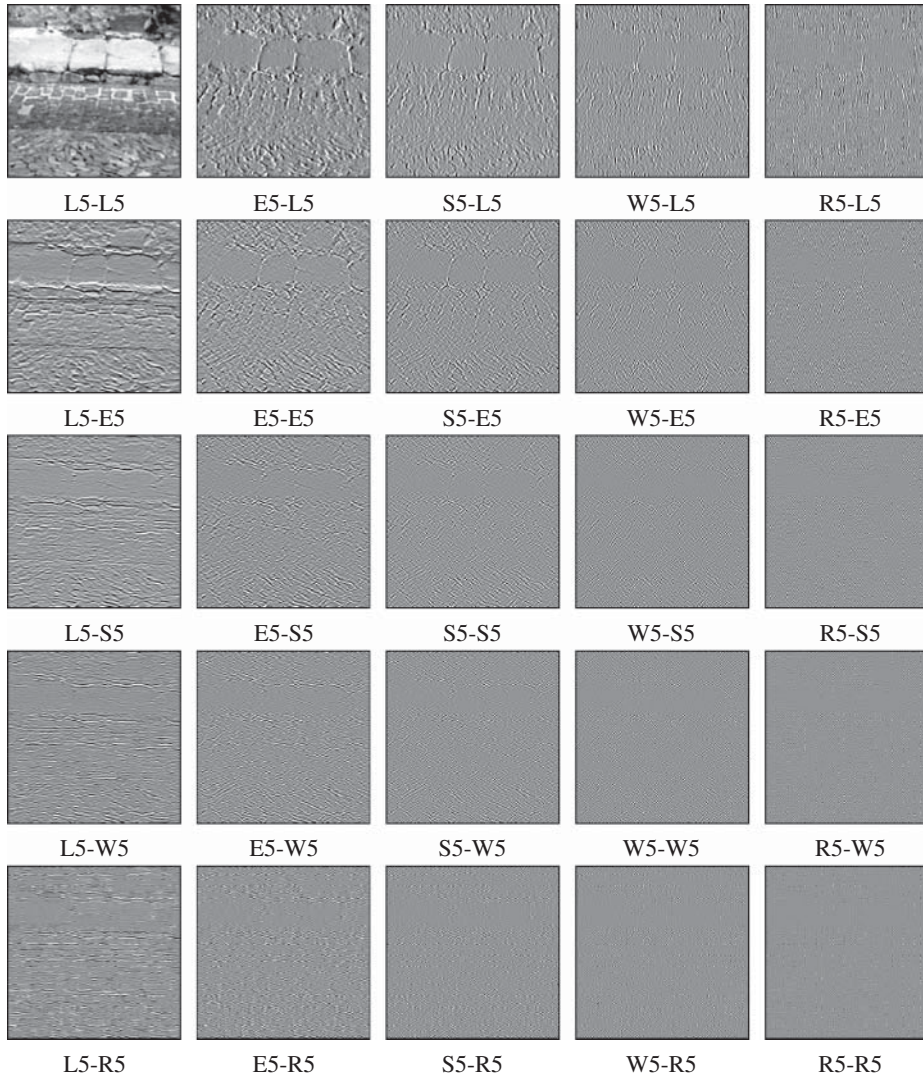
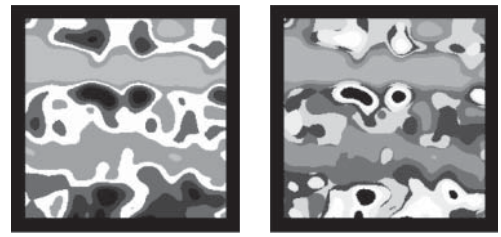


Figure 4.168 Outputs of the Laws' masks of size 5. The 2% extreme values of each panel were saturated and the rest were scaled to the range [0,255] for visualisation. Each panel $X-Y$ represents the output obtained when filter X is applied to the rows of the original image and filter Y is applied to the columns of the output.

Figure 4.169 Segmentation results obtained using the deterministic annealing algorithm for the energy features produced from the filtered outputs shown in Figure 4.168, using a Gaussian window of size 29×29 with standard deviation $\sigma = 7$. The deterministic annealing algorithm was run for two different values of the expected number K of distinct clusters. Source: Maria Petrou.

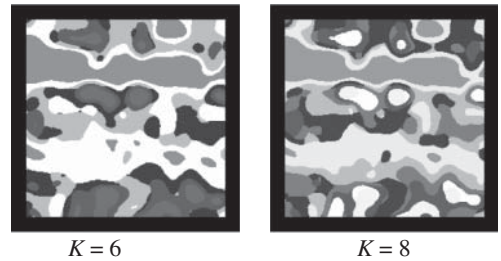


$K = 6$ $K = 8$

Example 4.229

Use the Laws' masks of Figure 4.165 of size 7×7 to compute features for the image of Figure 4.81 and segment it using the deterministic annealing algorithm.
The segmentation result is shown in Figure 4.170.

Figure 4.170 Segmentation results obtained using the deterministic annealing algorithm for the energy features computed from the filtered image outputs derived by convolving the image with the 7×7 Laws' masks. The energy features were constructed using a Gaussian window of size 29×29 with standard deviation $\sigma = 7$. K is the number of clusters specified for running the deterministic annealing algorithm. Source: Maria Petrou.



$K = 6$ $K = 8$

Example 4.230

Use all features you extracted in Examples 4.227- 4.229 to segment the image of Figure 4.81. Reduce the number of features first by performing principal component analysis and keeping only the first three principal components.
To perform principal component analysis, we consider that each pixel is a point in the 70-dimensional feature space. Let us denote the value of the lth feature at pixel (i,j) by $F_l(i,j)$. Values $F_l(i,j)$ for all values of l constitute the feature vector of pixel (i,j) and define its position in the 70-dimensional feature space. Let us say that the image is of size $M \times N$. Then the MN pixels of the image form a ***cloud*** *of points in the feature space. The idea is to use this cloud to identify axes centred in it so that the location of each pixel with respect to the cloud centre may be defined by using fewer numbers than 70.*

(Continued)

Example 4.230 (Continued)

First we identify the centre of the cloud by computing the average value of each feature over all pixels in the image:

$$\overline{F}_l = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}F_l(i,j). \tag{4.833}$$

Next we compute the covariance matrix of the data points that form the cloud. This is a 70×70 *matrix, expressing the correlation, if any, between any pair of features. An element* $C(k,l)$ *of this matrix is defined as follows:*

$$C(k,l) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[F_k(i,j) - \overline{F}_k][F_l(i,j) - \overline{F}_l]. \tag{4.834}$$

We compute the eigenvalues λ_m *of this matrix, arrange them in decreasing order (see Figure 4.171) and choose to keep the largest three of them. Then we compute the corresponding eigenvectors* v_1*,* v_2 *and* v_3*. Each such vector is 70 components long. The three principal components are the projections of each feature vector on the directions of the three eigenvectors:*

$$\begin{aligned}
P_1(i,j) &= v_1 \cdot [F(i,j) - \overline{F}]^T = \sum_{l=1}^{70} v_1(l)[F_l(i,j) - \overline{F}_l] \\
P_2(i,j) &= v_2 \cdot [F(i,j) - \overline{F}]^T = \sum_{l=1}^{70} v_2(l)[F_l(i,j) - \overline{F}_l] \\
P_3(i,j) &= v_3 \cdot [F(i,j) - \overline{F}]^T = \sum_{l=1}^{70} v_3(l)[F_l(i,j) - \overline{F}_l].
\end{aligned} \tag{4.835}$$

These values are now the features in terms of which image segmentation will take place in the new 3D feature space. Note that these features take positive and negative real values.

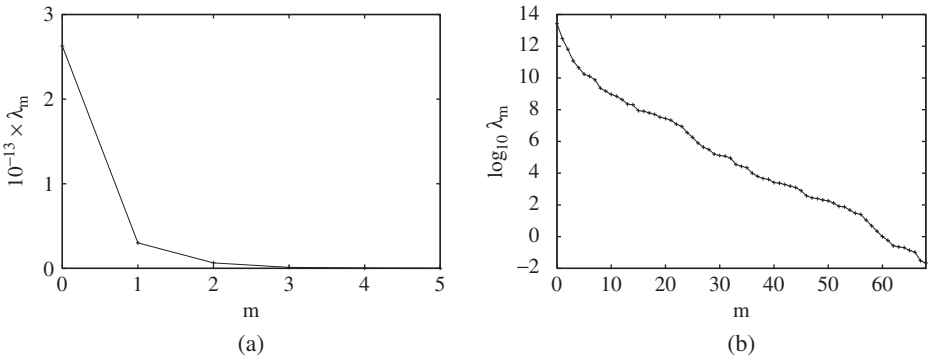


Figure 4.171 (a) The first six eigenvalues of matrix $C(k,l)$. (b) All eigenvalues of matrix $C(k,l)$ plotted in logarithmic scale.

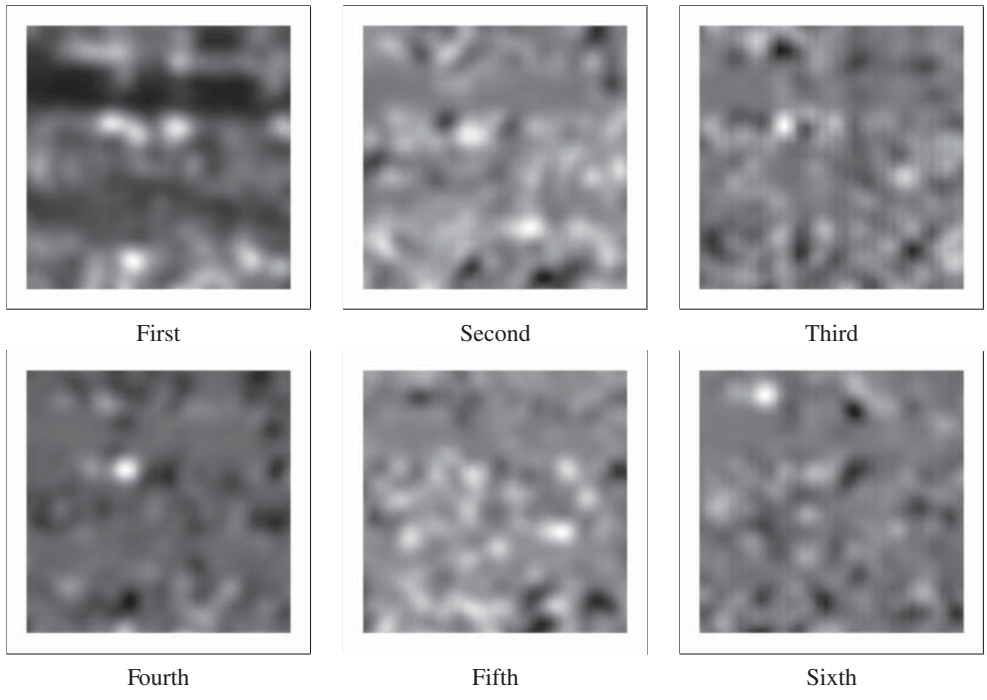
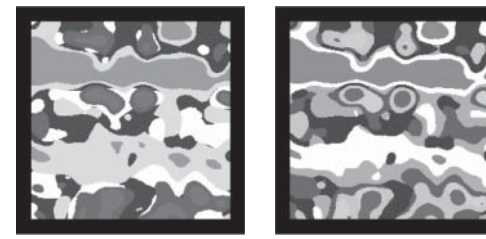


Figure 4.172 First six feature maps obtained using PCA. Each principal component map was individually scaled to the range [0,255] for presentation. Source: Maria Petrou.

Figure 4.173 Segmentation results obtained using the first three features constructed with PCA, shown in Figure 4.172, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.



Is there a systematic way to construct features that span the "what looks like where" space completely?

Yes, one may use the Walsh functions for that job. For an introduction to Walsh functions the reader is referred to Book I [75]. The Walsh functions may be used to construct a complete and orthonormal basis in terms of which any matrix of a certain size may be expanded. The Walsh matrices consist of only 1s and −1s, so they capture the binary characteristics of an image in terms of contrast of the local structure, depending on the size of the window we use to define what "local" means. For example, if "local" means a 3×3 neighbourhood around each pixel, the Walsh filters we must use must correspond to the three-sample long discrete versions of the Walsh function. If "local"

means a 5×5 neighbourhood, the Walsh filters we must use must correspond to the five-sample long discrete versions of the Walsh function, and so on. Each set of these filters expands the 3×3 or the 5×5 image patch in terms of a complete basis of elementary images. These elementary images may be constructed as the outer vector products of the discrete versions of the Walsh function. The discrete versions of the Walsh function of any order may be constructed by using the definition of the Walsh function by the recursive equation

$$W_{2j+q}(t) = (-1)^{\lfloor \frac{j}{2} \rfloor + q}[W_j(2t) + (-1)^{j+q} W_j(2t-1)] \tag{4.836}$$

where $\left\lfloor \frac{j}{2} \right\rfloor$ means the integer part of $j/2$, q is either 0 or 1, $j = 0, 1, 2, \ldots$, and

$$W_0(t) = \begin{cases} 1 \text{ for } 0 \le t < 1 \\ 0 \quad \text{elsewhere} \end{cases}. \tag{4.837}$$

In Book I [75] it was shown that the first three Walsh functions that may be computed from this equation are

$$W_1(t) = \begin{cases} -1 \text{ for } 0 \le t < \frac{1}{2} \\ 1 \text{ for } \frac{1}{2} \le t < 1 \end{cases} \tag{4.838}$$

$$W_2(t) = \begin{cases} -1 \text{ for } 0 \le t < \frac{1}{4} \\ 1 \text{ for } \frac{1}{4} \le t < \frac{3}{4} \\ -1 \text{ for } \frac{3}{4} \le t < 1 \end{cases} \tag{4.839}$$

$$W_3(t) = \begin{cases} 1 \text{ for } 0 \le t < \frac{1}{4} \\ -1 \text{ for } \frac{1}{4} \le t < \frac{1}{2} \\ 1 \text{ for } \frac{1}{2} \le t < \frac{3}{4} \\ -1 \text{ for } \frac{3}{4} \le t < 1 \end{cases}. \tag{4.840}$$

Each of these functions may be appropriately scaled and sampled to produce a vector which may be used to span the "what looks like where" space. These vectors are combined by outer vector product multiplication to form 2D elementary neighbourhoods, some examples of which are presented in Figures 4.174a and 4.175.

Example 4.231

Use the Walsh functions defined by equations (4.837)-(4.839) to construct basis vectors three samples long.

Since we are interested in three-sample long vectors, we scale the independent variable t in Equations (4.837)–(4.839) by multiplying it by 3, so we obtain:

$$W_0(t) = \begin{cases} 1 \text{ for } 0 \le t < 3 \\ 0 \text{ elsewhere} \end{cases} \tag{4.841}$$

$$W_1(t) = \begin{cases} -1 \text{ for } 0 \le t < \frac{3}{2} \\ 1 \text{ for } \frac{3}{2} \le t < 3 \end{cases} \tag{4.842}$$

$$W_2(t) = \begin{cases} -1 \text{ for } 0 \le t < \frac{3}{4} \\ 1 \text{ for } \frac{3}{4} \le t < \frac{9}{4} \\ -1 \text{ for } \frac{9}{4} \le t < 3 \end{cases}. \tag{4.843}$$

We then sample each function at the integer points only, so we produce the following vectors:

$$\begin{aligned} W_0^T &= (1,1,1) \\ W_1^T &= (-1,-1,1) \\ W_2^T &= (-1,1,1). \end{aligned} \tag{4.844}$$

Example 4.232

Combine the vectors produced in Example 4.231 in all possible ways in order to produce basis images of size 3×3 in terms of which you can analyse any 3×3 image neighbourhood. Present the elementary images you will construct in a numerical as well as in a pictorial way, as binary images.

$$\begin{aligned} W_0W_0^T &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}(1,1,1) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \\ W_0W_1^T &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}(-1,-1,1) = \begin{pmatrix} -1 & -1 & 1 \\ -1 & -1 & 1 \\ -1 & -1 & 1 \end{pmatrix} \\ W_0W_2^T &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}(-1,1,1) = \begin{pmatrix} -1 & 1 & 1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \end{aligned} \tag{4.845}$$

$$\begin{aligned} W_1W_0^T &= \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}(1,1,1) = \begin{pmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ 1 & 1 & 1 \end{pmatrix} \\ W_1W_1^T &= \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}(-1,-1,1) = \begin{pmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix} \\ W_1W_2^T &= \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}(-1,1,1) = \begin{pmatrix} 1 & -1 & -1 \\ 1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix} \end{aligned} \tag{4.846}$$

$$\begin{aligned} W_2W_0^T &= \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}(1,1,1) = \begin{pmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \\ W_2W_1^T &= \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}(-1,-1,1) = \begin{pmatrix} 1 & 1 & -1 \\ -1 & -1 & 1 \\ -1 & -1 & 1 \end{pmatrix} \\ W_2W_2^T &= \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}(-1,1,1) = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \end{pmatrix}. \end{aligned} \tag{4.847}$$

Figure 4.174a shows these images as binary shapes, with black corresponding to -1 and white to 1.

(Continued)

Example 4.232 (Continued)

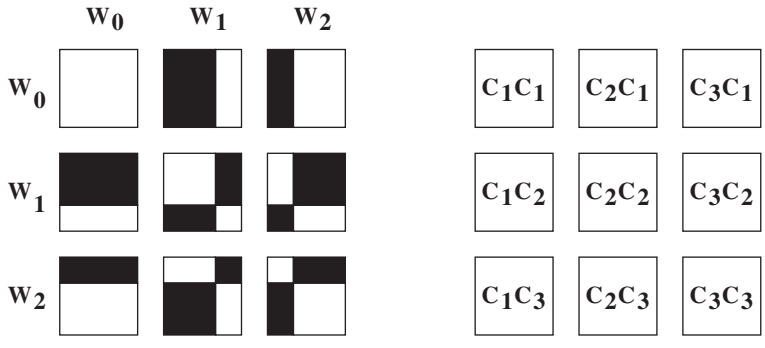


Figure 4.174 (a) If we process a 3×3 image with the inverse of the Walsh matrix, we expand it in terms of the nine elementary images shown here. Black means -1 and white means 1. These images were produced by taking the vector outer product of the Walsh vectors indicated above and on the left of the arrangement. Each of these images represents a particular type of local structure, which we may easily describe in linguistic terms. That is why we say that they constitute a basis for the "what looks like where" space for a 3×3 neighbourhood. (b) The coefficients of the expansion of an image neighbourhood in terms of these basis images may also be produced by pseudo-convolving the images with the filters indicated in the corresponding positions (i.e. applying them as they are, i.e. without reading them in reverse order), using the first one for horizontal pseudo-convolution and the second one for vertical pseudo-convolution. Here C_i stands for the ith column of the inverse of the Walsh matrix, i.e. matrix (4.874).

Example 4.233

Use Equations (4.836)–(4.839) to define the Walsh function $W_4(t)$.

To derive function $W_4(t)$, we must apply Equation (4.836) for $j = 2$ and $q = 0$:

$$W_4(t) = (-1)^{\lfloor \frac{2}{2} \rfloor + 0}[W_2(2t) + (-1)^{2+0} W_2(2t-1)] = -[W_2(2t) + W_2(2t-1)] \tag{4.848}$$

To decide what values $W_2(2t)$ and $W_2(2t-1)$ take, we must examine the values of their arguments, in conjunction with the definition of function $W_2(t)$ given by Equation (4.839):

$$\begin{aligned} &\text{For } 0 \le t < \frac{1}{8} \Rightarrow 0 \le 2t < \frac{1}{4} \text{ and } -1 \le 2t-1 < -\frac{3}{4} \\ &\quad \Rightarrow W_2(2t) = -1 \text{ and } W_2(2t-1) = 0 \Rightarrow W_4(t) = 1 \end{aligned} \tag{4.849}$$

$$\begin{aligned} &\text{For } \frac{1}{8} \le t < \frac{1}{4} \Rightarrow \frac{1}{4} \le 2t < \frac{1}{2} \text{ and } -\frac{3}{4} \le 2t-1 < -\frac{1}{2} \\ &\quad \Rightarrow W_2(2t) = 1 \text{ and } W_2(2t-1) = 0 \Rightarrow W_4(t) = -1 \end{aligned} \tag{4.850}$$

$$\begin{aligned} &\text{For } \frac{1}{4} \le t < \frac{3}{8} \Rightarrow \frac{1}{2} \le 2t < \frac{3}{4} \text{ and } -\frac{1}{2} \le 2t-1 < -\frac{1}{4} \\ &\quad \Rightarrow W_2(2t) = 1 \text{ and } W_2(2t-1) = 0 \Rightarrow W_4(t) = -1 \end{aligned} \tag{4.851}$$

$$\text{For } \frac{3}{8} \le t < \frac{4}{8} \Rightarrow \frac{3}{4} \le 2t < 1 \text{ and } -\frac{1}{4} \le 2t-1 < 0$$
$$\Rightarrow W_2(2t) = -1 \text{ and } W_2(2t-1) = 0 \Rightarrow W_4(t) = 1 \tag{4.852}$$

$$\text{For } \frac{1}{2} \le t < \frac{5}{8} \Rightarrow 1 \le 2t < \frac{5}{4} \text{ and } 0 \le 2t-1 < \frac{1}{4}$$
$$\Rightarrow W_2(2t) = 0 \text{ and } W_2(2t-1) = -1 \Rightarrow W_4(t) = 1 \tag{4.853}$$

$$\text{For } \frac{5}{8} \le t < \frac{6}{8} \Rightarrow \frac{5}{4} \le 2t < \frac{6}{4} \text{ and } \frac{1}{4} \le 2t-1 < \frac{1}{2}$$
$$\Rightarrow W_2(2t) = 0 \text{ and } W_2(2t-1) = 1 \Rightarrow W_4(t) = -1 \tag{4.854}$$

$$\text{For } \frac{6}{8} \le t < \frac{7}{8} \Rightarrow \frac{6}{4} \le 2t < \frac{7}{4} \text{ and } \frac{1}{2} \le 2t-1 < \frac{3}{4}$$
$$\Rightarrow W_2(2t) = 0 \text{ and } W_2(2t-1) = 1 \Rightarrow W_4(t) = -1 \tag{4.855}$$

$$\text{For } \frac{7}{8} \le t < 1 \Rightarrow \frac{7}{4} \le 2t < 2 \text{ and } \frac{3}{4} \le 2t-1 < 1$$
$$\Rightarrow W_2(2t) = 0 \text{ and } W_2(2t-1) = -1 \Rightarrow W_4(t) = 1. \tag{4.856}$$

Therefore, we conclude that:

$$W_4(t) = \begin{cases} 1 \text{ for } 0 \le t < \frac{1}{8} \\ -1 \text{ for } \frac{1}{8} \le t < \frac{3}{8} \\ 1 \text{ for } \frac{3}{8} \le t < \frac{5}{8} \\ -1 \text{ for } \frac{5}{8} \le t < \frac{7}{8} \\ 1 \text{ for } \frac{7}{8} \le t < 1 \end{cases}. \tag{4.857}$$

Example 4.234

Use Equations (4.837)–(4.840) and (4.857) to construct a set of elementary images in terms of which any 5×5 neighbourhood of an image may be expanded in the "what looks like where" space.

To construct these images, we multiply the independent variable by 5 and sample the functions at integer positions only. We obtain the following functions:

$$W_0(t) = \begin{cases} 1 \text{ for } 0 \le t < 5 \\ 0 \text{ elsewhere} \end{cases} \tag{4.858}$$

$$W_1(t) = \begin{cases} -1 \text{ for } 0 \le t < \frac{5}{2} \\ 1 \text{ for } \frac{5}{2} \le t < 5 \end{cases} \tag{4.859}$$

$$W_2(t) = \begin{cases} -1 \text{ for } 0 \le t < \frac{5}{4} \\ 1 \text{ for } \frac{5}{4} \le t < \frac{15}{4} \\ -1 \text{ for } \frac{15}{4} \le t < 5 \end{cases} \tag{4.860}$$

(Continued)

Example 4.234 (Continued)

$$W_3(t) = \begin{cases} 1 \text{ for } 0 \le t < \frac{5}{4} \\ -1 \text{ for } \frac{5}{4} \le t < \frac{5}{2} \\ 1 \text{ for } \frac{5}{2} \le t < \frac{15}{4} \\ -1 \text{ for } \frac{15}{4} \le t < 5 \end{cases} \tag{4.861}$$

$$W_4(t) = \begin{cases} 1 \text{ for } 0 \le t < \frac{5}{8} \\ -1 \text{ for } \frac{5}{8} \le t < \frac{15}{8} \\ 1 \text{ for } \frac{15}{8} \le t < \frac{25}{8} \\ -1 \text{ for } \frac{25}{8} \le t < \frac{35}{8} \\ 1 \text{ for } \frac{35}{8} \le t < 5 \end{cases}. \tag{4.862}$$

We then sample each function at the integer points only (i.e. at $t = 0, 1, 2, 3, 4$), to produce the following vectors:

$$\begin{aligned} W_0^T &= (1,1,1,1,1) \\ W_1^T &= (-1,-1,-1,1,1) \\ W_2^T &= (-1,-1,1,1,-1) \\ W_3^T &= (1,1,-1,1,-1) \\ W_4^T &= (1,-1,1,1,-1). \end{aligned} \tag{4.863}$$

These vectors may be combined in all possible pairs to produce 2D basis images for the "what looks like where" space for 5×5 neighbourhoods. There are 25 such images:

$$W_0W_0^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad W_0W_1^T = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \end{bmatrix}$$

$$W_0W_2^T = \begin{bmatrix} -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \end{bmatrix} \quad W_0W_3^T = \begin{bmatrix} 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

$$W_0W_4^T = \begin{bmatrix} 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \end{bmatrix} W_1W_0^T = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$W_1W_1^T = \begin{bmatrix} 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \end{bmatrix} W_1W_2^T = \begin{bmatrix} 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

$$
W_1W_3^T = \begin{bmatrix} -1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \end{bmatrix}
W_1W_4^T = \begin{bmatrix} -1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \end{bmatrix}
$$

$$
W_2W_0^T = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}
W_2W_1^T = \begin{bmatrix} 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 \end{bmatrix}
$$

$$
W_2W_2^T = \begin{bmatrix} 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 \end{bmatrix}
W_2W_3^T = \begin{bmatrix} -1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & 1 \end{bmatrix}
$$

$$
W_2W_4^T = \begin{bmatrix} -1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 \end{bmatrix}
W_3W_0^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}
$$

$$
W_3W_1^T = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 \end{bmatrix}
W_3W_2^T = \begin{bmatrix} -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 \end{bmatrix}
$$

$$
W_3W_3^T = \begin{bmatrix} 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & 1 \end{bmatrix}
W_3W_4^T = \begin{bmatrix} 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 \end{bmatrix}
$$

$$
W_4W_0^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}
W_4W_1^T = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 \end{bmatrix}
$$

$$
W_4W_2^T = \begin{bmatrix} -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 \end{bmatrix}
W_4W_3^T = \begin{bmatrix} 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & 1 \end{bmatrix}
$$

$$
W_4W_4^T = \begin{bmatrix} 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 \end{bmatrix}. \tag{4.864}
$$

(Continued)

Example 4.234 (Continued)

Figure 4.175 shows these 25 images in binary form.

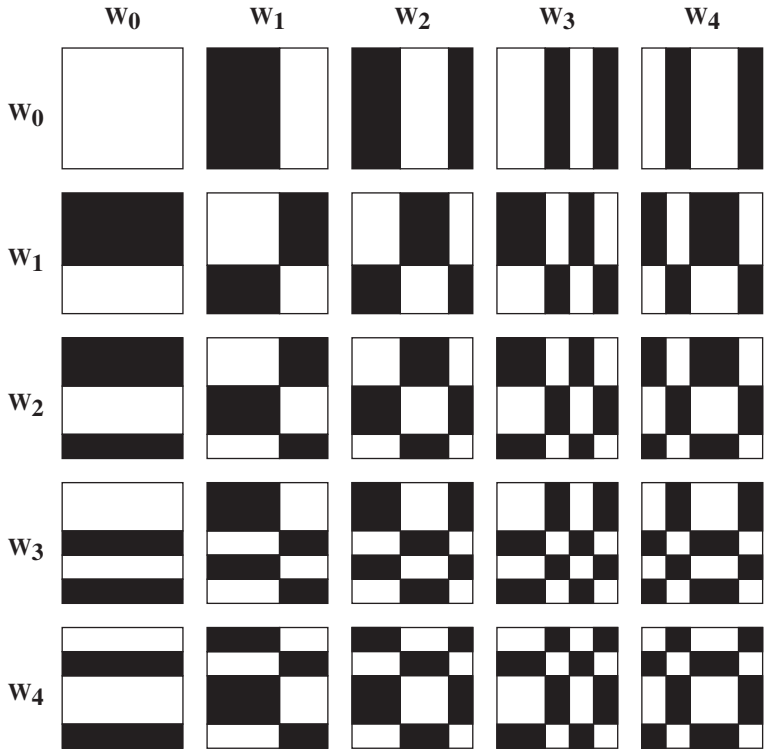


Figure 4.175 If we process a sub-image of size 5×5 with the inverse Walsh matrix of the same size, the numbers we compute are the coefficients of the expansion of the original sub-image in terms of these elementary images. We may say that these 25 elementary neighbourhoods span the 5×5 "what looks like where" space. Black means -1 and white means 1. These images were produced by taking the vector outer product of the Walsh vectors indicated above and on the left.

How can we expand a local image neighbourhood in terms of the Walsh elementary images?

When we wish to expand an image or a sub-image in terms of Walsh elementary functions of **even** size, all we have to do is to write the Walsh vectors we derived one under the other as **rows** of a transformation matrix W, and multiply the image (or sub-image) f from left with it and from right with its transpose:

$$g = WfW^T. \tag{4.865}$$

In this expression, the transformation matrix W, the input image neighbourhood f and the output transform g are all of the same size, say $N \times N$, where N is even. An even size transformation matrix constructed from Walsh functions is orthogonal and so its inverse is its transpose. Then the original image f may be recovered from its Walsh transform g very easily, by multiplying both sides of the above equation with W^T from the left and W from the right:

$$W^T gW = f. \tag{4.866}$$

In Book I [75] it was shown that this equation may be written as

$$f = g_{ij} W_i W_j^T \tag{4.867}$$

where g_{ij} is the ij element of transform g, and it is the coefficient that multiplies the elementary basis image formed by the outer product of Walsh vectors W_i and W_j^T. These coefficients may be thought of as local features of the image computed from its sub-image (= neighbourhood) f. The trouble is, however, that if this neighbourhood is of even size, it does not have a centre, and so there is an ambiguity as to which pixel the constructed features actually refer to. Usually some convention is adopted as to which pixel will be endowed with the computed features. On the other hand, if we insist on using image windows of odd size, such as 3×3 or 5×5, we must use odd-sized Walsh vectors, which yield an odd-sized Walsh transformation matrix. This matrix is produced by writing the Walsh vectors one under the other as rows. For example, for the case of a 5×5 neighbourhood, this matrix is produced by writing vectors (4.863) one under the other:

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \end{bmatrix}. \tag{4.868}$$

Such a matrix, however, is no longer orthogonal. We may check that very easily. For a matrix to be orthogonal, the dot product of any two of its row vectors must be zero. Only in that case the product of the matrix with its transpose will produce the unit matrix. This is not the case if we multiply two odd-sized vectors that consist of -1s and 1s only: we shall never have an equal number of -1s and $+1$s to cancel out. So, the transformation matrix created by writing those vectors one under the other as rows is not orthogonal, and in order to invert Equation (4.865), we must make use of the inverse of matrix W, which has to be worked out. Then we shall have:

$$W^{-1}g(W^T)^{-1} = f. \tag{4.869}$$

Note that this inversion leads to the representation of image f as a linear superposition of elementary images created by the **columns** of the **inverse** of matrix W, and not the **rows** of matrix W. Such elementary images do not necessarily have any meaning in terms of local image structure as we perceive it and as we would like to express it. On the other hand, the elementary images created by the rows of matrix W (examples of which are shown in Figures 4.174a and 4.175), can be seen to express perceptually identifiable structures: contrast between left and right of the image, cornerness, contrast between top and bottom of the image, stripiness, and so on. So, to circumvent this problem, we may use the inverse of matrix W, i.e. matrix W^{-1}, or $(W^{-1})^T$ to process the image. Then we have:

$$g = (W^{-1})^T f W^{-1}. \tag{4.870}$$

Let us multiply now this equation from the left with $((W^{-1})^T)^{-1}$ and from the right with W, to obtain:

$$((W^{-1})^T)^{-1}gW = ((W^{-1})^T)^{-1}(W^{-1})^T f W^{-1}W = f. \tag{4.871}$$

Note that $W^{-1}W = I \Rightarrow (W^{-1}W)^T = I^T = I \Rightarrow W^T(W^{-1})^T = I$. In other words, the inverse of the transpose of a matrix is the transpose of its inverse. So, the inverse of the transpose of W^{-1}, i.e. the inverse of $(W^{-1})^T$, is the transpose of W. This means that Equation (4.871) may be written as

$$f = W^T g W. \tag{4.872}$$

This is the same as Equation (4.866) which implies that the neighbourhood f by this transformation has been expanded in terms of the vector outer products created by the Walsh vectors in all possible combinations. These are the elementary images we wanted to use because they make sense to us as to what sort of local structure they represent (see Figures 4.174a and 4.175). The coefficients of this expansion, i.e. the elements of matrix g, are features that characterise the central pixel of the neighbourhood.

Example 4.235

Derive the inverse of transformation matrix W for the case of image patches of size 3×3.

From Equations (4.844) we have that the 3×3 Walsh transformation matrix is

$$W = \begin{pmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix}. \tag{4.873}$$

The determinant of this matrix is $1 \times [(-1) \times 1 - 1 \times 1] - 1 \times [(-1) \times 1 - 1 \times (-1)] + 1 \times [(-1) \times 1 - (-1) \times (-1)] = -2 - 2 = -4$. Therefore, by applying the formula for matrix inversion, we obtain:

$$W^{-1} = \frac{1}{-4} \begin{pmatrix} -2 & 0 & -2 \\ 0 & 2 & -2 \\ 2 & -2 & 0 \end{pmatrix}^T \Rightarrow W^{-1} = \begin{pmatrix} 0.5 & 0 & -0.5 \\ 0 & -0.5 & 0.5 \\ 0.5 & 0.5 & 0 \end{pmatrix}. \tag{4.874}$$

Example 4.236

Derive the inverse of transformation matrix W for the case of image patches of size 5×5.

The determinant of the Walsh matrix W of size 5×5 given by Equation (4.868) is 32.

The inverse matrix is:

$$W^{-1} = \begin{bmatrix} 0 & 0 & -0.5 & 0 & 0.5 \\ 0.25 & -0.25 & 0.25 & 0.25 & -0.5 \\ 0.25 & -0.25 & 0.25 & -0.25 & 0 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0 \\ 0.25 & 0.25 & -0.25 & -0.25 & 0 \end{bmatrix}. \tag{4.875}$$

Example 4.237

Derive the elementary images in terms of which an original 3×3 image is expanded if it is processed by matrix W according to Equation (4.865).

*These elementary images are the vector outer products of the **columns** of the matrix that multiplies the transformation matrix from the left in Equation (4.869), i.e. matrix W^{-1}. Matrix W^{-1} is given by Equation (4.874). So, the elementary images are:*

$$\begin{pmatrix} 0.5 \\ 0 \\ 0.5 \end{pmatrix} (0.5, 0, 0.5) = \begin{pmatrix} 0.25 & 0 & 0.25 \\ 0 & 0 & 0 \\ 0.25 & 0 & 0.25 \end{pmatrix}$$

$$\begin{pmatrix} 0.5 \\ 0 \\ 0.5 \end{pmatrix} (0, -0.5, 0.5) = \begin{pmatrix} 0 & -0.25 & 0.25 \\ 0 & 0 & 0 \\ 0 & -0.25 & 0.25 \end{pmatrix}$$

$$\begin{pmatrix} 0.5 \\ 0 \\ 0.5 \end{pmatrix} (-0.5, 0.5, 0) = \begin{pmatrix} -0.25 & 0.25 & 0 \\ 0 & 0 & 0 \\ -0.25 & 0.25 & 0 \end{pmatrix}$$

$$
\begin{pmatrix} 0 \\ -0.5 \\ 0.5 \end{pmatrix}(0.5,0,0.5) = \begin{pmatrix} 0 & 0 & 0 \\ -0.25 & 0 & -0.25 \\ 0.25 & 0 & 0.25 \end{pmatrix}
$$

$$
\begin{pmatrix} 0 \\ -0.5 \\ 0.5 \end{pmatrix}(0,-0.5,0.5) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.25 & -0.25 \\ 0 & -0.25 & 0.25 \end{pmatrix}
$$

$$
\begin{pmatrix} 0 \\ -0.5 \\ 0.5 \end{pmatrix}(-0.5,0.5,0) = \begin{pmatrix} 0 & 0 & 0 \\ 0.25 & -0.25 & 0 \\ -0.25 & 0.25 & 0 \end{pmatrix}
$$

$$
\begin{pmatrix} -0.5 \\ 0.5 \\ 0 \end{pmatrix}(0.5,0,0.5) = \begin{pmatrix} -0.25 & 0 & -0.25 \\ 0.25 & 0 & 0.25 \\ 0 & 0 & 0 \end{pmatrix}
$$

$$
\begin{pmatrix} -0.5 \\ 0.5 \\ 0 \end{pmatrix}(0,-0.5,0.5) = \begin{pmatrix} 0 & 0.25 & -0.25 \\ 0 & -0.25 & 0.25 \\ 0 & 0 & 0 \end{pmatrix}
$$

$$
\begin{pmatrix} -0.5 \\ 0.5 \\ 0 \end{pmatrix}(-0.5,0.5,0) = \begin{pmatrix} 0.25 & -0.25 & 0 \\ -0.25 & 0.25 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{4.876}
$$

These elementary images are shown in Figure 4.176a, where black stands for negative, grey for 0 and white for positive.

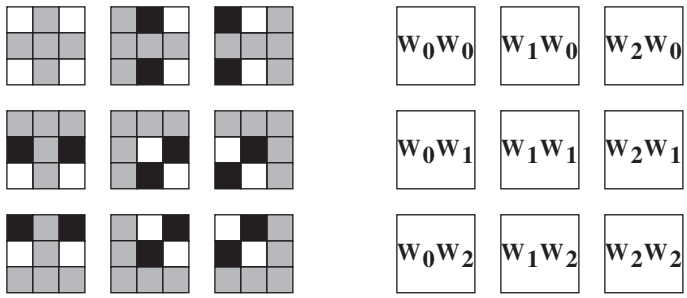


Figure 4.176 (a) The nine elementary neighbourhoods in terms of which the 3×3 Walsh transform expands the "what looks like where" space. Black means -0.25, grey means 0 and white means 0.25. These images were produced by taking the vector outer product of the column vectors of the inverse Walsh matrix. The local structure expressed by these images does not correspond to the local elementary structure we would probably choose intuitively. (b) The coefficients of the expansion of an image neighbourhood in terms of these basis images may also be produced by pseudo-convolving the images with the filters indicated in the corresponding positions, using the first one for horizontal pseudo-convolution and the second one for vertical pseudo-convolution. Here W_i stands for the ith Walsh filter given by Equation (4.844). (Reminder: pseudo-convolution means that the filter is applied as read, not inverted first.)

Example 4.238

Derive the elementary images in terms of which an original 5×5 image is expanded if it is processed by matrix W according to Equation 4.865.

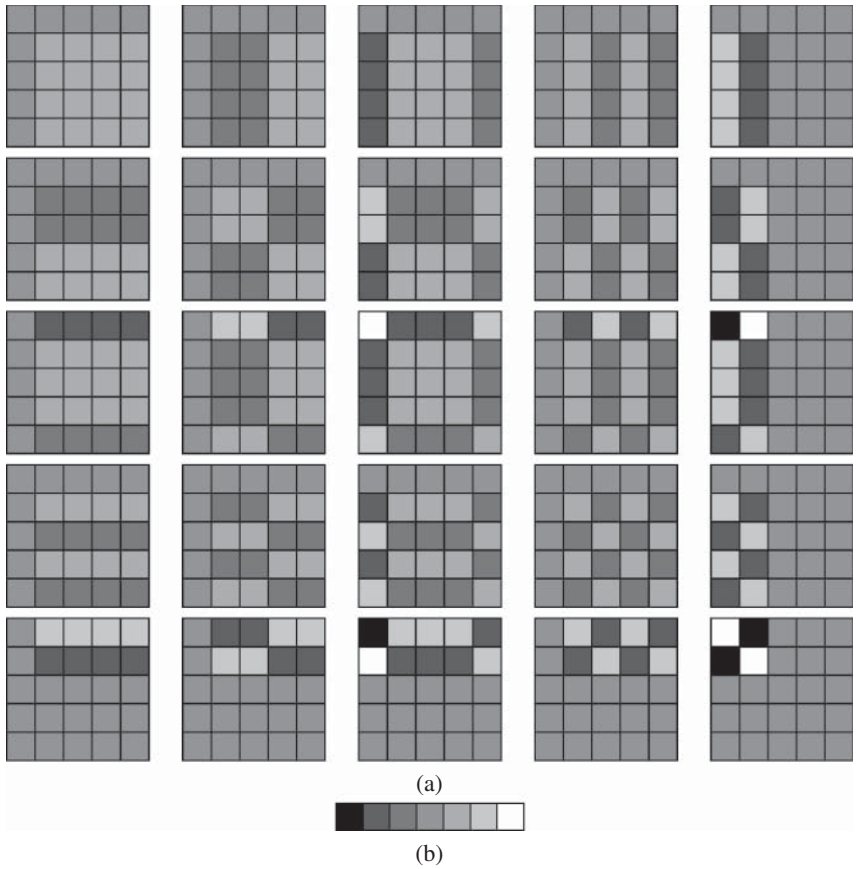


Figure 4.177 (a) The 25 elementary neighbourhoods in terms of which the 5×5 Walsh transform expands the "what looks like where" space. These images were produced by taking the vector outer product of the column vectors of the inverse Walsh matrix. (b) Grey level code used. From left to right: -0.25, -0.125, -0.0625, 0, 0.0625, 0.125 and 0.25.

To do that we take the columns of the inverse of matrix W computed in Example 4.236 and treat them as vectors. Each vector is combined with all other vectors to produce their vector outer product. The 25 elementary images produced that way are presented in Figure 4.177, where the values of the pixels are grey level coded.

Can we use convolution to compute the coefficients of the expansion of a sub-image in terms of a set of elementary images?

Yes, we can. The convolution filters we must use should be the columns of the expansion matrix. This is best demonstrated with an example. Let us consider a 6×6 image:

$$G = \begin{pmatrix} g_{11} & g_{12} & g_{13} & g_{14} & g_{15} & g_{16} \\ g_{21} & g_{22} & g_{23} & g_{24} & g_{25} & g_{26} \\ g_{31} & g_{32} & g_{33} & g_{34} & g_{35} & g_{36} \\ g_{41} & g_{42} & g_{43} & g_{44} & g_{45} & g_{46} \\ g_{51} & g_{52} & g_{53} & g_{54} & g_{55} & g_{56} \\ g_{61} & g_{62} & g_{63} & g_{64} & g_{65} & g_{66} \end{pmatrix}. \tag{4.877}$$

Let us consider the 3×3 neighbourhood patch around pixel g_{44}, and let us expand it in terms of the Walsh basis images of Figure 4.174a. We must process it from the right with matrix W^{-1} given by Equation (4.874) and from the left by the transpose of the same matrix:

$$\begin{pmatrix} 0.5 & 0 & -0.5 \\ 0 & -0.5 & 0.5 \\ 0.5 & 0.5 & 0 \end{pmatrix}^T \begin{pmatrix} g_{33} & g_{34} & g_{35} \\ g_{43} & g_{44} & g_{45} \\ g_{53} & g_{54} & g_{55} \end{pmatrix} \begin{pmatrix} 0.5 & 0 & -0.5 \\ 0 & -0.5 & 0.5 \\ 0.5 & 0.5 & 0 \end{pmatrix} =$$

$$\begin{pmatrix} 0.5 & 0 & 0.5 \\ 0 & -0.5 & 0.5 \\ -0.5 & 0.5 & 0 \end{pmatrix} \begin{pmatrix} 0.5g_{33} + 0.5g_{35} & -0.5g_{34} + 0.5g_{35} & -0.5g_{33} + 0.5g_{34} \\ 0.5g_{43} + 0.5g_{45} & -0.5g_{44} + 0.5g_{45} & -0.5g_{43} + 0.5g_{44} \\ 0.5g_{53} + 0.5g_{55} & -0.5g_{54} + 0.5g_{55} & -0.5g_{53} + 0.5g_{54} \end{pmatrix} =$$

$$\begin{pmatrix} 0.25g_{33} + 0.25g_{35} + 0.25g_{53} + 0.25g_{55} & -0.25g_{34} + 0.25g_{35} - 0.25g_{54} + 0.25g_{55} \\ -0.25g_{43} - 0.25g_{45} + 0.25g_{53} + 0.25g_{55} & 0.25g_{44} - 0.25g_{45} - 0.25g_{54} + 0.25g_{55} \\ -0.25g_{33} - 0.25g_{35} + 0.25g_{43} + 0.25g_{45} & 0.25g_{34} - 0.25g_{35} - 0.25g_{44} + 0.2.5g_{45} \end{pmatrix}$$

$$\begin{pmatrix} -0.25g_{33} + 0.25g_{34} - 0.25g_{53} + 0.25g_{54} \\ 0.25g_{43} - 0.25g_{44} - 0.25g_{53} + 0.25g_{54} \\ 0.25g_{33} - 0.25g_{34} - 0.25g_{43} + 0.25g_{44} \end{pmatrix}. \tag{4.878}$$

We notice that the first multiplication we perform produces results that could have been produced by pseudo-convolving the image **horizontally** with the **columns** of matrix W^{-1} (i.e. treating each column as a filter applied as read, without inverting its order). The second set of results could have been produced if we were pseudo-convolving the output of the previous convolution with the **columns** again of matrix W^{-1}, using the same convention, but this time vertically. So, the coefficients of the expansion of each image patch in terms of the elementary images produced by the Walsh functions, may be extracted by pseudo-convolving the image horizontally and then vertically with the columns of the inverse of the Walsh transformation matrix, used as filters. Which convolution filters must be used in what order to produce the coefficients of the expansion of a local image neighbourhood in terms of the elementary images of Figure 4.174a, is shown schematically in Figure 4.174b.

Example 4.239

Verify that the coefficients of the expansion computed in Equation (4.878) are indeed the coefficients of the expansion of the original patch in terms of the images shown in Figure 4.174a.

Each element of matrix (4.878) is the coefficient with which we must multiply the corresponding basis image in order to recover the original sub-image. The basis images were computed in Example 4.232 and they are shown pictorially in Figure 4.174. Synthesising the image from this expansion yields:

$$
\begin{aligned}
&(0.25g_{33}+0.25g_{35}+0.25g_{53}+0.25g_{55})\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}\\
&+(-0.25g_{34}+0.25g_{35}-0.25g_{54}+0.25g_{55})\begin{pmatrix}-1&-1&1\\-1&-1&1\\-1&-1&1\end{pmatrix}\\
&+(-0.25g_{33}+0.25g_{34}-0.25g_{53}+0.25g_{54})\begin{pmatrix}-1&1&1\\-1&1&1\\-1&1&1\end{pmatrix}\\
&+(-0.25g_{43}-0.25g_{45}+0.25g_{53}+0.25g_{55})\begin{pmatrix}-1&-1&-1\\-1&-1&-1\\1&1&1\end{pmatrix}\\
&+(0.25g_{44}-0.25g_{45}-0.25g_{54}+0.25g_{55})\begin{pmatrix}1&1&-1\\1&1&-1\\-1&-1&1\end{pmatrix}\\
&+(0.25g_{43}-0.25g_{44}-0.25g_{53}+0.25g_{54})\begin{pmatrix}1&-1&-1\\1&-1&-1\\-1&1&1\end{pmatrix}\\
&+(-0.25g_{33}-0.25g_{35}+0.25g_{43}+0.25g_{45})\begin{pmatrix}-1&-1&-1\\1&1&1\\1&1&1\end{pmatrix}\\
&+(0.25g_{34}-0.25g_{35}-0.25g_{44}+0.25g_{45})\begin{pmatrix}1&1&-1\\-1&-1&1\\-1&-1&1\end{pmatrix}\\
&+(0.25g_{33}-0.25g_{34}-0.25g_{43}+0.25g_{44})\begin{pmatrix}1&-1&-1\\-1&1&1\\-1&1&1\end{pmatrix}\\
&=\begin{pmatrix}g_{33}&g_{34}&g_{35}\\g_{43}&g_{44}&g_{45}\\g_{53}&g_{54}&g_{55}\end{pmatrix}.
\end{aligned}
\tag{4.879}
$$

Example 4.240

Identify the elementary images in terms of which each 3×3 neighbourhood of an image is expanded when the image is convolved with the three-tap Laws' masks.

The convolution of an image by all nine combinations of the three-tap Laws' filters, applied in the vertical and horizontal dimensions, produces for each pixel nine numbers. These numbers are the coefficients of the expansion of the 3×3 neighbourhood, around the pixel, in terms of elementary images that may be constructed as follows. The transformation matrix that corresponds to the Laws' masks used is created by writing the masks one under the other:

$$\begin{pmatrix} 1 & 2 & 1 \\ -1 & 0 & 1 \\ -1 & 2 & -1 \end{pmatrix}. \tag{4.880}$$

The inverse of this matrix is:

$$\begin{pmatrix} 0.25 & -0.5 & -0.25 \\ 0.25 & 0 & 0.25 \\ 0.25 & 0.5 & -0.25 \end{pmatrix}. \tag{4.881}$$

The elementary images in terms of which the convolutions with Laws' masks expand each local neighbourhood are formed by the vector outer products of the columns of the above matrix:

$$\begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \end{pmatrix} \begin{pmatrix} 0.25 & 0.25 & 0.25 \end{pmatrix} = \begin{pmatrix} 0.0625 & 0.0625 & 0.0625 \\ 0.0625 & 0.0625 & 0.0625 \\ 0.0625 & 0.0625 & 0.0625 \end{pmatrix}$$

$$\begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \end{pmatrix} \begin{pmatrix} -0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} -0.125 & 0 & 0.125 \\ -0.125 & 0 & 0.125 \\ -0.125 & 0 & 0.125 \end{pmatrix}$$

$$\begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \end{pmatrix} \begin{pmatrix} -0.25 & 0.25 & -0.25 \end{pmatrix} = \begin{pmatrix} -0.0625 & 0.0625 & -0.0625 \\ -0.0625 & 0.0625 & -0.0625 \\ -0.0625 & 0.0625 & -0.0625 \end{pmatrix}$$

$$\begin{pmatrix} -0.5 \\ 0 \\ 0.5 \end{pmatrix} \begin{pmatrix} 0.25 & 0.25 & 0.25 \end{pmatrix} = \begin{pmatrix} -0.125 & -0.125 & -0.125 \\ 0 & 0 & 0 \\ 0.125 & 0.125 & 0.125 \end{pmatrix}$$

$$\begin{pmatrix} -0.5 \\ 0 \\ 0.5 \end{pmatrix} \begin{pmatrix} -0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.25 & 0 & -0.25 \\ 0 & 0 & 0 \\ -0.25 & 0 & 0.25 \end{pmatrix}$$

$$\begin{pmatrix} -0.5 \\ 0 \\ 0.5 \end{pmatrix} \begin{pmatrix} -0.25 & 0.25 & -0.25 \end{pmatrix} = \begin{pmatrix} 0.125 & -0.125 & 0.125 \\ 0 & 0 & 0 \\ -0.125 & 0.125 & -0.125 \end{pmatrix}$$

$$\begin{pmatrix} -0.25 \\ 0.25 \\ -0.25 \end{pmatrix} \begin{pmatrix} 0.25 & 0.25 & 0.25 \end{pmatrix} = \begin{pmatrix} -0.0625 & -0.0625 & -0.0625 \\ 0.0625 & 0.0625 & 0.0625 \\ -0.0625 & -0.0625 & -0.0625 \end{pmatrix}$$

$$\begin{pmatrix} -0.25 \\ 0.25 \\ -0.25 \end{pmatrix} \begin{pmatrix} -0.5 & 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.125 & 0 & -0.125 \\ -0.125 & 0 & 0.125 \\ 0.125 & 0 & -0.125 \end{pmatrix}$$

(Continued)

Example 4.240 (Continued)

$$\begin{pmatrix} -0.25 \\ 0.25 \\ -0.25 \end{pmatrix} \begin{pmatrix} -0.25 & 0.25 & -0.25 \end{pmatrix} = \begin{pmatrix} -0.0625 & 0.0625 & -0.0625 \\ 0.0625 & -0.0625 & 0.0625 \\ -0.0625 & 0.0625 & -0.0625 \end{pmatrix}. \tag{4.882}$$

These elementary images are shown pictorially in Figure 4.178a.

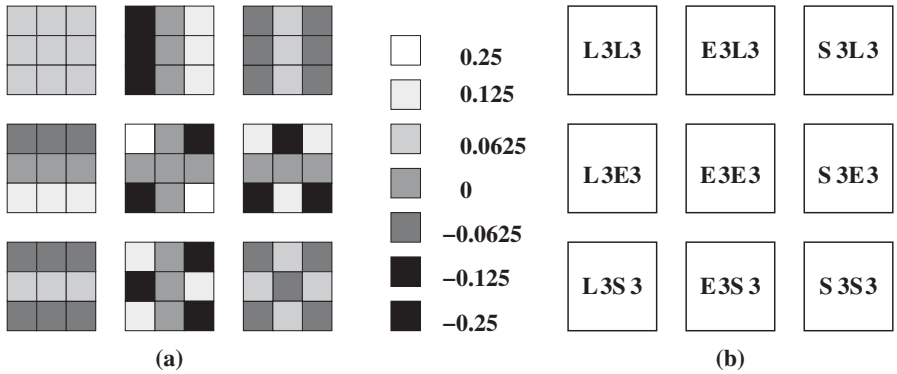


Figure 4.178 (a) Convolution of an image along the vertical and horizontal directions with all combinations of three-tap Laws' masks produces nine outputs that are the coefficients of the expansion of each image 3×3 patch in terms of these basis images. (b) The coefficients of the expansion of an image neighbourhood in terms of these basis images may also be produced by pseudo-convolving the images with the filters indicated in the corresponding positions, using the first one for horizontal convolution and the second one for vertical convolution. These are Laws' masks as defined in Figure 4.163.

Example 4.241

Use the three-tap Laws' masks to produce nine elementary images.

These elementary images are computed as follows:

$$\begin{aligned} L3L3^T &= \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} (1,2,1) = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \\ L3E3^T &= \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} (-1,0,1) = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \\ L3S3^T &= \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} (-1,2,-1) = \begin{pmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{pmatrix} \\ E3L3^T &= \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} (1,2,1) = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \end{aligned} \tag{4.883}$$

$$E3E3^T = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}(-1,0,1) = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

$$E3S3^T = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}(-1,2,-1) = \begin{pmatrix} 1 & -2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{pmatrix} \tag{4.884}$$

$$S3L3^T = \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix}(1,2,1) = \begin{pmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{pmatrix}$$

$$S3E3^T = \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix}(-1,0,1) = \begin{pmatrix} 1 & 0 & -1 \\ -2 & 0 & 2 \\ 1 & 0 & -1 \end{pmatrix}$$

$$S3S3^T = \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix}(-1,2,-1) = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}. \tag{4.885}$$

They are shown pictorially in Figure 4.179a.

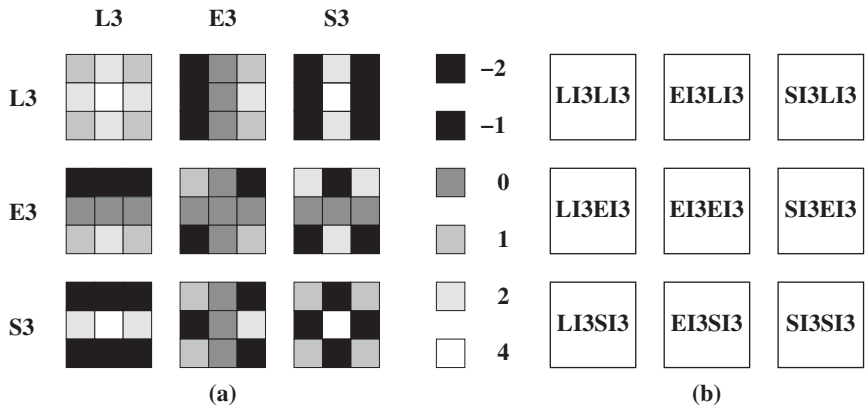


Figure 4.179 (a) Laws' masks of size 3 may be combined in all possible ways to form 3×3 basis images in terms of which any 3×3 image patch may be expanded. The 1D masks indicated on the left are combined with the 1D masks indicated along the top to form the corresponding elementary image. The pixel values of these elementary images are coded in tones of grey. (b) The coefficients of the expansion of an image neighbourhood in terms of these basis images may also be produced by pseudo-convolving the images with the filters indicated in the corresponding positions, using the first one for horizontal convolution and the second one for vertical convolution. These are the masks given by Equation (4.886).

Example 4.242

Derive the filters you should use in order to express each 3×3 image patch as a linear superposition of the elementary images shown in Figure 4.179a.

The elementary images in Figure 4.179a were created from the three-tap Laws' masks. The transformation matrix formed by these masks is matrix (4.880). The inverse of this matrix is (4.881). If

(Continued)

Example 4.242 (Continued)

we want to expand each 3×3 *image neighbourhood in terms of the basis images of Figure 4.179a, we must pseudo-convolve the images using as filters the columns of matrix (4.881), in all possible combinations along the vertical and horizontal directions, as indicated in Figure 4.179b:*

$$\begin{aligned} LI3 &\equiv (0.25, 0.25, 0.25) \\ EI3 &\equiv (-0.5, 0, 0.5) \\ SI3 &\equiv (-0.25, 0.25, -0.25). \end{aligned} \tag{4.886}$$

Example 4.243

Expand the 3×3 **neighbourhoods of the image of Figure 4.81 in terms of the basis images shown in Figure 4.174a.**

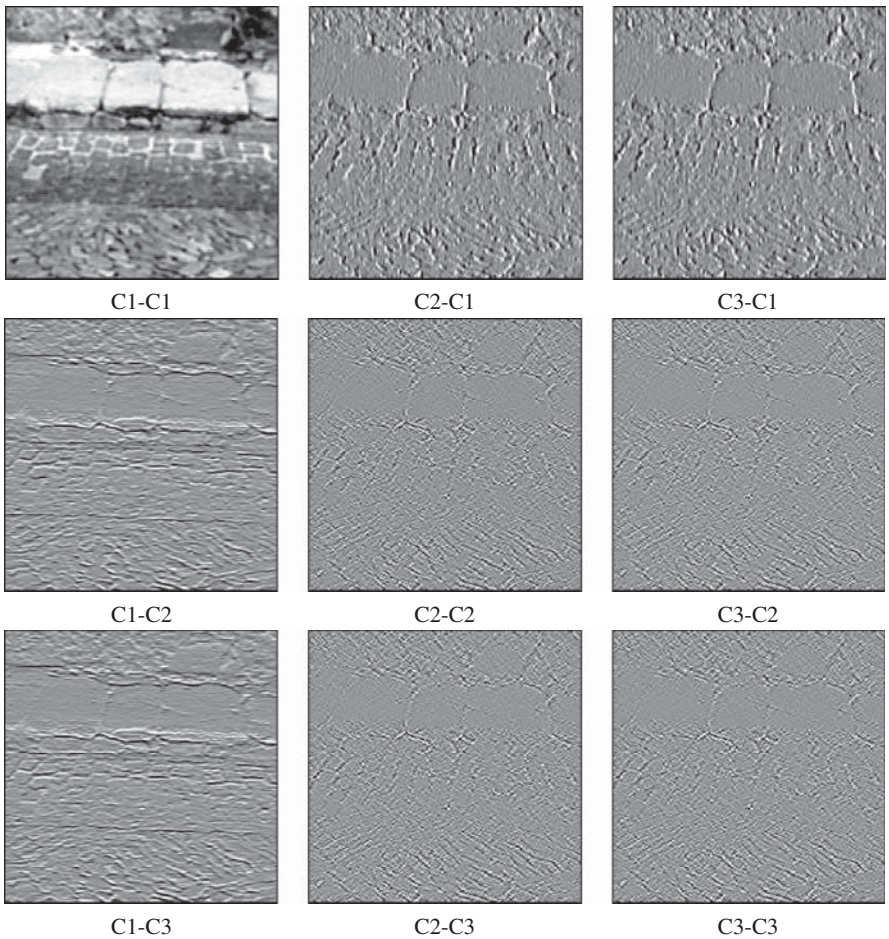


Figure 4.180 The expansion of each 3×3 neighbourhood of the image in Figure 4.81 using pseudo-convolutions. Each panel X-Y represents the output obtained when filter X is applied to the rows of the original image and filter Y is applied to the columns of the output. Each panel had its extreme 2% values saturated and the rest of its values scaled to the range [0,255] for visualisation. Source: Maria Petrou.

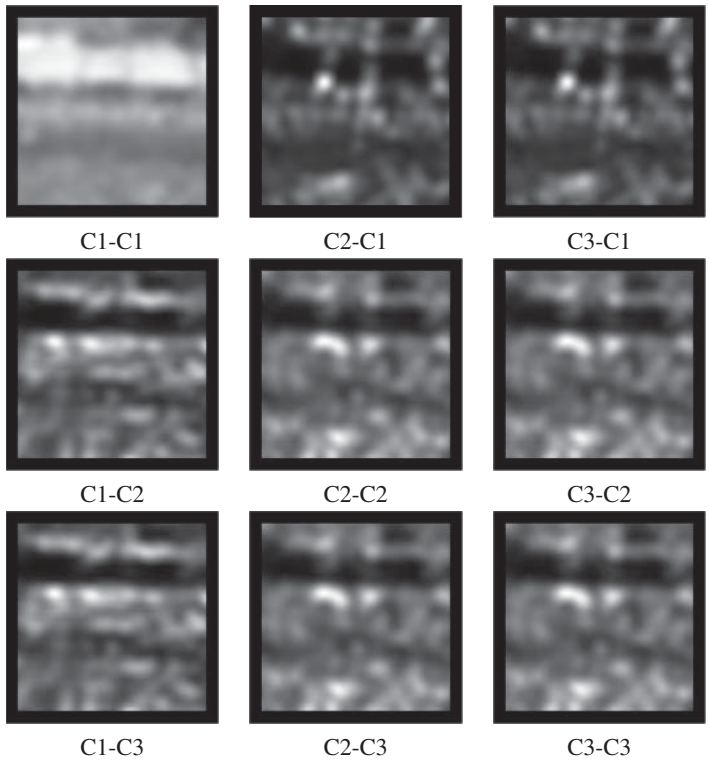
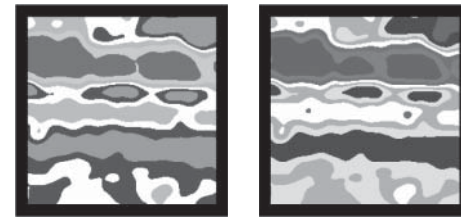


Figure 4.181 Energies of the outputs of Figure 4.180 computed using a Gaussian mask of size 29×29 with standard deviation $\sigma = 7$. Source: Maria Petrou.

Example 4.244

Use the expansion of Example 4.243 to segment the image of Figure 4.81.

Figure 4.182 Segmentation results obtained using the deterministic annealing algorithm over the energy features shown in Figure 4.181, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.



Example 4.245

Expand the 5×5 neighbourhoods of the image of Figure 4.81 in terms of the basis images shown in Figure 4.175.

(Continued)

Example 4.245 (Continued)

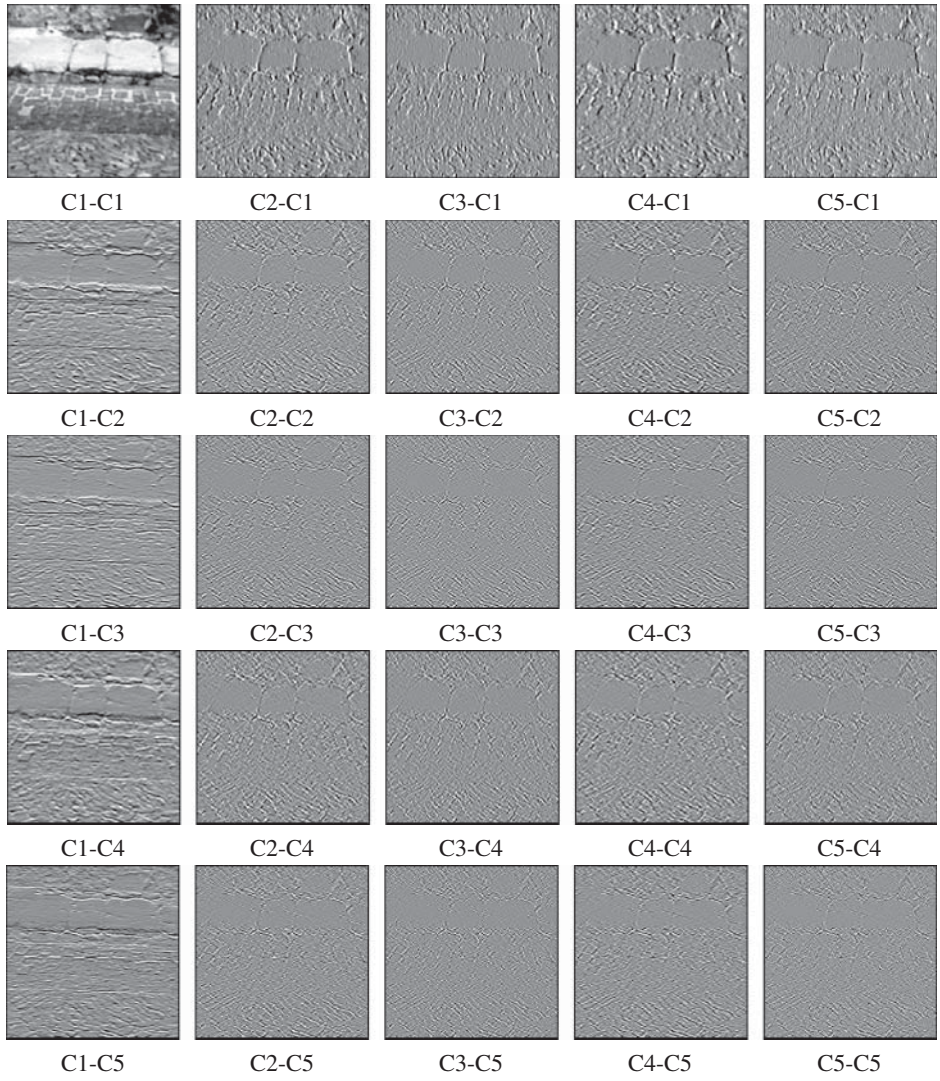
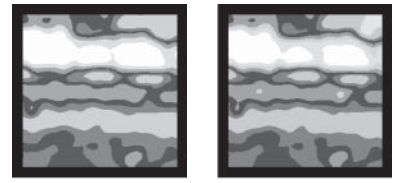


Figure 4.183 Expanding the 5×5 neighbourhoods of the image in Figure 4.81 in terms of the 5×5 Walsh basis images. Each panel $X-Y$ represents the output obtained when filter X is applied to the rows of the original image and filter Y is applied to the columns of the output. The 2% extreme values of each panel were saturated and the rest scaled to the range [0,255] for visualisation. Source: Maria Petrou.

Example 4.246

Use the expansion of Example 4.245 to segment the image of Figure 4.81.

Figure 4.184 Segmentation results obtained using the deterministic annealing algorithm over the 5×5 Walsh energy features, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.

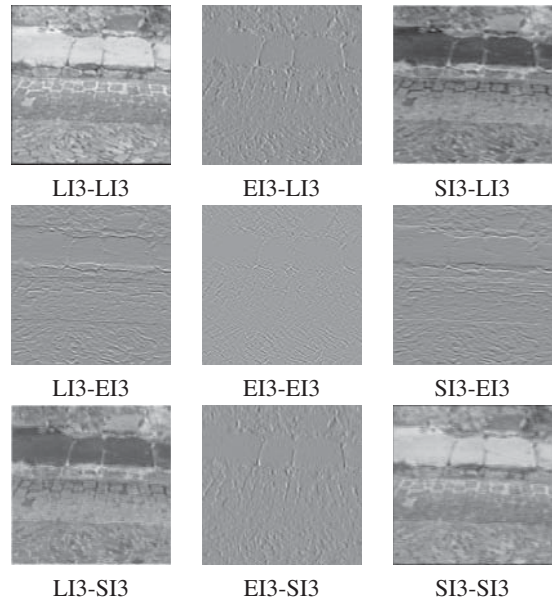


$K = 6$ $K = 8$

Example 4.247

Expand the 3×3 neighbourhoods of the image of Figure 4.81 in terms of the basis images shown in Figure 4.179a.

Figure 4.185 Expanding the 3×3 neighbourhoods of the image of Figure 4.81 in terms of the basis images created from Laws' masks. Each panel $X-Y$ represents the output obtained when filter X was applied to the rows of the original image and filter Y was applied to the columns of the output. Source: Maria Petrou.



LI3-LI3 EI3-LI3 SI3-LI3

LI3-EI3 EI3-EI3 SI3-EI3

LI3-SI3 EI3-SI3 SI3-SI3

All panels in Figures 4.185 and 4.186 have been individually scaled to the range [0,255]. *This means that comparing intensity values between panels is meaningless. For further processing,*

(Continued)

Example 4.247 (Continued)

such scaling should not be used, as really what matters is the relative values of the various coefficients of the local expansion, that tell us by how much each pattern in Figure 4.179a contributes locally. The energy features of Figure 4.186 were created from unscaled values and then scaled for visualisation. Note that patterns involving SI3 do not have 0 dc, so they capture part of the mean value of the image, and that is evident from Figure 4.185. Figure 4.187 shows enlarged versions of two such panels.

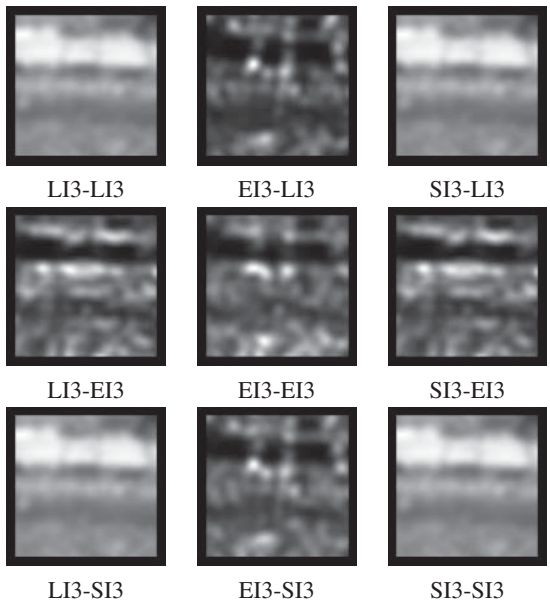


Figure 4.186 Energies of the outputs of Figure 4.185 computed using a Gaussian window of size 29×29 with standard deviation $\sigma = 7$. Source: Maria Petrou.

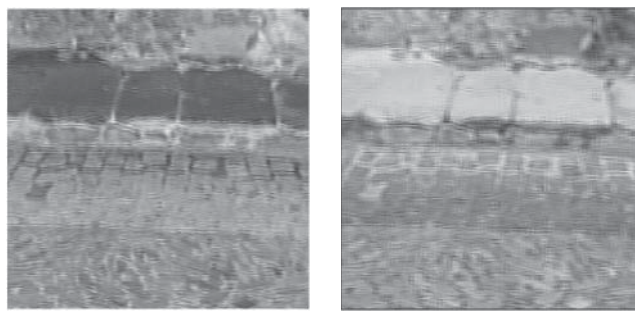
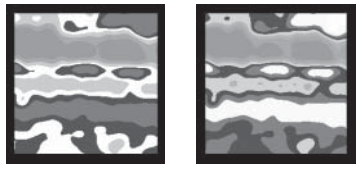


Figure 4.187 Panels LI3–SI3 (left) and SI3–SI3 (right) of Figure 4.185, enlarged. Source: Maria Petrou.

Example 4.248

Use the expansion you did in Example 4.247 to segment the image of Figure 4.81.
The result is shown in Figure 4.188.



$K = 6$ $K = 8$

Figure 4.188 Segmentation results obtained using the deterministic annealing algorithm over the energy features shown in Figure 4.186, for two different values of the expected number K of distinct clusters. Note that the energies depicted in Figure 4.186 are scaled to the range [0,255], but for segmentation the unscaled energy features were used, as what matters is the relative energy for each one of the local patterns. Source: Maria Petrou.

Is there any other way to express the local structure of the image?

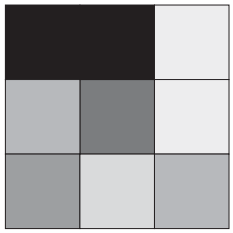
All methods we have discussed so far have been linear methods. There are some non-linear methods that may also be used to express the local image structure. For example, the local structure of the image may be expressed by the local symmetry of the image. An implicit way of expressing the local image symmetry is through the phase of the local Fourier transform. We know that the Fourier transform allows us to expand a signal in terms of sines and cosines. The sine components express the antisymmetric part of the signal, while the cosine components express the symmetric part. If a signal is purely symmetric, it has zero sine components, i.e. its Fourier transform is real, i.e. its phase is either 0° or 180°. If a signal is antisymmetric, it is expressed purely in terms of the sine components, i.e. its Fourier transform is purely imaginary, i.e. its phase is either 90° or 270°. So, we may say that the phase measures the degree of symmetry a function exhibits about the origin, as it expresses the balance of the antisymmetric and symmetric parts of the function. However, it is difficult to extract the phase of the Fourier transform, due to the problems mentioned in Chapter 3. Nevertheless, it is possible to use a particular signal representation, which encodes implicitly the phase information, and which may be thought of as an alternative way to represent the local structure of the image. This is the **Wigner distribution**.

Another way to express the local image structure is to try to encode the local neighbourhoods after they have been maximally simplified, i.e. binarised. This leads to the method of **local binary patterns (LBP)**.

4.10 Local Binary Patterns

What is the local binary pattern approach to texture representation?

The basic idea of this approach is demonstrated in Figure 4.189. We consider a 3×3 neighbourhood around each pixel. All neighbours that have values higher than the value of the central pixel are given value 1 and all those that have values lower than or equal to the value of the central pixel are given value 0. The eight binary numbers associated with the eight neighbours are then read sequentially in the clockwise direction to form a binary number. This binary number (or its equivalent in the decimal system) may be assigned to the central pixel and it may be used to characterise the local texture.



(a)

0	0	1
1		1
1	1	1

(b)

00111111 = 63 11110011 = 243
01111110 = 126 11100111 = 231
11111100 = 252 11001111 = 207
11111001 = 249 10011111 = 159

(c)

Figure 4.189 The pixels around the central pixel in (a) are given value 1 if they are brighter than the central pixel and value 0 otherwise. These values are shown in (b). If we read these values sequentially in the clockwise direction, depending which pixel is the starting pixel, we may form the binary numbers and their decimal equivalents shown in (c).

This is equivalent to assigning weights to the eight neighbouring pixels according to their relative position with respect to the central pixel. These weights are 2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1 and 2^0, with the first assigned to the neighbour that contributes the most significant digit, the second assigned to the neighbour that contributes the second most significant digit, and so on. Obviously, then, some neighbours are given more significance than others and this makes this representation sensitive to rotation, i.e. sensitive to which neighbour we consider to be the first. Another drawback of this approach is that it captures only the very local structure of the texture, and therefore it is appropriate only for micro-textures, but not macro-textures.

How can we make this representation rotationally invariant?

If we consider all possible binary numbers we can create from the local binary pattern, by starting the sequence from all eight neighbours in turn, and choose the smallest of the constructed numbers, we shall have a rotation-invariant representation. In the example of Figure 4.189, this number is 63, i.e. the number constructed by starting the sequence from the top left corner of the 3×3 neighbourhood.

How can we make this representation appropriate for macro-textures?

We can achieve this by applying the methodology to circular neighbourhoods of increasing radii. This leads to a multi-resolution representation of the local texture pattern, as for each radius a different binary pattern will be produced and therefore a different number representing it. As the radius of the neighbourhood increases, the number of neighbours increases and this may lead to prohibitively large values of the local binary pattern. This may be avoided by selecting only neighbours that are at the chosen radius and at a certain angular distance from each other. This is demonstrated in Figure 4.190.

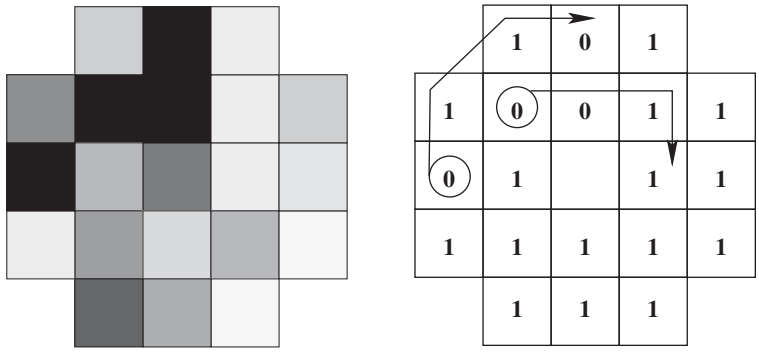


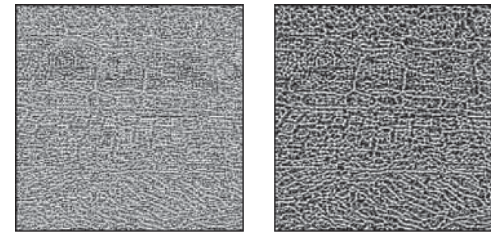
Figure 4.190 Two circular neighbourhoods with radii 1 and 2 are considered around the central pixel in (a). Each neighbour in these neighbourhoods is given value 1 if it is brighter than the central pixel and value 0 otherwise, as shown in (b). From each neighbourhood we construct the smallest binary number. For the neighbourhood with radius 1 this number is 00111111 = 63. For the neighbourhood with radius 2 this number is 011011111111 = 1791. In both cases this number is constructed by reading the digits in the clockwise direction starting from the digit marked by the circle. In the neighbourhood of radius 2 we may choose to consider only every other neighbour. Then the smallest binary number that we can construct is 011111 = 31.

Example 4.249

Compute the LBP for each pixel of the image of Figure 4.81, for radii 1 and 2.

For radius 1 there are 8 neighbours of each pixel, so the range of values of LBP is from 0 to $2^8 - 1$*, i.e.* $[0,255]$*. For radius 2 there are 12 neighbours, so the range of values is* $[0, 4095]$*. Because of the way we construct the rotationally invariant numbers, there is an overpopulation of the lower values in each range. So, in order to visualise the LBP "images", we histogram equalise them first. The results are shown in Figure 4.191.*

Figure 4.191 Local binary pattern maps for the image of Figure 4.81. Source: Maria Petrou.



radius = 1 radius = 2

How can we use the local binary patterns to characterise textures?

The local binary pattern approach may be used to characterise stationary textures, or segment non-stationary textures. The texture patch around a pixel may be characterised by the histogram of the LBP values of the pixels in a window around the central pixel. Such a histogram may be treated as the signature of the local texture of the image. To compare two textures then or two pixel neighbourhoods, we have to compare their corresponding histograms of LBP values using an appropriate **metric** or **pseudo-metric**.

What is a metric?

A metric is a function $d(A, B)$ that allows us to measure the distance from A to B. A function may only be used as a metric, if it obeys the following properties for all A, B and C:

$$\begin{aligned} &d(A, A) = 0 \\ &d(A, B) \geq 0 \\ &d(A, B) = d(B, A) \\ &d(A, B) \leq d(A, C) + d(C, B). \end{aligned} \tag{4.887}$$

The last one of the above properties is known as the **triangle inequality**. The most commonly used metric is the Euclidean metric, which allows us to measure the distance between two points.

What is a pseudo-metric?

A pseudo-metric is a function $d(A, B)$ that does not always obey the triangle inequality, but it may be used to measure the distance from A to B.

Why should one wish to use a pseudo-metric and not a metric?

Sometimes it is very difficult to define a proper metric, or it is very difficult to prove that the function we have adopted to measure distances obeys the triangle inequality. So sometimes we use a function that makes sense intuitively and it exhibits the right behaviour for most values of its arguments. Such a function is the one we use here to measure the "distance" between two distributions or histograms.

How can we measure the difference between two histograms?

There are a number of functions we may use. We shall present here one commonly used in image processing. It is based on the extension of the definition of the **Kullback–Leibler divergence** so that it obeys the property of symmetry (the third of properties (4.887)). Suppose that the histograms we wish to compare have been normalised so they may be treated as probability density functions. Let us call them $f_1(i)$ and $f_2(i)$, where i is the ith bin of each histogram. The two histograms are assumed to have an identical number of bins N and bins with identical widths. Then we may measure the distance of the two histograms as:

$$d(f_1, f_2) \equiv \sum_{i=1}^{N} f_1(i) \log f_1(i) + \sum_{i=1}^{N} f_2(i) \log f_2(i) - \sum_{i=1}^{N} (f_1(i) + f_2(i)) \log(f_1(i) + f_2(i)) + 2 \log 2. \tag{4.888}$$

Example B4.250

Prove that function (4.888) obeys the property $d(f_1, f_1) = 0$.
We apply definition (4.888) for $f_2 = f_1$

$$\begin{aligned} d(f_1,f_1) &= 2\sum_{i=1}^{N} f_1(i)\log f_1(i) - 2\sum_{i=1}^{N} f_1(i)\log(2f_1(i)) + 2\log 2 \\ &= 2\sum_{i=1}^{N} f_1(i)\log f_1(i) - 2\sum_{i=1}^{N} f_1(i)\log 2 - 2\sum_{i=1}^{N} f_1(i)\log f_1(i) + 2\log 2 \\ &= 0 \end{aligned} \tag{4.889}$$

where we made use of the property of logarithms that $\log(ab) = \log a + \log b$ *and the fact that the histograms are normalised, so* $\sum_{i=1}^{N} f_1(i) = 1$.

Example B4.251

Prove that function (4.888) obeys the property $d(f_1, f_2) \geq 0$.
We observe that $\sum_{i=1}^{N} f_1(i) = \sum_{i=1}^{N} f_2(i) = 1$. *So, factor 2 in the last term of definition (4.888) may be replaced by* $\sum_{i=1}^{N} f_1(i) + \sum_{i=1}^{N} f_2(i)$*:*

$$\begin{aligned} d(f_1,f_2) &= \sum_{i=1}^{N} f_1(i)\log f_1(i) + \sum_{i=1}^{N} f_2(i)\log f_2(i) - \\ &\quad \sum_{i=1}^{N} (f_1(i)+f_2(i))\log(f_1(i)+f_2(i)) + \left(\sum_{i=1}^{N} f_1(i) + \sum_{i=1}^{N} f_2(i)\right)\log 2 \\ &= \sum_{i=1}^{N} \Big\{ f_1(i)\log f_1(i) + f_2(i)\log f_2(i) - \\ &\quad (f_1(i)+f_2(i))\log(f_1(i)+f_2(i)) + (f_1(i)+f_2(i))\log 2. \Big\} \end{aligned} \tag{4.890}$$

In order to prove that $d(f_1, f_2) \geq 0$ *it is enough to prove that each term of the above sum is non-negative. A term in the above sum has the form*

$$f(a,b) \equiv a\log a + b\log b - (a+b)\log(a+b) + (a+b)\log 2 \tag{4.891}$$

where $1 \geq a, b \geq 0$. *If we take the partial derivatives of this function with respect to* a *and* b *and set them to 0 we shall be able to identify its extreme values. Differentiation with respect to* a *yields:*

$$\frac{\partial f}{\partial a} = \log a + \frac{a}{a} - \log(a+b) - \frac{a+b}{a+b} + \log 2 = \log\frac{2a}{a+b}. \tag{4.892}$$

So the two partial derivatives are:

$$\frac{\partial f}{\partial a} = \log\frac{2a}{a+b} \qquad \frac{\partial f}{\partial b} = \log\frac{2b}{a+b}. \tag{4.893}$$

(Continued)

Example B4.251 (Continued)

An extremum should be sought among the roots of the system $\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b} = 0$*. These equations are satisfied only when* $a = b$*. So, function* $f(a, b)$ *takes its extreme value for* $a = b$*. For all pairs of values* (a, b) *with* $a \neq b$*, function* $f(a, b)$ *must take values either larger than its value at* $a = b$*, or lower than its value at* $a = b$*. The value at* $a = b$ *is* $f(a, a) = 0$*. The value at a randomly picked point with* $a \neq b$*, say at point* $(1, 0)$*, will then tell us whether the function is always positive or always negative. We can easily verify that* $f(1, 0) = \log 2 > 0$*, so* $f(a, b)$ *is either 0 (when* $a = b$*) or positive (when* $a \neq b$*). Then by extension the same is true for the sum of N such functions, which proves the non-negativity of pseudo-metric (4.888).*

Example B4.252

Prove that function (4.888) obeys the property $d(f_1, f_2) = d(f_2, f_1)$**.**
This is trivial. We observe that definition (4.888) remains the same if we exchange the roles of $f_1(i)$ *and* $f_2(i)$*.*

Example B4.253

Consider a 25×25 **neighbourhood around each pixel of image 4.81. For each pixel construct the normalised histogram of the LBP values for radius 1. Then pick up at random 1000 triplets of pixels,** A**,** B **and** C**, and use function (4.888) to measure the distances between** A **and** B**,** A **and** C**, and** C **and** B**. Comment on the fraction of these triplets for which the triangle inequality (4.887) is obeyed. Repeat for radius 2.**
It was found that in 99.4% *of the cases the triangle inequality was obeyed for radius 1 and in* 100% *of the cases for radius 2. These percentages did not change when 10000 triplets were used. So, this pseudo-metric is almost a perfect metric.*

How can we use the local binary patterns to segment textures?

So far, when we clustered pixels in the feature space using the deterministic annealing algorithm, we measured their distances using the Euclidean metric. This need not be the case. We may measure pixel distances using any other metric or pseudo-metric that is appropriate for the type of feature we have. So, in this case, as the features are whole histograms of LBP values, all we have to do is to apply the deterministic annealing algorithm using function (4.888) instead of the Euclidean metric.

We may have some savings in memory space if we realise that a large number of bins of these histograms are empty, due to the way the LBP numbers are constructed. For example, for radius 1, only bins 0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 37, 39, 43, 45, 47, 51, 53, 55, 59, 61, 63, 85, 87, 91, 95, 111, 119, 127, 255 are non-empty. This means that for this case each histogram is 36 bins long.

Figure 4.192 shows some segmentation results obtained for image 4.81. The histograms were constructed using windows of size 25×25. We can see that these results are not at all satisfactory. This example demonstrates some other shortcomings of the LBP approach.

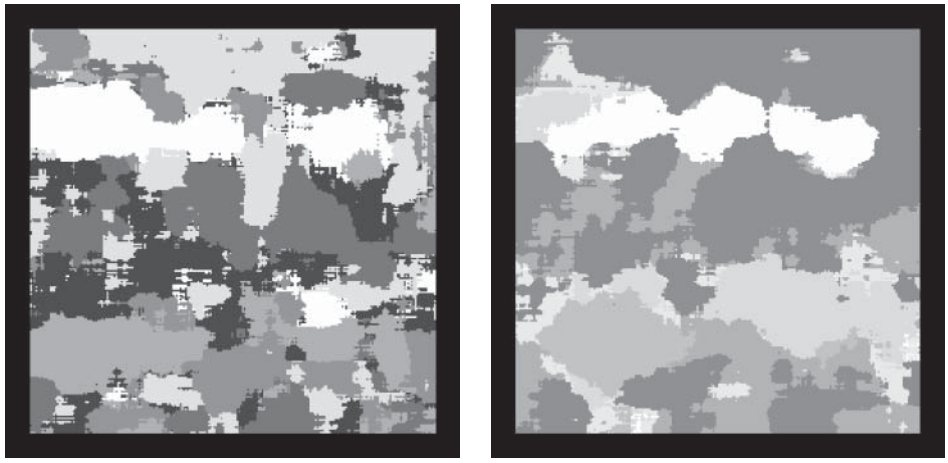


Figure 4.192 Segmentation of image 4.81 using the LBP histograms for radius 1 (left) and 2 (right) and six clusters in the deterministic annealing algorithm. Source: Maria Petrou.

How can we overcome the shortcomings of the LBP segmentation?

There are two points of improvement.

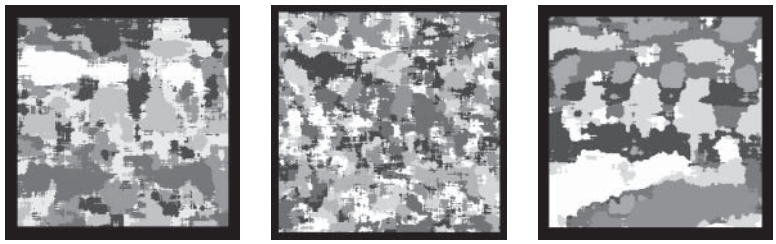
First, it is easy to realise that due to the way the LBP numbers are created, they are very sensitive to noise. The slightest fluctuation above or below the value of the central pixel is treated the same way as a major contrast between the central pixel and its surroundings. If we were using tertiary numbers instead of binary (i.e. we were allowing three distinct values), we might have been able to cope with this, by saying, for example, that any neighbour with value $\pm\delta$ different from the central pixel is considered as having value equal to the central pixel. Then we would have been able to have the neighbouring pixels take value 0 if they were darker than the central pixel, value 1 if they had value equal to the central pixel (within the chosen threshold) and value 2 if they were brighter than the central pixel. The unique number we would be creating then would have not been in the binary system, but in the system with base 3. This is an option that we shall not discuss any further here. If we wish to stick with the binary system, we may try to avoid noisy patterns by simply ignoring them: a noisy pattern due to its randomness will create neighbours that fluctuate above and below the value of the central pixel, with the 0s and the 1s frequently succeeding each other. So, one improvement we may make is to consider from each histogram only the so-called **stable** patterns, defining them as the patterns that have all their 1s in succession. These patterns are shown in Figure 4.193. We may see that these patterns correspond to some sort of corners of various angles having their vertices at the central pixel. If we consider only these patterns, it means that only nine bins of the histogram of LBP values are of relevance, namely those with values 0, 1, 3, 7, 15, 31, 63,127 and 255. The result of segmenting the image of Figure 4.81 using only the rotation-invariant stable patterns for radius 1 with a 25×25 window used for the calculation of the local histogram is shown in Figure 4.194a. Reducing the window to size 15×15 in order to improve localisation yields the result shown in 4.194b. This shows that the 15×15 window is too small for the reliable calculation of the histogram. Finally, if we do not try to make the stable patterns rotation-invariant (after all directionality is an important texture characteristic) and use a window of size 25×25 for computing the histogram, we obtain the result shown in Figure 4.194c. It is even more important for the window to be large when we retain the directionality of the features, because the histogram consists of more bins than in the case of rotationally invariant features, so a

0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0

0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1
0	0	1	0	1	1	1	1	1

0	0	1	1	0	1	1	1	1
1	0	1	1	0	1	1	0	1
1	1	1	1	1	1	1	1	1

Figure 4.193 The stable patterns in a 3×3 neighbourhood. All rotational variations of them are also accepted as stable.



(a) (b) (c)

Figure 4.194 (a) Segmentation result using the LBP histograms of only the rotation-invariant stable patterns for radius 1, computed inside a 25×25 local window. (b) As in (a), but with a 15×15 window used. (c) As in (a) but the directionality of the stable points retained. In all cases the deterministic annealing algorithm was run for $K = 6$. Source: Maria Petrou.

larger number of pixels is needed for its reliable calculation. We see that the last result shows some improvement over the previous segmentations.

The second important correction we can make to recover the loss of information created by computing the LBP value is to compute the local contrast of each pattern. This can be very easily done. We calculate the average of the grey values of the pixels labelled with 1 and the average of the grey values of the pixels labelled with 0 and find the difference between these two numbers. The result of the segmentation we can achieve by using only these contrast pixel values is shown in Figure 4.195. We can immediately see that this segmentation result is much better than any of the results obtained by LBP alone. The results did not differ much either we used the stable patterns only or all patterns to compute the contrast values.

Finally, we may combine the contrast values with the LBP values to perform segmentation. These results are shown in Figure 4.196. These segmentation results are comparable with what we obtained by other methods. It is clear, however, that most of the information came from the contrast feature rather than the value of LBP. This is perhaps because LBP over-simplifies the local structure. It is also noticeable that the use of radius 2 is preferable. When we use LBP and contrast we have 2D histograms, with even more bins than when we use either of these features alone. Using a radius larger than 1 necessitates then the use of a subset of neighbours at that distance, placed at selected orientations, so that we have fewer histogram bins and the pixels inside the local windows are enough to populate them reliably.

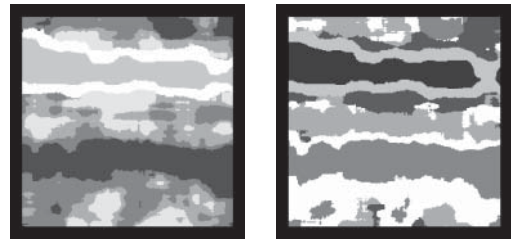
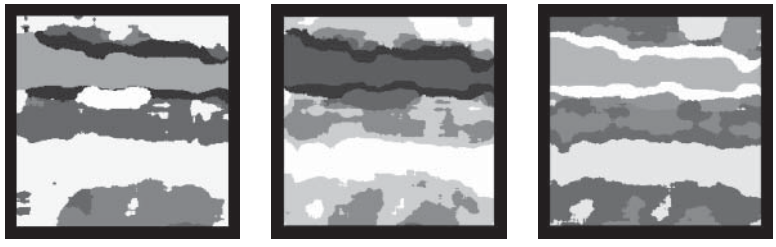


Figure 4.195 Segmentation results using the contrast values only for radius 1 (left) and 2 (right) and six clusters in the deterministic annealing algorithm. In both cases the histograms of the contrast values were computed inside a 25×25 window. Source: Maria Petrou.



(a) (b) (c)

Figure 4.196 (a) Segmentation result using the LBP histograms of the stable directional patterns and the contrast values for radius 1 with a local window of size 25×25. (b) As in (a) but rotation invariant LBP patterns used. (c) As in (b) with the features computed for radius 2. In all cases six clusters were specified in the deterministic annealing algorithm. Source: Maria Petrou.

4.11 The Wigner Distribution

What is the Wigner distribution?

Consider a signal $f(t)$. At a fixed point t_0, construct the following function:

$$r(t_0;\alpha) \equiv f\left(t_0 + \frac{\alpha}{2}\right) f^*\left(t_0 - \frac{\alpha}{2}\right). \tag{4.894}$$

Take the Fourier transform of this function with respect to α:

$$W(t_0;u) = \int_{-\infty}^{\infty} r(t_0;\alpha) \mathrm{e}^{-\mathrm{j}\alpha u} \mathrm{d}\alpha = \int_{-\infty}^{\infty} f\left(t_0 + \frac{\alpha}{2}\right) f^*\left(t_0 - \frac{\alpha}{2}\right) \mathrm{e}^{-\mathrm{j}\alpha u} \mathrm{d}\alpha. \tag{4.895}$$

This is the **Wigner distribution** of the signal at point t_0. In general, the Wigner distribution is computed at all points of the signal, so it is a 2D representation of the signal. This function assigns to each signal point a "spectrum", the **Wigner spectrum**, $W(t;u)$, which represents the signal fully, unlike the Fourier **spectrum**. Indeed, if we know the Fourier spectrum of a signal, we cannot recover the original signal because the phase information is missing. For a real signal, on the other hand, the Wigner distribution is a reversible transform, allowing one to recover the original signal up to a sign (see Examples 4.255 and 4.257). That is why we say that the Wigner distribution encodes **implicitly** the phase information of the signal.

Example B4.254

Show that if $F(\omega)$ is the Fourier transform of a signal $f(t)$, then the Wigner distribution of the signal may be expressed in an equivalent way as:

$$W(\tau;\omega)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F\left(\omega+\frac{\xi}{2}\right)F^{*}\left(\omega-\frac{\xi}{2}\right)\mathrm{e}^{\mathrm{j}\xi\tau}\mathrm{d}\xi. \tag{4.896}$$

Let us start from the definition of the Fourier transform of $f(t)$

$$F(\omega)=\int_{-\infty}^{\infty}f(t)\mathrm{e}^{-\mathrm{j}t\omega}\mathrm{d}t \tag{4.897}$$

and let us substitute this into the right-hand side of (4.896):

$$\begin{aligned}&\frac{1}{2\pi}\int_{-\infty}^{\infty}F\left(\omega+\frac{\xi}{2}\right)F^{*}\left(\omega-\frac{\xi}{2}\right)\mathrm{e}^{\mathrm{j}\xi\tau}\mathrm{d}\xi=\\&\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(t)\mathrm{e}^{-\mathrm{j}t\left(\omega+\frac{\xi}{2}\right)}\mathrm{d}t\int_{-\infty}^{\infty}f^{*}(\tilde{t})\mathrm{e}^{\mathrm{j}\tilde{t}\left(\omega-\frac{\xi}{2}\right)}\mathrm{d}\tilde{t}\mathrm{e}^{\mathrm{j}\xi\tau}\mathrm{d}\xi=\\&\frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(t)\mathrm{e}^{-\mathrm{j}t\omega}f^{*}(\tilde{t})\mathrm{e}^{\mathrm{j}\tilde{t}\omega}\left\{\int_{-\infty}^{\infty}\mathrm{e}^{\mathrm{j}\xi\left(-\frac{t}{2}-\frac{\tilde{t}}{2}+\tau\right)}\mathrm{d}\xi\right\}\mathrm{d}t\mathrm{d}\tilde{t}.\end{aligned} \tag{4.898}$$

The integral inside the curly brackets is equal to $2\pi\delta(-\frac{t}{2}-\frac{\tilde{t}}{2}+\tau)$ according to the result of Example 4.10. If we perform the integral over $\tilde{t}$ next, we have:

$$\begin{aligned}&\frac{1}{2\pi}\int_{-\infty}^{\infty}F\left(\omega+\frac{\xi}{2}\right)F^{*}\left(\omega-\frac{\xi}{2}\right)\mathrm{e}^{\mathrm{j}\xi\tau}\mathrm{d}\xi=\\&\int_{-\infty}^{\infty}f(t)\mathrm{e}^{-\mathrm{j}t\omega}\left\{\int_{-\infty}^{\infty}f^{*}(\tilde{t})\mathrm{e}^{\mathrm{j}\tilde{t}\omega}\delta\left(-\frac{t}{2}-\frac{\tilde{t}}{2}+\tau\right)\mathrm{d}\tilde{t}\right\}\mathrm{d}t.\end{aligned} \tag{4.899}$$

We introduce a new variable of integration $y\equiv-\frac{t}{2}-\frac{\tilde{t}}{2}+\tau\Rightarrow\tilde{t}=2\tau-t-2y\Rightarrow\mathrm{d}\tilde{t}=-2\mathrm{d}y$:

$$\begin{aligned}&\frac{1}{2\pi}\int_{-\infty}^{\infty}F\left(\omega+\frac{\xi}{2}\right)F^{*}\left(\omega-\frac{\xi}{2}\right)\mathrm{e}^{\mathrm{j}\xi\tau}\mathrm{d}\xi=\\&-2\int_{-\infty}^{\infty}f(t)\mathrm{e}^{-\mathrm{j}t\omega}\left\{\int_{\infty}^{-\infty}f^{*}(2\tau-t-2y)\mathrm{e}^{\mathrm{j}(2\tau-t-2y)\omega}\delta(y)\mathrm{d}y\right\}\mathrm{d}t=\\&2\int_{-\infty}^{\infty}f(t)\mathrm{e}^{-\mathrm{j}t\omega}f^{*}(2\tau-t)\mathrm{e}^{\mathrm{j}(2\tau-t)\omega}\mathrm{d}t=\\&2\int_{-\infty}^{\infty}f(t)f^{*}(2\tau-t)\mathrm{e}^{\mathrm{j}2(\tau-t)\omega}\mathrm{d}t.\end{aligned} \tag{4.900}$$

Let us define a new variable of integration again, $z\equiv2(t-\tau)\Rightarrow t=\tau+\frac{z}{2}$ and $\mathrm{d}t=\frac{1}{2}\mathrm{d}z$:

$$\begin{aligned}&\frac{1}{2\pi}\int_{-\infty}^{\infty}F\left(\omega+\frac{\xi}{2}\right)F^{*}\left(\omega-\frac{\xi}{2}\right)\mathrm{e}^{\mathrm{j}\xi\tau}\mathrm{d}\xi=\\&\int_{-\infty}^{\infty}f\left(\tau+\frac{z}{2}\right)f^{*}\left(\tau-\frac{z}{2}\right)\mathrm{e}^{-\mathrm{j}\omega z}\mathrm{d}z=\\&W(\tau;\omega).\end{aligned} \tag{4.901}$$

The last equality was obtained by direct comparison with the definition of the Wigner distribution, given by Equation (4.895). This is the result we had to prove.

Example B4.255

Given the Wigner distribution $W(t;u)$ of a real signal $f(t)$, recover the original signal.

The Wigner distribution is the Fourier transform of function $r(t;\alpha) = f\left(t+\frac{\alpha}{2}\right) f\left(t-\frac{\alpha}{2}\right)$. So, if we take the inverse Fourier transform of $W(t;u)$ we shall obtain function $r(t;\alpha)$. By setting then $\alpha = 0$, we have $r(t;0) = f(t)^2$, which means that we may recover the original signal $f(t)$ up to a sign, by taking the square root of $r(t;0)$: $f(t) = \pm\sqrt{r(t;0)}$.

Example B4.256

Compute the Wigner distribution of the signal

$$g(t) = \mathrm{e}^{-\frac{t^2}{2\sigma^2}} \tag{4.902}$$

where σ is some positive constant.

By definition, for a real signal:

$$\begin{aligned} W_g(t;u) &= \int_{-\infty}^{\infty} g\left(t+\frac{\alpha}{2}\right) g\left(t-\frac{\alpha}{2}\right) \mathrm{e}^{-\mathrm{j}u\alpha}\mathrm{d}\alpha \\ &= \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(t+\frac{\alpha}{2}\right)^2} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(t-\frac{\alpha}{2}\right)^2} \mathrm{e}^{-\mathrm{j}u\alpha}\mathrm{d}\alpha \\ &= \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(t^2+\frac{\alpha^2}{4}+\frac{2t\alpha}{2}+t^2+\frac{\alpha^2}{4}-\frac{2t\alpha}{2}+ju\alpha 2\sigma^2\right)}\mathrm{d}\alpha \\ &= \mathrm{e}^{-\frac{t^2}{\sigma^2}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(\frac{\alpha^2}{2}+ju\alpha 2\sigma^2\right)}\mathrm{d}\alpha. \end{aligned} \tag{4.903}$$

To compute this integral we have to complete the square in the exponent of the integrand:

$$\begin{aligned} W_g(t;u) &= \mathrm{e}^{-\frac{t^2}{\sigma^2}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(\left(\frac{\alpha}{\sqrt{2}}\right)^2+2\frac{\alpha}{\sqrt{2}}(\sqrt{2}ju\sigma^2)-(\sqrt{2}u\sigma^2)^2+(\sqrt{2}u\sigma^2)^2\right)}\mathrm{d}\alpha \\ &= \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\frac{1}{2\sigma^2}(\sqrt{2}u\sigma^2)^2} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(\frac{\alpha}{\sqrt{2}}+\sqrt{2}ju\sigma^2\right)^2}\mathrm{d}\alpha. \end{aligned} \tag{4.904}$$

We define a new variable of integration $y \equiv \frac{1}{\sqrt{2}\sigma}\left(\frac{\alpha}{\sqrt{2}} + j\sqrt{2}u\sigma^2\right)$, which leads to $\mathrm{d}y = \frac{1}{2\sigma}\mathrm{d}\alpha$ or $\mathrm{d}\alpha = 2\sigma\mathrm{d}y$. Then

$$\begin{aligned} W_g(t;u) &= \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-u^2\sigma^2} 2\sigma \int_{-\infty}^{\infty} \mathrm{e}^{-y^2}\mathrm{d}y \\ &= 2\sigma\mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-u^2\sigma^2}\sqrt{\pi} \end{aligned} \tag{4.905}$$

where we made use of the result of Example 3.36.

So, we may say that the Wigner distribution of the Gaussian signal (4.902) is:

$$W_g(t;u) = 2\sqrt{\pi}\sigma\mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\sigma^2u^2}. \tag{4.906}$$

Example B4.257

The Wigner distribution of a real signal is given by Equation (4.906). Recover the original signal.

Let the original signal be $g(t)$. To recover it, we shall first take the inverse Fourier transform of the Wigner distribution, to recover function $r(t;\alpha) \equiv g\left(t+\frac{\alpha}{2}\right)g\left(t-\frac{\alpha}{2}\right)$:

$$\begin{aligned} r(t;\alpha) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} W_g(t;u)\mathrm{e}^{\mathrm{j}u\alpha}\mathrm{d}u \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} 2\sqrt{\pi}\sigma\mathrm{e}^{-\frac{t^2}{\sigma^2}}\mathrm{e}^{-\sigma^2u^2}\mathrm{e}^{\mathrm{j}u\alpha}\mathrm{d}u \\ &= \frac{1}{2\pi}2\sqrt{\pi}\sigma\mathrm{e}^{-\frac{t^2}{\sigma^2}}\int_{-\infty}^{\infty}\mathrm{e}^{-\sigma^2u^2}\mathrm{e}^{\mathrm{j}u\alpha}\mathrm{d}u. \end{aligned} \tag{4.907}$$

To compute this integral we must first complete the square in the exponent of the integrand:

$$\begin{aligned} r(t;\alpha) &= \frac{\sigma}{\sqrt{\pi}}\mathrm{e}^{-\frac{t^2}{\sigma^2}}\int_{-\infty}^{\infty}\mathrm{e}^{-\left((\sigma u)^2-2\sigma u\left(\frac{1}{2\sigma}j\alpha\right)-\left(\frac{1}{2\sigma}\alpha\right)^2+\left(\frac{1}{2\sigma}\alpha\right)^2\right)}\mathrm{d}u \\ &= \frac{\sigma}{\sqrt{\pi}}\mathrm{e}^{-\frac{t^2}{\sigma^2}}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\sigma u-\frac{1}{2\sigma}j\alpha\right)^2}\mathrm{e}^{-\left(\frac{1}{2\sigma}\alpha\right)^2}\mathrm{d}u \\ &= \frac{\sigma}{\sqrt{\pi}}\mathrm{e}^{-\frac{t^2}{\sigma^2}}\mathrm{e}^{-\left(\frac{1}{2\sigma}\alpha\right)^2}\int_{-\infty}^{\infty}\mathrm{e}^{-\left(\sigma u-\frac{1}{2\sigma}j\alpha\right)^2}\mathrm{d}u. \end{aligned} \tag{4.908}$$

We define a new variable of integration $y \equiv \sigma u - \frac{1}{2\sigma}j\alpha$, which means that $\mathrm{d}y = \sigma\mathrm{d}u$, or $\mathrm{d}u = \mathrm{d}y/\sigma$. Then:

$$\begin{aligned} r(t;\alpha) &= \frac{1}{\sqrt{\pi}}\mathrm{e}^{-\frac{t^2}{\sigma^2}}\mathrm{e}^{-\frac{\alpha^2}{4\sigma^2}}\int_{-\infty}^{\infty}\mathrm{e}^{-y^2}\mathrm{d}y \\ &= \frac{1}{\sqrt{\pi}}\mathrm{e}^{-\frac{t^2}{\sigma^2}}\mathrm{e}^{-\frac{\alpha^2}{4\sigma^2}}\sqrt{\pi} \\ &= \mathrm{e}^{-\frac{t^2}{\sigma^2}}\mathrm{e}^{-\frac{\alpha^2}{4\sigma^2}}. \end{aligned} \tag{4.909}$$

Here we made use of the result of Example 3.36.

To recover the original signal, we set $\alpha = 0$ in $r(t;\alpha)$:

$$g(t)^2 = r(t;0) \Rightarrow g(t) = \pm\sqrt{r(t;0)} \Rightarrow g(t) = \pm\mathrm{e}^{-\frac{t^2}{2\sigma^2}}. \tag{4.910}$$

So, we see that the original signal was recovered up to a sign.

Example B4.258

Compute the Wigner distribution of the signal

$$g(t) = \mathrm{e}^{-\frac{t^2}{2\sigma^2}-\mathrm{j}t} \tag{4.911}$$

where σ is some positive constant.

Now the signal is complex, and we must remember to use also the complex conjugate of the signal where appropriate. By definition:

$$\begin{aligned}
W_g(t;u) &= \int_{-\infty}^{\infty} g\left(t+\frac{\alpha}{2}\right) g^*\left(t-\frac{\alpha}{2}\right) \mathrm{e}^{-\mathrm{j}u\alpha}\mathrm{d}\alpha \\
&= \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(t+\frac{\alpha}{2}\right)^2 - \mathrm{j}\left(t+\frac{\alpha}{2}\right)} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(t-\frac{\alpha}{2}\right)^2 + j\left(t-\frac{\alpha}{2}\right)} \mathrm{e}^{-\mathrm{j}u\alpha}\mathrm{d}\alpha \\
&= \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(t^2+\frac{\alpha^2}{4}+\frac{2t\alpha}{2}+2\sigma^2 jt+2\sigma^2 j\frac{\alpha}{2}+t^2+\frac{\alpha^2}{4}-\frac{2t\alpha}{2}-2\sigma^2 jt+2\sigma^2 j\frac{\alpha}{2}+ju\alpha 2\sigma^2\right)}\mathrm{d}\alpha \\
&= \mathrm{e}^{-\frac{t^2}{\sigma^2}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(\frac{\alpha^2}{2}+j(u+1)\alpha 2\sigma^2\right)}\mathrm{d}\alpha. \qquad (4.912)
\end{aligned}$$

To compute this integral we must complete the square in the exponent of the integrand:

$$\begin{aligned}
W_g(t;u) &= \mathrm{e}^{-\frac{t^2}{\sigma^2}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(\left(\frac{\alpha}{\sqrt{2}}\right)^2 + 2\frac{\alpha}{\sqrt{2}}(\sqrt{2}j(u+1)\sigma^2) - (\sqrt{2}(u+1)\sigma^2)^2 + (\sqrt{2}(u+1)\sigma^2)^2\right)}\mathrm{d}\alpha \\
&= \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\frac{1}{2\sigma^2}(\sqrt{2}(u+1)\sigma^2)^2} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(\frac{\alpha}{\sqrt{2}}+\sqrt{2}j(u+1)\sigma^2\right)^2}\mathrm{d}\alpha. \qquad (4.913)
\end{aligned}$$

We define a new variable of integration $y \equiv \frac{1}{\sqrt{2}\sigma}\left(\frac{\alpha}{\sqrt{2}} + j\sqrt{2}(u+1)\sigma^2\right)$, *which leads to* $\mathrm{d}y = \frac{1}{2\sigma}\mathrm{d}\alpha$ *or* $\mathrm{d}\alpha = 2\sigma\mathrm{d}y$. *Then*

$$\begin{aligned}
W_g(t;u) &= \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-(u+1)^2\sigma^2} 2\sigma \int_{-\infty}^{\infty} \mathrm{e}^{-y^2}\mathrm{d}y \\
&= 2\sigma \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-(u+1)^2\sigma^2}\sqrt{\pi} \qquad (4.914)
\end{aligned}$$

where we made use of the result of Example 3.36.

So, we may say that the Wigner distribution of signal (4.911) is

$$W_g(t;u) = 2\sqrt{\pi}\sigma \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\sigma^2(u+1)^2}. \qquad (4.915)$$

Example B4.259

The Wigner distribution of a signal is given by Equation (4.915). Compute the original signal.

Let the original signal be $g(t)$. *Since we are not told that it is real, we must assume that it is complex. To recover it, we shall first take the inverse Fourier transform of the Wigner distribution, to recover function* $r(t;\alpha) \equiv g\left(t+\frac{\alpha}{2}\right) g^*\left(t-\frac{\alpha}{2}\right)$:

$$\begin{aligned}
r(t;\alpha) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} W_g(t;u)\mathrm{e}^{\mathrm{j}u\alpha}\mathrm{d}u \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} 2\sqrt{\pi}\sigma \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\sigma^2(u+1)^2} \mathrm{e}^{\mathrm{j}u\alpha}\mathrm{d}u \\
&= \frac{\sigma}{\sqrt{\pi}} \mathrm{e}^{-\frac{t^2}{\sigma^2}} \int_{-\infty}^{\infty} \mathrm{e}^{-\sigma^2u^2-\sigma^2-2\sigma^2u+ju\alpha}\mathrm{d}u
\end{aligned}$$

(Continued)

Example B4.259 (Continued)

$$
\begin{aligned}
&= \frac{\sigma}{\sqrt{\pi}} \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\sigma^2} \int_{-\infty}^{\infty} \mathrm{e}^{-\sigma^2 u^2 - u(2\sigma^2 - \mathrm{j}\alpha)} \mathrm{d}u \\
&= \frac{\sigma}{\sqrt{\pi}} \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\sigma^2} \int_{-\infty}^{\infty} \mathrm{e}^{-\left((\sigma u)^2 + 2\sigma u \frac{2\sigma^2 - \mathrm{j}\alpha}{2\sigma} + \left(\frac{2\sigma^2 - \mathrm{j}\alpha}{2\sigma}\right)^2 - \left(\frac{2\sigma^2 - \mathrm{j}\alpha}{2\sigma}\right)^2\right)} \mathrm{d}u \\
&= \frac{\sigma}{\sqrt{\pi}} \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\sigma^2} \int_{-\infty}^{\infty} \mathrm{e}^{-\left(\sigma u + \frac{2\sigma^2 - \mathrm{j}\alpha}{2\sigma}\right)^2} \mathrm{d}u \mathrm{e}^{\left(\frac{2\sigma^2 - \mathrm{j}\alpha}{2\sigma}\right)^2}.
\end{aligned} \tag{4.916}
$$

We define a new variable of integration $y \equiv \sigma u + \frac{2\sigma^2 - \mathrm{j}\alpha}{2\sigma}$, *which means that* $\mathrm{d}y = \sigma \mathrm{d}u$, *or* $\mathrm{d}u = \mathrm{d}y/\sigma$. *Then:*

$$
\begin{aligned}
r(t;\alpha) &= \frac{1}{\sqrt{\pi}} \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\sigma^2} \mathrm{e}^{\left(\frac{2\sigma^2 - \mathrm{j}\alpha}{2\sigma}\right)^2} \int_{-\infty}^{\infty} \mathrm{e}^{-y^2} \mathrm{d}y \\
&= \frac{1}{\sqrt{\pi}} \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\sigma^2} \mathrm{e}^{\frac{4\sigma^4 - \alpha^2 - 4\sigma^2 \mathrm{j}\alpha}{4\sigma^2}} \sqrt{\pi} \\
&= \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\frac{\alpha^2}{4\sigma^2}} \mathrm{e}^{-\mathrm{j}\alpha}.
\end{aligned} \tag{4.917}
$$

Here we made use of the result of Example 3.36.

Since the signal is assumed complex, we may assume that it has the form $g(t) \equiv M(t)\mathrm{e}^{\mathrm{j}\Phi(t)}$ *where* $M(t)$ *and* $\Phi(t)$ *are real functions which we must define. Then from the definition of* $r(t;\alpha)$, *we have*

$$
r(t;\alpha) = M\left(t + \frac{\alpha}{2}\right) \mathrm{e}^{\mathrm{j}\Phi\left(t + \frac{\alpha}{2}\right)} M\left(t - \frac{\alpha}{2}\right) \mathrm{e}^{-\mathrm{j}\Phi\left(t - \frac{\alpha}{2}\right)} = \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\frac{\alpha^2}{4\sigma^2}} \mathrm{e}^{-\mathrm{j}\alpha} \tag{4.918}
$$

from which we deduce that

$$
M\left(t + \frac{\alpha}{2}\right) M\left(t - \frac{\alpha}{2}\right) = \mathrm{e}^{-\frac{t^2}{\sigma^2}} \mathrm{e}^{-\frac{\alpha^2}{4\sigma^2}} \tag{4.919}
$$

and

$$
\Phi\left(t + \frac{\alpha}{2}\right) - \Phi\left(t - \frac{\alpha}{2}\right) = -\alpha. \tag{4.920}
$$

By setting $\alpha = 0$ *in (4.919), we deduce that* $M(t) = \pm \mathrm{e}^{-\frac{t^2}{2\sigma^2}}$.
To recover function $\Phi(t)$ *we rewrite Equation (4.920) in the form*

$$
\frac{\Phi\left(t + \frac{\alpha}{2}\right) - \Phi\left(t - \frac{\alpha}{2}\right)}{\alpha} = -1. \tag{4.921}
$$

This equation is valid for all values of α, *so it must also be valid for the limit when* $\alpha \to 0$. *For this limit we recognise on the left-hand side of (4.921) the derivative of* $\Phi(t)$, *so we have:*

$$
\frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} = -1 \Rightarrow \Phi(t) = -t + C \tag{4.922}
$$

where C is any real constant.

We may say, therefore, that Wigner distribution (4.915) was produced from signal:

$$
g(t) = \pm \mathrm{e}^{-\frac{t^2}{2\sigma^2}} \mathrm{e}^{-\mathrm{j}t + \mathrm{j}C}. \tag{4.923}
$$

This shows that the original complex signal may be recovered from its Wigner distribution up to a factor $\pm \mathrm{e}^{\mathrm{j}C}$.

Example B4.260

Show that all functions of the form (4.923) have the same Wigner distribution, irrespective of the value of real constant C.

To compute the Wigner distribution of function (4.923) we must first construct function $r(t;\alpha)$:

$$r(t;\alpha) \equiv g\left(t+\frac{\alpha}{2}\right)g^*\left(t-\frac{\alpha}{2}\right). \tag{4.924}$$

Upon substitution, we obtain:

$$\begin{aligned} r(t;\alpha) &= (\pm)^2 e^{-\frac{1}{2\sigma^2}\left(t+\frac{\alpha}{2}\right)^2} e^{-j\left(t+\frac{\alpha}{2}\right)} e^{jC} e^{-\frac{1}{2\sigma^2}\left(t-\frac{\alpha}{2}\right)^2} e^{j\left(t-\frac{\alpha}{2}\right)} e^{-jC} \\ &= e^{-\frac{1}{2\sigma^2}\left(t+\frac{\alpha}{2}\right)^2} e^{-\frac{1}{2\sigma^2}\left(t-\frac{\alpha}{2}\right)^2} e^{-j\alpha}. \end{aligned} \tag{4.925}$$

The last expression is the same function $r(t;\alpha)$ as in Example 4.258. So it will lead to the same Wigner distribution as the one we computed for function $g(t) = e^{-\frac{t^2}{2\sigma^2}} e^{-jt}$. So, all functions (4.923) have the same Wigner distribution, namely that of (4.915). Since sign $\pm$ in front of Equation (4.923) may also be expressed as a phase factor of the form $e^{j\pi k}$ for $k = 0$ or $k = 1$, we may say that this example demonstrates that the Wigner distribution defines the original function fully, up to a constant in the phase of the function. Note that this phase has nothing to do with the phase we talked about earlier when we demonstrated that phase represents the symmetry of the function. That phase was referring to the phase of the Fourier transform of the function, which somehow measures the balance between the real and the imaginary parts of its Fourier transform, i.e. the balance between the symmetric and the antisymmetric part of the function, while here the word "phase" refers to the function itself, which is assumed complex. (Due to duality, the phase of the signal measures the symmetry or antisymmetry of the Fourier transform of the signal.) In image processing we do not deal with complex signals, so we may ignore this detail according to which the Wigner distribution defines the original signal up to a factor e^{jC}. As far as image processing is concerned, we may safely assume that the Wigner distribution defines the original signal fully, up to a sign.

Example B4.261

Compute the ambiguity function of a signal $f(t)$ that is defined as the Fourier transform of the Wigner distribution of the signal.

Let us call the Fourier transform of the Wigner distribution $A(\xi;\tau)$. The Wigner distribution of the signal is by definition:

$$W(t;u) \equiv \int_{-\infty}^{\infty} f\left(t+\frac{\alpha}{2}\right) f^*\left(t-\frac{\alpha}{2}\right) e^{-j\alpha u} d\alpha. \tag{4.926}$$

This is a 2D function, the first argument of which is either a spatial coordinate, or time, and the second is a frequency. When we compute its Fourier transform, therefore, we must treat each independent variable accordingly, so we are consistent with the conventions of Fourier analysis. This means that the exponent of the kernel used for the frequency variable will be positive, while for the time variable it will be negative, and we shall have a $1/(2\pi)$ factor in front coming from

(Continued)

Example B4.261 (Continued)

the frequency variable:

$$A(\xi;\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} W(t;u)\mathrm{e}^{-\mathrm{j}t\xi}\mathrm{e}^{\mathrm{j}u\tau}\mathrm{d}t\mathrm{d}u = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f\left(t+\frac{\alpha}{2}\right)f^*\left(t-\frac{\alpha}{2}\right)\mathrm{e}^{-\mathrm{j}\alpha u}\mathrm{e}^{-\mathrm{j}t\xi}\mathrm{e}^{\mathrm{j}u\tau}\mathrm{d}\alpha\mathrm{d}t\mathrm{d}u = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f\left(t+\frac{\alpha}{2}\right)f^*\left(t-\frac{\alpha}{2}\right)\mathrm{e}^{-\mathrm{j}t\xi}\left\{\int_{-\infty}^{\infty}\mathrm{e}^{\mathrm{j}(\tau-\alpha)u}\mathrm{d}u\right\}\mathrm{d}\alpha\mathrm{d}t. \tag{4.927}$$

The integral inside the curly brackets by Example 4.10 is $2\pi\delta(\tau-\alpha)$*. Then*

$$A(\xi;\tau) = \int_{-\infty}^{\infty}\mathrm{e}^{-\mathrm{j}t\xi}\left\{\int_{-\infty}^{\infty} f\left(t+\frac{\alpha}{2}\right)f^*\left(t-\frac{\alpha}{2}\right)\delta(\tau-\alpha)\mathrm{d}\alpha\right\}\mathrm{d}t = \int_{-\infty}^{\infty}\mathrm{e}^{-jt\xi} f\left(t+\frac{\tau}{2}\right)f^*\left(t-\frac{\tau}{2}\right)\mathrm{d}t. \tag{4.928}$$

This is the ***ambiguity function*** *of the signal, which was defined as the Fourier transform of the Wigner distribution with respect to* both *its arguments. We can see from (4.928) that it is the Fourier transform of function* $r(t;\alpha)\equiv f\left(t+\frac{\alpha}{2}\right)f^*\left(t-\frac{\alpha}{2}\right)$ *with respect to its* ***first*** *argument only. The Wigner distribution is the Fourier transform of the same function with respect to its* ***second*** *argument. The ambiguity function takes its name from radar applications where it represents a wave travelling forward and backward while Doppler shifted due to the relative movement of the wave source and the target. In* $A(\xi;\tau)$*, frequency parameter* ξ *is associated with the Doppler shift, while parameter* τ *with the time delay of the wave, i.e. with the target range.*

How is the Wigner distribution used for the analysis of digital signals?

The Wigner distribution as defined by Equation (4.895) is not applicable to digital signals. To use it for the analysis of digital signals we have first to adapt it for them. The version of the Wigner distribution appropriate for digital signals is called **pseudo-Wigner distribution**.

What is the pseudo-Wigner distribution?

The most important use of the Wigner distribution for the analysis of digital signals is the extraction of local signal information. Locality implies the use of a window, so the first modification is that the Wigner spectrum is computed from a windowed part of the signal. A commonly used window for that is the **Kaiser window**. However, any other window might also be used. In the simplest of cases the rectangular window may be used. The second adaptation is that instead of computing the continuous Fourier transform of the signal, we compute the discrete Fourier transform. The third adaptation is that we cannot have shifts in the argument of the signal $\alpha/2$ since the signal is defined at integer positions only. So, the pseudo-Wigner distribution for a $2N+1$ long 1D digital signal $f(n)$, where $n=-N,-N+1,\ldots,-1,0,1,\ldots,N-1,N$, is defined as

$$PWD(m;p) = \sum_{k=-N}^{N} h(k)f(m+k)f^*(m-k)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{2N+1}pk} \tag{4.929}$$

where $h(k)$ is a window.

The use of the discrete Fourier transform has significant implications. For a start, the use of the rectangular window causes the creation of extra frequencies that are due to the window and nothing else. We have seen examples of such distortions in Chapter 3. The use of other than the rectangular window may ameliorate this problem. The other implication concerns the sampling frequency of the signal. When we deal with digital signals, we know that we have sampled them with certain frequency. The signals are band-limited and the sampling frequency is such that aliasing is avoided. To compute the Wigner distribution we form first function $r(t;\alpha)$ defined by Equation (4.894). This is essentially a multiplication of the original signal with itself. When two signals are multiplied, their Fourier transforms are convolved, so the Fourier transform of $r(t;\alpha)$ is the result of convolving the Fourier transform of the original signal with itself. This means that function $r(t;\alpha)$ has bandwidth double that of the original signal. Doubling the bandwidth may create aliasing, unless the original signal had been sampled twice as densely as it was necessary to avoid aliasing, i.e. at twice the **Nyquist frequency**.

What is the Kaiser window?

The Kaiser window is defined as

$$h_N(k) = \begin{cases} \frac{I_0\left[\alpha\sqrt{1-\left(\frac{k}{N}\right)^2}\right]}{I_0(\alpha)} & \text{for } |k| \le N \\ 0 & \text{otherwise} \end{cases} \tag{4.930}$$

where $I_0(x)$ is the modified Bessel function of the first kind and of zero order, and α is a parameter that allows us to trade between the central and the side lobes of the window. This window is appropriate for isolating segments of signals $2N+1$ long.

Bessel function $I_0(x)$ may be approximated by a polynomial, with error less than 1.9×10^{-7}, as follows:

$$I_0(x) = \begin{cases} 1 + 3.5156229t^2 + 3.0899424t^4 + 1.2067492t^6 \\ +0.2659732t^8 + 0.0360768t^{10} + 0.0045813t^{12} & \text{for } -3.75 \le x \le 3.75 \\ \\ \frac{1}{\sqrt{x}}e^x\left(0.39894228 + 0.01328592t^{-1} + 0.00225319t^{-2}\right. & \\ -0.00157565t^{-3} + 0.00916281t^{-4} - 0.02057706t^{-5} & \\ \left.+0.02635537t^{-6} - 0.01647633t^{-7} + 0.00392377t^{-8}\right) & \text{for } 3.75 \le x < \infty \end{cases} \tag{4.931}$$

In these formulae $t \equiv x/3.75$.

Figure 4.197 shows the Kaiser window for various values of α and for $N = 31$.

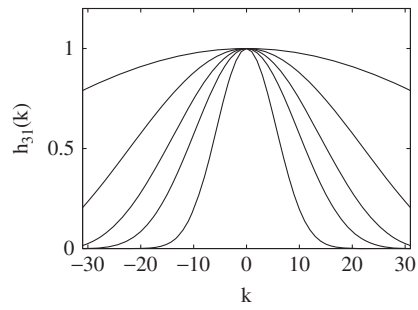


Figure 4.197 Kaiser window for $\alpha = 1,\ 3,\ 6,\ 11,\ 31$, from top to bottom, respectively.

Example B4.262

Compute the complex Fourier series of a train of delta functions defined as:

$$\delta_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \qquad \text{for} -\frac{T}{2} < t < \frac{T}{2}. \tag{4.932}$$

This may be considered as a periodic function with period T as shown in Figure 4.198, defined as:

$$f(t) = \delta(t) \qquad \text{for} -\frac{T}{2} < t < \frac{T}{2}. \tag{4.933}$$

Then its complex Fourier series is

$$\delta_T(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t} \tag{4.934}$$

where $\omega_0 = 2\pi/T$ and the coefficients c_n are by definition equal to:

$$c_n \equiv \frac{1}{T}\int_{-T/2}^{T/2} f(t)e^{-jn\omega_0 t}dt = \frac{1}{T}\int_{-T/2}^{T/2} \delta(t)e^{-jn\omega_0 t}dt = \frac{1}{T}. \tag{4.935}$$

So, the Fourier series of a train of delta functions is:

$$\delta_T(t) = \frac{1}{T}\sum_{n=-\infty}^{\infty} e^{jn\frac{2\pi}{T}t}. \tag{4.936}$$

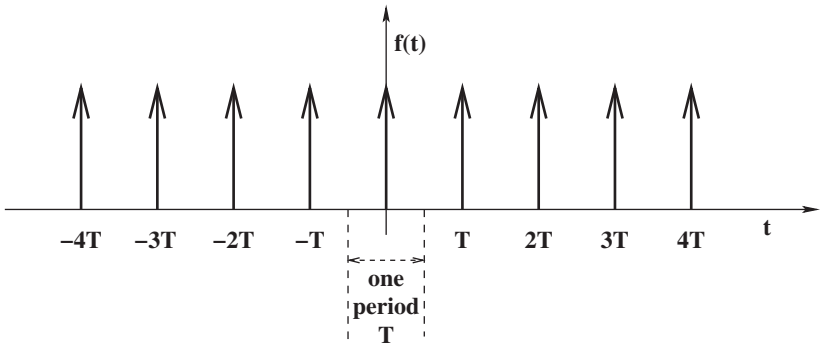


Figure 4.198 A train of delta functions T distance apart may be thought of as a periodic function consisting of a single δ function and having period T.

Example B4.263

Compute the Fourier transform of a train of delta functions defined by (4.932).
Let us take the Fourier transform of both sides of (4.936):

$$\int_{-\infty}^{\infty} \delta_T(t)e^{-j\omega t}dt = \frac{1}{T}\sum_{n=-\infty}^{\infty}\int_{-\infty}^{\infty} e^{jn\frac{2\pi}{T}t}e^{-j\omega t}dt$$
$$= \frac{1}{T}\sum_{n=-\infty}^{\infty}\int_{-\infty}^{\infty} e^{jt\left(\frac{2\pi n}{T}-\omega\right)}dt$$

$$= \frac{1}{T}\sum_{n=-\infty}^{\infty} 2\pi\delta\left(\frac{2\pi n}{T} - \omega\right)$$
$$= \frac{2\pi}{T}\sum_{n=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi n}{T}.\right) \tag{4.937}$$

Here we made use of the result of Example 4.10 and of the fact that the δ function is an even function. So, the Fourier transform of a train of delta functions at a distance T apart is also a train of delta functions at frequency $2\pi/T$ apart.

What is the Nyquist frequency?

The Nyquist frequency is the minimum frequency with which a band-limited function should be sampled so that it can be fully recovered from its sampled values.

This is better explained with the help of Figure 4.199. Let us consider the band-limited continuous signal $f(t)$ shown in 4.199a. Its Fourier spectrum is shown in 4.199b. Let us also consider the comb (or train) of delta functions shown in 4.199c. The Fourier spectrum of this comb is also a comb of delta functions shown in 4.199d. Note that if the teeth of the comb are at a distance T apart, the teeth of the comb in the Fourier domain are $2\pi/T$ apart, i.e. the denser the teeth are in the real space, the further apart they are in the frequency space. If we multiply function $f(t)$ with the comb, we convert it from a continuous function to a sampled digital function, as shown in Figure 4.199e. Multiplication of two functions in the real domain is equivalent to convolution of their Fourier spectra. The result of convolving the band-limited spectrum of the original function $f(t)$ with the comb of Figure 4.199c produces the spectrum of Figure 4.199f. So, if we assume that in practice we have at our disposal only the sampled function 4.199c, we shall be able to have full information about the continuous function, only if the teeth of the comb in the Fourier space were far enough apart for the repetitions of the Fourier spectrum of the original function not to overlap. If the teeth of the comb in the frequency domain were too close, the replicas of the spectrum would overlap as shown in Figure 4.200b. Then the high frequencies of the original signal would be destroyed. This is intuitively obvious. Imagine that we sample the function very sparsely, i.e. imagine that the teeth of the comb in the real domain are very far apart. Then we shall miss the details of the function, and the sampled function will not be identical with the continuous one. Not only will the details of the continuous function be missing, but in addition false details may appear, since points which in reality were well separated may come next to each other due to undersampling. This is equivalent to saying that the sampled function does not capture the high frequencies of the continuous function correctly, and in signal processing terms it is expressed by saying that the function has been **aliased** because the teeth of the sampling comb in the **frequency** domain were too close to each other. When the teeth of the sampling comb in the real domain are exactly as far apart as they should be for the repeated spectra of the function just to touch, without any overlap (see Figure 4.199f), we say that we have sampled the signal with the **Nyquist interval** or with the **Nyquist frequency**. Referring to Figure 4.199, we see that the Nyquist frequency corresponds to the case when the distance between the teeth of the sampling comb in the frequency space is equal to the bandwidth of the signal:

$$\frac{2\pi}{T} = 2\omega_0 \Rightarrow \frac{2\pi}{T} = 4\pi f_0 \Rightarrow \frac{1}{T} = 2f_0 \Rightarrow f_N = 2f_0. \tag{4.938}$$

So, the Nyquist frequency f_N is twice the highest frequency present in the signal. If the signal is sampled with the Nyquist frequency we say that it is **critically sampled**. It is best when the signal is sampled at higher frequencies, as shown in Figure 4.200a. Sampling at lower frequencies leads to aliasing, i.e. destruction of high-frequency details of the signal.

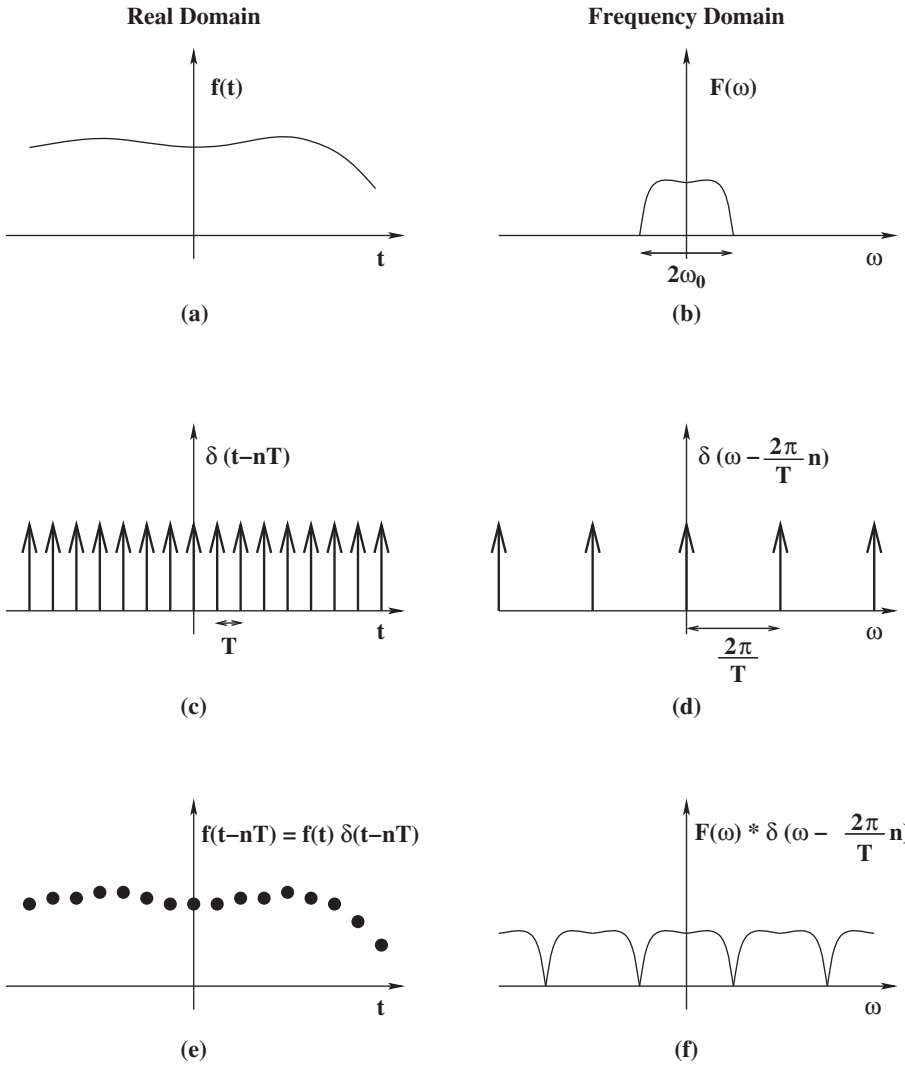


Figure 4.199 (a) A continuous band-limited signal and its Fourier spectrum shown in (b). When sampled with the comb in (c) its Fourier spectrum is convolved with the Fourier spectrum of the comb shown in (d). The result is shown in (f). In this example the teeth of the sampling comb have been chosen to be close enough, for the teeth of the comb in the frequency space to be exactly as far apart as is necessary for the repeated spectra of the function not to overlap. This is the case of sampling at the Nyquist frequency.

Why does the use of the pseudo-Wigner distribution require signals that have been sampled at twice their Nyquist frequency?

Let us consider a band-limited signal $f(t)$ with bandwidth $2\omega_0$. Let us say that its Fourier transform is $\mathcal{F}(\omega)$, and its Fourier spectrum is $F(\omega)$. Then the Fourier transform of $f(t-a)$ will be $\mathcal{F}(\omega)e^{ja}$. The Fourier transform of $f(t+a)$ will be $\mathcal{F}(\omega)e^{-ja}$. So, both $f(t-a)$ and $f(t+a)$ will have the same Fourier spectrum as $f(t)$, namely $F(\omega)$. Since we multiply these two functions in order to compute the pseudo-Wigner distribution, their Fourier spectra are convolved, i.e. the Fourier spectrum of $f(t-a)f(t+a)$ is $F(\omega)\bigstar F(\omega)$. $F(\omega)\bigstar F(\omega)$, however, will have double the bandwidth of $F(\omega)$, i.e. its

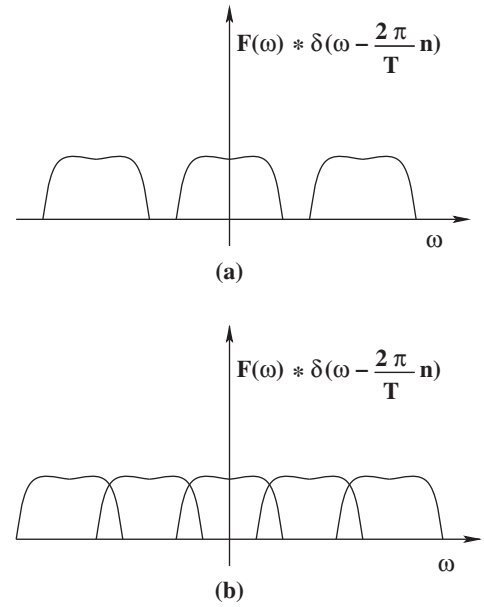


Figure 4.200 If the teeth of the sampling comb in the real domain are closer than the Nyquist interval, the repeated spectra of the function in the frequency space are well separated, as shown in (a). If the teeth of the sampling comb in the real domain are further apart than the Nyquist interval, the repeated spectra of the function in the frequency space overlap, causing aliasing, as shown in (b).

bandwidth will be $4\omega_0$. This means that when this function is sampled, unless it is sampled at twice the Nyquist frequency of $f(t)$, there will be aliasing. This is shown in Figure 4.201.

Should we worry about aliasing when we use the pseudo-Wigner distribution for texture analysis?

Aliasing is always a problem. However, if we are not really interested in reproducing the signal, but simply computing some numbers that characterise it, even if these numbers characterise its aliased version, they may still be appropriate for distinguishing the signal from other signals (i.e. one texture from other textures). Aliasing may become a problem only if the aliased versions of two textures are identical. Then, of course, the features we shall compute from them will not be able to distinguish the two textures. It is rather unlikely that the data will contrive in such a way that the two aliased textures will become the same, so aliasing may not be of too much concern when using the Wigner distribution for texture segmentation.

How is the pseudo-Wigner distribution defined for the analysis of images?

To use the pseudo-Wigner distribution for images we must first define it in 2D:

$$PWD(m,n;p,q) = \sum_{k=-M}^{M}\sum_{l=-N}^{N} h(k,l)f(m+k,n+l)f(m-k,n-l)e^{-j\frac{2\pi}{2N+1}pk-j\frac{2\pi}{2M+1}ql}. \quad (4.939)$$

Here $h(k,l)$ is a 2D window, for example a 2D Kaiser window consisting of two 1D Kaiser windows multiplied with each other. The sub-image this function is applied to is supposed to be of size $(2M+1)\times(2N+1)$.

How can the pseudo-Wigner distribution be used for texture segmentation?

We can compute the Wigner spectrum for each point in the image by considering a window around each pixel. The components of the Wigner spectra then may be used to construct features. It is even possible to perform feature reduction by using principal component analysis.

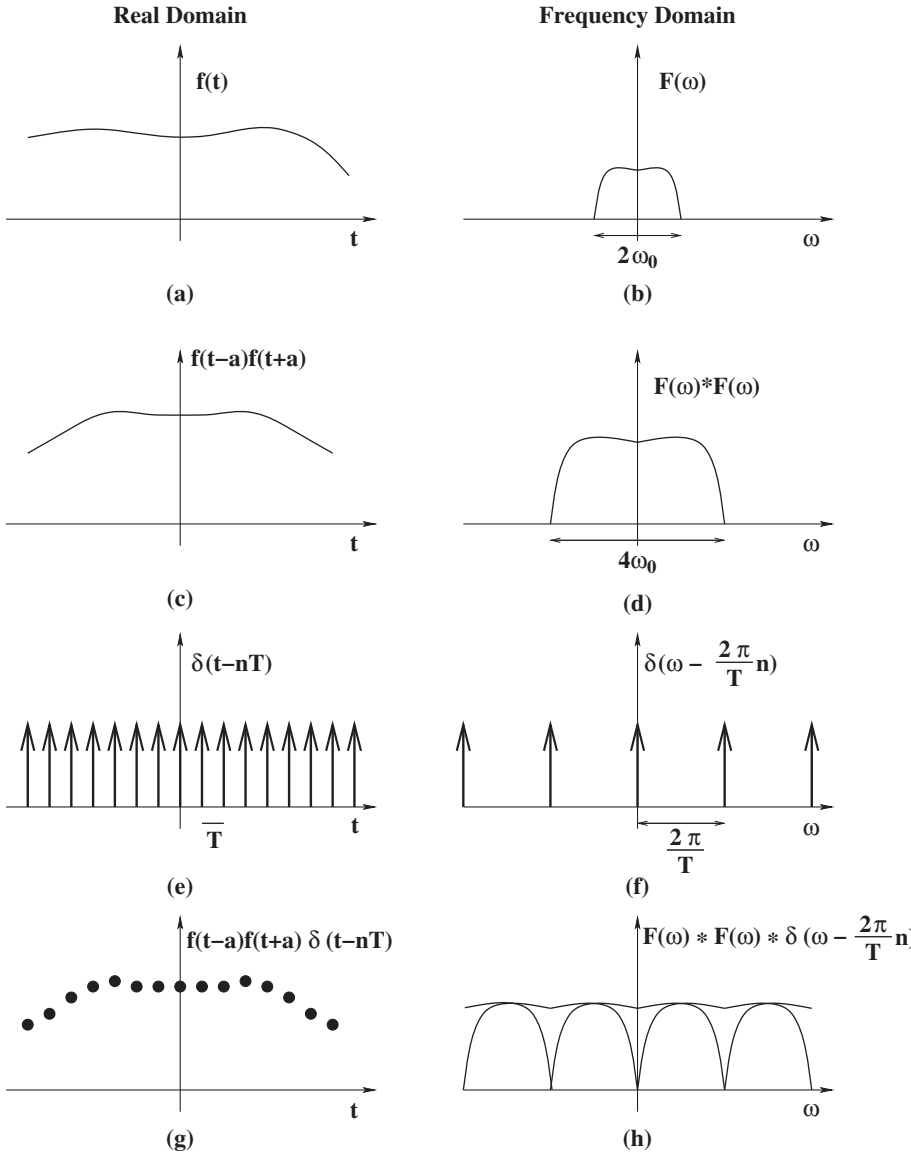


Figure 4.201 (a) A continuous band-limited signal $f(t)$ and its Fourier spectrum in (b). Function $f(t-a)f(t+a)$ in (c) and its Fourier spectrum in (d). The sampling comb in (e) and its Fourier spectrum in (f). The sampled function $f(t-a)f(t+a)$ and its aliased Fourier spectrum if the sampling comb used is at the Nyquist frequency of the original signal. If the sampling comb used were at twice the Nyquist frequency of the original signal, no aliasing would have arisen.

Example 4.264

Prove that

$$PWD(m,n;p,q) = PWD(m,n;-p,-q). \tag{4.940}$$

Let us start from the definition of the pseudo-Wigner distribution (4.939) and use it to compute $\text{PWD}(m, n; -p, -q)$*:*

$$\text{PWD}(m,n;-p,-q) = \sum_{k=-M}^{M}\sum_{l=-N}^{N} h(k,l)f(m+k,n+l)f(m-k,n-l)e^{j\frac{2\pi}{2N+1}pk+j\frac{2\pi}{2M+1}ql}. \quad (4.941)$$

We define new variables of summation in (4.941), $\tilde{k} \equiv -k$ *and* $\tilde{l} \equiv -l$*. Then:*

$$\text{PWD}(m,n;-p,-q) = \sum_{\tilde{k}=M}^{-M}\sum_{\tilde{l}=N}^{-N} h(k,l)f(m+k,n+l)f(m-k,n-l)e^{-j\frac{2\pi}{2N+1}p\tilde{k}-j\frac{2\pi}{2M+1}q\tilde{l}}. \quad (4.942)$$

Variables $\tilde{k}$ *and* $\tilde{l}$ *are dummy, and the result of a summation does not depend on the order in which we sum the terms, so the double sum on the right-hand side of (4.942) is the same as the double sum on the right-hand side of (4.939), which proves property (4.940).*

Example 4.265

Use features constructed from the Wigner distribution to segment the image of Figure 4.81.

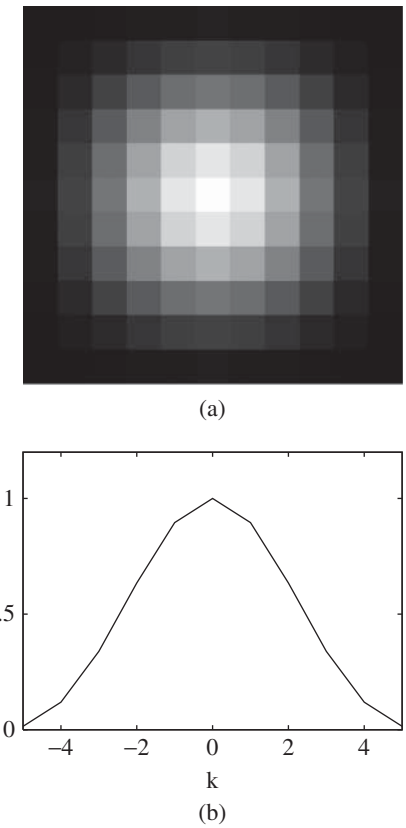


Figure 4.202 In (a) the 2D Kaiser window created by multiplying two 1D Kaiser windows of size 11 with $\alpha = 6$, shown in (b). Source: Maria Petrou.

(Continued)

Example 4.265 (Continued)

Figure 4.202 shows the 2D Kaiser window used in the experiments. Its size is 11×11 *and* $\alpha = 6$. *Since the window we use is* 11×11, *each Wigner spectrum we use to characterise a pixel consists of 121 components. We may use one component at a time to create feature maps that represent the image in 121 different ways. However, because of the symmetry of the Wigner spectrum, 60 of those maps are identical to the remaining 60, so we have only 61 independent feature maps. These are shown in Figure 4.203. Figure 4.204 shows the segmentation result obtained by using all 61 features to segment the image.*

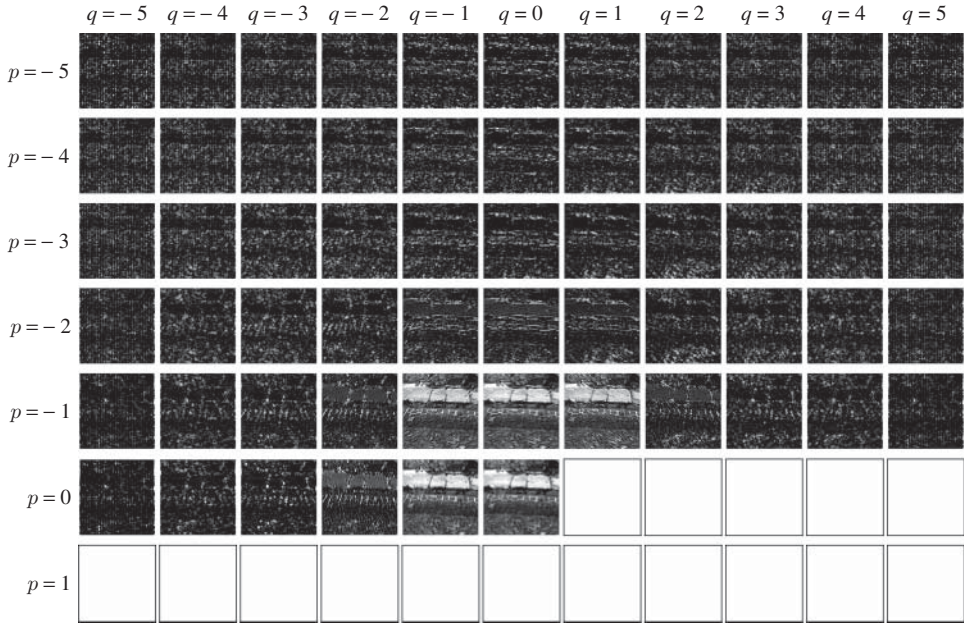
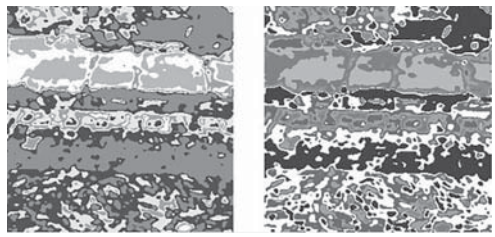


Figure 4.203 Wigner feature maps obtained for the image of Figure 4.81 using the Kaiser window shown in Figure 4.202. Each frame has been produced by keeping for each pixel a single value for its Wigner spectrum, identified by the values of p and q. Because of property (4.940) only half of the frames are shown. For displaying purposes alone, the 2% extreme values of each feature were mapped either to 0 or 255 and the remaining values were linearly scaled to the range [0,255]. The empty frames indicate that they are not displayed because they are identical with the frames at symmetric positions about the last frame displayed, the one that corresponds to $p = 0$ and $q = 0$. Source: Maria Petrou.



$K = 6$ $K = 8$

Figure 4.204 Segmentation results obtained using the deterministic annealing algorithm for the 61 Wigner features shown in Figure 4.203, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.

We may use principal component analysis (see Example 4.230) to achieve feature reduction. Figure 4.203 shows the feature maps produced from the first six eigenvectors of the covariance matrix of the pixels in the 61-dimensional feature space. Figure 4.206 shows the segmentation result obtained if we use only the first three of these PCA features. Figure 4.207 shows the segmentation result obtained if no Kaiser window were used, i.e. if the 11×11 *Wigner spectrum were computed using just a rectangular* 11×11 *neighbourhood around each pixel. Figure 4.208 shows the segmentation result obtained for this case if we perform PCA and use only the first three eigen-features.*

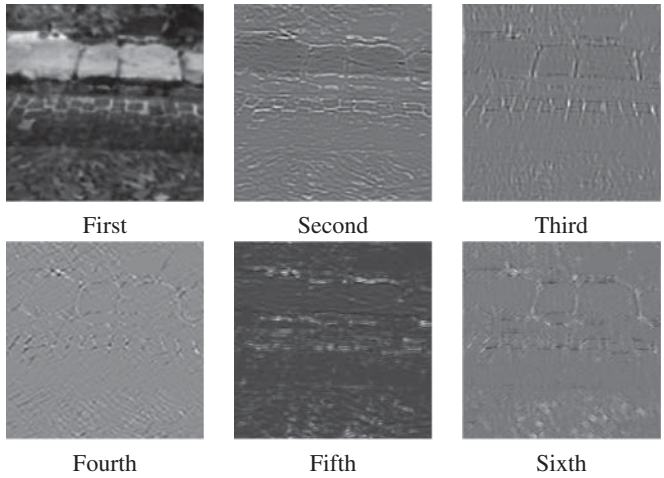


Figure 4.205 First six feature maps constructed using principal component analysis of the 61 Wigner features shown in Figure 4.203. Source: Maria Petrou.

Figure 4.206 Segmentation results obtained using the deterministic annealing algorithm for the first three feature maps shown in Figure 4.205, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.

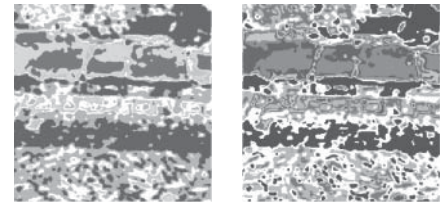
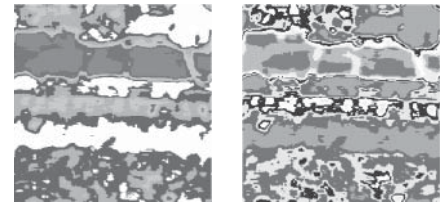
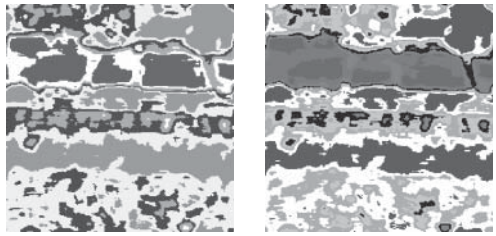


Figure 4.207 Segmentation results obtained using the deterministic annealing algorithm for the 61 Wigner features computed using a rectangular flat window, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.



(Continued)

Example 4.265 (Continued)



$K = 6$ $K = 8$

Figure 4.208 Segmentation results obtained using the deterministic annealing algorithm with the first three eigen-features computed from the 61 Wigner features constructed using a rectangular flat window around each pixel, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.

Further, we may not use the Wigner features directly, but compute energies from them, by averaging their values over a local neighbourhood around each pixel. The energy feature maps created this way are shown in Figure 4.209. The segmentations obtained by using these features are shown in Figure 4.210. We may again perform feature selection using PCA. The eigen-feature maps obtained which correspond to the six largest eigenvalues are shown in Figure 4.209 and the segmentation obtained from the first three eigen-features is shown in Figure 4.212.

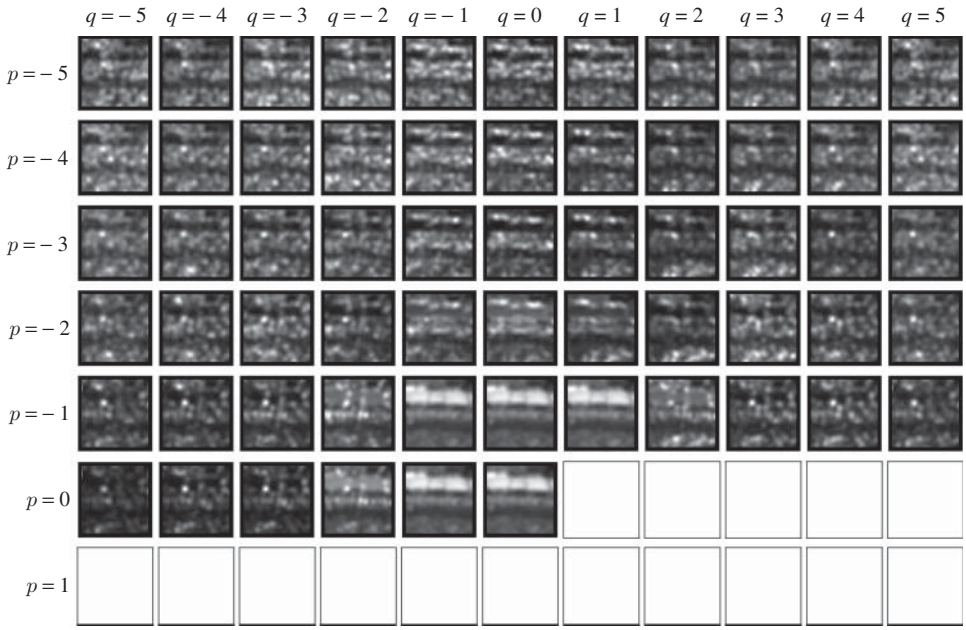
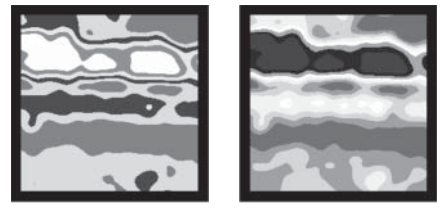
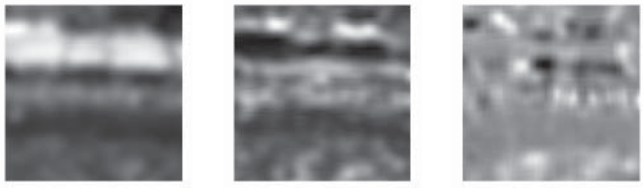


Figure 4.209 Energy features constructed from the components of the Wigner spectrum of each pixel, obtained by using the Kaiser window shown in Figure 4.202. The local energy values were computed from the corresponding feature maps of Figure 4.203 by using a Gaussian window of size 29×29 with $\sigma = 7$. For displaying purposes alone, the 2% extreme values of each feature were mapped either to 0 or 255 and the remaining values were linearly scaled to the range [0,255]. The missing or empty panels are not shown because each of them is identical with a panel already on display. Source: Maria Petrou.

Figure 4.210 Segmentation results obtained using the deterministic annealing algorithm with the 61 Wigner energy features shown in Figure 4.209, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.



$K = 6$ $K = 8$

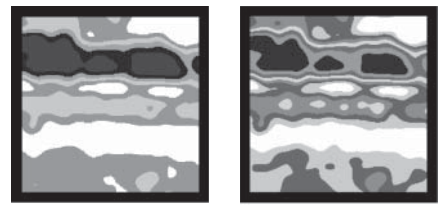


First Second Third

Fourth Fifth Sixth

Figure 4.211 The first six eigen-feature maps produced by PCA for the 61 Wigner energy features shown in Figure 4.209. Source: Maria Petrou.

Figure 4.212 Segmentation results obtained using the deterministic annealing algorithm for the first three eigen-features, shown in Figure 4.211, produced from the 61 Wigner energy features that were computed with the help of a Kaiser window, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.

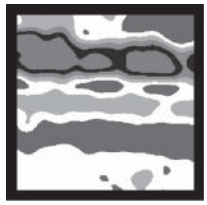


$K = 6$ $K = 8$

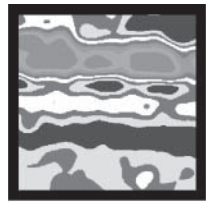
Finally, we may compute energy features from Wigner features computed without the use of Kaiser window. The segmentation result in this case is shown in Figure 4.213 when all 61 energy features were used, and in Figure 4.214 when only the first three eigen-features were used.

(Continued)

Example 4.265 (Continued)

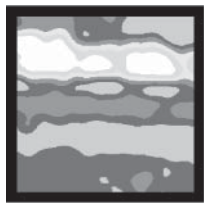


$K = 6$

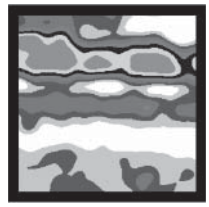


$K = 8$

Figure 4.213 Segmentation results obtained using the deterministic annealing algorithm with the 61 Wigner energy features computed without the Kaiser window, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.



$K = 6$



$K = 8$

Figure 4.214 Segmentation results obtained using the deterministic annealing algorithm with the first three eigen-features of the Wigner energy features computed without a Kaiser window, for two different values of the expected number K of distinct clusters. Source: Maria Petrou.

4.12 Convolutional Neural Networks for Textures Feature Extraction

Why do we require convolutional neural networks for texture analysis?

With the rise of deep learning, there was a drastic change in research and development of computer vision, in around 2010. It has brought higher levels of performance to classify big visual data at a competition contest, the Imagenet Large Scale Visual Recognition Challenge (ILSVRC). Table 4.8 shows the performance of classification accuracy using the large-scale image dataset called Imagenet. At ILSVRC2010 and ILSVRC201, SIFT, LBP and SVM were traditional trends to use feature extraction and classification tasks. However, in 2012, a methodology, so-called **deep learning, Alexnet**, based on convolutionl neural networks (CNNs) achieved state-of-the- art accuracy, in which error rate was 16.4%, in the object classification task. After three years, in 2015, the classification accuracy of Resnet was greatly improved to reach the error rate 3.5%, while the human error rate was 5.1% [85]. Although the CNN is an extension of neural networks that developed in the 1980s and 1990s, ILSVRC2012 should be deemed as a turning point for neural networks. In addition, from 2013 onwards, the CNN-based methods greatly developed in the field of computer vision, machine learning and their many applications. Therefore the CNN immediately became a powerful tool for feature extraction of images, and in around 2013, several researchers began to apply it for texture classification.

Although the analysis of several texture features has been developed continually so far, deep learning innovation has also contributed to the development of texture analysis research. On the other hand, deep learning performance is always dependent on the quality of the large-scale dataset. Li mentioned that "the contest itself is made possible by the Imagenet database, an immense collection of more than 14 million images that have been identified by humans" [85].

Table 4.8 Classification results at ILSVRC (http://www.image-net.org/).

Conpetition	Error rate	Methods
ILSVRC2010	28.2%	Combination of SIFT and LBP features
ILSVRC2011	25.8%	High-dimensional image signatures, one-versus-all linear SVMs, compression using product quantization
ILSVRC2012	16.4%	**Large-scale deep neural networks** with 60 million parameters,
.		hidden-unit dropout method
ILSVRC2013	11.7%	Several large deep convolutional networks averaged together
ILSVRC2014	6.7%	Improved CNN architecture, multi-scale method from Hebbian theory (allowed to use other CNN-training data that are not provided)

According to the contest reports, there are two kinds of contributions to international societies in the following:

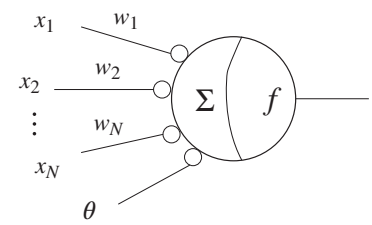
1. Precision improvement by learning deep neural networks
2. Construction of the large-scale data set.

Up to this section, we have introduced several feature extraction methods, a so-called a hand-crafted approach that is made by explicit theories or logics. On the other hand, the CNN-based methods are called a neural network-based approach that is an imitation of the human visual system. While the former is visually explainable, the latter has a kind of blackbox property for extracting features from images. In this section, we introduce some basic models on deep learning or neural networks and several texture datasets. Firstly a neuron model, the Rosenblat learning machine called Perceptron, Hebbian theory, and the back-propagation learning method will be explained.

The basis of a neural network is the Mcculloch and Pitts neuron model

The human brain has approximately 86 billion neurons on average, which was estimated by Herculano-Houzel in 2009[37]. In the 1940s, biophysical or psychological researchers put more effort into research and development on modelling neurons in the brain as mathematical functions. Through analysis of behavior, they recognised that there exists a gap between biological and artificial neuron models. However, they set a target on modeling neurons and their connections for analysing visual information processing in the brain. Unquestionably, an artificial neuron model is based on the CNN. In 1943, Mcculloch and Pitts modelled a computation of neurons as shown in Figure 4.215.

Figure 4.215 A neuron model: $g(\mathbf{x})$.



Box 4.19 Mcculloch and Pitts neuron model

Let us represent a mathematical model of the artificial neuron proposed by Mcculloch and Pitts [65]. Starting from a linear combination of a weight vector $\mathbf{w} = (w_1, \cdots, w_i, \cdots, w_N)^T$ and a pattern vector $\mathbf{x} = (x_1, \cdots, x_i, \cdots, x_N)^T$, we formulate the mathematical model as follows:

$$g(\mathbf{x}) = f\left(\sum_{i=1}^{N} w_i x_i + \theta\right) = f(\mathbf{w} \cdot \mathbf{x} + \theta) \tag{4.943}$$

where f is an activation function called a threshold logic unit, and θ is a threshold value. There are several activation functions, which will be discussed later, but the simplest $f(z)$ is a function expressed as:

$$f(z) = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ otherwise} \end{cases}. \tag{4.944}$$

The psychologist Hebb investigated the behaviour of neuron connections described in his book published in 1936 [36]. His neuropsychological theory is called **Hebbian theory** or **Hebb's rule**. Many researchers commonly used the Hebbian theory for long-term potentiation.

Box 4.20 Hebbian theory

Hebbian theory and the basis learning models are based on the following assumption as seen on page 67 of his book:

"When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased".

This neuron activation is the basis of memory occurring in the brain. If an activation occurred at two neurons A and B, the connection with a synapse between A and B is strengthened. The study of several learning machines and neural networks was pushed forward using the Hebbian theory.

What is the Rosenblatt learning model: Perceptron?

In 1958, based on the Hebbian theory, a learning machine called the **Perceptron** was proposed by Rosenblat [82]. The simple Perceptron is shown in Figure 4.216. A pattern vector $\mathbf{x}$ in a sensory

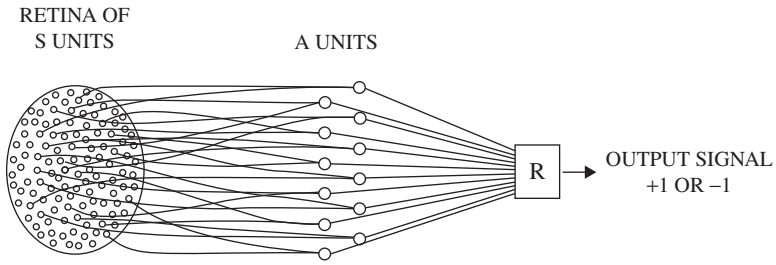


Figure 4.216 A simple Perceptron model [83].

layer S is mapped to an association-layer A with randomly connected wires. That means each node in A has a set of randomly located origin points in the retina. Although the connections between the sensory layer S and the association layer A are random, we assume this mapping keeps a linear separability property. The Perceptron excluded the analysis of these connections between S and A from consideration in the learning process.

A receptive layer R consists of a categorical node with a linear combination between hidden nodes values and their connection weights. The input pattern vector in a sensory layer is normalised to a unit vector, and the output signal in the receptive layer has a value of -1 or 1. At first, the initial values of all weights are set randomly.

The Perceptron is able to classify two pattern classes, C_1 and C_2. Let a set of L training samples (vectors) be $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L\}$, and a set of the corresponding labels be $\tau = \{\tau_1, \tau_2, \cdots, \tau_L\}$, where for any i:

$$\tau_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \in C_1 \\ -1 & \text{if } \mathbf{x}_i \in C_2 \end{cases}. \tag{4.945}$$

For training, we use $\mathbf{X}$ repeatedly to generate a set of training data $\{\mathbf{x}^{(t)}, \tau^{(t)} | t = 1, 2, \cdots\}$, where $\mathbf{x}^{(1)} = \mathbf{x}_1, \mathbf{x}^{(2)} = \mathbf{x}_2, \cdots, \mathbf{x}^{(L)} = \mathbf{x}_L, \mathbf{x}^{(L+1)} = \mathbf{x}_1, \ldots, \mathbf{x}^{(2L)} = \mathbf{x}_L, \ldots$. For an input pattern vector $\mathbf{x}$ of the network, we define an evaluation function of an error between the target τ and the output of the network *out* for learning the weight vector $\mathbf{w}$ and the threshold θ as:

$$\text{err} = \tau - \text{out} = \tau - \text{Sign}(\mathbf{w} \cdot \mathbf{x} + \theta). \tag{4.946}$$

The learning procedure of the simple Perceptron is described as follows:

Step 1: Initialize the weight vector $\mathbf{w}$ and the threshold value θ at random.
Step 2: Set time $t = 1$.
Step 3: Repeat steps 4–6 at time t until it converges.
Step 4: Calculate the error using $\text{err}_t \leftarrow \tau^{(t)} - \text{Sign}(\mathbf{w} \cdot \mathbf{x}^{(t)} + \theta)$.
Step 5: if $(\text{err}_t \neq 0)$, then update the parameters as follows:
$\mathbf{w} \leftarrow \mathbf{w} + \eta \times err \times \mathbf{x}^{(t)}$, and $\theta \leftarrow \theta + \eta \times \text{err}$.
Step 6: Set $t \leftarrow t + 1$.

The activation function in Perceptron uses a sign function Sign denoted as:

$$\text{Sign}(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{otherwise} \end{cases}. \tag{4.947}$$

At step 3, "converge" means that if the weight vector $\mathbf{w}$ and the threshold value θ are unchanged for all training samples $\mathbf{X}$, then the procedure terminates. After training these parameters, $\mathbf{w}$ and θ, the output signal for an unknown vector $\mathbf{x}$ is obtained as a classification result:

$$\text{Out}(\mathbf{x}) = \text{Sign}(\mathbf{w} \cdot \mathbf{x} + \theta) \tag{4.948}$$

Finally the classification boundary is given by $\mathbf{w} \cdot \mathbf{x} + \theta = 0$.

Example 4.266

Let us consider two-dimensional samples of an AND gate, as shown in Table 4.9. Describe the behavior of the simple Perceptron for learning the AND gate.

(Continued)

Example 4.266 (Continued)

Table 4.9 AND gate in two dimensional data.

Number	x_1	x_2	τ
1	−1	−1	−1
2	−1	1	−1
3	1	−1	−1
4	1	1	1

The learning process is described as follows:

Step 1: *We set randomly the weight parameters and the threshold value:*
$w_1 = 0.405$, $w_2 = 0.109$, $\theta = 0.117$
Step 2: *Set* $t = 1$.
Step 3: *The network does not converge.*
Step 4: *We calculate the output of the network:*
$\text{Out} = w_1 \times x_1^{(1)} + w_2 \times x_2^{(1)} + \theta = 0.405 \times (-1) + 0.109 \times (-1) + 0.117 = -0.396$,
The error is $\text{err} = \tau^{(1)} - Sign(Out) = -1 - (-1) = 0$.
Step 5: *The weight vector and the threshold do not change.*
Step 6: *Set* $t = 2$.
Step 3: *The network does not converge.*
Step 4: *We calculate the output of the network:*
$\text{Out} = 0.405 \times (-1) + 0.109 \times (1) + 0.117 = -0.177$,
The error is $\text{err} = \tau^{(2)} - Sign(Out) = -1 - (-1) = 0$.
Step 5: *The weights and the threshold do not change.*
Step 6: *Set* $t = 3$.
Step 3: *The network does not converge.*
Step 4: *We calculate the output of the network,*
$\text{Out} = -0.195 \times (1) + 0.709 \times (-1) + (-0.482) = 0.412$,
The error is $\text{err} = \tau^{(2)} - \text{Sign(Out)} = -1 - (1) = -2$
Step 5: *The weight vector and the threshold value are updated:*
$\mathbf{w} \leftarrow (0.405, 0.109) + 0.3 \times (-2) \times (1, -1) = (-0.195, 0.709)$,
$\theta \leftarrow 0.117 + 0.3 \times (-2) = -0.482$.
Step 6: *Set* $t = 4$.
Step 3: *The network does not converge.*
Step 4: *We calculate the output of the network,*
$\text{Out} = -0.195 \times (1) + 0.709 \times (1) + (-0.482) = 0.322$,
The error is $\text{err} = \tau^{(2)} - \text{Sign(Out)} = 1 - (1) = 0$
Step 5: *The weights and the threshold do not change.*
Step 6: *Set* $t = 5$.
Step 3: *The network does not converge.*
Repeating this process, we obtain the classification result as shown in Table 4.10. Finally the classification boundary is

$$1.005 \times x_1 + 0.709 \times x_2 - 0.482 = 0.$$

Table 4.10 Perceptron training process (AND gate).

t	(x_1, x_2)	tau	w_1	w_2	θ	Out	err
1	(−1,−1)	−1	0.405	0.109	0.117	−0.396	0
2	(−1,1)	−1	0.405	0.109	0.117	−0.177	0
3	(1,−1)	−1	−0.195	0.709	−0.482	0.412	−2
4	(1,1)	1	−0.195	0.709	−0.482	0.322	0
5	(−1,−1)	−1	−0.195	0.709	−0.482	−0.996	0
6	(−1,1)	−1	0.405	0.109	−1.082	0.422	−2
7	(1,−1)	−1	0.405	0.109	−1.082	−0.787	0
8	(1,1)	1	1.005	0.709	−0.482	−0.567	2
9	(−1,−1)	−1	1.005	0.709	−0.482	−2.196	0
10	(−1,1)	−1	1.005	0.709	−0.482	−0.777	0

Minsky and Papert pointed out that the simple Perceptron cannot solve exclusive OR (XOR) problem as shown in 4.14 published in 1969 [66]. This is a classical two-class classification problem, but the XOR problem is not linearly separable, which will be discussed later. That means we cannot draw a straight line as a boundary to classify the two classes in the two-dimensional space.

What is the Widrow and Hoff learning model?

In 1960, Widrow and Hoff proposed adaptive switching circuits as a learning method [97]. The Widrow–Hoff model has a single layer in this circuit with ADAptive LInear NEuron (ADALINE). The ADALINE is based on a two layers model, which have no hidden layers. After the proposal of ADALINE, they also proposed three-layer models, called MADALINE (Many ADALINE) [98], which is combined with several ADALINEs. Among them, the simple MADALINE is shown in Figure 4.217.

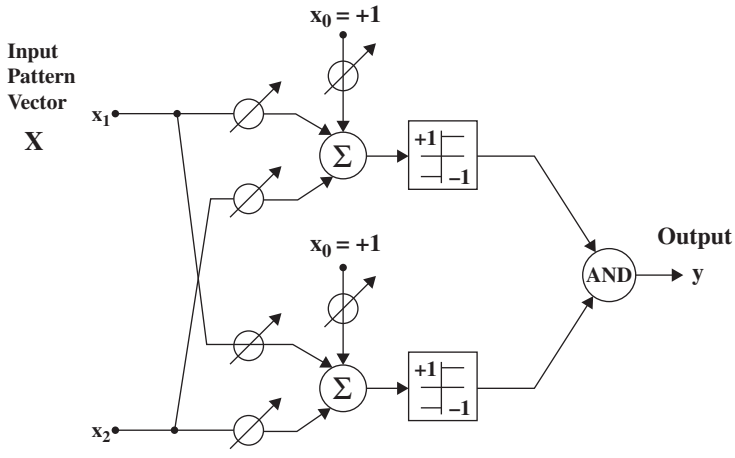


Figure 4.217 A two ADALINE model (three-layer MADALINE) [98].

The MADALINE can also solve linear separable problems, but it cannot solve XOR problems. A simple example of linear separable problem of an OR gate is shown in Table 4.11. The MADALINE network used in this example as shown in Figure 4.217 has one single output node.

The MADALINE has two input nodes x_1, x_2, four weight parameters $w_{11}, w_{12}, w_{21}, w_{22}$ between the input layer and the hidden layer, and two weight parameters v_1, v_2 between the hidden layer and output layer. In this case, v_1, v_2 are fixed equal to 1 for network training. And there are two threshold values, θ_1, θ_2, as the training parameters.

Using L training samples, $\mathbf{X}$, we prepare a set of data sequence $\mathbf{Z}$ for network training as follows:

$$
\begin{aligned}
\mathbf{Z} = \{ & (x_1^{(1)}, x_2^{(1)}, \tau^{(1)}) = (1, 1, -1), \\
& (x_1^{(2)}, x_2^{(2)}, \tau^{(2)}) = (1, -1, 1), \\
& (x_1^{(3)}, x_2^{(3)}, \tau^{(3)}) = (-1, 1, 1), \\
& (x_1^{(4)}, x_2^{(4)}, \tau^{(4)}) = (-1, -1, -1), \\
& (x_1^{(5)}, x_2^{(5)}, \tau^{(5)}) = (1, 1, -1), \\
& (x_1^{(6)}, x_2^{(6)}, \tau^{(6)}) = (1, -1, 1), \\
& (x_1^{(7)}, x_2^{(7)}, \tau^{(7)}) = (-1, 1, 1), \\
& (x_1^{(8)}, x_2^{(8)}, \tau^{(8)}) = (-1, -1, -1), \\
& \vdots \\
& (x_1^{(t)}, x_2^{(t)}, \tau^{(t)}) = (x_1^{(t+4)}, x_2^{(t+4)}, \tau^{(t+4)}), \\
& \vdots \\
\}.
\end{aligned}
$$

Given the data sequence $\mathbf{Z}$, the MADALINE learning procedure is described as follows:

Step 1: Initialize the weight parameters $w_{11}, w_{12}, w_{21}, w_{22}$ and the threshold values θ_1, θ_2 randomly.
Step 2: Set time $t = 1$.
Step 3: Repeat steps 4–7 at the time t until it converges.
Step 4: Given the input $x_1^{(t)}$ and $x_2^{(t)}$, the output y is calculated as follows:
$z_1' \leftarrow x_1^{(t)} \times w_{11} + x_2^{(t)} \times w_{12} + \theta_1$,
$z_2' \leftarrow x_1^{(t)} \times w_{21} + x_2^{(t)} \times w_{22} + \theta_2$,
$z_1 \leftarrow \text{Sign}(z_1')$,
$z_2 \leftarrow \text{Sign}(z_2')$,
$y \leftarrow z_1 \; AND \; z_2$.
Step 5: Calculate the error as $\text{err} \leftarrow \tau^{(t)} - y$.
Step 6: If $(\text{err} \neq 0)$, then update the weight parameters by
$w_{11} \leftarrow w_{11} + \eta \times (\tau^{(t)} - z_1') \times x_1^{(t)}$,
$w_{12} \leftarrow w_{12} + \eta \times (\tau^{(t)} - z_1') \times x_2^{(t)}$,
$w_{21} \leftarrow w_{21} + \eta \times (\tau^{(t)} - z_2') \times x_1^{(t)}$,
$w_{22} \leftarrow w_{22} + \eta \times (\tau^{(t)} - z_2') \times x_2^{(t)}$.
and also update the threshold values by
$\theta_1 \leftarrow \theta_1 + (\tau^{(t)} - z_1')$.
$\theta_2 \leftarrow \theta_2 + (\tau^{(t)} - z_2')$.
Step 7: Set $t \leftarrow t + 1$.

Example 4.267

Let us consider two-dimensional samples of an OR gate as shown in Table 4.11. Here $L = 4$. Describe the behavior of the MADALINE for learning the OR gate.

Table 4.11 OR gate in two-dimensional data.

Number	x_1	x_2	τ
1	−1	−1	−1
2	−1	1	1
3	1	−1	1
4	1	1	1

Step 1: *We set randomly the weight parameters* $w_{11} = 0.221$, $w_{12} = -0.393$, $w_{21} = 0.153$, $w_{22} = -0.005$ *and the threshold value* $\theta_1 = -0.021$, $\theta_2 = -0.084$.
Step 2: *Set* $t = 1$.
Step 3: *The network does not converge.*
Step 4: *We calculate the output y as follows:*

$$z_1' = w_{11} \times x_1^{(1)} + w_{12} \times x_2^{(1)} + \theta_1$$
$$= 0.221 \times (-1) + (-0.393) \times (-1) + (-0.021) = 0.151,$$
$$z_2' = w_{21} \times x_1^{(1)} + w_{22} \times x_2^{(1)} + \theta_2$$
$$= 0.153 \times (-1) + (-0.005) \times (-1) + (-0.084) = -0.232,$$
$$z_1 = \text{Sign}(x_1') = 1,\ z_2 = \text{Sign}(x_2') = -1, \textit{ So } y = z_1 \textit{ AND } z_2 = -1$$

Step 5: *The error is* $\text{err} = \tau^{(1)} - y = -1 - (-1) = 0$.
Step 6: *So the weight parameters and the threshold values do not change.*
Step 7: *Set* $t = 2$.
Step 3: *The network does not converge.*
Step 4: *We calculate the output y as follows:*

$$z_1' = 0.221 \times (-1) + (-0.393) \times (1) + (-0.021) = 0.635,$$
$$z_2' = 0.153 \times (-1) + (-0.005) \times (1) + (-0.084) = -0.244,$$
$$z_1 = \text{Sign}(x_1') = 1,\ z_2 = \text{Sign}(x_2') = -1, \textit{ So } y = z_1 \textit{ AND } z_2 = -1$$

Step 5: $\text{err} = \tau^{(2)} - (-1) = 1 - (-1) = 2$.
Step 6: *The weight parameters and the threshold values are updated as follows:*

$$w_{11} \leftarrow w_{11} + \eta \times (\tau^{(2)} - z_1') \times x_1^{(2)}$$
$$= 0.221 + 0.3 \times (1 - 0.635) \times (-1) = -0.378,$$
$$w_{12} \leftarrow w_{12} + \eta \times (\tau^{(2)} - z_1') \times x_2^{(2)}$$
$$= -0.393 + 0.3 \times (1 - 0.635) \times (1) = 0.206,$$
$$w_{21} \leftarrow w_{21} + \eta \times (\tau^{(2)} - z_2') \times x_1^{(2)}$$
$$= 0.153 + 0.3 \times (1 - (-0.244)) \times (-1) = -0.446,$$
$$w_{22} \leftarrow w_{22} + \eta \times (\tau^{(2)} - z_2') \times x_2^{(2)}$$
$$= -0.005 + 0.3 \times (1 - (-0.244)) \times (1) = 0.594,$$
$$\theta_1 \leftarrow \theta_1 + (\tau^{(2)} - z_1') = -0.021 + 0.3 \times (1 - 0.635) = 0.579;$$
$$\theta_2 \leftarrow \theta_2 + (\tau^{(2)} - z_2') = -0.084 + 0.3 \times (1 - (-0.244)) = 0.515.$$

(Continued)

Example 4.267 (Continued)

Step 7: *Set* $t = 3$.
Step 3: *The network does not converge.*
After repeating this process, we obtain the classification result as shown in Table 4.12.

Table 4.12 OR training process.

t	(x_1, x_2)	w_{11}	w_{12}	w_{21}	w_{22}	θ_1	θ_2	z'_1	z'_2	$t-y$
1	(−1,−1)	0.221	−0.393	0.153	−0.005	−0.021	−0.084	0.151	−0.232	0
2	(−1,1)	0.221	−0.393	0.153	−0.005	−0.021	−0.084	0.635	−0.244	2
3	(1,−1)	−0.378	0.206	−0.446	0.594	0.579	0.515	−0.006	−0.525	2
4	(1,1)	0.221	−0.393	0.153	−0.005	1.179	1.115	1.007	1.263	0
5	(−1,−1)	0.221	−0.393	0.153	−0.005	1.179	1.115	1.351	0.967	−2
6	(−1,1)	0.821	0.206	0.753	0.594	0.579	0.515	−0.035	0.355	2
7	(1,−1)	0.221	0.806	0.153	1.194	1.179	1.115	0.593	0.746	0
8	(1,1)	0.221	0.806	0.153	1.194	1.179	1.115	2.207	2.462	0
9	(−1,−1)	0.221	0.806	0.153	1.194	1.179	1.115	0.151	−0.233	0
10	(−1,1)	0.221	0.806	0.153	1.194	1.179	1.115	1.764	2.155	0

Let us consider a two-class classification problem, which finds a boundary between C_1 and C_2 as well. The evaluation of the MADALINE is performed by Least Mean Square (LMS) algorithm which is developed using a gradient steepest method. This learning approach is also called a delta rule. The delta rule minimizes the LMS error between the output value and the target value, which is also called the LMS error minimization rule.

We have two kinds of categorical data, $\mathbf{X}_1 = \{\mathbf{x}|\mathbf{x} \in C_1\}$ and $\mathbf{X}_2 = \{\mathbf{x}|\mathbf{x} \in C_2\}$, where $\mathbf{X} = \mathbf{X}_1 \bigcup \mathbf{X}_2$. Our target is to find the optimum weight parameters and threshold values by the delta rule, which is applied for the parameter updates from time t to $t+1$. This update rule is based on the the small changes of the weight parameters $\delta\mathbf{w}_t(\mathbf{x})$:

$$\mathbf{w}_{t+1}(\mathbf{x}) = \mathbf{w}_t(\mathbf{x}) + \delta\mathbf{w}_t(\mathbf{x}). \tag{4.949}$$

Let the LMS error function be $F(\mathbf{w}^{(t)})$. We approximate the changes of the function F from time t to $t+1$:

$$F(\mathbf{w}_{t+1}) = F(\mathbf{w}_t + \Delta\mathbf{w}_t) \sim F(\mathbf{w}_t) + \nabla F(\mathbf{w}_t) \cdot \Delta\mathbf{w}_t. \tag{4.950}$$

If $\nabla F(\mathbf{w}_t) \cdot \Delta\mathbf{w}_t < 0$, then we verify $F(\mathbf{w}_{t+1}) < F(\mathbf{w}_t)$. In Equation (4.949), $\delta\mathbf{w}_t(\mathbf{x})$ is a gradient descent vector that is written as:

$$\delta\mathbf{w}_t(\mathbf{x}) = -\eta\nabla F(\mathbf{x}) \text{ and } \nabla F(\mathbf{w}) = \left(\frac{\partial F(\mathbf{w})}{\partial w_1}, \frac{\partial F(\mathbf{w})}{\partial w_2}, \cdots\right). \tag{4.951}$$

Finally the weight parameters are updated by the following rule:

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t & \text{if } \mathbf{x} \text{ is correctly classified} \\ \mathbf{w}_t - \eta \times \nabla F(\mathbf{w}) & \text{if } \mathbf{x} \in C_1 \text{and } \mathbf{w}_t\mathbf{x} \geq 0 \\ \mathbf{w}_t + \eta \times \nabla F(\mathbf{w}) & \text{if } \mathbf{x} \in C_2 \text{and } \mathbf{w}_t\mathbf{x} < 0 \end{cases}. \tag{4.952}$$

Box 4.21 Steepest descent method

We have a function $F(\mathbf{w})$ with an N-dimensional vector $\mathbf{w}$, in which exists the gradient $\nabla F(\mathbf{w})$. Let us find the best solution with the function F by an iteration procedure starting from an initial location $\mathbf{w}^{(0)}$. The iteration procedure from t to $t+1$ is expressed as:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \times \nabla F(\mathbf{w}^{(t)}), \text{ for } t = 0, 1, 2, \ldots \tag{4.953}$$

where η is a learning rate of the step with the gradient.

Example 4.268

Let us consider a function $F(w_1, w_2) = (w_1 - a_1)^2 + (w_2 - a_2)^2$ with $a_1 = 2, a_2 = 1$. When the initial settings are $w_1^{(0)} = 0$, $w_2^{(0)} = 0$ and $\eta = 0.2$, calculate Equation (4.953).
The gradient of F is given by $\nabla F(w_1, w_2) = (2(w_1 - a_1), 2(w_2 - a_2))$, and the iteration function is calculated by:

$$(w_1^{(t+1)}, w_2^{(t+1)}) \leftarrow (w_1^{(t)}, w_2^{(t)}) - \eta \cdot \nabla F(w_1^{(t)}, w_2^{(t)}) \text{ for } i = 1, 2, 3, \ldots . \tag{4.954}$$

We apply the initial location for this iteration. The result is shown in Table 4.13.

Table 4.13 The step by step caluclation of the gradient method. $\eta = 0.2$.

t	$w_1^{(t)}$	$w_2^{(t)}$	$F(w_1^{(t)}, w_2^{(t)})$	$\nabla F(w_1^{(t)}, w_2^{(t)})$
0	0	0	5	(−2,−4)
1	0.4	0.8	1.8	(−1.2,−2.4)
2	0.64	1.28	0.648	(−0.72,−1.44)
3	0.784	1.568	0.233	(−0.432,−0.864)
4	0.870	1.741	0.084	(−0.259,−0.518)
5	0.922	1.844	0.03	(−0.155,−0.311)
6	0.953	1.906	0.011	(−0.093,−0.187)
7	0.972	1.944	0.004	(−0.056,−0.112)
8	0.983	1.966	0.001	(−0.033,−0.067)
9	0.989	1.979	0.0005	(−0.020,−0.040)
10	0.994	1.989	0.0001	(−0.012,−0.024)
⋮	⋮			
20	1.000	2.000	0.0	(0.0,0.0)

What are the findings from physiology?

Box 4.22 Hubel and Wiesel discoveries

Hubel and Wiesel discovered the existence of neurons that have a function of detecting the contour of objects in the cerebral cortex of cat eyes, and modelled neuron behaviour

(Continued)

Box 4.22 (Continued)

over 1959–1962 [38][39]. In 1965, they clarified hyper complex neurons that have a function of detecting a certain length of lines in vision. This was called a hierarchical model from simple neurons to complex neurons[39]. In 1981, Hubel and Wiesel received a Nobel Prize in Physiology or Medicine for the contribution of their discoveries concerning information processing in the visual system.

When looking at an object, humans can extract some information from it. However, it is very difficult to describe what kind of information is extracted and how it is done. In the fields of physiology, psychology, engineering, etc., with the development of computers and measuring instruments, numerous achievements have been reported since 1950s. Ratriff and Hartline found the lateral inhibitory effect in the eye of limulus [34]. Campbell and Robson revealed the presence of multi-channel hypothesis for visibility of grating [10]. And Marr defined a tri-level hypothesis that has computational, algorithmic and implementational levels in simple modelling of low-level processing [64].

Modelling human information processing covers various physiological findings obtained so far. Of course, the visual model in brain has not been elucidated. We need to construct a model that might simulate visual information processing. In 1980,Fukushima proposed the Cognitron and the Neocognitron for modelling the visual information process. This work has been referred to as a basic framework on multi-layer convolutional neural networks [28].

What are local patterns: primitives and texton

In general, there are no explicit definitions of texture itself in computer vision, but textures and color are two important features to describe a picture. Texture contains a clue of identifying or segmenting patterns in an image. If we need to select and extract global or local features directly that determine the quality of classification, we commonly use features include color, texture, shape and spatial correlations.

The color represents as the features of spatial information of the image. Color features have characteristics that are insensitive to changes in direction and size, and do not capture the local features of objects in an image.

Texture is a ubiquitous visual phenomenon, which is a macroscopic visual scene that appears by primitive patterns repeated in space. As for a visual feature describing the homomorphism in a picture, it embodies the organization of the surface structure of the object that changes periodically.

Shape is one of the discriminative features for describing information on objects to be recognized in the image. Several shape-based recognition methods effectively detect a target, but they all have a common problem, that is, they are easily affected by deformation. When a target is deformed, the detection result will be very unreliable.

The spatial relationship also represents a feature that the target to be identified has in spatial relationship. Spatial information expresses limited information. It is usually impossible to accurately determine an image by virtue of spatial relationships. Therefore, it is necessary to combine other feature information to effectively perform image classification and retrieval.

There are two main types of textures: the first one is regular texture, which is formed by a combination of texture primitives in a regular and orderly manner. It is often called artificial texture, a common weaving object in life, a plate roll, a brick wall all reflect the artificial texture. The second one is quasi-regular texture, which is not formed by regular texture primitives, but is represented by the grey scale of the pixel by presenting some form of distribution. The overall ruled and partially messy nature is also known as natural texture.

Box 4.23 Julesz texton

Julesz revealed the existence of grey level spatial dependency. Different from color features, texture refers to the spatial information of a set of basic elements or primitives (i.e. textons), the fundamental micro-structures in natural images and the atoms of pre-attentive human visual perception [41]. Textures may range from purely stochastic to perfectly regular and everything in between. Texture is an important characteristic of many types of images. Figure 4.218 describes pre-attentively distinguishable texture pairs based either on global statistics or local conspicuous elements, or both: (a) with different first-order statistics and difference in element size; (b) with different second-order statistics and different element orientation;

Figure 4.218 Pre-attentively distinguishable texture pairs by Juletz [41].

What kind of information do we extract as local patterns or features?

When humans look at an object and instantly extract certain information about it, it is presumed what is being estimated or identified. For example, when an apple is placed on a white floor, it is clearly distinguishable by the color, white and red, and then something like an apple is recognised from the red shape.

From the viewpoint of physiology, Mach effect phenomenon or Mach bands can be observed in the visual system [78]. The edge parts in a visual scene are enhanced by suppression of the side band. With differences of intensities, these parts make it easier to extract important components of visual information. This characteristic is considered to reduce the amount of information. An important finding is the existence of an edge extraction mechanism that is likely to have information at the edge, but why this extraction mechanism was formed is an important issue when considering pattern recognition.

To discuss with this problem, let us consider a feature extraction mechanism (model) from the viewpoint of the amount of information. Here, the information is referred to as "event obtained when the prediction is contrary to the expected assumption". At this time, from the previous example, the presence of a red object in a white background is interpreted as red being present in some places, contrary to the assumption of the existence of a place that is all white. Assuming uniformity, when this assumption is broken, we notice that there is information in that part. From this fact, "prediction" would be reasonable to assume uniformity that no change would occur. For an observation area, S, we simply define the amount of information using entropy. In this area S, it is assumed that the quantised function $f(\mathbf{x})$ is represented by B-level features. Let n_α be the number of pixels at feature level α and M be the total number of pixels in region S. The normalized histogram p_α is expressed by $p_\alpha = \frac{n_\alpha}{M}, \alpha = 1, 2, \cdots, B$. It could be regarded as a feature distribution. Here $p_\alpha \geq 0$ and $\sum_{\alpha=1}^{B} p_\alpha = 1$. Using this feature distribution, the average amount of information

is defined by

$$H(p) = \frac{-1}{\log B}\sum_{\alpha=1}^{B} p_\alpha \log p_\alpha, \tag{4.955}$$

where $0 \le H(p) \le 1$.

For example, when $B = 2$, there are two feature levels of -1 and 1. When a pattern f is uniform for all points in S, i.e. $f(\mathbf{x}) = -1$ or 1, the average information amount $H(p)$ comes to 0. On the other hand, in S, when the number of 1s is the same as the number of -1s, that is $p_1 = p_2 = 1/2$, the average information amount reaches to the maximum. Further, it is easy to guess that when $H(p)$ is maximized at the Bth level feature, $p_i = p_j, i \neq j$ holds for all i, j in S.

A visual discrimination model assumes a hierarchical mechanism. As we mentioned in 4.22, hierarchy in a visual system can be seen in the Hubel and Wiesel model in [39]. The bottom layer such as sensory layer deals with all pixels, but the next layer only has small regions with large differential values, and so forth. The physiologists discussed the existence of such a function in the model of the visual system.

If we implement low-level processing of this model, we need to clarify where the information amount is the maximum in the local area S. A black/white character pattern has pixel intensities of 0 (black) and 1 (white). A pattern with intensity 0 forms the shape of a group. The maximum amount of information is located on the boundary in this binary pattern. Edge regions with several orientations (vertical, horizonal, etc.) can be represented in the second layer. Representative patterns are based on the frequency of occurrence of local patterns. We come to the next problem arising as to what kind of local patterns should be selected.

It is required mathematically to introduce an evaluation measure. As a similar concept to distance, a similarity measure is defined by an inner product. This measure is widely used in pattern recognition. If two patterns are similar, the similarity should be $r = 1$, and if they are dissimilar, the similarity is $r = -1$. Edge regions are extracted from a set of images or patterns, and clustering is performed to obtain M representative clusters. If we use principal component analysis (PCA), what kind of primitive patterns appear in the local area.

Box 4.24 What is principal component analysis (PCA)?

We have L images. Let us denote each image vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{iN})^T, i = 1, 2, \cdots, L.(L < N)$. The images are normalised by subtracting the mean image:

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i - \overline{\mathbf{x}} = \mathbf{x}_i - \frac{1}{L}\sum_{i=1}^{L}\mathbf{x}_i.$$

We concatenate these vectors to make a matrix $\tilde{X}$ with size $N \times L$, i.e. $\tilde{X} = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots, \tilde{\mathbf{x}}_L)$. The covariance matrix with size $N \times N$ is calculated by $\Sigma = \tilde{X}\tilde{X}^T$. If we apply the PCA for the covariance matrix, there exist L eigenvectors corresponding to non-zero eigenvalues.

We calculate the eigenvalues and the corresponding eigenvectors using $\Sigma U = \Lambda U$, where $U = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_L)$ and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_L)$ $(\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_L)$.

For training, we project each normalised image $\tilde{\mathbf{x}}_i$ into the eigenspace using $\tilde{\mathbf{y}}_i = U^T\tilde{\mathbf{x}}_i$, $i = 1, 2, \cdots, L$. For testing, we project each normalised test image into the eigen space using

$$\tilde{\mathbf{z}}_j = \mathbf{z}_j - \overline{\mathbf{x}} = \mathbf{z}_j - \frac{1}{L}\sum_{i=1}^{L}\mathbf{x}_i,\ j = 1, 2, \ldots$$

and $\tilde{\mathbf{v}}_j = U^T\tilde{\mathbf{z}}_j, j = 1, 2, \ldots.$

Example 4.269

If you apply an eigenspace projection for local pattern vectors, show what kind of principal vectors appear in principal components. We have three training images and one test image with size 2×2 as follows:

$$\mathbf{x}_1 : \begin{pmatrix} 20 & 15 \\ 20 & 5 \end{pmatrix}, \mathbf{x}_2 : \begin{pmatrix} 30 & 10 \\ 15 & 20 \end{pmatrix}, \mathbf{x}_3 : \begin{pmatrix} 10 & 5 \\ 25 & 10 \end{pmatrix}, \mathbf{z}_1 : \begin{pmatrix} 15 & 15 \\ 15 & 5 \end{pmatrix}.$$

Calculate the eigen images using the PCA.

The mean image of $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ *is* $\overline{\mathbf{x}} = \begin{pmatrix} 20 & 10 \\ 20 & 15 \end{pmatrix}$, *and these images are converted to the vector form:*

$$\mathbf{x}_1 = \begin{pmatrix} 20 \\ 15 \\ 20 \\ 5 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} 30 \\ 10 \\ 15 \\ 30 \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 10 \\ 5 \\ 25 \\ 10 \end{pmatrix}, \mathbf{z}_1 = \begin{pmatrix} 15 \\ 15 \\ 15 \\ 5 \end{pmatrix}, \overline{\mathbf{x}} = \begin{pmatrix} 20 \\ 10 \\ 20 \\ 15 \end{pmatrix}.$$

The normalised vectors and the corresponding matrix form are as follows:

$$\tilde{\mathbf{x}}_1 = \begin{pmatrix} 0 \\ 5 \\ 0 \\ -10 \end{pmatrix}, \tilde{\mathbf{x}}_2 = \begin{pmatrix} 10 \\ 0 \\ -5 \\ 15 \end{pmatrix}, \tilde{\mathbf{x}}_3 = \begin{pmatrix} -10 \\ -5 \\ 5 \\ -5 \end{pmatrix}, \tilde{X} = \begin{pmatrix} 0 & 10 & -10 \\ 5 & 0 & -5 \\ 0 & -5 & 5 \\ -10 & 15 & -5 \end{pmatrix}.$$

The covariance matrix with size 4×4 *is calculated by*

$$\Sigma = \tilde{X}\tilde{X}^T = \begin{pmatrix} 200 & 50 & -100 & 200 \\ 50 & 50 & -25 & -25 \\ -100 & -25 & 50 & -100 \\ 200 & -25 & -100 & 350 \end{pmatrix}.$$

From the eigenvalue problem, $\Sigma U = \Lambda U$, *we obtain:*

$$U = \begin{pmatrix} 0.562 & 0.522 \\ 0.033 & 0.641 \\ -0.281 & -0.261 \\ 0.777 & -0.499 \end{pmatrix}, \Lambda = \begin{pmatrix} 529.6 & 0 & 0 & 0 \\ 0 & 120.4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Mapping these vectors into the eigenspace, we get:

$$U^T\tilde{X} = \begin{pmatrix} 0.562 & 0.522 \\ 0.033 & 0.641 \\ -0.281 & -0.261 \\ 0.777 & -0.499 \end{pmatrix} \times \begin{pmatrix} 0 & 10 & -10 \\ 5 & 0 & -5 \\ 0 & -5 & 5 \\ -10 & 15 & -5 \end{pmatrix} = \begin{pmatrix} -7.605 & 8.195 \\ 18.68 & -0.96 \\ -11.075 & -7.235 \end{pmatrix}^T.$$

The test image $\mathbf{z}_1$ *is normalised as follows:*

$$\tilde{\mathbf{z}}_1 = \mathbf{z}_1 - \overline{\mathbf{x}} = \begin{pmatrix} 15 \\ 15 \\ 15 \\ 5 \end{pmatrix} - \begin{pmatrix} 20 \\ 10 \\ 20 \\ 15 \end{pmatrix} = \begin{pmatrix} -5 \\ 5 \\ -5 \\ -10 \end{pmatrix}.$$

(Continued)

Example 4.269 (Continued)

And it is mapped into the eigenspace:

$$\tilde{\mathbf{v}}_1 = U^T\tilde{\mathbf{z}}_1 = \begin{pmatrix} 0.562 & 0.522 \\ 0.033 & 0.641 \\ -0.281 & -0.261 \\ 0.777 & -0.499 \end{pmatrix} \times \begin{pmatrix} -5 \\ 5 \\ -5 \\ -10 \end{pmatrix} = \begin{pmatrix} -9.010 \\ 6.895 \end{pmatrix}.$$

For each class, we compute a representative vector $\tilde{\mathbf{w}}_j$ in eigenspace for classification:

$$U^T\tilde{X} = (\tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2, \tilde{\mathbf{w}}_3) = \begin{pmatrix} -7.605 & 18.68 & -11.075 \\ 8.195 & -0.96 & -7.235 \end{pmatrix}.$$

We recognise which class the test pattern $\mathbf{z}_1$ belongs to by calculating the similarity measure as follows:

Class 1: *The similarity measure is* $\frac{\tilde{\mathbf{v}}_1 \cdot \tilde{\mathbf{w}}_1}{\|\tilde{\mathbf{v}}_1\| \times \|\tilde{\mathbf{w}}_1\|} = 0.98$.

Class 2: *The similarity measure is* $\frac{\tilde{\mathbf{v}}_1 \cdot \tilde{\mathbf{w}}_2}{\|\tilde{\mathbf{v}}_1\| \times \|\tilde{\mathbf{w}}_2\|} = -0.82$.

Class 3: *The similarity measure is* $\frac{\tilde{\mathbf{v}}_1 \cdot \tilde{\mathbf{w}}_3}{\|\tilde{\mathbf{v}}_1\| \times \|\tilde{\mathbf{w}}_3\|} = 0.33$. *Therefore as $\mathbf{z}_1$ is similar to $\mathbf{x}_1$, the test image belongs to class 1.*

Next we calculate the covariance matrix with size 3×3:

$$\Sigma' = \tilde{X}^T\tilde{X} = \begin{pmatrix} 125 & -150 & 25 \\ -150 & 350 & -200 \\ 25 & -200 & 175 \end{pmatrix}.$$

From $\Sigma' U' = \Lambda' U'$, we obtain

$$U' = \begin{pmatrix} 0.331 & 0.747 \\ -0.812 & -0.087 \\ 0.481 & -0.660 \end{pmatrix}, \Lambda' = \begin{pmatrix} 529.6 & 0 & 0 \\ 0 & 120.4 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\tilde{X}U' = \begin{pmatrix} 0 & 10 & -10 \\ 5 & 0 & -5 \\ 0 & -5 & 5 \\ -10 & 15 & -5 \end{pmatrix} \times \begin{pmatrix} 0.331 & 0.747 \\ -0.812 & -0.087 \\ 0.481 & -0.660 \end{pmatrix} = \begin{pmatrix} -12.93 & 5.73 \\ -0.75 & 7.035 \\ 6.465 & -2.865 \\ -17.895 & -5.475 \end{pmatrix}.$$

If we normalise $\begin{pmatrix} -12.93 & 5.73 \\ -0.75 & 7.035 \\ 6.465 & -2.865 \\ -17.895 & -5.475 \end{pmatrix}$, *then we obtain* $\begin{pmatrix} -0.562 & 0.507 \\ -0.033 & 0.640 \\ 0.281 & -0.260 \\ -0.778 & -0.498 \end{pmatrix} = U$. *Note that the first column vector is the inverse vector (each sign of the element is changed).*

Example 4.270

Apply the PCA for local pattern vectors. What kind of principal component vectors can we obtain as primitive patterns?

If you collect several texture images and extracted the local patterns, we use the PCA to generate the principal component vectors. The results are shown in Figure 4.219.

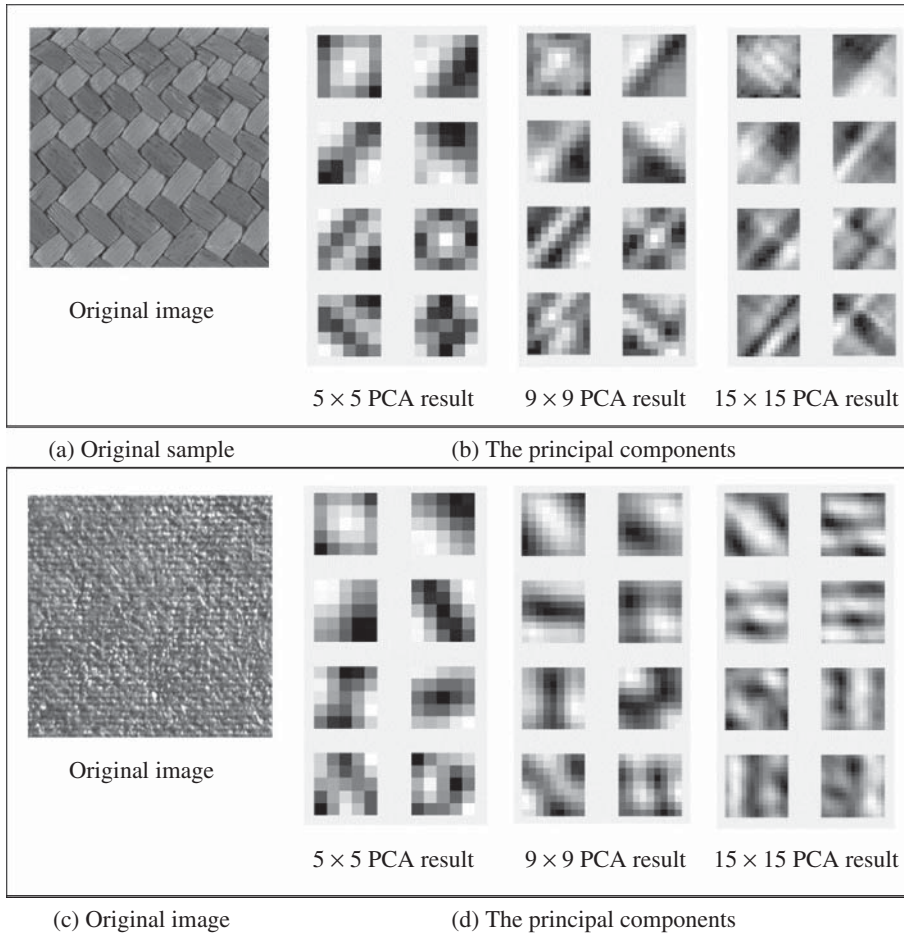


Figure 4.219 The principal component vectors. For each PCA result with size 5×5, 9×9 and 15×15, the upper-left one corresponds to the primal component, the upper-right one to the second component, and lower-right one to the eighth component, etc. Source: Maria Petrou.

How do we find primitive patterns?

Selection of primitive patterns depends on a property of input patterns. It is obvious that patterns you always see are selected as primitive patterns, while patterns that rarely appear are not. This dependency is based on the frequency of appearance of patterns. For example, a new-born cat wears glasses and draws vertical bars on the glasses [39]. In the experiments, this cat is put in a white room. When the cat only see the vertical bars, it has been discovered that the "vertical bar

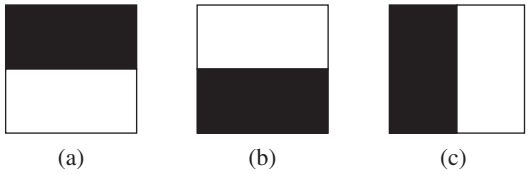


Figure 4.220 Similarity definition of edges.

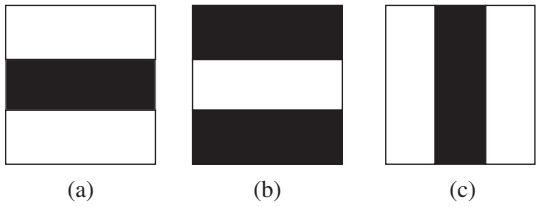


Figure 4.221 Similarity definition of lines.

detection cells" found are many, but "horizontal bar detection cells" seldom appear. Also, if a character set is limited to Chinese characters, mainly vertical and horizontal edge patterns appeared, while in the case of alphabets and numbers, many oblique edge patterns appear. Therefore the problem is that we should clarify conceptually similar or dissimilar patterns as primitive patterns corresponding to edges or lines.

For example, the pattern similarity to be considered is shown in Figure 4.220. Figures 4.220(a) and (b) correspond to the relationship between -1 and 1 in terms of similarity, which is called a dual pattern. Here, the pattern as in (c) is orthogonal to the one in (a). That is, the similarity r is close to $r = 0$.

When the local area S described above is further enlarged (this is referred to as a larger area S'), $S \subset S'$, where is the area S' that mamimises the amount of information in the edge layer? The locations of this kind are like straight line segments as shown in Figure 4.221. It is considered to be equivalent to the locations where some changes occurred in the edge layer. From the observation of whether or not they are the same, we would find the dual pattern as shown in figures 4.221(a) and (b). And the pattern as in (c) is an orthogonal pattern ($r = 0$).

When you look at the results of the PCA in Figure 4.219, you can find a gaussian like pattern in the primal component. And then edge patterns are appeared in the second and third components.

What is the back-propagation learning method?

Although it is quite difficult to define primitive patterns, an innovation that is called the back-propagation (BP) learning method was advanced in 1986 [84]. At that time, many researchers immediately implemented this BP method for several recognition applications. Among them, Lecun proposed a character recognition method using the BP[52].

Box 4.25 Sigmoid function

Sig(x) is a sigmoid function as shown in Figure 4.222 expressed by

$$\mathrm{Sig}(x) = \frac{1}{1 + \mathrm{e}^{-x}}. \tag{4.956}$$

And the differential of this function is calculated by

$$\frac{\mathrm{dSig}(x)}{\mathrm{d}x} = \frac{\mathrm{e}^{-x}}{(1+\mathrm{e}^{-x})^2} = \mathrm{Sig}(x)(1-\mathrm{Sig}(x)). \tag{4.957}$$

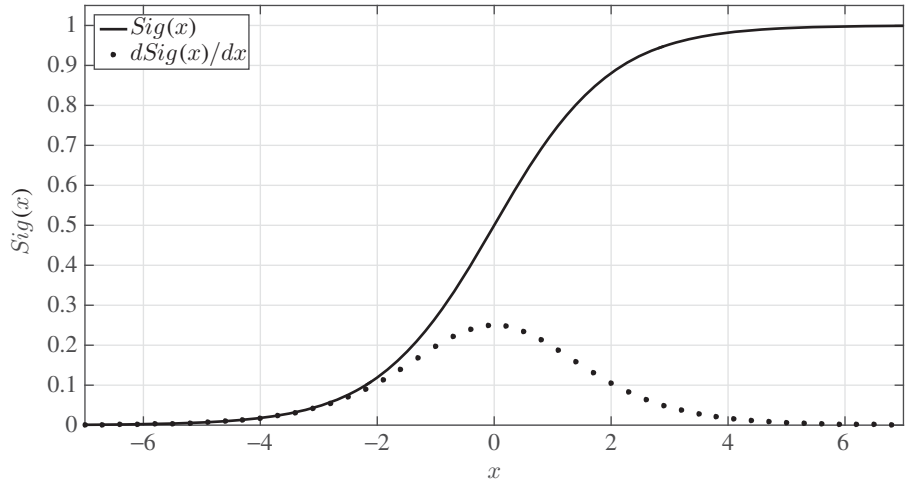


Figure 4.222 A sigmoid function and the first derivative.

Box 4.26 Three-layer neural networks

Runmelhart et al. proposed an extension of the delta rule called a generalized delta rule[84]. Let us explain the three-layer neural networks (NNs) as shown in Figure 4.223. For training, we prepare a set of training pattern vectors $\mathbf{X} = \{(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L)\}$, and $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{in}, \cdots, x_{iN})$, $i = 1, 2, \cdots, L$.

A pattern vector $\mathbf{x}_i$ is mapped to hidden nodes with $h(\mathbf{x}_i)$ and then to output nodes with $\mathbf{c}(\mathbf{x}_i)$. The target (teaching) vector is $\tau(\mathbf{x}_i)$, which corresponds to $\mathbf{c}(\mathbf{x}_i)$. These parameters are summarized as follows,

$$\mathbf{c}(\mathbf{x}) = (c_1(\mathbf{x}), c_2(\mathbf{x}), \cdots, c_k(\mathbf{x}), \cdots, c_K(\mathbf{x})) \tag{4.958}$$

$$\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), h_2(\mathbf{x}), \cdots, h_m(\mathbf{x}), \cdots, h_M(\mathbf{x})) \tag{4.959}$$

$$\boldsymbol{\tau}(\mathbf{x}) = (\tau_1(\mathbf{x}), \tau_2(\mathbf{x}), \cdots, \tau_K(\mathbf{x})) \tag{4.960}$$

where

$$c_k(\mathbf{x}) = \mathrm{Sig}\left(\sum_{m=1}^{M} v_{mk} h_m(\mathbf{x}) + \theta_k^2\right) \tag{4.961}$$

$$h_m(\mathbf{x}) = \mathrm{Sig}\left(\sum_{n=1}^{N} w_{nm} x_{in} + \theta_m^1\right). \tag{4.962}$$

(Continued)

Box 4.26 (Continued)

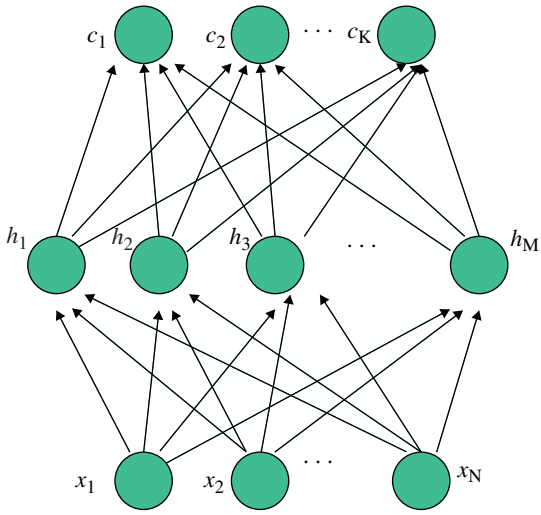


Figure 4.223 Three-layer neural networks.

The LMS error between the output $\mathbf{c}(\mathbf{x}_i)$ and the target $\boldsymbol{\tau}(\mathbf{x}_i)$ is defined by

$$S(\mathbf{X}) = \frac{1}{2L}\sum_{i=1}^{L}\sum_{k=1}^{K}(c_k(\mathbf{x}_i) - \tau_k(\mathbf{x}_i))^2. \tag{4.963}$$

Let us derive partial derivatives of S with respect to v_{mk} and w_{nm},

$$\frac{\partial S(\mathbf{X})}{\partial v_{mk}} = 0, \tag{4.964}$$

$$\frac{\partial S(\mathbf{X})}{\partial w_{nm}} = 0. \tag{4.965}$$

$$\begin{aligned}\frac{\partial S(\mathbf{X})}{\partial v_{mk}} &= \frac{1}{2L}\frac{\partial}{\partial v_{mk}}\sum_{i=1}^{L}\sum_{k=1}^{K}(c_k(\mathbf{x}_i) - \tau_k(\mathbf{x}_i))^2 \\ &= \frac{1}{L}\sum_{i=1}^{L}\sum_{k=1}^{K}(c_k(\mathbf{x}_i) - \tau_k(\mathbf{x}_i))\frac{\partial c_k(\mathbf{x}_i)}{\partial v_{mk}} \\ &= \frac{1}{L}\sum_{i=1}^{L}\left\{(\sum_{k=1}^{K}(c_k(\mathbf{x}_i) - \tau_k(\mathbf{x}_i))c_k(\mathbf{x}_i)(1 - c_k(\mathbf{x}_i))\right\}h_m(\mathbf{x}_i)\end{aligned} \tag{4.966}$$

$$\begin{aligned}\frac{\partial S(\mathbf{X})}{\partial w_{nm}} &= \frac{1}{L}\sum_{i=1}^{L}\sum_{k=1}^{K}(c_k(\mathbf{x}_i) - \tau_k(\mathbf{x}_i))\frac{\partial c_k(\mathbf{x}_i)}{\partial w_{nm}} \\ &= \frac{1}{L}\sum_{i=1}^{L}\sum_{k=1}^{K}(c_k(\mathbf{x}_i) - \tau_k(\mathbf{x}_i))c_k(\mathbf{x}_i)(1 - c_k(\mathbf{x}_i))v_{mk}\frac{\partial h_m(\mathbf{x}_i))}{\partial w_{nm}} \\ &= \frac{1}{L}\sum_{i=1}^{L}\left\{\sum_{k=1}^{K}(c_k(\mathbf{x}_i) - \tau_k(\mathbf{x}_i))c_k(\mathbf{x}_i)(1 - c_k(\mathbf{x}_i))v_{mk}\right\} \\ &\qquad \times h_m(\mathbf{x}_i)(1 - h_m(\mathbf{x}_i))x_{in}.\end{aligned} \tag{4.967}$$

We assume that the changes of these parameters in the time domain depend on the spatial changes in the function S. Using the steepest descent method, we derive the generalized delta rules as follows,

$$v_{mk} \leftarrow v_{mk} - \eta \frac{\partial S(\mathbf{X})}{\partial v_{mk}} \tag{4.968}$$

$$w_{nm} \leftarrow w_{nm} - \eta \frac{\partial S(\mathbf{X})}{\partial w_{nm}}. \tag{4.969}$$

The LMS error S can be decomposed into the error of each data, S_i, in the following.

$$\frac{\partial S(\mathbf{X})}{\partial v_{mk}} = \sum_{i=1}^{L} \frac{\partial S_i(\mathbf{x}_i)}{\partial v_{mk}} \tag{4.970}$$

$$\frac{\partial S(\mathbf{X})}{\partial w_{nm}} = \sum_{i=1}^{L} \frac{\partial S_i(\mathbf{x}_i)}{\partial w_{nm}} \tag{4.971}$$

where S_i is denoted by

$$S_i(\mathbf{x}_i) = \frac{1}{2} \sum_{k=1}^{K} (c_k(\mathbf{x}_i) - \tau_k(\mathbf{x}_i))^2. \tag{4.972}$$

The generalised delta rules for each data are derived in the following,

$$v_{mk}^{(t+1)} \leftarrow v_{mk}^{(t)} - \eta \frac{\partial S_{t \bmod L}(\mathbf{x}^{(t)})}{\partial v_{mk}} = v_{mk}^{(t)} - \eta (c_k(\mathbf{x}_i^{(t)}) - \tau_k(\mathbf{x}_i^{(t)})) h_m(\mathbf{x}_i^{(t)}) \tag{4.973}$$

$$\begin{aligned} w_{nm}^{(t+1)} &\leftarrow w_{nm}^{(t)} - \eta \frac{\partial S_{t \bmod L}(\mathbf{x}^{(t)})}{\partial w_{nm}} \\ &= w_{nm}^{(t)} - \eta (c_k(\mathbf{x}_i^{(t)}) - \tau_k(\mathbf{x}_i^{(t)})) v_{mk}^{(t)} \times h_m(\mathbf{x}_i^{(t)})(1 - h_m(\mathbf{x}_i^{(t)})) x_{in}^{(t)} \end{aligned} \tag{4.974}$$

$$\theta_k^{2,(t+1)} \leftarrow \theta_k^{2,(t)} - \eta \frac{\partial S_{t \bmod L}(\mathbf{x}^{(t)})}{\partial \theta_k^2} = \theta_k^{2,(t)} - \eta (c_k(\mathbf{x}_i^{(t)}) - \tau_k(\mathbf{x}_i^{(t)})), \tag{4.975}$$

$$\begin{aligned} \theta_m^{1,(t+1)} &\leftarrow \theta_m^{1,(t)} - \eta \frac{\partial S_{t \bmod L}(\mathbf{x}^{(t)})}{\partial \theta_m^1} \\ &= \theta_m^{1,(t)} - \eta (c_k(\mathbf{x}_i^{(t)}) - \tau_k(\mathbf{x}_i^{(t)})) v_{mk}^{(t)} \times h_m(\mathbf{x}_i^{(t)})(1 - h_m(\mathbf{x}_i^{(t)})). \end{aligned} \tag{4.976}$$

Example 4.271

Let us give as an example of three-layer NN as shown in Figure 4.224 This three-layer NN can solve an XOR problem as shown in Table 4.14. As Perceptron and ADALINE are linear classifierd, it is difficult to solve it.

The three-layer NN converges on a solution, as shown in Figure 4.225. In this figure, the vertical axis is a learning error between the training signal and the target signal. The step by step calculation of the three-layer NN is shown in Tables 4.15 and 4.16. Here the number of total epochs is up to 1000, and the learning rate is $\eta = 0.2$

(Continued)

Example 4.271 (Continued)

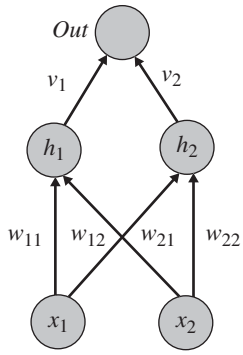


Figure 4.224 The three-layer NN.

Table 4.14 XOR problem.

Number	x_1	x_2	τ
1	−1	−1	−1
2	−1	1	1
3	1	−1	1
4	1	1	−1

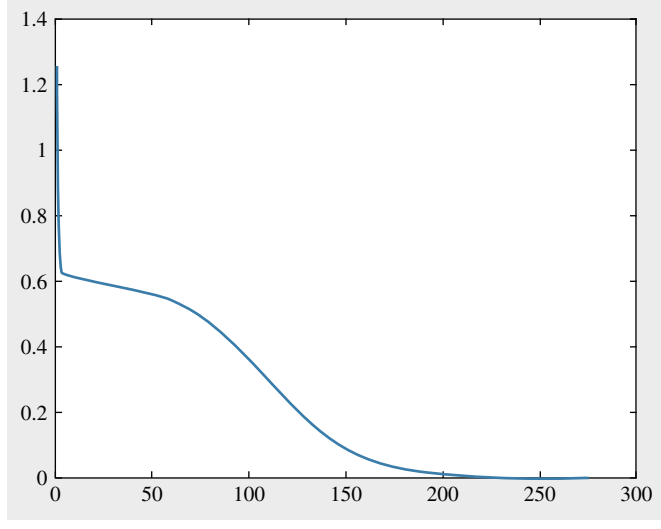


Figure 4.225 Error in the three-layer NN.

Table 4.15 The step by step calculation of three-layer NN(1).

t	w_{11}	w_{12}	w_{21}	w_{22}	v_1	v_2	θ_1	θ_2	θ_3	err	*out*
0	0.200	0.800	−0.700	0.200	0.100	−0.100	1.000	0.800	−1.500		
1	0.199	0.801	−0.701	0.201	0.138	−0.079	1.001	0.799	−1.454	0.107	−1.463
2	0.191	0.806	−0.692	0.196	0.265	0.054	1.009	0.794	−1.211	2.940	−1.425
3	0.197	0.808	−0.698	0.195	0.434	0.210	1.015	0.796	−1.017	1.877	−0.937
⋮				⋮							
100	−0.322	0.754	−0.765	0.330	0.624	0.323	0.968	0.826	−0.670	0.653	0.143
200	−0.814	0.787	−0.954	0.499	0.817	0.384	1.015	0.844	−0.789	0.521	0.021
300	−1.266	0.923	−1.231	0.777	1.094	0.610	1.253	0.907	−1.102	0.399	−0.106
400	−1.663	1.195	−1.572	1.185	1.431	1.020	1.588	1.089	−1.612	0.295	−0.232
500	−1.983	1.538	−1.897	1.592	1.784	1.511	1.888	1.382	−2.231	0.183	−0.395
600	−2.209	1.829	−2.139	1.888	2.085	1.932	2.102	1.653	−2.788	0.090	−0.576
700	−2.353	2.026	−2.298	2.075	2.304	2.228	2.239	1.841	−3.196	0.037	−0.727
800	−2.439	2.149	−2.396	2.184	2.450	2.416	2.323	1.958	−3.463	0.014	−0.832
900	−2.488	2.224	−2.456	2.247	2.540	2.531	2.373	2.027	−3.628	0.005	−0.899
1000	−2.516	2.269	−2.493	2.283	2.595	2.600	2.403	2.068	−3.727	0.002	−0.940

Table 4.16 The step by step calculation of three-layer NN(2).

t	h_1	h_2	Δv_1	Δv_2	$\Delta\theta_3$	Δw_{11}	Δw_{12}	Δw_{21}	Δw_{22}	$\Delta\theta_1$	$\Delta\theta_2$
0											
1	0.818	0.450	0.038	0.021	0.046	−0.001	0.001	−0.001	0.001	0.001	−0.001
2	0.525	0.550	0.127	0.133	0.242	−0.008	0.005	0.008	−0.005	0.008	−0.005
3	0.869	0.803	0.168	0.156	0.194	0.006	0.002	−0.006	−0.002	0.006	0.002
⋮				⋮							
100	0.485	0.873	−0.055	−0.100	−0.114	−0.019	−0.005	−0.019	−0.005	−0.019	−0.005
200	0.333	0.895	−0.034	−0.091	−0.102	−0.019	−0.005	−0.019	−0.005	−0.019	−0.005
300	0.233	0.932	−0.021	−0.083	−0.089	−0.018	−0.004	−0.018	−0.004	−0.018	−0.004
400	0.168	0.970	−0.013	−0.075	−0.077	−0.016	−0.002	−0.016	−0.002	−0.016	−0.002
500	0.124	0.989	−0.007	−0.060	−0.061	−0.012	−0.001	−0.012	−0.001	−0.012	−0.001
600	0.098	0.995	−0.004	−0.042	−0.042	−0.008	0.000	−0.008	0.000	−0.008	0.000
700	0.083	0.997	−0.002	−0.027	−0.027	−0.005	0.000	−0.005	0.000	−0.005	0.000
800	0.076	0.998	−0.001	−0.017	−0.017	−0.003	0.000	−0.003	0.000	−0.003	0.000
900	0.071	0.998	−0.001	−0.010	−0.010	−0.002	0.000	−0.002	0.000	−0.002	0.000
1000	0.069	0.999	0.000	−0.006	−0.006	−0.001	0.000	−0.001	0.000	−0.001	0.000

(Continued)

Example 4.271 (Continued)

For initial setting, input data is $x_1 = -1, x_2 = -1$ *and the target is* $y = -1$. *The network output is out. Four weights and two thresholds between the first layer and the hidden layer are denoted as* $w_{11} = 0.2, w_{12} = 0.8, w_{21} = -0.7, w_{22} = 0.2, \theta_1 = 1.0, \theta_2 = 0.8$, *and the hidden layer has two nodes* h_1, h_2. *Two weights and one threshold between the hidden layer and the output layer are denoted as* $v_1 = 0.1, v_2 = -0.1, \theta_3 = -1.5$, *and the learning rate is* $\eta = 0.1$.

For forward propagation, we calculate node values in the hidden layer using the Sigmoid function:

$$\begin{aligned} net_{h1} &= x_1 \times w_{11} + x_2 \times w_{21} + \theta_1 \\ &= (-1) \times 0.2 + (-1) \times (-0.7) + 1.0 = 1.5, \\ net_{h2} &= x_1 \times w_{12} + x_2 \times w_{22} + \theta_2 \\ &= (-1) \times 0.8 + (-1) \times 0.2 + 0.8 = -0.2, \\ h_1 &= \frac{1}{(1 + \exp(-net_{h1})} = \frac{1}{(1 + \exp(-1.5)} = 0.817, \\ h_2 &= \frac{1}{(1 + \exp(-net_{h2})} = \frac{1}{(1 + \exp(-(-0.2))} = 0.450. \end{aligned}$$

The output value is

$$out = h_1 \times v_1 + h_2 \times v_2 + \theta_3 = 0.817 \times 0.1 + 0.450 \times (-0.1) + (-1.5) = -1.463.$$

For backward propagation, the square error is

$$E_{\text{total}} = \frac{1}{2}(y - \text{out})^2 = 0.5 \times (-1 - (-1.463))^2 = 0.215.$$

For the parameter v_1, *we calculate the partial derivative of* E_{total} *with respect to* v_1.

$$\begin{aligned} \frac{\partial E_{\text{total}}}{\partial v_1} &= \frac{\partial E_{\text{total}}}{\partial \text{out}} \times \frac{\partial \text{out}}{\partial v_1}, \\ \frac{\partial E_{\text{total}}}{\partial \text{out}} &= 2 \times 1/2(y - \text{out}) = y - \text{out} = 0.463 \\ \frac{\partial \text{out}}{\partial v_1} &= h_1 = 0.817. \end{aligned}$$

Update the parameter value on v_1 *by*

$$v_1 \leftarrow v_1 + \eta \times (y - \text{out}) \times h_1 = v_1 + 0.038 = -0.138.$$

In the same way, the parameter value on v_2 *by*

$$v_2 \leftarrow v_2 + \eta \times (y - \text{out}) \times h_2 = v_2 + 0.021 = -0.079.$$

For θ_3 *by*

$$\theta_3 \leftarrow \theta_3 + \eta \times (y - \text{out}) = \theta_3 + 0.046 = -1.454.$$

For w_{11},

$$\begin{aligned} \frac{\partial E_{\text{total}}}{\partial w_{11}} &= \frac{\partial E_{\text{total}}}{\partial \text{out}} \times \frac{\partial \text{out}}{\partial h_1} \times \frac{\partial h_1}{\partial net_1} \times \frac{\partial net_1}{\partial w_{11}} \\ \frac{\partial E_{\text{total}}}{\partial \text{out}} &= 2 \times \frac{1}{2}(y - \text{out}) = y - \text{out} = 0.463. \end{aligned}$$

As $\text{out} = h_1 \times v_1 + h_2 \times v_2$, *we obtain*

$$\frac{\partial net_1}{\partial h_1} = v_1 = 0.1$$

As $h_1 = \frac{1}{1+\exp(-net_{h1})}$, *we obtain*

$$\frac{\partial h_1}{\partial net_1} = h_1(1 - h_1) = 0.150.$$

As $net_{h1} = x_1 \times w_{11} + x_2 \times w_{21}$, *we obtain*

$$\frac{\partial net_1}{\partial w_{11}} = x_1 = -1.$$

Therefore

$$\frac{\partial E_{\text{total}}}{\partial w_{11}} = (y - \text{out}) \times v_1 \times h_1(1 - h_1) \times x_1 = -0.001.$$

Renew the value in w_{11},

$$w_{11} \leftarrow w_{11} + \eta \times \frac{\partial E_{\text{total}}}{\partial w_{11}}.$$

Update the parameters w_{11}, w_{12}, w_{21} *and* w_{22} *as follows:*

$$w_{11} \leftarrow w_{11} + \eta \times (y - \text{out}) \times v_1 \times h_1(1 - h_1) \times x_1 = w_{11} - 0.007 = 0.199,$$
$$w_{12} \leftarrow w_{12} + \eta \times (y - \text{out}) \times v_2 \times h_2(1 - h_2) \times x_1 = w_{12} + 0.001 = 0.801,$$
$$w_{21} \leftarrow w_{21} + \eta \times (y - \text{out}) \times v_1 \times h_1(1 - h_1) \times x_2 = w_{21} - 0.001 = -0.701,$$
$$w_{22} \leftarrow w_{22} + \eta \times (y - \text{out}) \times v_2 \times h_2(1 - h_2) \times x_2 = w_{22} + 0.001 = 0.201.$$

We also update θ_1 *and* θ_2 *as follows:*

$$\theta_1 \leftarrow \theta_1 + \eta \times (y - \text{out}) \times v_1 \times h_1(1 - h_1) = \theta_1 + 0.001 = 1.001.$$
$$\theta_2 \leftarrow \theta_2 + \eta \times (y - \text{out}) \times v_2 \times h_2(1 - h_2) = \theta_2 - 0.001 = 0.799.$$

How do we solve the vanishing gradient problem?

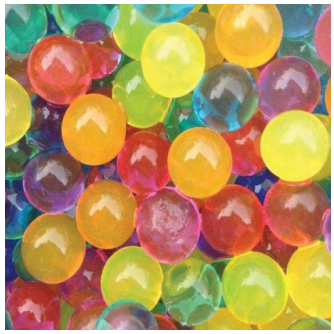
If we train a deep neural network, we suffer from a vanishing gradient problem, which is an unstable behavior. In a deep neural network it is difficult to propagate gradient information backward from the output nodes to the input nodes. Some specific activation functions such as the sigmoid function make the input information in the small input space of [0,1] vanish. Therefore, the big change in the input of the sigmoid function causes a small change in the output. And then the derived function becomes small. If we check the first derivative of the sigmoid function as shown in Figure 4.222, it is noticed that the output range of the first derivative turns into [0,0.3).

There are several ways to fix this problem. Several researchers have proposed solving it by finding effective methods, such as the alternative activation function, pre-training, and gradient descent approaches, etc. The most common change is the use of the rectified linear activation function $\text{Rect}(x) = \max(0, x)$, instead of the sigmoid activation function. When a layer using a specific activated function is added to a neural network, the gradients of the loss function become closer to zero, and it turns out that the training of the network converges to the optimum.

Difficulties in texture classification: intra-class variations and inter-class similarities

The difficulties in texture classification mainly involve large intra-class variations and inter-class similarities. Since the size and color are not stable even in one structure, it is hard to define a texton as mentioned in Box 4.23. The large intra-class appearance variations are caused by changes in illumination, rotation, scale, blur, noise, occlusion, etc. For example, as shown in Figure 4.226, these intra-class images have noticeably distinctive elements of "bubble" as: (a) toy balloon-like bubbles with almost same size and rich colors, and (b) beer foam-like bubbles with huge diversity in size and shape but all in yellow with a white foam background.

Inter-class images may also appear to have similar patterns due to the fuzzy definition of texture species, thus making the cluster task difficult due to unclear boundaries between classes that result in limited accuracy. Therefore, when dealing with the texture classification tasks, it is also required to reduce the features enhancing inter-class similarities. For example, as shown in Figure 4.227, (a) is woven, but the image has a similar pattern, (b) is from the braided class, and this kind of similar sample may even be difficult for a human to discriminate, thus confusing or upsetting the texture classification task.



(a) One example of bubbly texture

(b) Another example of a bubbly class

Figure 4.226 Very large intra-class variations caused by changes in illumination, rotation, scale, blur, noise, occlusion, etc. Source: Flickr.

(a) Woven texture example

(b) Cross-hatched texture example

Figure 4.227 Inter-class images may also appear to have a similar pattern due to the fuzzy definition of each texture species.

What is Fisher's discriminating feature space?

Texture features require a characteristic of discriminating classes. The problem is how to construct the boundaries between classes. Let us introduce the famous Fisher's amount of information, in terms of Fisher mapping, which requires that the variance within classes should be reduced, while the variance between classes should be increased.

Let $\mathbf{X}$ be an entire set of patterns, and $\mathbf{X}_i \subset \mathbf{X}$ be the subset in the ith class. If we have K classes, $\mathbf{X} = \cup_{i=1}^{K} \mathbf{X}_i$. For the ith class, the mean vector, $\mathbf{u}_i$, and the covariance matrix, Σ_i, in R^N are denoted as:

$$\mathbf{u}_i = E_{\mathbf{x}\in\mathbf{X}_i}[\mathbf{x}] \tag{4.977}$$

$$\Sigma_i = E_{\mathbf{x}\in\mathbf{X}_i}[(\mathbf{x}-\mathbf{u}_i)(\mathbf{x}-\mathbf{u}_i)^T]. \tag{4.978}$$

Let us assume a linear mapping from an N-dimensional vector, $\mathbf{x} \in \mathbf{X}$, to the M-dimensional vector, $\mathbf{x}$, $(N > M)$:

$$\mathbf{x} = \Phi \cdot \mathbf{x}. \tag{4.979}$$

Thus, the degenerated M-dimensional space is the discriminative feature space, which separates K classes of patterns.

Let construct the discriminant feature space, where matching is performed using similarity. In order to separate K classes, The Fisher mapping considers an intra-class covariance matrix and an inter-class covariance matrix in distribution of class patterns. First, let the intra-class covariance matrix be Σ_W and the inter-class covariance matrix be Σ_B in the following:

$$\Sigma_W = \sum_{i=1}^{K} w_i \Sigma_i \tag{4.980}$$

$$\Sigma_B = \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1}^{K} w_i w_j ((\mathbf{u}_i - \mathbf{u}_j)^T \Sigma_w^{-1} (\mathbf{u}_i - \mathbf{u}_j)). \tag{4.981}$$

After the linear mapping in Equation (4.979), the mean vector and the covariance matrix in $\hat{\mathbf{R}}^M$ are denoted as:

$$\hat{\mathbf{u}}_i = \Phi^T \mathbf{u}_i \;\; \hat{\Sigma}_i = \Phi^T \Sigma_i \Phi.$$

We obtain the mapped intra- and inter-classes covariance matrices:

$$\hat{\Sigma}_W = \Phi^T \Sigma_W \Phi, \;\; \hat{\Sigma}_B = \Phi^T \Sigma_B \Phi.$$

As an evaluation criterion, we utilise the average Maharanobis distance MD[Φ] of K classes as:

$$\mathrm{MD}[\Phi] = \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1}^{K} w_i w_j ((\hat{\mathbf{u}}_i - \hat{\mathbf{u}}_j)^T \hat{\Sigma}_w^{-1} (\hat{\mathbf{u}}_i - \hat{\mathbf{u}}_j)) \tag{4.982}$$

where the probability of ith class pattern occurrence is w_i, Let us find the mapping matrix Φ that maximizes MD[Φ]. Since this is the maximum optimisation problem for Φ, we solve the following generalised eigenvalue problem given by:

$$\Sigma_B \Phi = \Sigma_W \Phi \Lambda_M \tag{4.983}$$

where λ_i is the ith eigenvalue, and the eigen matrix is $\Lambda_M = \mathrm{diag}[\lambda_1, \lambda_2, \cdots, \lambda_M]$.

In order to investigate the meaning of this transformation, the normalized space from R^N is $\hat{R}^M$. $\boldsymbol{\xi}_i$ is an eigenvector for the ith eigenvalue λ_i. Therefore, the mapping from $\mathbf{x}$ to $\mathbf{x}$ is

$$P_M = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \cdots, \boldsymbol{\xi}_M), \tag{4.984}$$

where P_M is an $N \times M$ matrix. From the equation (4.984), the following equation holds

$$\Lambda_M = \Sigma_W^{-1/2} P_M. \tag{4.985}$$

It can be considered in two stages. Here, we map the pattern space into $\check{R}^M$ space. Then we apply the intra-class covariance matrix in $\tilde{R}^M$ for this space:

$$\check{\mathbf{x}} = \Sigma_W^{-1/2}\mathbf{x}, \ \mathbf{x} = P_M^T \mathbf{x}. \tag{4.986}$$

This means that the variance in each class should be constant. Therefore, $\check{R}^N$ is called a normalized space. In this space, the scattering within each class is given by:

$$\check{\Sigma}_W = \Sigma_W^{-1/2} \Sigma_W \Sigma_W^{-1/2} = I_M. \tag{4.987}$$

Since each class variance is constant, the separation between classes is strengthened while keeping this property. Specifically, we consider the discrimination between two classes based on the Mahalanobis distance.

Example 4.272

At first, we have two classes that have 2D data:

$$C_1 = \{(3.5, 4.6), (4.2, 3.8), (3.2, 3.6), (3.1, 3.4), (4.5, 2.6)\}$$
$$C_2 = \{(0.9, 1.5), (1.7, 3.0), (1.2, 1.9), (2.2, 2.9)\}.$$

We illustrate the data as shown in Figure 4.228. Here "·" indicates class 1 and "★" indicates class 2.

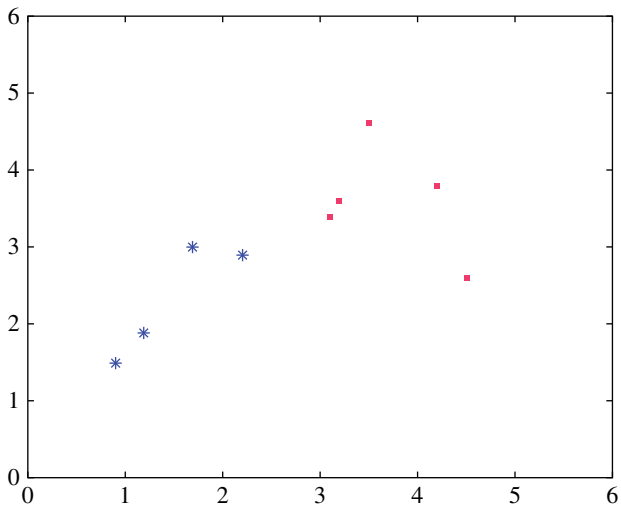


Figure 4.228 The data for Fisher discriminant analysis.

Suppose two classes of observations have means μ_1, μ_2:

$$\mu_1 = \left(\frac{1}{5}(3.5 + 4.2 + 3.2 + 3.1 + 4.5), \frac{1}{5}(4.6 + 3.8 + 3.6 + 3.4 + 2.6)\right)^T = (3.7, 3.6)^T.$$

In the same way, we obtain $\mu_2 = (1.5, 2.325)^T$.

And we calculate the mean vector for all data:

$$\mu = \left(\frac{1}{9}(3.7 \times 5 + 1.5 \times 4), \frac{1}{9}(3.6 \times 5 + 2.325 \times 4)\right)^T = (2.722, 3.033)^T.$$

Let us derive the intra-class dispersion matrix S_w :

$$S_1 = \sum_{i=1}^{5}(C_{1,i} - \mu_1)^T \times (C_{1,i} - \mu_1) = \begin{pmatrix} 1.540 & -0.780 \\ -0.780 & 2.080 \end{pmatrix}$$

$$S_2 = \sum_{i=1}^{4}(C_{2,i} - \mu_2)^T \times (C_{2,i} - \mu_2) = \begin{pmatrix} 0.980 & 1.160 \\ 1.160 & 1.648 \end{pmatrix}$$

$$S_w = \frac{5 \times S_1 + 4 \times S_2}{9} = \begin{pmatrix} 1.291 & 0.082 \\ 0.082 & 1.888 \end{pmatrix}.$$

And we calculate the inter-class dispersion matrix S_b :

$$S_b = \frac{1}{9}(5 \times (\mu - \mu_1)^T \cdot (\mu - \mu_1) + 4 \times (\mu - \mu_2)^T \cdot (\mu - \mu_2)) = \begin{pmatrix} 1.195 & 0.693 \\ 0.692 & 0.401 \end{pmatrix}.$$

We find the maximum eigenvalue and eigenvector for:

$$\left(S_w + \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}\right)^{-1} \cdot S_b = \begin{pmatrix} 0.841 & 0.487 \\ 0.314 & 0.182 \end{pmatrix}.$$

We obtain the eigenvalues and eigenvectors:

$$\lambda_1 = 1.022, \ v_1 = \begin{pmatrix} 0.937 \\ 0.350 \end{pmatrix}$$

$$\lambda_2 = -2.775, \ v_2 = \begin{pmatrix} -0.501 \\ 0.865 \end{pmatrix}.$$

We only use the biggest eigenvalue. So the slope of the line is:

$$W = v_1 = \begin{pmatrix} 0.937 \\ 0.350 \end{pmatrix}$$

$$k = \frac{0.350}{0.937} = 0.3732.$$

Newly mapped 1D data become:

$$C_{1,\text{new}} = C_1 \cdot W = \{4.887, 5.263, 4.257, 4.093, 5.125\}$$

$$C_{2,\text{new}} = C_2 \cdot W = \{1.368, 2.642, 1.789, 3.075\}.$$

We draw the data on the line as shown in Figure 4.229: Here "∗" indicates class 1 and "∘" indicates class 2.

(Continued)

Example 4.272 (Continued)

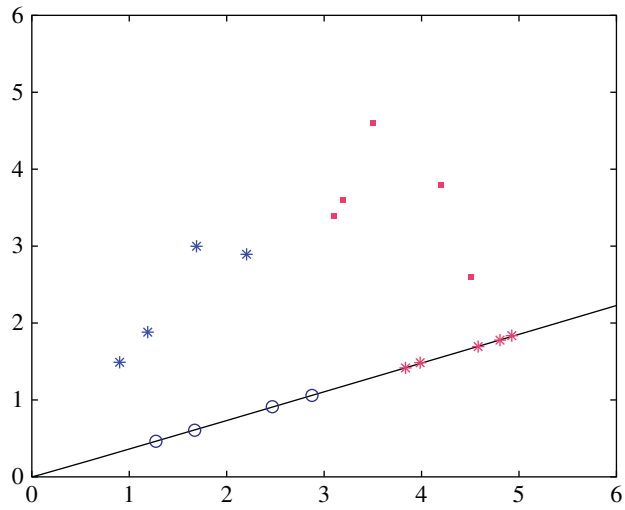


Figure 4.229 The result for Fisher discriminant analysis.

How was texture analysis developed in the past?

Texture classification includes pre-processing, feature extraction, and learning and classification. For training the neural networks, we need to collect a large number of texture images with classes, and store them with labels as training and testing datasets. After pre-processing these images, we extract typical discriminative features from them. Finally, we classify those texture images into different classes.

The history of the development of traditional texture analysis methods is shown in Table 4.17. Gibson investigated visual perception of wallpapers with different angles and proposed a concept of texture gradients as shape-from-texture [30]. In his book, he mentioned, "In the retinal image ab, there exists a gradient of texture from coarse to fine, $\cdots$". Traditional methods include GLCM, the MRF model, the Gabor filter, wavelet, fractal, SIFT, etc. Among these methods, the bags of features (BoF) [68], a vector of locally aggregated descriptors (VLAD) [40], and Fisher vector (FV) encoding [86] methods are three widely used traditional methods on texture recognition. BoF [68] is simply using the k-means methods to cluster image features like scale-invariant feature transform (SIFT) features and then replace the feature points with a clustering center close to the feature points.

VLAD [40], like BoF, only considers the cluster centre closest to the feature point, and VLAD preserves the distance of each feature point to the nearest cluster centre; similar to the FV encoding, VLAD takes into account the value of each dimension of the feature point, and has a more detailed picture of the local information of the image; and the VLAD feature has no loss of information.

The Fisher vector [86] is obtained by modelling the feature points in a generative model (e.g. Gaussian mixture model, GMM). Then the Fisher vector is input into a discriminant classifier (e.g. SVM) to obtain an image classification result. The fisher vector is a representation of the sample features in the fisher kernel that represents a picture as a vector, extending the BoF by encoding higher order statistics (first and second order), retaining information about the fitting

Table 4.17 Traditional methods for the development of texture analysis.

1950	Shape from texture [30]
1962	Texture perception model (random texture) [42]
1973	Grey-level co-occurrence matrix [33]
1980	Laws' filter masks [33]
1981	The texton theory [41]
1983	MRF texture model [17]
1985	Gaussian MRF [13]
1989	Gabor filter [27]
1997	Wavelet [58]
1999	Fractal model [45]
2000	Gabor Wavelets [4]
2001	Bag of Textons [79]
2002	Multi-scale LBP [59]
2004	SIFT [18]
2007	LBP for Facial Texture [102]
2009	Texton-boost [89]
2010	Improved Fisher Vector [55]

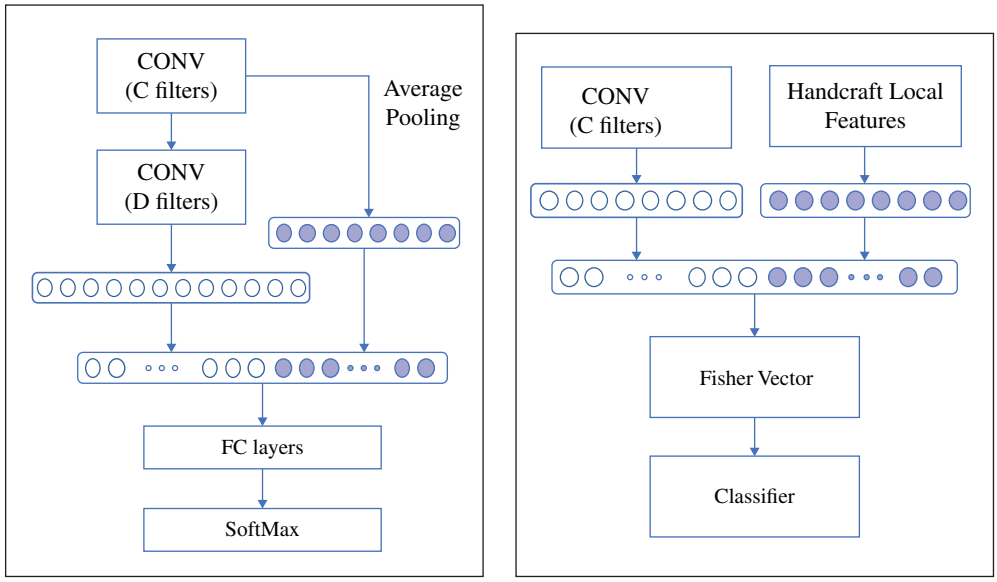
error of the best fit. We will introduce the concept of the Fisher information matrix which is the basis of the Fisher vector.

The time-line introducing development of deep-learning based texture analysis methods are shown as in Table 4.18. Generally, such a deep learning methodology learns network parameters to reduce the least square error with an input signal and the target signal.

Since 2012, the CNN-based method has been greatly developed and has achieved the state-of-the-art in many computer vision tasks. CNN network has powerful feature extraction for images, thus in 2015, researchers began to apply CNN in texture recognition. The frameworks of two CNN based methods are shown as Figure 4.230.

Table 4.18 Time-line of deep-learning based methods.

2012	Deep CNN for ImageNet [50]
2013	Scattering Convolution Network [8]
2014	DeCAF and IFV [15]
2015	CNN for Texture Synthesis [14]
2015	FV-CNN [14]
2016	Texture CNN [3]
2016	Bilinear CNN [56]
2017	Deep TEN [101]
2017	FASON [19]



(a) Texture CNN (b) FV- CNN

Figure 4.230 The framework of TCNN and FV-CNN.

Andrearczyk et al. (2016) proposed texture CNN (T-CNN) [3], utilizing global average pooling extracting an energy measure from the last convolution layer which is connected to a fully connected layer.

Cimpoi et al. (2015) proposed FV-CNN [14] and Song et al. (2017) proposed LFV-CNN [93]. Stating the FV descriptor shows higher recognition performance than FC-CNN, even if the pre-trained VGG-VD [90] model is fine-tuned on the texture dataset, and this method obtained a state-of-the-art result. The convolutional layer using the pre-trained neural network extracted the depth features and used the Fisher vector encoder to refresh the state-of-the-art (middle) at the time.

A method utilizing co-occurrence matrices in a CNN structure is a cascade model adding a co-occurrence feature learning layer to the basic CNN framework [100]. The co-occurrence vectors computed from CNN features of different layers are fused via a fully connected layer obtaining the prediction results.

What is the Fisher information matrix (FIM)?

Let a probability density function be $f(\mathbf{x}, \theta)$ and a log-likelihood of this function be

$$\log L(\theta, \mathbf{x}') = \log f(\mathbf{x}', \theta) \tag{4.988}$$

where we have a sample $\mathbf{x}'$. A score function, which is a partial derivative of the log-likelihood function, is defined by

$$\frac{\partial}{\partial \theta} \log L(\theta, \mathbf{x}') = \frac{1}{L(\theta, \mathbf{x}')} \frac{\partial}{\partial \theta} L(\theta, \mathbf{x}'). \tag{4.989}$$

Note that the expectation of this score function is $E\left[\frac{\partial}{\partial \theta} \log L(\theta, \mathbf{x}')\right] = 0$. The Fisher information amount is a variance of the score function which is denoted by

$$E\left[\left(\frac{\partial}{\partial \theta} \log L(\theta, \mathbf{x}')\right)^2\right] = Var\left[\frac{\partial}{\partial \theta} \log L(\theta, \mathbf{x}')\right]. \tag{4.990}$$

And the Fisher information matrix is provided by $\mathrm{FIM} = [\mathrm{FIM}_{ij}]$, which element is denoted by

$$\mathrm{FIM}_{ij} = E\left[\left(\frac{\partial}{\partial\theta_i}\log L(\theta,\mathbf{x}')\right)\left(\frac{\partial}{\partial\theta_j}\log L(\theta,\mathbf{x}')\right)\right] = E[G_i \times G_j] \tag{4.991}$$

where we define a score vector by

$$\mathbf{G} = (G_1, G_2, \cdots, G_K)^T, G_i = \frac{\partial}{\partial\theta_i}\log L(\theta,\mathbf{x}').$$

Example 4.273

If the probability density function is

$$f(x,\mu,\sigma) = \frac{1}{(2\pi)^{1/2}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

calculate the Fisher information matrix of this function.

From Equation 4.273, we calculate the log-likelihood as follows,

$$\begin{aligned}\log L(\mu,\sigma,x') &= -\frac{1}{2}\log(2\pi) - \log\sigma - \frac{(x'-\mu)^2}{2\sigma^2}\\ &= -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma^2 - \frac{(x'-\mu)^2}{2\sigma^2}\end{aligned}$$

and derive two score functions, which take the derivative with μ and σ^2,

$$\frac{\partial}{\partial\mu}\log L(\mu,\sigma,x') = \frac{x'-\mu}{\sigma^2}$$

$$\frac{\partial}{\partial\sigma^2}\log L(\mu,\sigma,x') = -\frac{1}{\sigma^2} + \frac{(x'-\mu)^2}{2\sigma^4}.$$

From Equation (4.990), the Fisher information amount is

$$E\left[\left(\frac{\partial}{\partial\mu}\log L(\mu,\sigma,x')\right)^2\right] = E\left[\frac{(x'-\mu)^2}{\sigma^4}\right] = \frac{1}{\sigma^2},$$

$$E\left[\left(\frac{\partial}{\partial\sigma^2}\log L(\mu,\sigma,x')\right)^2\right] = E\left[\left(-\frac{1}{\sigma^2} + \frac{(x'-\mu)^2}{2\sigma^4}\right)^2\right] = \frac{1}{2\sigma^4}.$$

$$E\left[\left(\frac{\partial}{\partial\mu}\log L(\mu,\sigma,x')\right)\left(\frac{\partial}{\partial\sigma^2}\log L(\mu,\sigma,x')\right)\right] = 0.$$

Finally, the Fisher information matrix is

$$\mathrm{FIM} = \begin{pmatrix}\frac{1}{\sigma^2} & 0\\ 0 & \frac{1}{2\sigma^4}\end{pmatrix}.$$

How do we implement the co-occurrence matrix feature map?

Let us take up the basic method GLCM from the conventional methods, and consider the connection between the feature map and the CNN. For the construction of a co-occurrence matrix it is required to consider all pairs of pixels that are at a fixed distance from each other. Irrespective of the relative orientations, the line that joins them forms in the reference direction of the image.

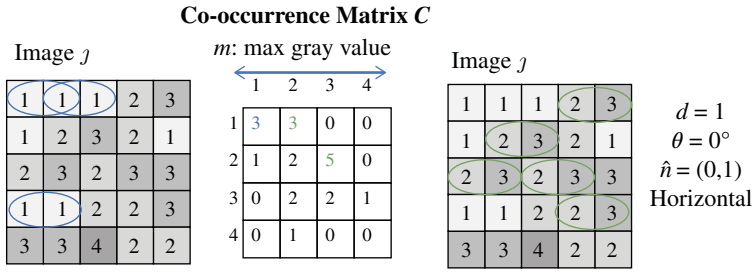


Figure 4.231 The calculation process of the co-occurrence matrix.

Such matrices are parametric by the distance d only, and the system may have as many matrices as it chooses to use Equation 3.267. The calculation process of co-occurrence matrix is shown in Figure 4.231.

As all pairs connect to one certain pixel, the network could construct a digital circle to represent all directions, for example, setting $d = 4$, the network may calculate GLCM in 16 directions. According to Table 4.19, we have all the pairs of pixels as following offsets [-4,1], [-4,2], [-3,2], [-3,3], [-2,3], [-2,4], [-1,4], [0,4], [1,4], [2,4], [2,3], [3,3], [3,2], [4,2], [4,2], [4,0]. By calculating the GLCM inn all the directions with grey level $k = 56$, the network could get a $16 \times 56 \times 56$ size feature map for each texture image. Different angle θ features for a texture image of distance $d = 1$ are shown in Figure 4.232.

The RGB to grey conversion is calculated by:

$$\text{Grey} = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{4.992}$$

where Grey indicates the computation result of grey image conversion, and R, G, B indicate the red, green, blue channel colour numerical values of the original image, respectively.

Figure 4.233 shows the visualization of GLCM features with parameter $d = 4$, horizonal direction $\theta = 0°$, setting gray level $k = 56$ of banded, blotchy, bumpy, and gauzy texture images. Different textures show distinguishing features, thus proving the GLCM is an applicable descriptor for textures.

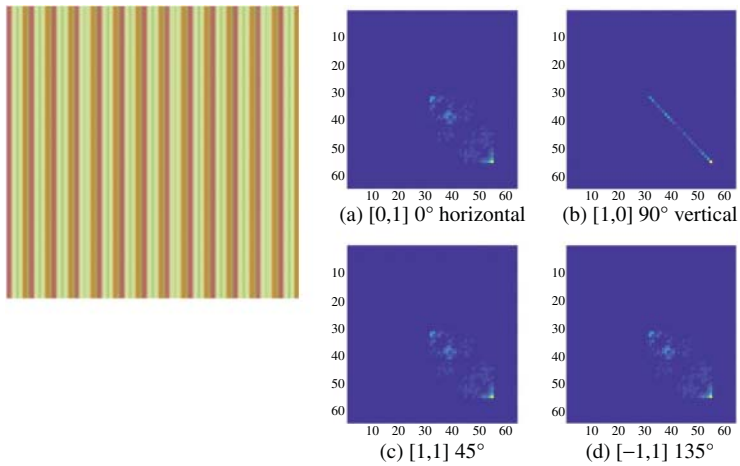


Figure 4.232 Example of different angle θ features for a texture image of distance $d = 1$. Source: Sei-ichiro Kamata.

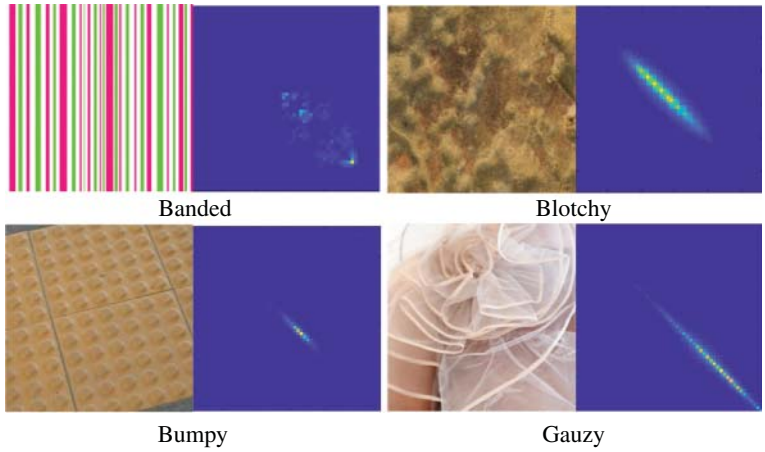


Figure 4.233 The visualization of GLCM features with parameter $d = 4$, horizontal direction $\theta = 0^{\circ}$, setting grey level $k = 56$ of banded, blotchy, bumpy, and gauzy texture images; different textures shows distinguishing features, thus proving the GLCM is an applicable descriptor for textures. Source: Sei-ichiro Kamata.

Table 4.19 The coordinates of the pixels that make up the digital circle according to certain distance.

Distance d	Direction θ
$\mathbf{d = 1}$	$[-1, 0], [-1, 1], [0, 1], [1, 1]$
$\mathbf{d = 2}$	$[-2, 0], [-2, 1], [-1, 2], [0, 2],$ $[1, 2], [2, 1], [1, -2], [0, -2],$ $[-1, -2], [-2, -1]$
$\mathbf{d = 4}$	$[-4, 1], [-4, 2], [-3, 2], [-3, 3],$ $[-2, 3], [-2, 4], [-1, 4], [0, 4],$ $[1, 4], [2, 4], [2, 3], [3, 3],$ $[3, 2], [4, 2], [4, 2], [4, 0]$

In order to find the good parameter setting for GLCM feature extraction, the network set experiment using distances respectively, with all the setting directions according to Table 4.19.

The GLCM approach is based on the statistics of pixel intensity distributions that express the relative frequencies (or probabilities) $P(i, j|d, \theta)$. Two pixels having relative polar coordinates (d, θ) appear with intensities i, j. The co-occurrence matrices provide raw numerical data on the texture.

Basebone CNN structure

VGG-Net explores the relationship between the depth of the convolutional neural network and its performance by repeatedly stacking 3×3 small convolution kernels and 2×2 maximum pooling layers (Figure 4.234). VGG-Net successfully constructed a 16–19-layer deep convolutional neural network. VGGNet has a significantly lower error rate than the previous state-of-the-art network architecture. The VGG-Net paper uses 3×3 small convolution kernels and 2×2 maximum pooling cores to improve performance by continuously deepening the network structure [90].

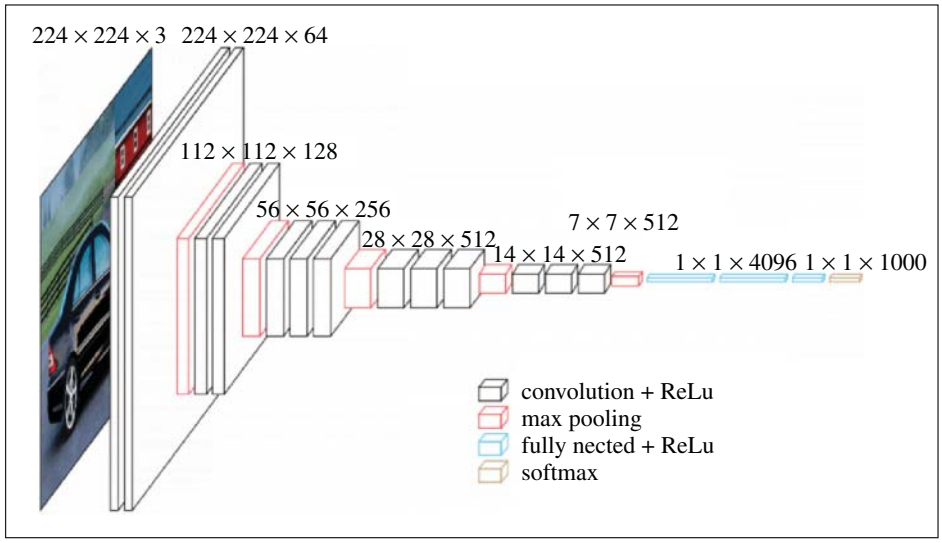


Figure 4.234 The framework and feature map dimension of Vgg-16 Network[90].

The problem of degradation of deep networks at least indicates that deep networks are not easy to train. The ResNet network is referenced to the VGG-19 network, modified on the basis of it, and the residual unit is added through the short circuit mechanism. The change is mainly reflected in the fact that ResNet directly uses the convolution of stride= 2 for down-sampling, and replaces the fully connected layer with the global average pool layer. An important design principle of ResNet is that when the feature map size is reduced by half, the number of feature maps is doubled, which preserves the complexity of the network layer [35].

There are several texture datasets available

The network evaluate the result using three widely accepted texture and material datasets in texture analysis: The Describable Textures Dataset (DTD) image database and the Flickr Material Database (FMD), which are introduced as follows.

(1) DTD Dataset

The DTD [15] is an evolving collection of textural images in the wild, annotated with a series of human-centric attributes, inspired by the perceptual properties of textures. This dataset is made available to the computer vision community for research purposes. The DTD contains 120 images for each of the 47 texture classes, 5640 images in total. It is considered as the most challenging data set because it contains wild images. Using the protocol published with the dataset, the dataset is divided into 10 splits randomly, and every split is divided into three parts, one is used for training, one is used for verification, and final part is used for testing. Examples of the dataset are shown in Figure 4.235.

(2) FMD Dataset

The FMD [88] was constructed with the specific purpose of capturing a range of real world appearances of common materials (e.g. glass, plastic, etc.), ensuring a variety of illumination conditions, compositions, colors, texture and material sub-types. The FMD data set contains 100 images in each of 10 material classes, that is 1000 images in amount. During the experiment, the network randomly select half dataset images for training, with one quarter of the training images set as

(a) Braided (b) Bumpy (c) Cracked

Figure 4.235 Examples of the DTD dataset.

(a) Fabric (b) Foliage (c) Glass (d) Metal

Figure 4.236 Examples of the FMD dataset.

the validation set and the other half used for testing step. Examples of the dataset are shown in Figure 4.236.

We set up several deep networks for texture classification

This example shows how to create a deep learning neural network with residual connections and train it on DTD database. Residual connections are a popular element in convolutional neural network architectures. Using residual connections improves gradient flow through the network and enables training of deeper networks.

Environment

A test environment is utilised on Matlab 2019a using NVIDIA CUDA 10.2 and the Deep Learning Toolbox: https://www.mathworks.com/products/deep-learning.html Hardware system includes: CPU: AMD Ryzen 5 3600, and GPU: Nvidia GeForce GTX 1080Ti with 32 GB memory. We use open dtd_dataset_tool and adjust the dataset path and save path in the file. Image sizes range from 300×300 to 640×640, and the images contain at least 90% of the surface representing the category attribute. All the images in the dataset are resized into 224×224. Before running dtd_dataset_tool, the program reads all the images and labels in the dataset, and all the images are divided into train, validation and test subsets evenly. Each subset has 1880 images.

Define network architecture

The residual network architecture consists of three components: main branch with convolutional units, batch normalization and ReLU functions. Residual connections bypass the convolutional units in the main branch. The ResNHere "·" indicate class 1 and "★" indicate class 2. A flowchart is shown in Figure 4.237.

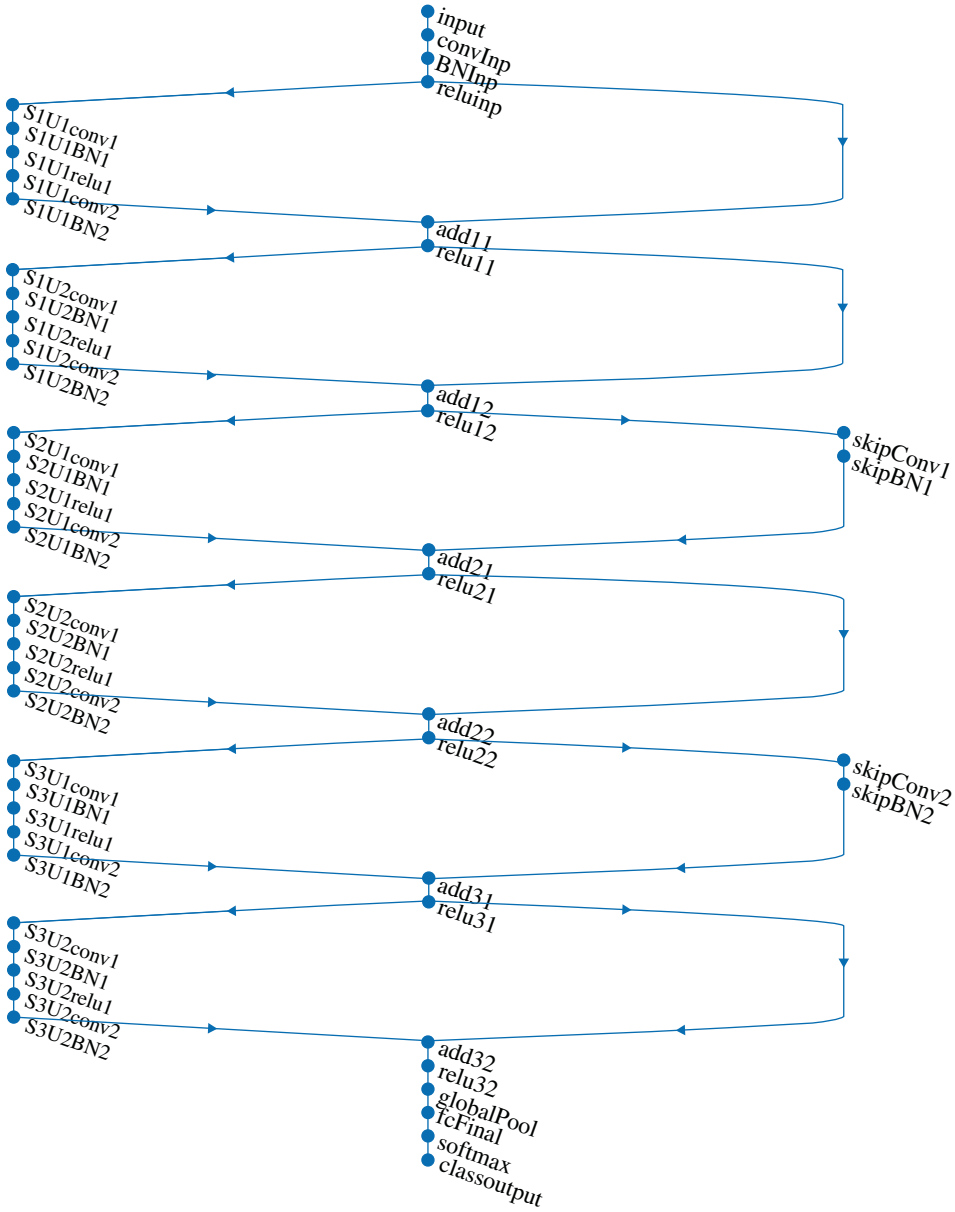


Figure 4.237 ResNet flowchart.

Create main branch

The main branch contains five sections in the figure. The initial section includes image input layer INPUT, initial convolution CONV, batch normalisation BN and ReLU function RELU. The next three sections of convolutional layer have different size of features. Each section contains N convolutional units. In this case, $N = 2$. Each convolutional unit contains two 3×3 convolutional layers. The netWidth parameter is the network width, defined as the number of convolutional filters in one convolutional layer. The netWidth is set to 64. In the figure, S1U1 means that the section

number is 1 and the unit number is 1. The final section has global average pooling, fully connected, softmax, and classification layers.

Create residual connections

The core idea of ResNet is the so-called "identity shortcut connection" that skips one or more layers. We add residual connections around the convolutional units. Most residual connections perform no operations and simply add element-wise to the outputs of the convolutional units.

Training networks

We train the network for 150 epochs. And we select the learning rate as 0.05, which is proportional to the mini-batch size and we reduce the learning rate by a factor of 5 after 60 epochs. We validate the network once per epoch using the validation data. miniBatchSize is set as 50 in this example. Training the network takes about 1000 minutes. The error convergence for each epochs is shown in Figure 4.238.

Evaluate trained networks

We compute the final accuracy of several networks using the datasets: DTD and FMD. The baseline experiments are constructed on the "very deep VGG network" (VGG-16) [90]. The numbers of gray level of the co-occurrence matrices are set to 56, thus the network can get a sized co-occurrence feature for each image. The accuracy of several networks are shown in Table 4.20. Since the state of art methods contain several steps that cannot be trained in single network. Results show the ResNet achieves a little better performance than the baseline. All the networks do not include pre-trained network parameters using Imagenet or other datasets.

What is the "take-home" message of this chapter?

Texture segmentation tries to answer the question "what is where" in an image. This question may be answered by creating a basis of elementary answers in terms of which we express the full answer

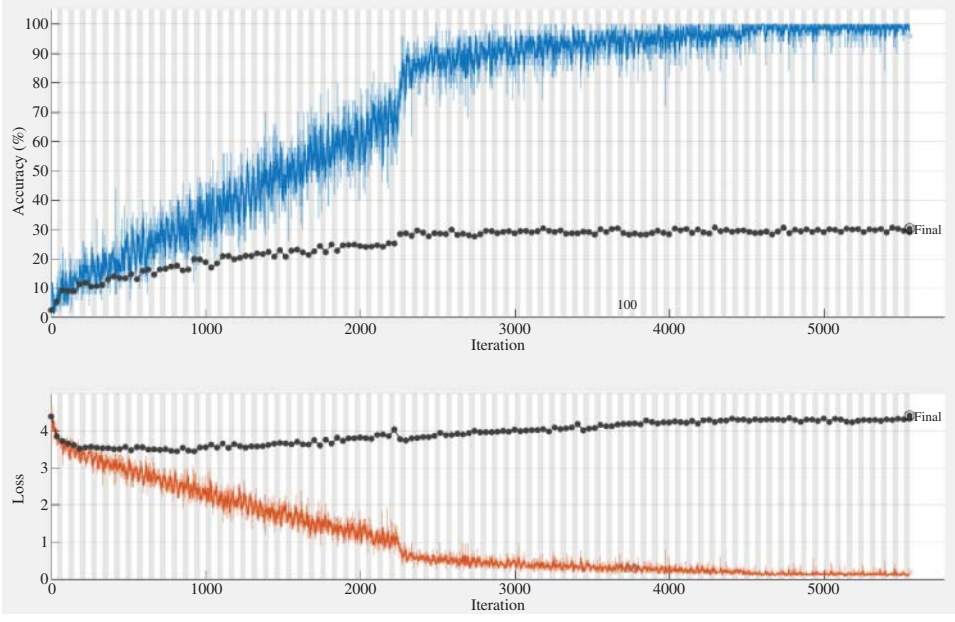


Figure 4.238 ResNet convergence.

Table 4.20 Results compared with other methods.

Method datasets	DTD	FMD
Baseline[90]	15.96	31.67
Texture CNN[3]	15.80	32.00
GLCM+CNN	16.81	26.67
ResNet	30.72	45.00

a) The unit of accuracy is per cent.

to the problem. Both Gabor functions and wavelets produce such bases appropriate either for use in conjunction with the image itself, or with its Fourier transform. The most appropriate of these bases is the one produced by using the prolate spheroidal sequence functions, which are a type of wavelets specifically designed for this purpose. The conventional wavelet bases have been primarily designed to allow the most compact representation of a signal and they are not necessarily the most appropriate for texture segmentation.

These techniques answer in a formal way the question "what is where" in the image without trying to make sense of what the "what" part of the answer means. Alternative approaches try to capture perceptually salient image characteristics. Such approaches do not refer to the frequency domain at all, but they transform the image with the help of filters that capture something of the local structure. Such filters are the masks developed by Laws and the coefficients of the expansion of a local neighbourhood in terms of Walsh functions. Instead of linar filters, a non-linear approach may be used, namely the one that extracts from the image local binary patterns to characterise the texture.

A compromise between the two approaches is based on the use of the Wigner distribution, which is also a non-linear approach. The Gabor and wavelet approaches answer the "what" part of the question by considering the magnitude of the local Fourier transform of the signal. The magnitude, however, is known to convey much less information than the phase. The Wigner distribution encodes implicitly local phase information that is nothing else than a measure of local image symmetry. So, the Wigner distribution offers a frequency as well as a structural image representation. Finally, deep CNN tries to capture information on textures in some way, but it is quite hard to figure out what information in the network is important in a texture image.

Bibliographical Notes

There are parts of this book that constitute original work. However, most of it is based on material contained in several other books and scientific papers. There are thousands of published papers on texture analysis and no attempt has been made here to list them let alone to review them. The material in the book constitutes the core of what is behind and beneath the majority of these papers. The list that follows consists of the major sources used to develop this book, and some of them are peripheral to image processing. For example, reference [1] is an invaluable source of formulae used to calculate approximately certain functions, like the error function in Chapter 2 and the Bessel function in Chapter 4. Reference [73] is irreplaceable for anybody who wants to understand stochastic processes. In certain places reference to Book I is made. This is the book "Image Processing, the Fundamentals" by M. Petrou and P. Bosdogianni [75].

Some bibliographical notes concerning individual chapters follow. The list of references given is by no means complete. There are hundreds of books and publications from which the authors have been benefited over the years in addition to this list. It would not be possible to identify and include all of them. It is hoped that the list will serve as a springboard for the interested reader to access the sources that are not listed.

Chapter 1: The algorithm on texture boundary detection described here was first published in [71].

Chapter 2: The binary images used were scanned from the copyright-free book [103]. The algorithm on Hilbert curves was kindly provided by Sei-ichiro Kamata and it is based on [44]. The parameter estimation methods for the Boolean model are based on the excellent book on the subject by Ilya Molchanov [67]. Finally, the seminal book by Jean Serra [87] is a major source for the subject of mathematical morphology.

Chapter 3: Images 3.1a,c,e,f,g were taken from [105], while images 3.1b,d,h were downloaded from [104]. Significant sources for the section on fractals were the original work by Mandelbrot [63], the excellent paper by Voss [95] and the books by Falconer [24] and Turner et al. [94]. The Ransac algorithm used for robust estimation was taken from [26]. The use of Markov random fields to texture creation and characterisation was pioneered by Cross and Jain [17]. For estimation of the Markov parameters very useful references were [7] and [31]. The relationship between fractals and Markov random fields was first drawn to the attention of the first author by Dr Sarah Bell, at a meeting in London [5]. The proof about the significance of the critical temperature is taken from the book by Kindermann and Snell [46] which is an excellent tutorial on MRFs. The proof of the Hammersley–Clifford theorem is based on Besag's seminal work [6]. Also, useful insight into the subject was gained from references [12, 22] and [47]. The discussion on maximum likelihood estimation is based on the excellent pattern recognition book by Duda et al [23]. The algorithm of simulated annealing was first introduced to image processing by Geman and Geman [29]. The

Image Processing: Dealing with Texture, Second Edition. Maria Petrou and Sei-ichiro Kamata.

Companion Website: www.wiley.com/go/kamataText2

algorithm described here for phase unwrapping was taken from [70]. For the original definition of co-occurrence matrices and features constructed from them, the major source has been [32]. For using ratios of co-occurrence matrix elements as features and generalised co-occurrence matrices see [48] and [49]. Finally, for the use of co-occurrence matrices for illumination direction identification the reader should look at [54].

Chapter 4: The section on uncertainty and 2D Gabor functions is largely based on the seminal work by Daugman [20, 21]. The section on prolate spheroidal functions is based on the work presented in [91, 51] and [92]. In this respect [99] is also a useful reference. References [16] and [9] were very useful in understanding the use of Gabor functions in texture segmentation. The ψ transform for feature extraction from Gabor coefficients originated in [25] and was further investigated in [77]. The book by Akansu and Haddad [2] proved to be an excellent textbook on wavelets, while the book by Mallat [61] gives a comprehensive coverage of the subject. The idea of the structure tree was first published in [11]. The idea of using a distance histogram for feature selection was first published in [72]. The deterministic annealing algorithm is based on references [80] and [81]. Laws' masks were first proposed in [53]. The rest of that section has not been published anywhere else. The work on local binary patterns has been championed by the team at Oulu University in a series of papers and a good web-based tutorial [60, 76, 106]. The use of Wigner distributions in texture analysis has been pioneered by Wechsler [96], and the original reference for the Kaiser window is [43]. Finally, the overview of several deep learning based classification methods was introduced in the paper [57].

References

1 M Abramowitz and I A Stegun (eds), 1970. *Handbook of Mathematical Functions*, Dover Publications, Inc, New York, ISBN 486 61272 4, Library of Congress Catalogue 65–12253.

2 A N Akansu and R A Haddad, 1992. *Multiresolution Signal Decomposition*, Academic Press, ISBN 0 12 047141 8.

3 V Andrearczyk and P F Whelan, 2016. "Using filter banks in convolutional neural networks for texture classification", *Pattern Recognition Letters*, Vol 84, pp 63–69.

4 S Arivazhagan, L Ganesan and S P Priyal, 2006. "Texture classification using Gabor wavelets based rotation invariant features", *Pattern Recognition Letters*, Vol 27, No 16, pp 1976–1982.

5 S B M Bell, A talk on "Markov Models and Fractals" presented at the one-day technical meeting on "Markov Models in Computer Vision" organised by the British Machine Vision Association on May 22, 1990 in London.

6 J Besag, 1974. "Spatial interaction and the statistical analysis of lattice systems", *Journal of the Royal Statistical Society, London B*, Vol 36, No 2, pp 192–225.

7 C F Borges, 1999. "On the estimation of Markov random field parameters", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 21, pp 216–224.

8 J Bruna and S Mallat, 2013. "Invariant scattering convolution networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 35, No 8, pp 1872–1886.

9 J M H du Buf, 1990. "Gabor phase in texture discrimination", *Signal Processing*, Vol 21, pp 221–240.

10 F W Campbell and J G Robson, 1968. "Application of Fourier analysis to the visibility of gratings", *Journal of Physiology*, Vol 197, No 3, pp 551–566.

11 T Chang and C C J Kuo, 1993. "Texture analysis and classification with tree-structured wavelet transform", *IEEE Transactions on Image Processing*, Vol 2, pp 429–441.

12 D Chantler, 1987. *Introduction to Modern Statistical Mechanics*, Chapter 5: "Statistical mechanical theory of phase transitions", Oxford University Press, ISBN 0 19 504276 X.

13 R Chellappa and S Chatterjee, 1985. "Classification of textures using Gaussian Markov random fields", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol 33, No 4, pp 959–963.

14 M Cimpoi, S Maji and A Vedaldi, 2015. "Deep filter banks for texture recognition and segmentation", *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp 3828–3836.

15 M Cimpoi, S Maji, I Kokkinos, S Mohamed and A Vedaldi, 2014. "Describing textures in the wild", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp 3606–3613.

Image Processing: Dealing with Texture, Second Edition. Maria Petrou and Sei-ichiro Kamata.

Companion Website: www.wiley.com/go/kamataText2

16 M Clark, A C Bovik and W S Geisler, 1984. "Texture segmentation using Gabor modulation/demodulation", *Pattern Recognition Letters*, Vol 6, pp 261–267.

17 G R Cross and A K Jain, 1983. "Markov random field texture models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 5, pp 25–39.

18 G Csurka, C Dance, L Fan, J Willamowski and C Bray, 2004. "Visual categorization with bags of keypoints", *Workshop on statistical learning in computer vision, ECCV*, Vol 1, No 1-22, pp 1–2.

19 X Dai, N J Yue-Hei and L S Davis, 2017. "FASON: First and second order information fusion network for texture recognition", *IEEE Conference on Computer Vision and Pattern Recognition*, pp 7352–7360.

20 J G Daugman, 1985. "Uncertainty relation for resolution in space, spatial frequency and orientation optimised by 2D visual cortical filters", *Journal of the Optical Society of America*, Vol 2, pp 1160–1169.

21 J G Daugman, 1988. "Complete discrete 2D Gabor transforms by neural networks for image analysis and compression", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol 36, pp 1169–1179.

22 P A Devijver and J Kittler, 1987. "Decision making in context", Course notes of the Course on Statistical Pattern Recognition, The Cosener's House, Abingdon, October 8–9.

23 R O Duda, P E Hart and D G Stork, 2001. *Pattern Classification*, John Wiley and Sons, ISBN 0 471 05669 3.

24 K Falconer, 1990. *Fractal Geometry–Mathematical Foundations and Applications*, John Wiley and Sons, ISBN 0 471 92287 0.

25 F Farrokhnia and A K Jain, 1991. "A multichannel filtering approach to texture segmentation", Proceedings of the Computer Vision and Pattern Recognition Conference, CVPR 1991, pp 364–370.

26 M A Fischer and R C Bolles, 1981. "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of ACM*, Vol 24, pp 381–395.

27 I Fogel and D Sagi, 1989. "Gabor filters as texture discriminator", *Biological Cybernetics*, Vol 61, No 2, pp 103–113.

28 K Fukushima, 1980. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biological Cybernetics*, Vol 36, pp 193–202.

29 S Geman and D Geman, 1984. "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 6, pp 721–741.

30 J J Gibson, 1950. *The Perception of the Visual World*, Houghton Mifflin, Boston.

31 M Gurelli and L Onural, 1994. "On a parameter estimation method for Gibbs–Markov random fields", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 16, pp 424–430.

32 R M Haralick, 1986. "Statistical image texture analysis", *Handbook of Pattern Recognition and Image Processing*, T Y Young and K S Fu (eds), Academic Press, pp 247–279.

33 R M Haralick, K Shanmugam, et al., 1973. "Textural features for image classification", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol 3, No 6, pp 610–62.

34 H K Hartline and F Ratliff, 1957, "Inhibitory interaction of receptor units in the eye of Limulus", *Journal of General Physiology*, Vol 40, No 3, pp 357–376.

35 K He, X Zhang, S Ren and J Sun, 2016. "Deep residual learning for image recognition", *IEEE Conference on Computer Vision and Pattern Recognition*, pp 770–778.

36 D Hebb, 1949. *The Organization of Behavior: A Neuropsychological Theory*, Wiley and Sons, New York.

37 S Herculano-Houzel, 2009. "The human brain in numbers: a linearly scaled-up primate brain" *Frontiers in Human Neuroscience* https://doi.org/10.3389/neuro.09.031.2009.

38 D Hubel and T Wiesel, 1959. "Receptive fields of single neurones in the cat's striate cortex", *Journal of Physiology*, Vol 148, pp 574–591.

39 D Hubel and T Wiesel, 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". *Journal of Physiology*, Vol 160, pp 106–154.

40 H Jégou, M Douze, C Schmid and P Pérez, 2010. "Aggregating local descriptors into a compact image representation", *23rd IEEE Conference on Computer Vision & Pattern Recognition, CVPR 2010* pp 3304–3311.

41 B Julesz, 1981. "Textons, the elements of texture perception, and their interactions", *Nature*, Vol 290, No 5802, pp 91.

42 B Julesz, Visual pattern discrimination, 1962. *IRE Transactions on Information Theory*, Vol 8, No 2, pp 84–92.

43 J F Kaiser, 1966. Chapter 7: "Digital filters", in *System Analysis by Digital Computer*, F F Kuo and J F Kaiser (eds), John Wiley and Sons.

44 S Kamata, R O Eason and E Kawaguchi, 1993. "An implementation of the Hilbert scanning algorithm and its application to data compression", *The Institute of Electronics, Information and Communication Engineers IEICE Transactions on Information and Systems*, Vol E76-D, No 4, pp 420–428.

45 L M Kaplan, 1999. "Extended fractal analysis for texture classification and segmentation", *IEEE Transactions on Image Processing*, Vol 8, no 11, pp 1572–1585.

46 R Kindermann and J L Snell, 1980. *Markov Random Fields and Their Applications*, first book of the AMS soft-cover series in Contemporary Mathematics, American Mathematical Society.

47 J B Kogut, 1979. "An introduction to lattice gauge theory and spin systems", *Reviews of Modern Physics*, Vol 51, No 4, pp 659–713.

48 V Kovalev and M Petrou, 1996. "Multidimensional co-occurrence matrices for object recognition and matching", *Graphical Models and Image Processing*, Vol 58, pp 187–197.

49 V A Kovalev, M Petrou and Y S Bondar, 1998. "Using orientation tokens for object recognition", *Pattern Recognition Letters*, Vol 19, pp 1125–1132.

50 A Krizhevsky, I Sutskever and G E Hinton, 2012. "Imagenet classification with deep convolutional neural networks", *Advances in Neural Information Processing Systems*, pp 1097–1105.

51 H J Landau and H O Pollak, 1961. "Prolate spheroidal wave functions, Fourier analysis and uncertainty II", *The Bell System Technical Journal*, Vol 40, pp 65–84.

52 Y LeCun, B Boser, J S Denker, D Henderson, R E Howard, W Hubbard and L D Jackel, 1989. "Backpropagation applied to handwritten zip code recognition", *Neural Computation*, Vol 1, No 4, pp 541–551.

53 K I Laws, 1979. "Texture image segmentation", University of Southern California, Department of Electrical Engineering Systems, Technical Report USCIPI No 940.

54 X Llado, A Oliver, M Petrou, J Freixenet and J Marti, 2003. "Simultaneous surface texture classification and illumination tilt angle prediction", The British Machine Vision Conference, Norwich, UK, September 8–11, Harvey and Bangham (eds), pp 789–798.

55 J Li and P Cui, 2009. "Improved kernel fisher discriminant analysis for fault diagnosis", *Expert Systems with Applications*, Vol 36, No 2, pp 1423–1432.

56 T Y Lin, A RoyChowdhury and S Maji, 2015. "Bilinear CNN models for fine-grained visual recognition", *IEEE International Conference on Computer Vision*, pp 1449–1457.

57 L Liu, J Chen, P Fieguth, G Zhao, R Chellappa and M Pietikäinen, 2019. "From BoW to CNN: Two decades of texture representation for texture classification", *International Journal of Computer Vision*, Vol 127, No 1, pp 74–109.

58 S Livens, P Scheunders, G Van de Wouwer and D Van Dyck, 1997. "Wavelets for texture analysis, an overview", *1997 Sixth International Conference on Image Processing and Its Applications, Dublin*, pp 1–5.

59 T Mäenpää and M. Pietikäinen, 2003. "Multi-scale binary patterns for texture analysis", *Scandinavian Conference on Image Analysis*, pp 885–892.

60 T Maenpaa and M Pietikainen, 2005. "Texture analysis with local binary patterns", in *Handbook of Pattern Recognition and Computer Vision*, C H Cahen and P S P Wang (eds), World Scientific, pp 197–216.

61 S Mallat, 1998. *A wavelet tour of signal processing*, Academic Press, ISBN 0 12 466605 1.

62 P Mallikarjuna, A T Targhi, M Fritz, E Hayman, B Caputo and J O Eklundh, 2006. "The kth-tips2 database", *Computational Vision and Active Perception Laboratory* (CVAP), Stockholm, Sweden).

63 B B Mandelbrot and J W Van Ness, 1968. "Fractional Brownian motions, fractional noises and applications", *SIAM Review*, Vol 10, pp 422–437.

64 D. Marr, Vision, San Francisco: W. H. Freeman, 1982

65 W Mcculloch and W Pitts, 1943. "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, Vol 5, pp 115–133.

66 M Minsky and S Papert, 1969. *Perceptron: An Introduction to Computational Geometry*, MIT Press.

67 I Molchanov, 1997. *Statistics of the Boolean Model for Practitioners and Mathematicians*, John Wiley and Sons, ISBN 0 471 97102 2.

68 E Nowak, F Jurie and B Triggs, 2006. "Sampling strategies for bag-of-features image classification", *European Conference on Computer Vision*, pp 490–503.

69 T Ojala, M Pietikäinen and D Harwood, 1996. "A comparative study of texture measures with classification based on featured distributions", *Pattern Recognition*, Vol 29, No 1, pp 51–59.

70 A V Oppenheim and R W Schafer, 1975. *Digital Signal Processing*, Prentice–Hall, ISBN 0 13 214635 5.

71 P L Palmer and M Petrou, 1997. "Locating boundaries of textured regions", *IEEE Transactions on Geoscience and Remote Sensing*, Vol 35, pp 1367–1371.

72 P L Palmer, N Fatemi-Ghomi and M Petrou, 1995. "Feature selection for the tree-wavelet transform", 6th International Conference of Computer Analysis of Images and Patterns (CAIP), Prague, September, pp 820–825.

73 A Papoulis, 1965. *Probability, Random Variables and Stochastic Processes*, McGraw-Hill Kogakusha, Ltd, Library of Congress Catalogue Card Number 64-22956.

74 A Pentland, 1984. "Fractal-based description of natural scenes", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, Vol 6 No 6, pp 661–674.

75 M Petrou and P Bosdogianni, 1999. *Image Processing, the Fundamentals*, John Wiley and Sons, ISBN 0 471 99883 4.

76 M Pietikainen, T Ojala and Z Xu, 2000. "Rotation invariant texture classification using feature distributions", *Pattern Recognition*, Vol 33, pp 43–52.

77 T Randen and J H Husoy, 1994. "Multichannel filtering for image texture segmentation", *Optical Engineering*, Vol 33, pp 2617–2625.

78 F Ratliff, 1965. *Mach Bands: Quantitative Studies on Neural Networks in the Retina*, Holden-Day, San Francisco.

79 X Ren and J Malik, 2003. "Learning a classification model for segmentation", *IEEE International Conference on Computer Vision*, pp 1–10.

80 K Rose, E Gurewitz and G C Fox, 1992. "Vector quantisation by deterministic annealing", *IEEE Transactions on Information Theory*, Vol 38, pp 1249–1257.

81 K Rose, E Gurewitz and G C Fox, 1993. "Constrained clustering as an optimisation method", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 15, pp 785–794.

82 F Rosenblatt, 1958. "The Perceptron: A probabilistic model for information storage and organization in the brain", *Psychological Review*, Vol 65, No 6, pp 386–408.

83 F Rosenblatt, 1964. "Analytic techniques for the study of neural nets", *IEEE Transactions on Applications and Industry*, Vol 83, No 74, pp 285–292.

84 D E Rumelhart, G E Hinton and R J Williams, 1986. "Learning representations by back-propagating errors", *Nature*, Vol 323 (6088) pp 533–536.

85 O Russakovsky, et al., 2015. "ImageNet Large Scale Visual Recognition Challenge", *International Journal of Computer Vision*, Vol 115, pp 211–252.

86 J Sánchez, F Perronnin, T Mensink and J Verbeek, 2013. "Image classification with the fisher vector: Theory and practice", *International Journal of Computer Vision*, Vol 105, No 3, pp 222–245.

87 J Serra, 1982. *Image Analysis and Mathematical Morphology, Volume 1*, Academic Press, ISBN 0 12 637241 1.

88 L Sharan, R Rosenholtz and E Adelson, 2009. "Material perception: What can you see in a brief glance?", *Journal of Vision*, Vol 9, No 8, pp 784–784.

89 J Shotton, J Winn, C Rother and A Criminisi, 2009. "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context", *International Journal of Computer Vision*, Vol 81, No 1, pp 2–23.

90 K Simonyan and A Zisserman, 2014. "Very deep convolutional networks for large-scale image recognition", arXiv preprint arXiv:1409.1556.

91 D Slepian and H O Pollak, 1961. "Prolate spheroidal wave functions, Fourier analysis and uncertainty I", *The Bell System Technical Journal*, Vol 40, pp 43–63.

92 D Slepian, 1977. "Prolate spheroidal wave functions, Fourier analysis and uncertainty V: The discrete case", *The Bell System Technical Journal*, Vol 57, pp 1371–1430.

93 Y Song, F Zhang, Q Li, H Huang, L J O'Donnell and W Cai, 2017. "Locally-transferred fisher vectors for texture classification", *Proceedings of the IEEE International Conference on Computer Vision*, pp 4912–4920.

94 M J Turner, J M Blackledge and P R Andrews, 1998. *Fractal Geometry in Digital Imaging*, Academic Press, ISBN 0 12 703970 8.

95 R F Voss, 1985. "Random fractal forgeries", in *Fundamental Algorithms for Computer Graphics*, R A Earnshaw (ed), Springer-Verlag, NATO ASI Series, ISBN 3 540 13920 6, Proceedings of the NATO Advanced Study Institute on Fundamental Algorithms for Computer Graphics held in Ilkley, Yorkshire, England, March 30–April 12, pp 805–835.

96 H Wechsler, 1990. *Computational Vision*, Academic Press, ISBN 0 12 741245 X.

97 B Widrow and M E Hoff, 1960. "Adaptive switching circuits", *1960 IRE WESCON Convention Record, Part 4, New York*, pp 96–104.

98 B Widrow and M A Lehr, 1990. "30 years of adaptive neural networks: perceptron, Madaline, and backpropagation", *Proceedings of IEEE*, Vol 78, No 9, pp 1415–1442.

99 R Wilson and M Spann, 1988. *Image Segmentation and Uncertainty*, Research Studies Press Ltd, and John Wiley and Sons, ISBN 0 86380 067 X and ISBN 0 471 91826 1.

100 S Ya-Fang, Y Yang-Ming, Yen-Yu, W Ming-Fang, L Yi-Chang and C Yung-Yu, 2017. "Deep co-occurrence feature learning for visual object recognition", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp 4123–4132.

101 H Zhang, J Xue, Jia and K Dana, 2017. "Deep ten: Texture encoding network", *IEEE Conference on Computer Vision and Pattern Recognition*, pp 708–717.

102 G Zhao and M Pietikainen, 2007. "Dynamic texture recognition using local binary patterns with an application to facial expressions", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, Vol 29, No 6, pp 915–928.

103 *artfile Patterns*, 1990. Phaidon Press, Oxford, ISBN 0 7148 2670 7.

104 http://sipi.usc.edu/services/database/Database.html

105 http://whitechapel.media.mit.edu/vismod/imagery/VisionTexture/vistex.html

106 http://www.oulu.fi/research/imag/texture/lbp/lbp.php

Index

Image Processing: Dealing with Texture, Second Edition. Maria Petrou and Sei-ichiro Kamata.

Companion Website: www.wiley.com/go/kamataText2

h

i

j

k

l

m

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.