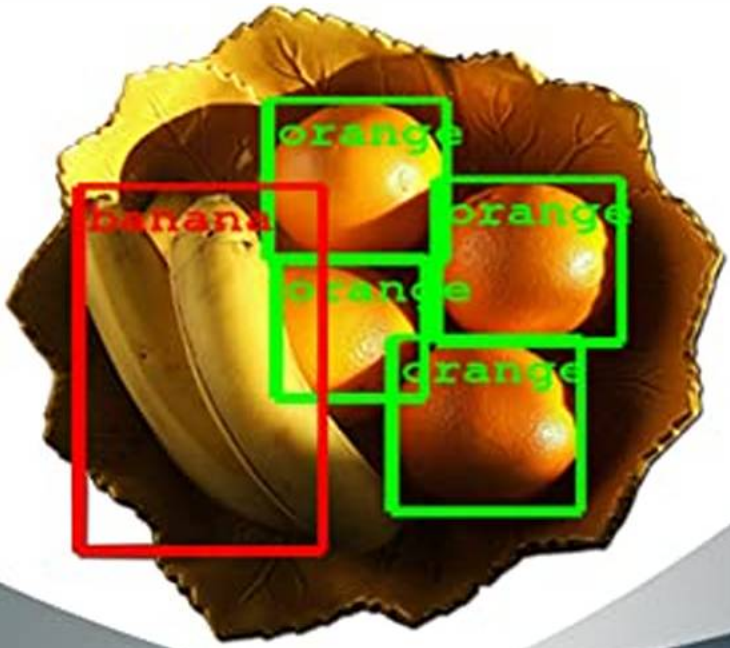
BEST amazon.com SELLER

Python with OpenCV3

- Computer Vision
- Image Processing
- Machine Learning & Object Detection



Richard Stallman

OpenCV3 with Python

Table of Contents

Chapter1: OpenCV3 with Python

5.5 Histogram
5.6 Workshop
5.7 Face Synthesis
5.8 Motion Detecting CCTV

6. Geometric Transform
   6.1 Translate, Scaling, Rotate
   6.2 Warping
   6.3 Lens Distortion
7. Workshop
   - Face Mosaic
   - Liquefy Tool
   - Distortion Camera
8. Filter
   8.1 Convolution Filter
   8.2 Blurring
   8.3 Edge detection
   8.4 Morphology
   8.5 Image pyramids
   8.6 Workshop
   - Mosaic2
   - Cartoonizing
9. Image Segmentation
   9.1 Contour
   9.2 Hough Transform
   9.3 Connected Component
   9.4 Workshop
   - Recognizing Shapes
   - Document Scanner
   - Coin Counter

10. Matching and Tracking
    - 10.1 Matching with a similar picture
    - 10.2 Feature and Key Point
    - 10.3 Descriptor Extractor
    - 10.4 Feature Matching
    - 10.5 Tracking
    - 10.6 Workshop
        - Panorama Picture Maker
        - Book cover searcher
11. Machine Learning and Object Detection
    - 11.1 k-Means
    - 11.2 k-NN
    - 11.3 SVM and HOG
    - 11.4 Cascade Classifier
    - 11.5 Workshop
        - Face Mosaic
        - Hanibal Mask (Snapchat Filter)
        - Face Distortion (Snapchat Filter)

Computer Vision

1. **Computer Vision**
2. Application Field

❖ Field of Computer Vision

- Image Processing
  - Process of getting better quality video using computer
  - Print new video after receiving video
    - Image Enhancement
    - Image Restoration
    - Image Compositing
    - Image Segmentation
- Computer Vision(CV)
  - Get meaningful info. from video
  - Print data from received video
    - Object Detection
    - Object Recognition
    - Object Tracking
- Computer Graphics(CG)
  - Generating video not exiting
  - Print new data from received data

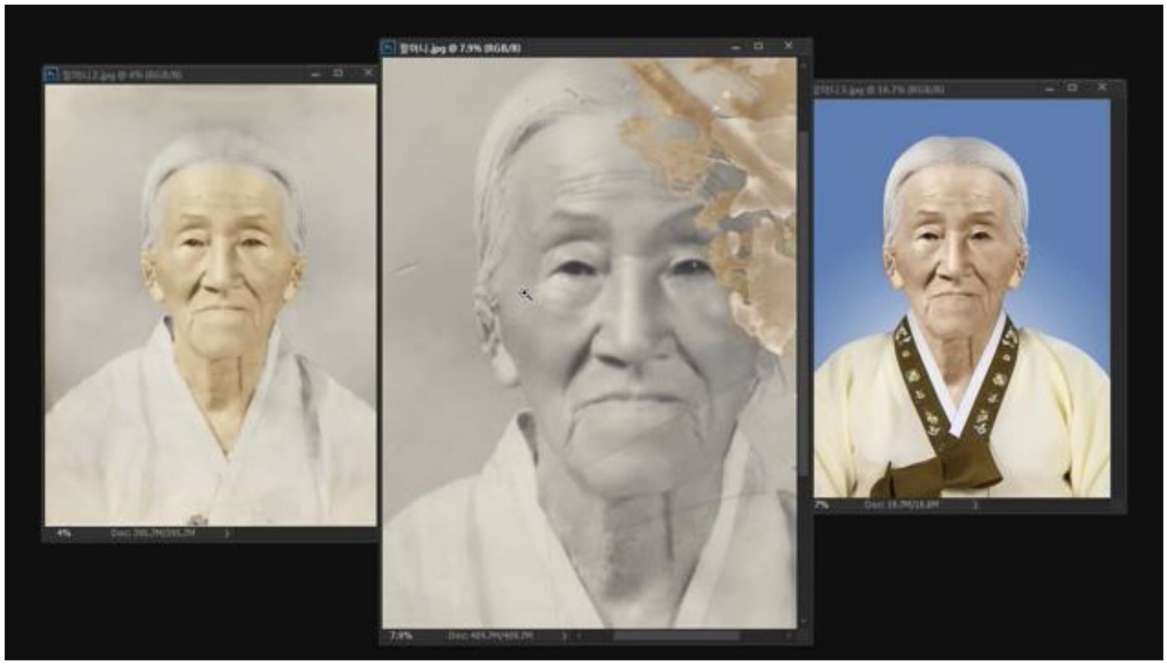
❖ Image Processing Case

- Improve Image

❖ Image Processing Case

- Improve Image

- Restore Image



❖ Image Processing Case

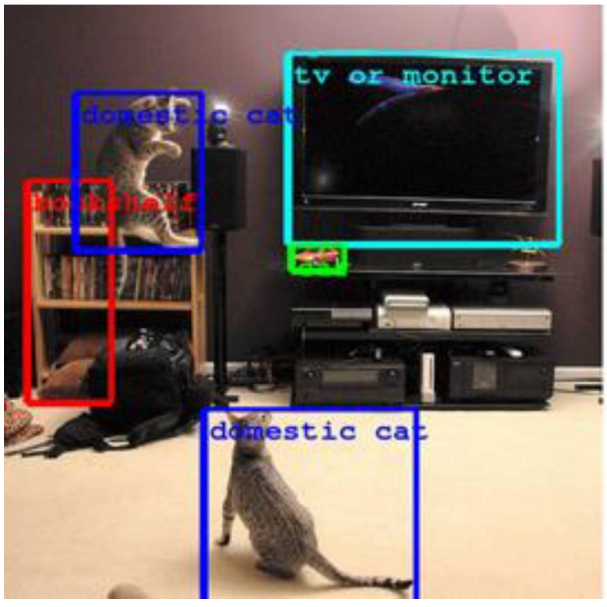
- Compose Images

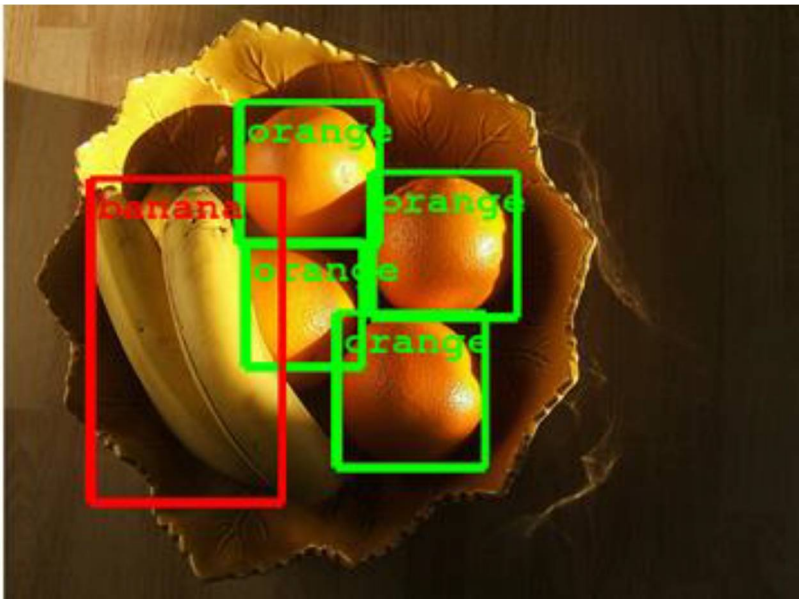
- Divide Image

❖ Computer Vision Case

- Detect Objects

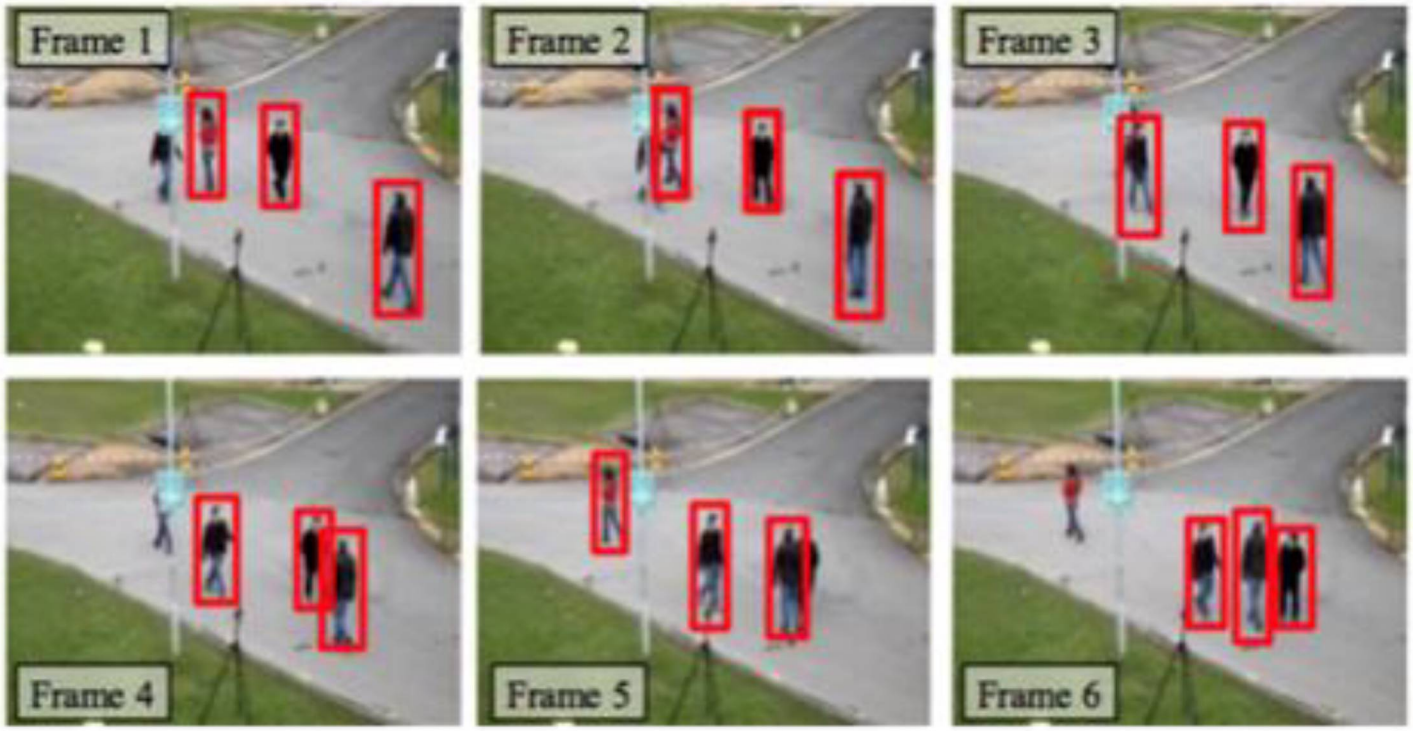
- Recognize Objects





❖ Computer Vision Case

- Trace Objects



Computer Vision

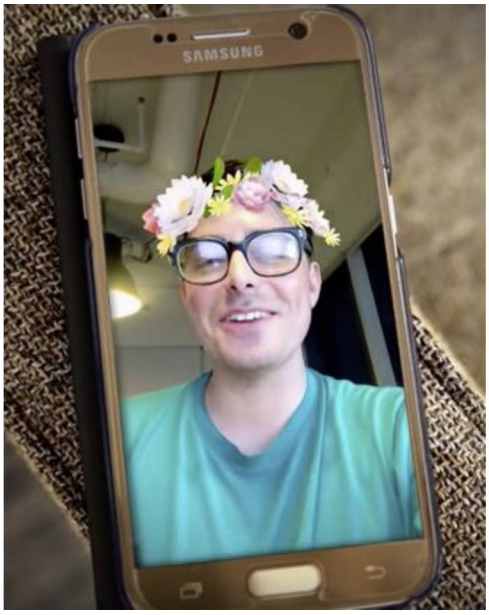
1. Computer Vision
2. **Application Field**

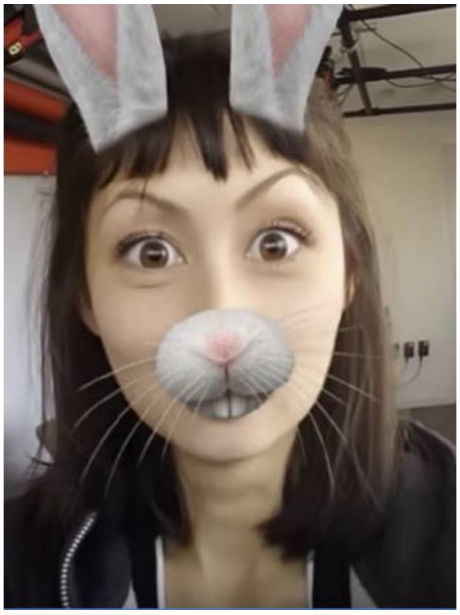
❖ Fields

- Mobile
  - Camera filter, image messenger
- Video, Broadcasting
  - Warping, Morphing, Face Swapping,
  - Chroma key, Caption
- Medical
  - Ultra Sound, X-Ray, CT, MRI Analysis Diagnose
- Sports
  - Posture analysis, Video Reading
- Automobile
  - Around View, Autonomous Cars
- Security, Safety, Traffics
  - Intruder alarm, trace suspect, protect, figure out accident types, control entrance
- Industry
  - Process inspection, defect inspection, sense mall-operation
- Robotics
  - Recognize objects and faces

❖ Mobile

- Snap Chat



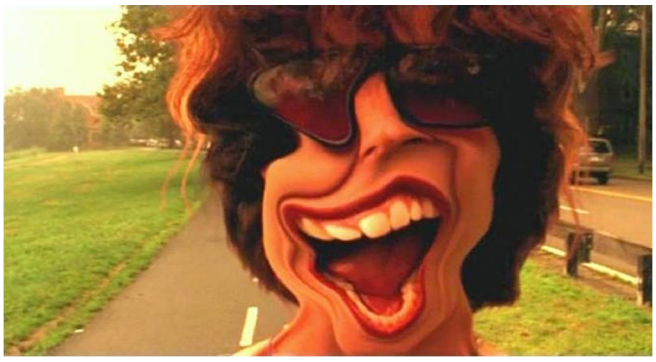


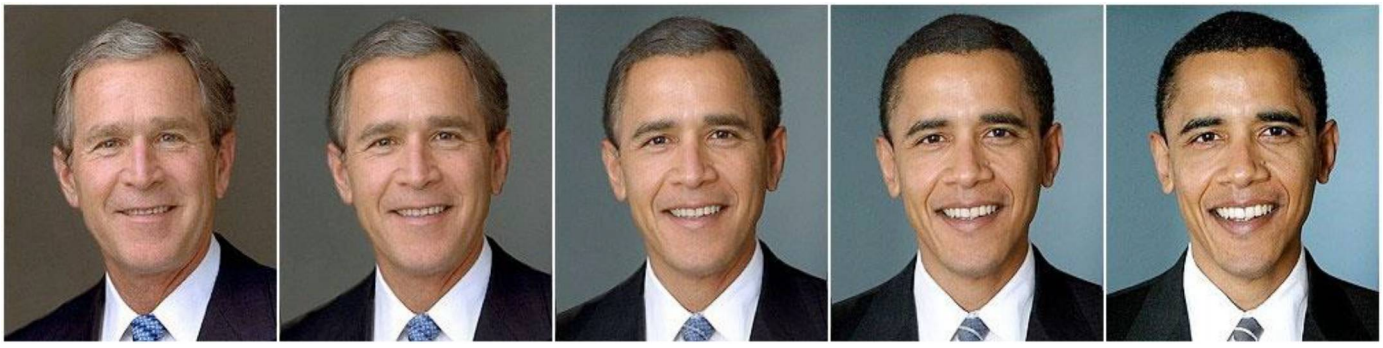
❖ Mobile

- Snow Camera

❖ Movie, Film

- Warping
- Morphing





❖ Movie, Film

- Face Swapping

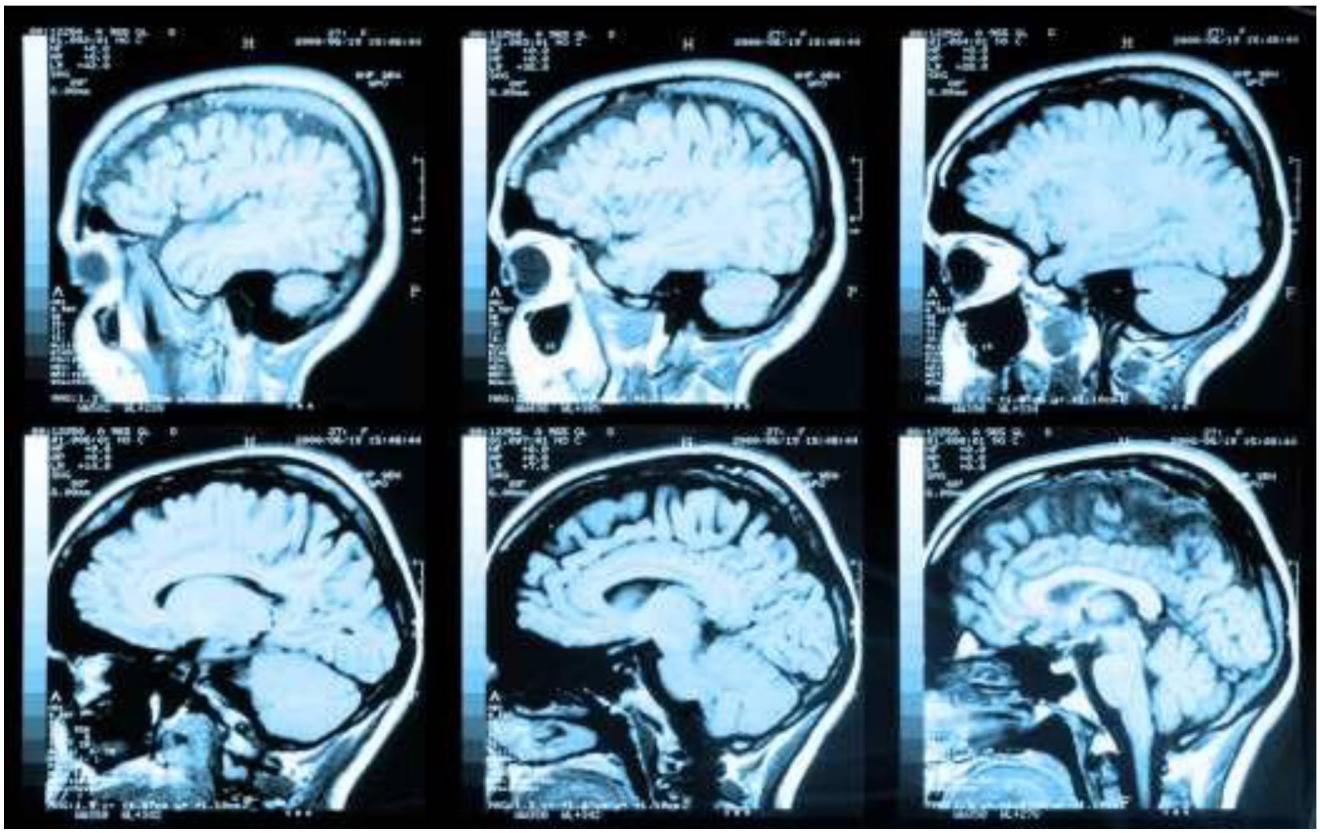
- Chroma key

❖ Movie, Film

- Chroma key

- Subtitles

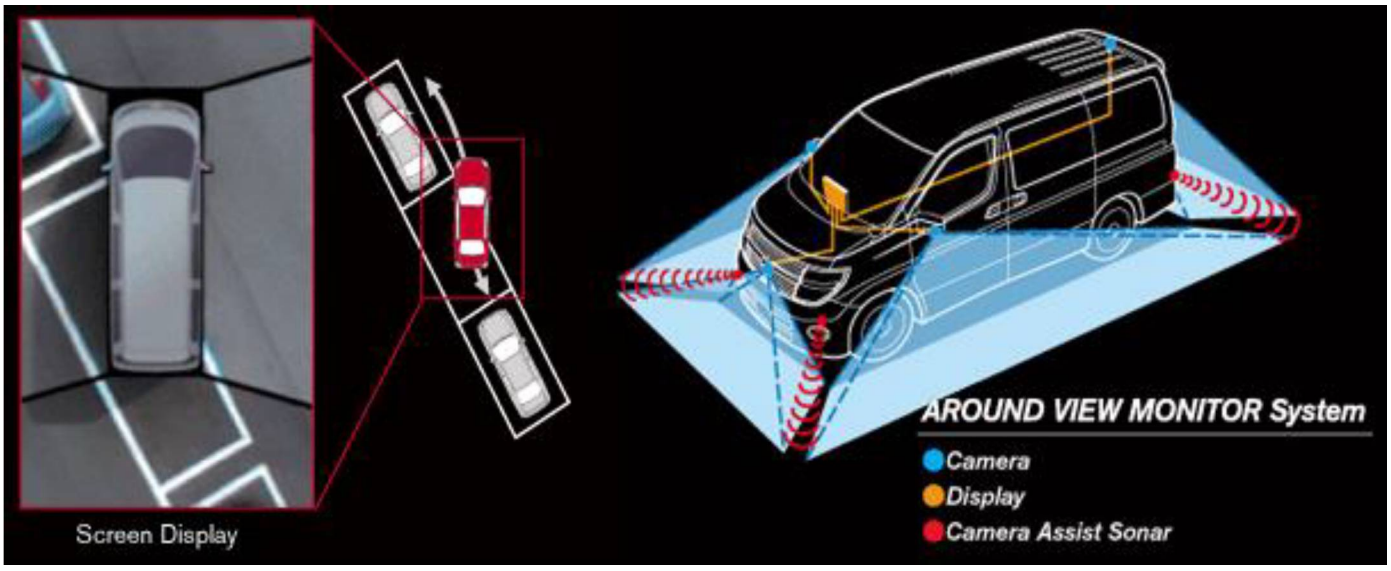
❖ Medical diagnosis



❖ Sports

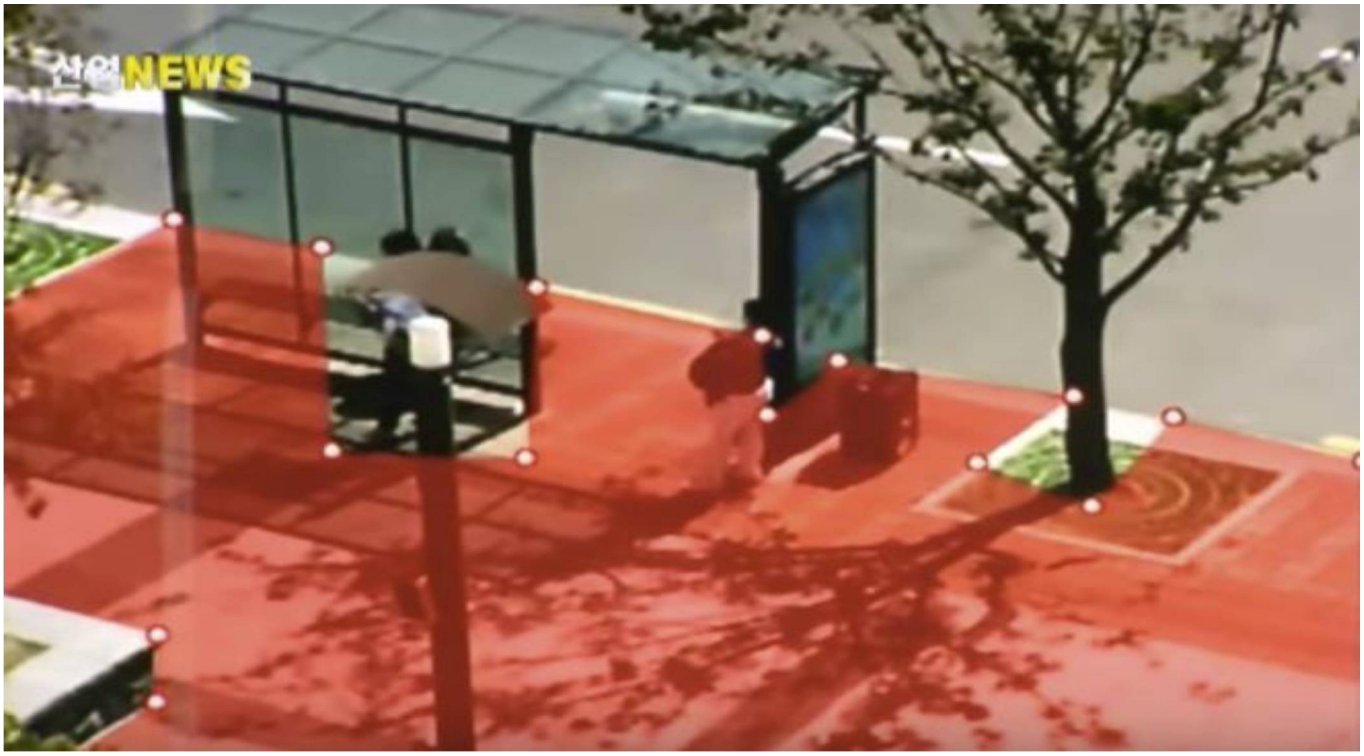
❖ Car

- Around View



- Autonomous Driving Cars

❖ Security, Traffic



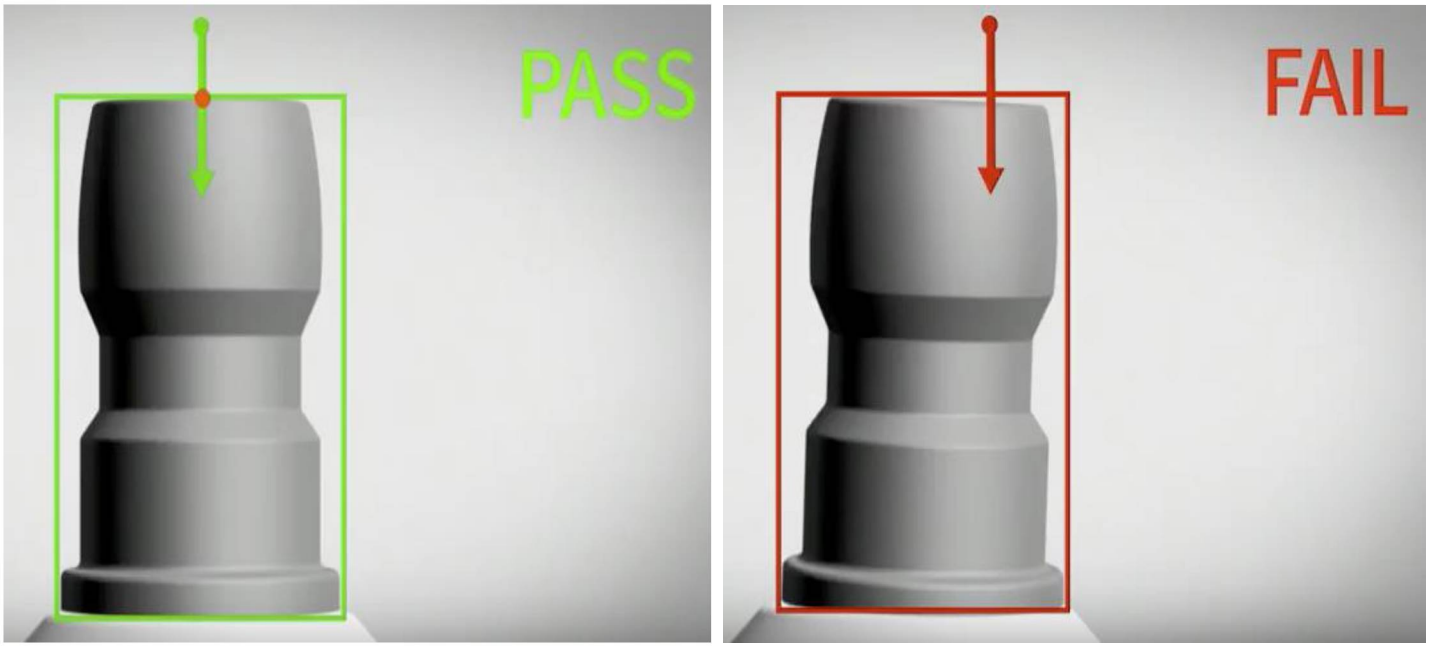
❖ Security, Traffic

❖ Industry

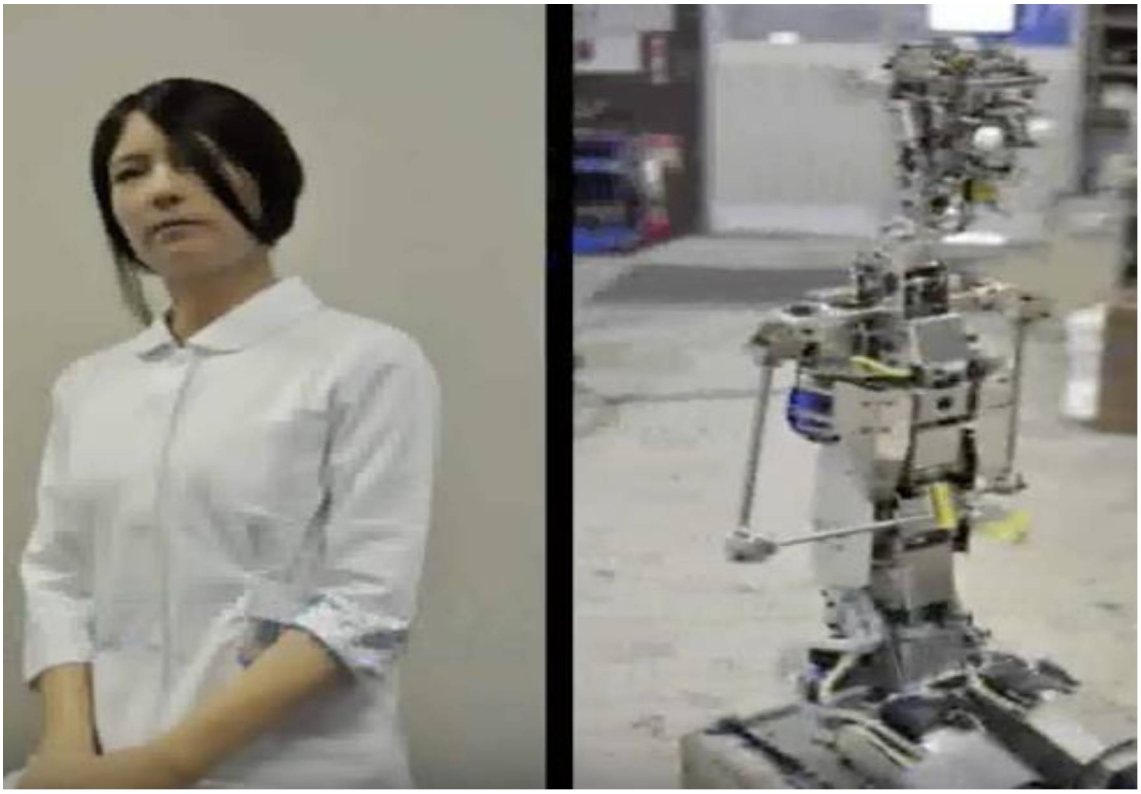
- Process Management

❖ Industry

- Quality, Defect Inspection



❖ Robotics



Opencv with Python

1. **OpenCV3 Intro**
2. Setup on Windows
3. Setup on Mac
4. Setup on Ubuntu
5. Setup on Raspberry-Pi

OpenCV

- http://opencv.org
- Open Source Computer Vision Library
- Official Support
  - Native : C/C++
  - Bindings: Python, Java
  - Support Platform : Windows, Linux, Mac, iOS, Android
- Support Acceleration of OpenCL Activation Hardware
- Intel Russia Team
  - Began by part of CPU intensive application performance enhancing research

History

- Launched : 1999' Jan, IPL(Image Processing Library) based, C language
- First Ver. alpha : 2000'
- 5 Betas : 2001 ~ 2005
- Ver 1.0 : 2006' Oct
- Precompiled Python module included
  - Borland C++ makefiles added
- Ver 2.0 : 2009' Sep
  - new C++ API , STL Based
  - new Python interface
- Ver2.2 :2010' Dec
  - package reorganized (core, imgproc, features2d, ml, contrib, ...)
  - new Python biding, NumPy Required
- Ver 2.3 : 2011' July
  - Android porting
  - Python module cv2 for OpenCV 2.x
- ver 3.4 : 2017' Dec , latest

❖ OpenCV

- Official Web Site : https://opencv.org
- Source Repository
  - Main Repository
    - https://github.com/opencv/opencv
    - Code storage for official distribution
  - Extra Repository
    - https://github.com/opencv/opencv_contrib
    - Include immature or non-popularized details
    - When completion level increases, move to main storage
    - Include SIFT, SURFS and other codes with patent issues
    - BSD License
  - Free
    - Regardless of research or industry
    - Exclude Extra Repository
  - No need for source open obligation
- OpenCV-Python: Support both python 2, 3 (No difference)
  - 2.7, 3.4+

❖ Environment

Platform	Version	HardWare
Windows	Windows 7 or later	WebCam
Mac	OS X Siera or later	WebCam
Ubuntu	16.04 LTS	WebCam
RaspberryPi	RaspberryPi3 B Model Rasbian Stretch	SD card 16GB PiCamera or USB WebCam(UVC)

❖ Pre Required

- Python 3.4+
  - This course Not Supports Python2.7
- NumPy 1.8+
- OpenCV-Python 3.4+, Extra(contrib) included

Opencv with Python

1. OpenCV3 Intro
2. **Setup on Windows**
3. Setup on Mac
4. Setup on Ubuntu
5. Setup on Raspberry-Pi

❖ Setup Python

- Python 3 Only
  - Python 3.4+
- Python 2 Only
  - Python 2.7
- Python 3 + Python 2
  - 설치 순서 중요
  - 1. Setup Python 3
  - 2. Setup Python 2

❖ Setup Python 3

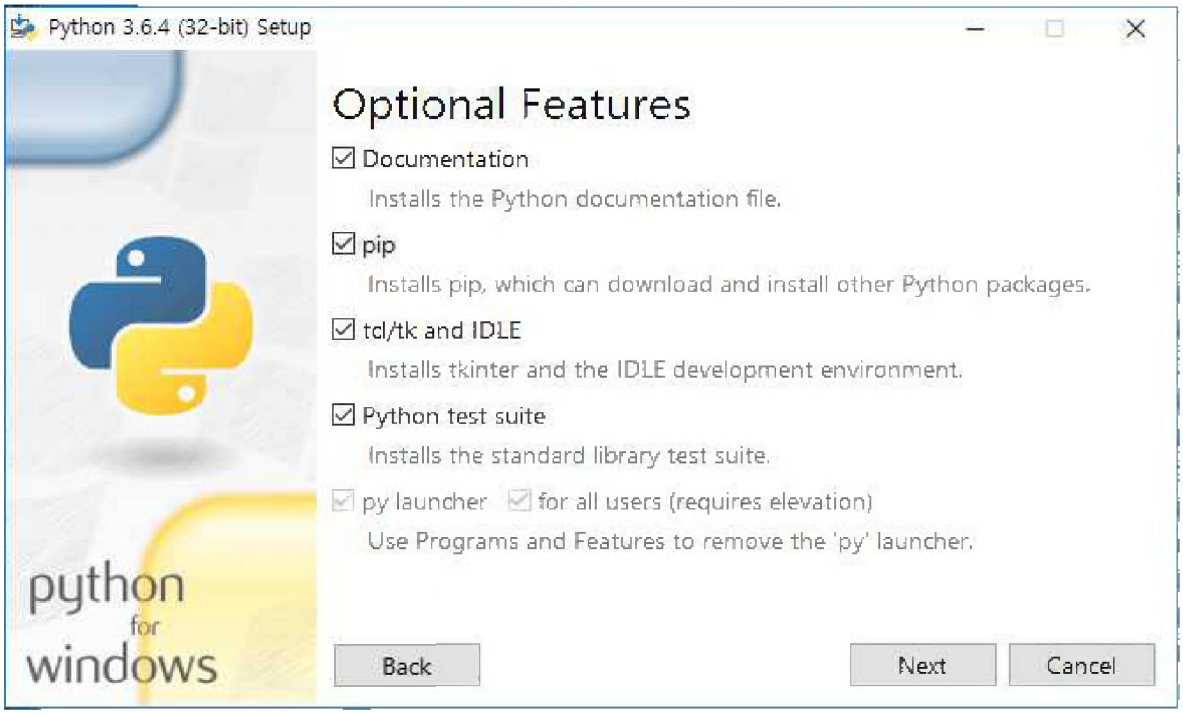
- https://www.python.org/downloads/
- Download Python 3.4+
  - Numpy not support Python 3.3-

❖ Setup Python 3.x

- Start Install wizard
- Check Click "Add Python 3.x to PATH"
- Click "Customize installation"

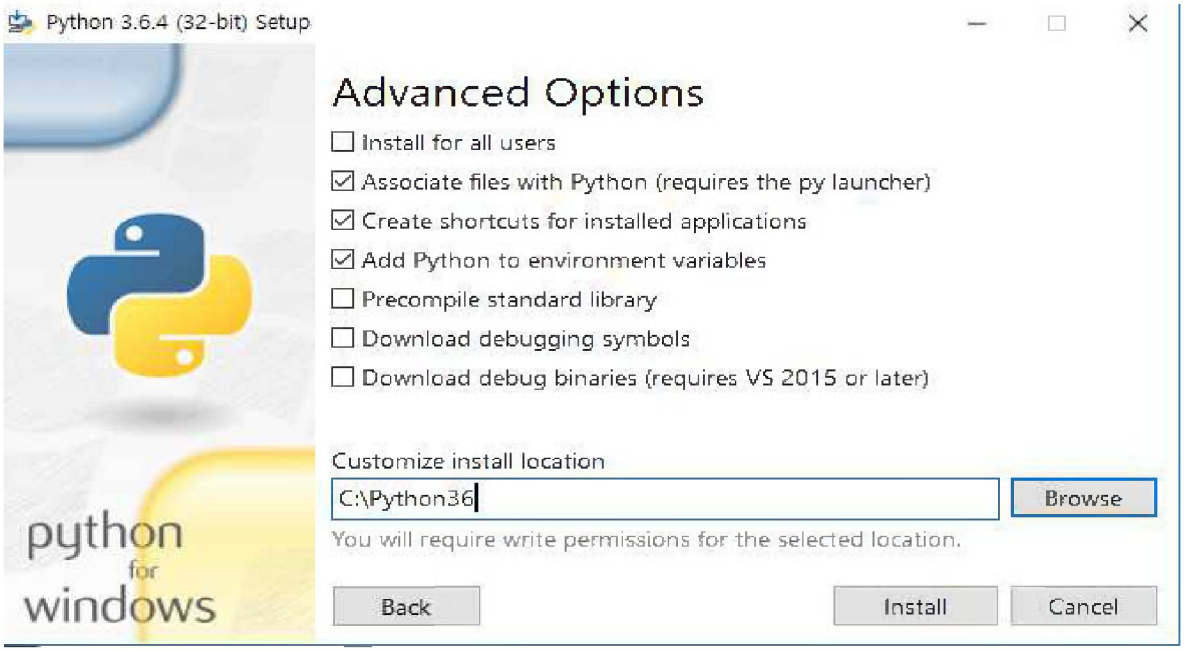
❖ Setup Python 3.x

- Optional Features



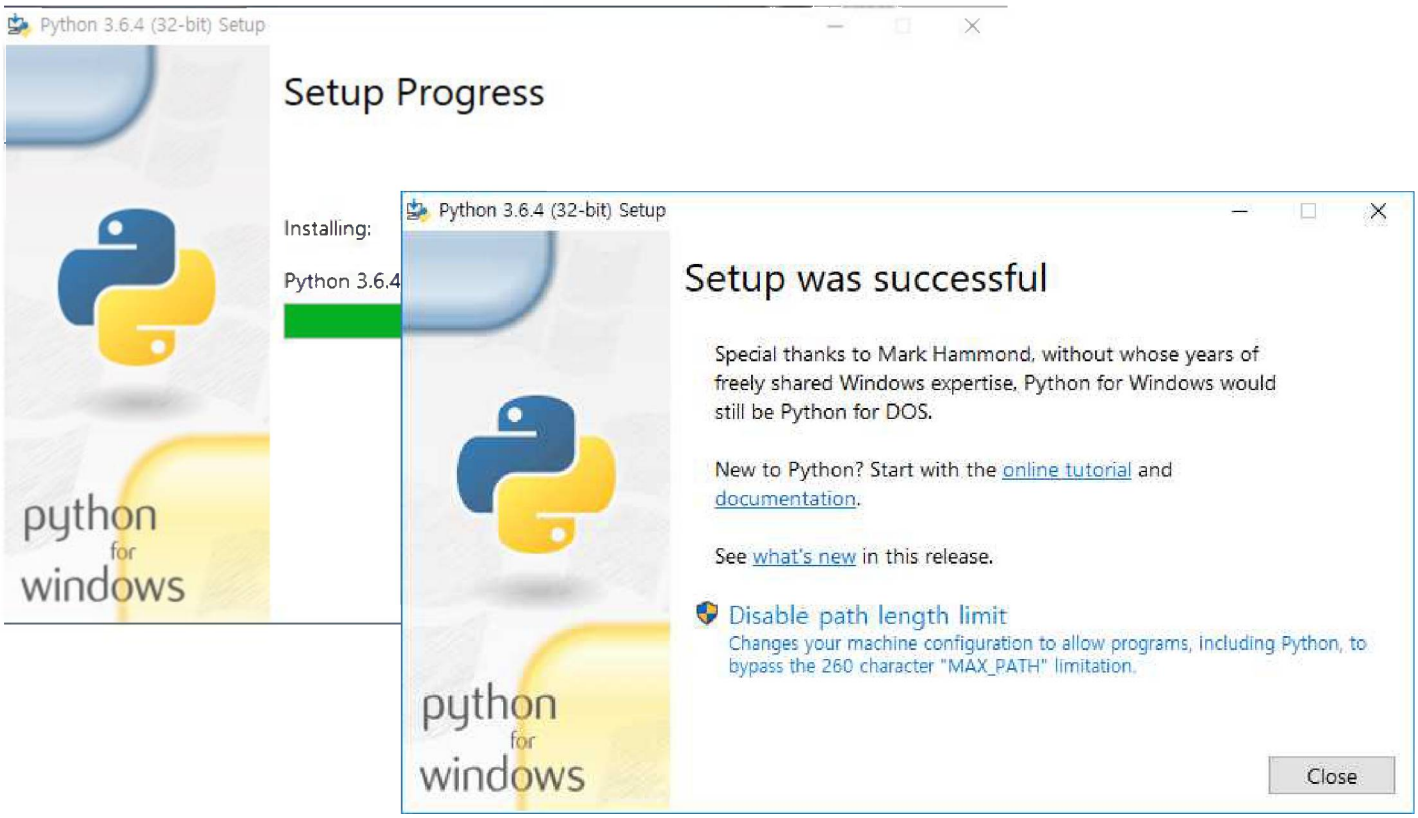
❖ Setup Python 3.x

- Change "Install location"
  - c:\Python36
  - Not necessary



❖ Setup Python 3.x

- Installation Complete

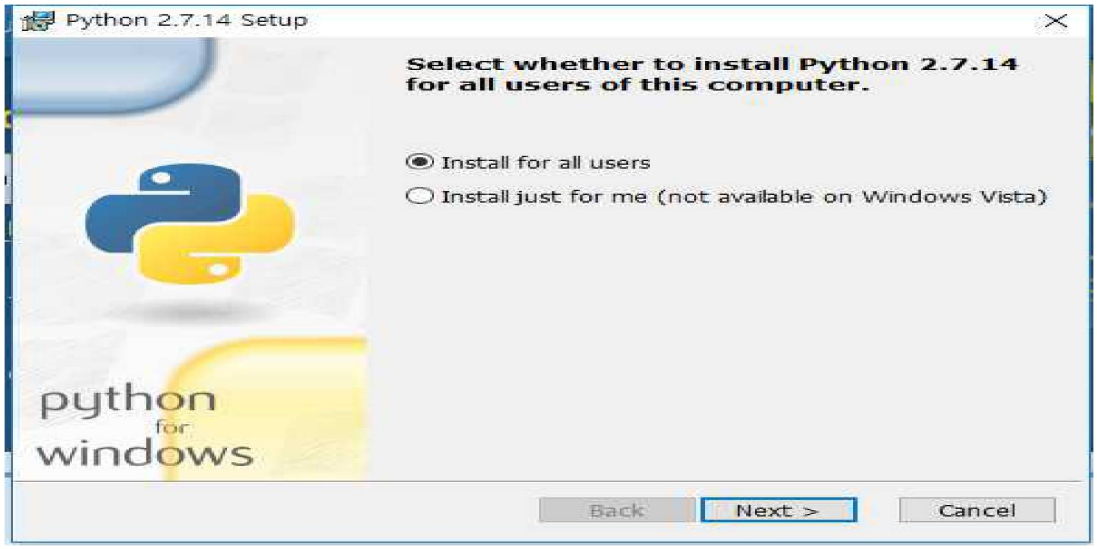


❖ Setup Python 2.7

- https://www.python.org/downloads/
- Download Python 2.7

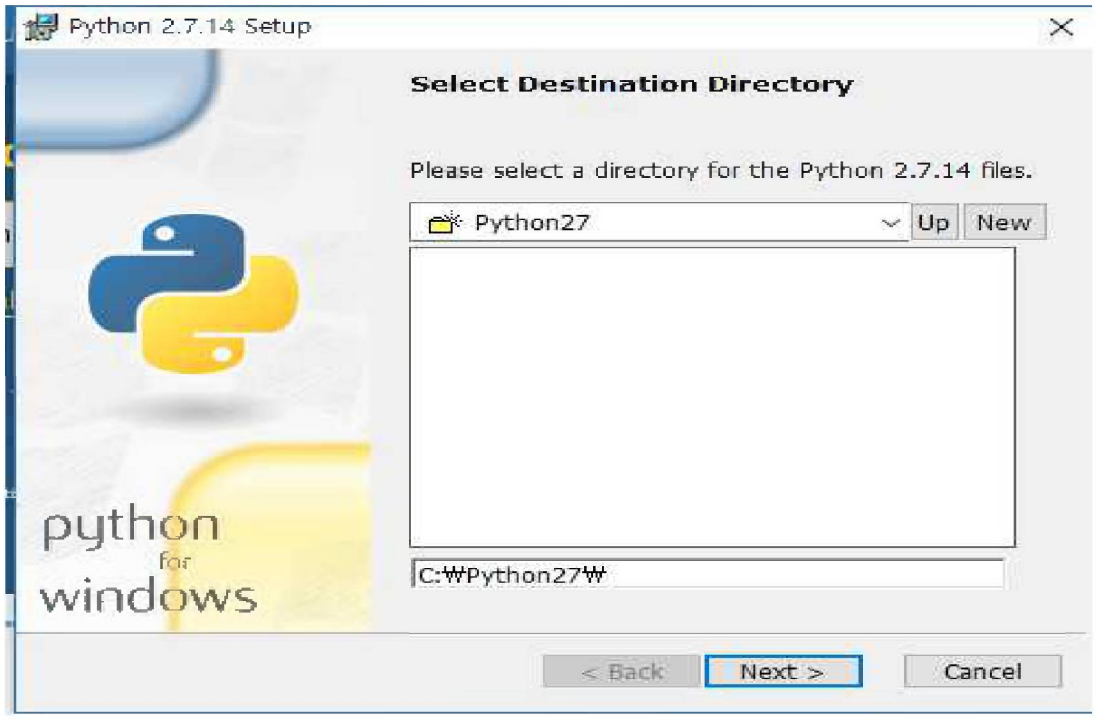
❖ Setup Python 2.7

- Star "Install wizard"
- Select users to install
  - "Install for all users"



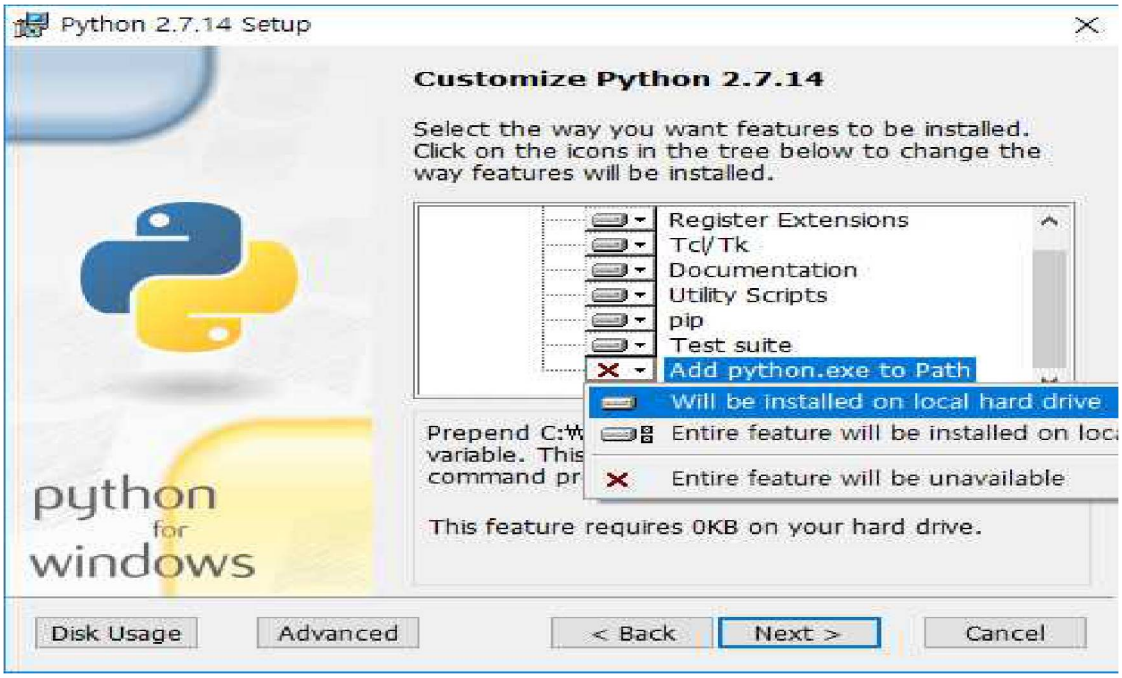
❖ Setup Python 2.7

- Select Destination Directory
  - Default Path



❖ Setup Python 2.7

- Enable "Add python.exe to path" Option

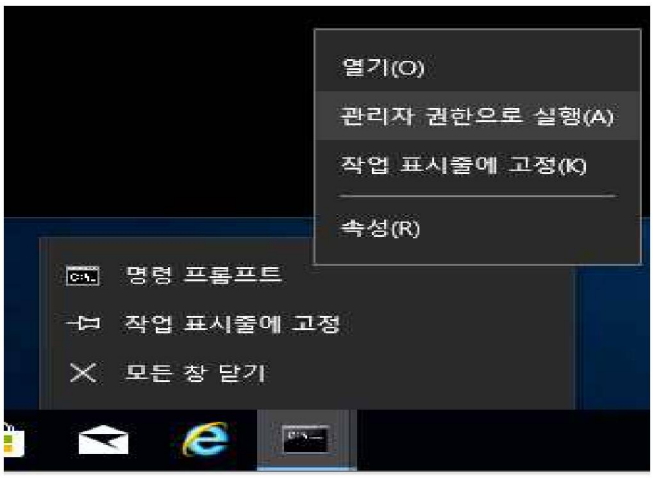


❖ Setup Python 2.7

- Installation Complete

❖ Python Command Setup

- Make Link
  - Open Window Command Line as Administator
  - `mklink c:\windows\python2.exe c:\python27\python.exe`
  - `mklink c:\windows\python3.exe c:\python36\python.exe`
- Check Python Version
  - Python 2
    - python --version
    - python2 --version
  - Python 3
    - python3 --version
- Check Pip Version
  - Python 2
    - pip --version
    - pip2 --version
  - Python 3
    - pip3 –version



❖ Python Command Setup

```
관리자: 명령 프롬프트
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:₩Windows₩system32>mklink c:₩windows₩python2.exe c:₩python27₩python.exe
c:₩windows₩python2.exe <<===>> c:₩python27₩python.exe에 대한 기호화된 링크를 만들었습니다.

C:₩Windows₩system32>mklink c:₩windows₩python3.exe c:₩python36₩python.exe
c:₩windows₩python3.exe <<===>> c:₩python36₩python.exe에 대한 기호화된 링크를 만들었습니다.

C:₩Windows₩system32>python --version
Python 2.7.14

C:₩Windows₩system32>python3 --version
Python 3.6.4

C:₩Windows₩system32>pip --version
pip 9.0.1 from c:₩python27₩lib₩site-packages (python 2.7)

C:₩Windows₩system32>pip3 --version
pip 9.0.1 from c:₩python36₩lib₩site-packages (python 3.6)
```

❖ Setup Dependent Library

- Numpy
  - Installation
    - pip install numpy
    - pip3 install numpy
  - Check in Python

```
import numpy as np
np.__version__
```

```
C:₩Users₩sw>pip install numpy
Collecting numpy
  Downloading numpy-1.14.0-cp27-none-win32.whl (9.8MB)
    100% |################################| 9.8MB 75kB/s
Installing collected packages: numpy
Successfully installed numpy-1.14.0
```

```
c:₩>pip3 install numpy
Collecting numpy
  Downloading numpy-1.14.0-cp36-none-win32.whl (9.8MB)
    100% |                                | 9.8MB 88kB/s
Installing collected packages: numpy
Successfully installed numpy-1.14.0
```

```
>>> import numpy as np
>>> np.__version__
'1.14.0'
```

```
>>> import numpy
>>> numpy.__version__
'1.14.0'
```

❖ Setup Dependent Library

- Matplotlib
  - Installation
    - pip install matplotlib
    - pip3 install matplotlib

```
C:₩Users₩sw>pip3 install matplotlib
Collecting matplotlib
  Downloading matplotlib-2.1.2-cp36-cp36m-win32.whl (8.5MB)
    100% |                                | 8.5MB 59kB/s
Collecting pytz (from matplotlib)
  Using cached pytz-2017.3-py2.py3-none-any.whl
```

```
C:₩Users₩sw>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-2.1.2-cp27-cp27m-win32.whl (8.2MB)
    100% |################################| 8.3MB 53kB/s
Requirement already satisfied: numpy>=1.7.1 in c:₩python27₩lib₩site-packages
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 (from matplotlib)
  Downloading pyparsing-2.2.0-py2.py3-none-any.whl (56kB)
    100% |################################| 61kB 190kB/s
Collecting backports.functools-lru-cache (from matplotlib)
```

  - Check in Python

```
import matplotlib
matplotlib.__version__
```

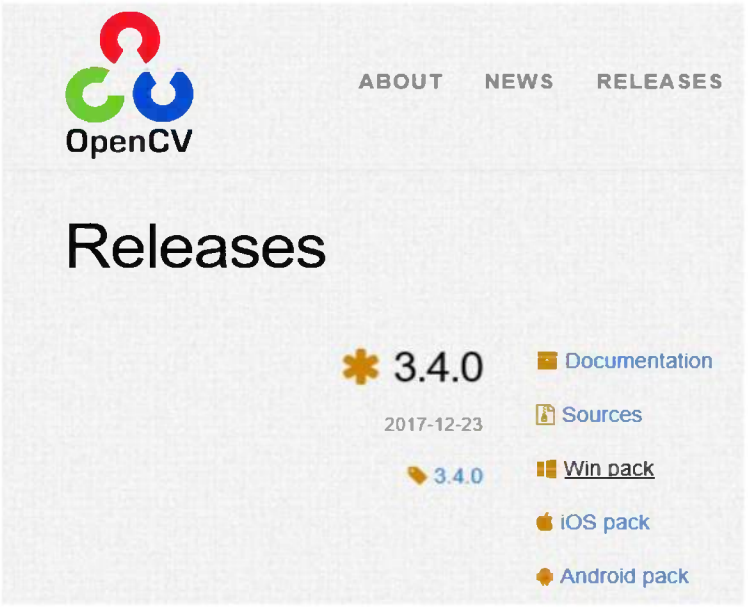
```
>>> import matplotlib
>>> matplotlib.__version__
'2.1.2'
```

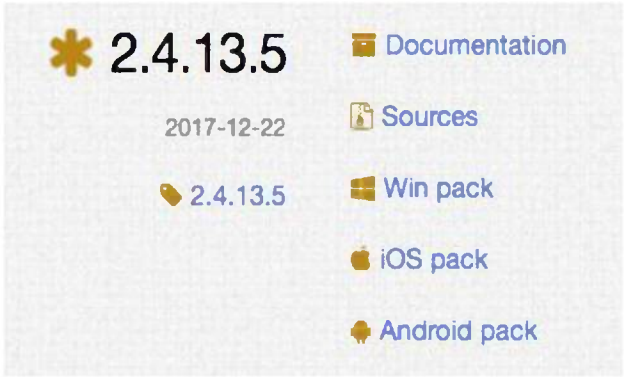
❖ Setup OpenCV-Python

- 3가지 방법
  - OpenCV Pre-Built Library Manually
    - Windows Only
    - Python 2.7 Only
    - Easy to select OpenCV version
      - OpenCV 2.4, 3.4
    - Not included contrib packages
  - PyPi Pre-Built Using pip
    - Almost platform support
    - Python 2, 3 Supported
    - OpenCV 3.x Only
    - contrib package provided separately
  - Source code build
    - VisualStudio / MinGW
    - CMake
    - https://docs.opencv.org/master/d3/d52/tutorial_windows_install.html

❖ OpenCV Pre-Built Library

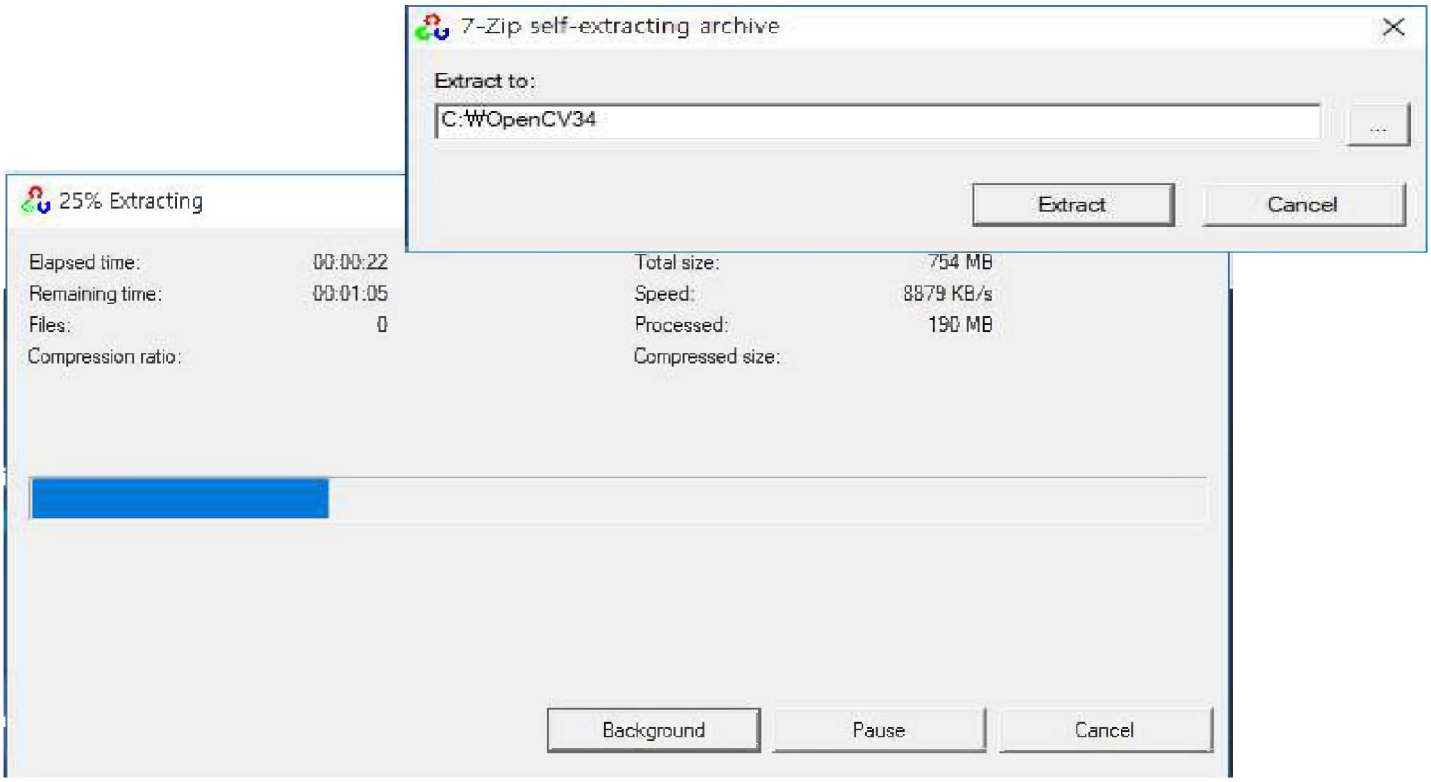
- Download Windows Pack
  - https://opencv.org/releases.html
    - 3.4 , 2.4





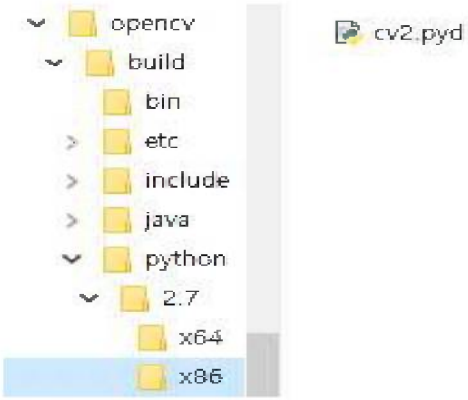
❖ OpenCV Pre-Built Library

- Self-Extracting Archive



❖ OpenCV Pre-Built Library

- Copy library to site-packages
  - From
    - opencv/build/bin/python/2.7/x64/cv2.pyd
    - opencv/build/bin/python/2.7/x86/cv2.pyd
  - To
    - c:\Python27\Lib/site-packages



❖ OpenCV Pre-Built Library

- Check out Installed OpenCV Version in Python 2

```
import cv2
cv2.__version__
```

```
>>> import cv2
>>> cv2.__version__
'3.4.0'
```

```
>>> import cv2
>>> cv2.__version__
'2.4.13.5'
```

- Not support in Python 3

```
C:₩Users₩sw>python3
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Inte
!)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: Module use of python27.dll conflicts with this version of Python
```

❖ OpenCV-Python Installation - PyPi

- Unofficial OpenCV packages for Python
- OpenCV-Python(Latest Version)
  - https://pypi.python.org/pypi/opencv-python
  - Core modules without Extra(contrib) modules
  - Install OpenCV-Python
    - pip install opencv-python
    - pip3 install opencv-python

```
C:₩Users₩sw>pip install opencv-python
Collecting opencv-python
  Downloading opencv_python-3.4.0.12-cp27-cp27m-win32.whl (22.8MB)
    100% |################################| 22.8MB 32kB/s
Requirement already satisfied: numpy>=1.11.1 in c:₩python27₩lib₩site-packages
 (from opencv-python)
Installing collected packages: opencv-python
Successfully installed opencv-python-3.4.0.12
```

```
C:₩Users₩sw>pip3 install opencv-python
Collecting opencv-python
  Downloading opencv_python-3.4.0.12-cp36-cp36m-win32.whl (22.8MB)
    100% |                                | 22.8MB 31kB/s
Requirement already satisfied: numpy>=1.11.3 in c:₩python36₩lib₩site-packages
 (from opencv-python)
Installing collected packages: opencv-python
Successfully installed opencv-python-3.4.0.12
```

❖ OpenCV-Python Installation - PyPi

- Unofficial OpenCV packages for Python
- OpenCV-Contrib-Python(Latest Version)
  - https://pypi.python.org/pypi/opencv-contrib-python
  - contains contrib modules
  - Install OpenCV-Contrib-Python
    - pip install opencv-contrib-python
    - pip3 install opencv-contrib-python

```
C:₩Users₩sw>pip install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-3.4.0.12-cp27-cp27m-win32.whl (27.5MB)
    100% |################################| 27.5MB 24kB/s
Requirement already satisfied: numpy>=1.11.1 in c:₩python27₩lib₩site-packages
 (from opencv-contrib-python)
Installing collected packages: opencv-contrib-python
Successfully installed opencv-contrib-python-3.4.0.12
```

```
C:₩Users₩sw>pip3 install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-3.4.0.12-cp36-cp36m-win32.whl (27.5MB)
    100% |                                | 27.5MB 13kB/s
Requirement already satisfied: numpy>=1.11.3 in c:₩python36₩lib₩site-packages
 (from opencv-contrib-python)
Installing collected packages: opencv-contrib-python
Successfully installed opencv-contrib-python-3.4.0.12
```

❖ OpenCV-Python Installation - PyPi

- Version Specific Installation
  - List up OpenCV Version (3.x only)
    - pip install opencv-python==?
  - Install specific version (example)
    - pip install opencv-python=3.1.x.x

```
C:₩Users₩sw>pip install opencv-python==?
Collecting opencv-python==?
  Could not find a version that satisfies the requirement opencv-python==? (f
rom versions: 3.1.0.0, 3.1.0.1, 3.1.0.2, 3.1.0.3, 3.1.0.5, 3.2.0.6, 3.2.0.7,
3.2.0.8, 3.3.0.8, 3.3.0.10, 3.3.1.11, 3.4.0.12)
No matching distribution found for opencv-python==?
```

❖ OpenCV-Python Installation - PyPi

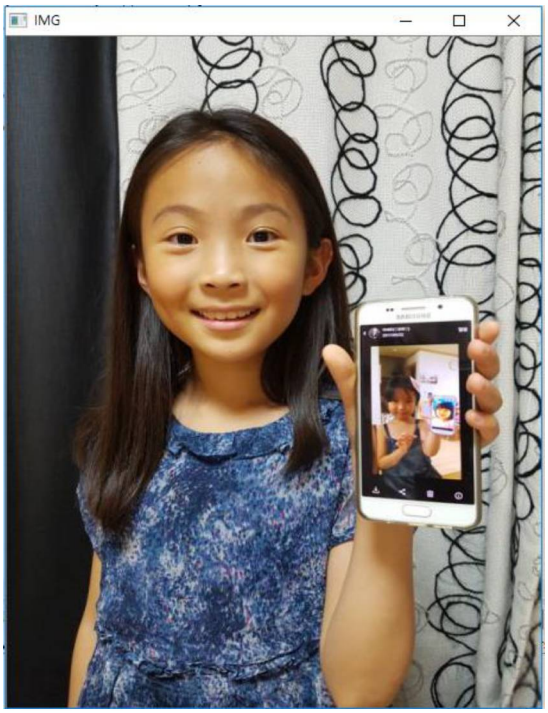
- Check out Installed Version
  - import cv2
  - cv2.__version__

```
C:₩Users₩sw>python
Python 2.7.14 (v2.7.14:84471935ed
(Intel)] on win32
Type "help", "copyright", "credit
>>> import cv2
>>> cv2.__version__
'3.4.0'
```

```
C:₩Users₩sw>python3
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2
1)] on win32
Type "help", "copyright", "credits" or
>>> import cv2
>>> cv2.__version__
'3.4.0'
>>>
```

- Test Code

```
import cv2

img_file = "../img/model.jpg"
img = cv2.imread(img_file)

if not img is None:
     cv2.imshow('IMG', img)
     while True:
          key = cv2.waitKey(0) #& 0xFF
          if key == 27:
               cv2.destroyAllWindows()
               break
else:
print('No image file.')
```



Opencv with Python

1. OpenCV3 Intro
2. Setup on Windows
3. **Setup on Mac**
4. Setup on Ubuntu
5. Setup on Raspberry-Pi

❖ Setup Python

- Pre-installed Python not support pip
- Python 3 Only
  - Python 3.4+
- Python 2 Only
  - Python 2.7
- Python 3 + Python 2
  - Installation order does not matter
  - 1. Setup Python 3
  - 2. Setup Python 2

❖ Setup Python 3

- https://www.python.org/downloads/
- Download Python 3.4+
  - Numpy not support Python 3.3

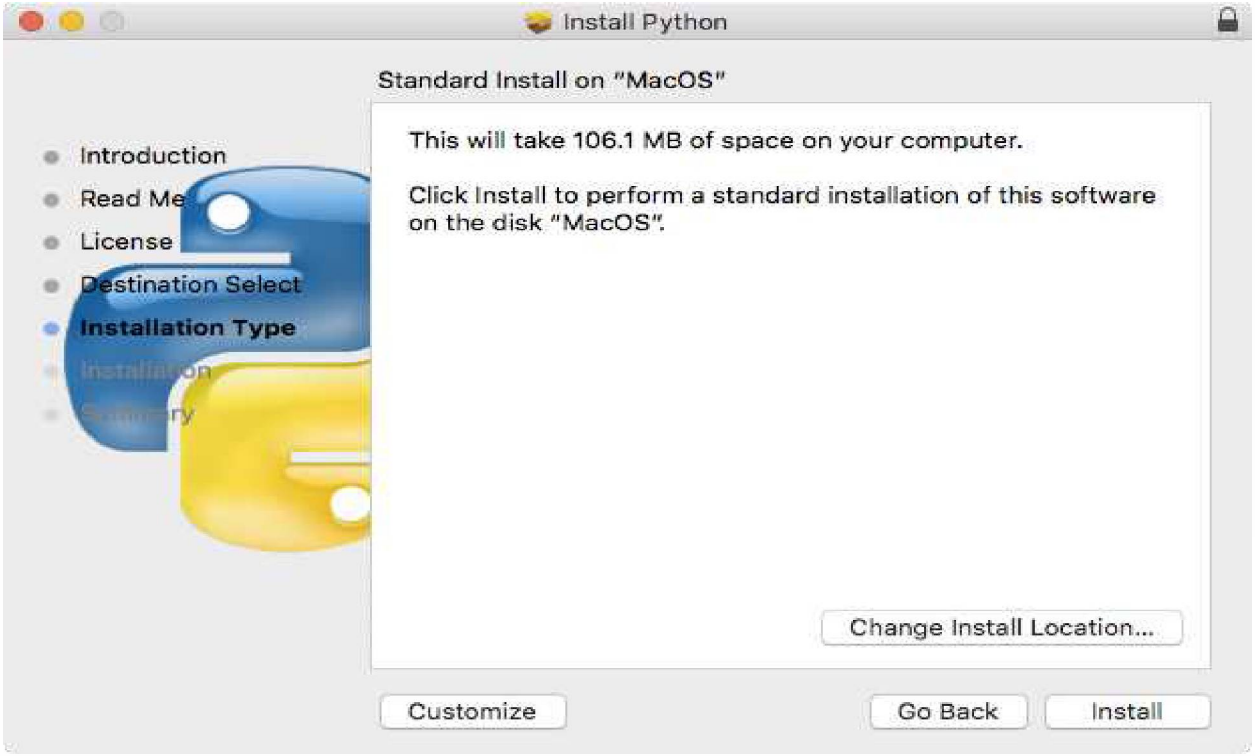
❖ Setup Python 3.x

- Start Install wizard

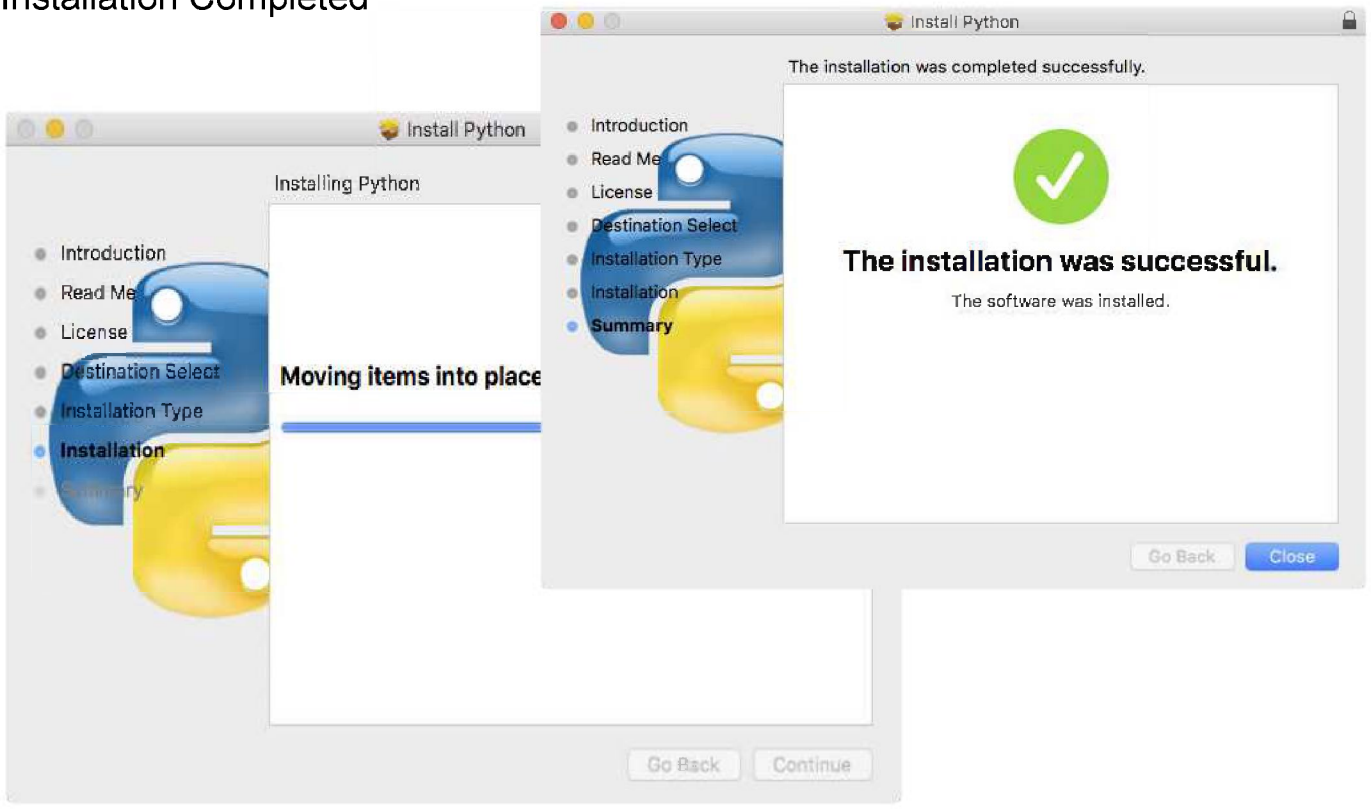
- License Agreement

❖ Setup Python 3.x

- Standard Install



- Installation Completed



❖ Check up Python 3

- python3 --version

❖ Check Pip Version

- pip3 --version

```
rainer@swMBA:~$ python3 --version
Python 3.6.4
rainer@swMBA:~$ pip3 --version
pip 9.0.1 from /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/s
ite-packages (python 3.6)
```

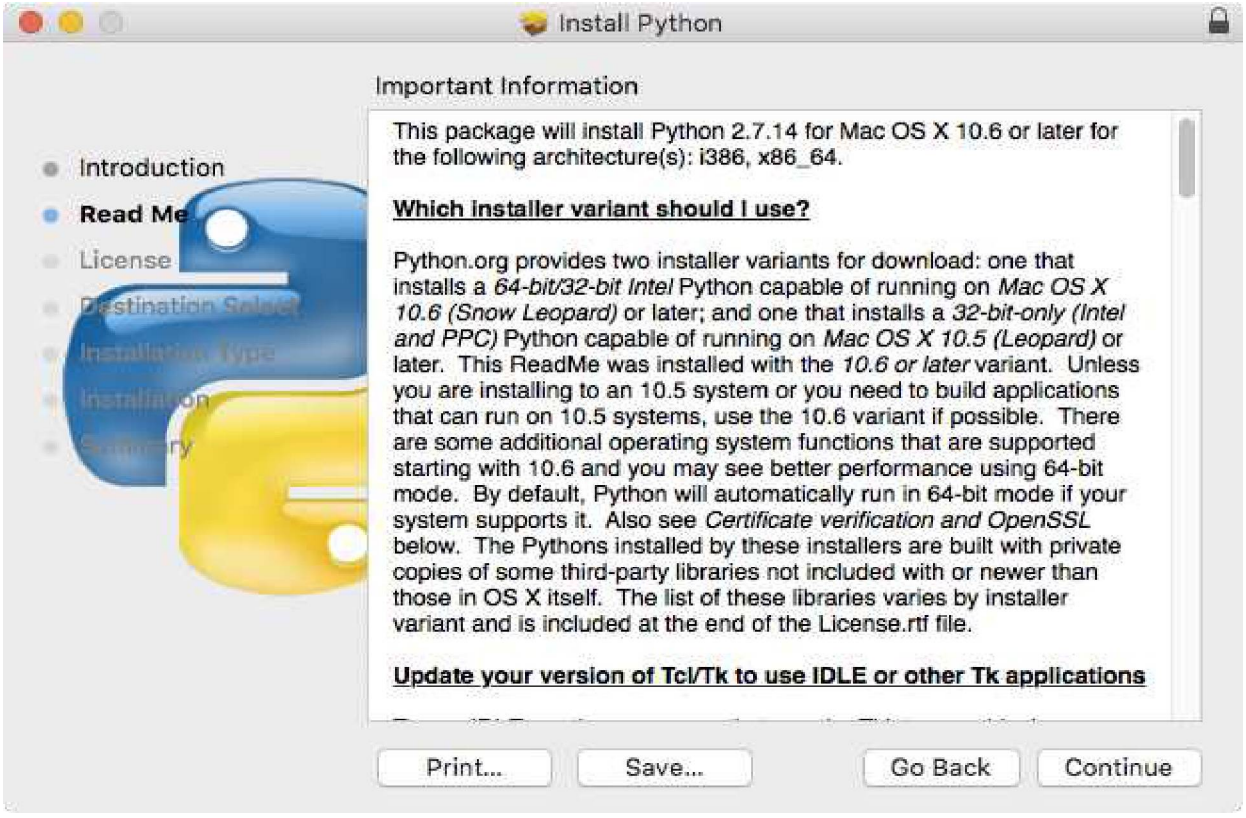
❖ Setup Python 2.7

- https://www.python.org/downloads/
- Download Python 2.7

❖ Setup Python 2.7

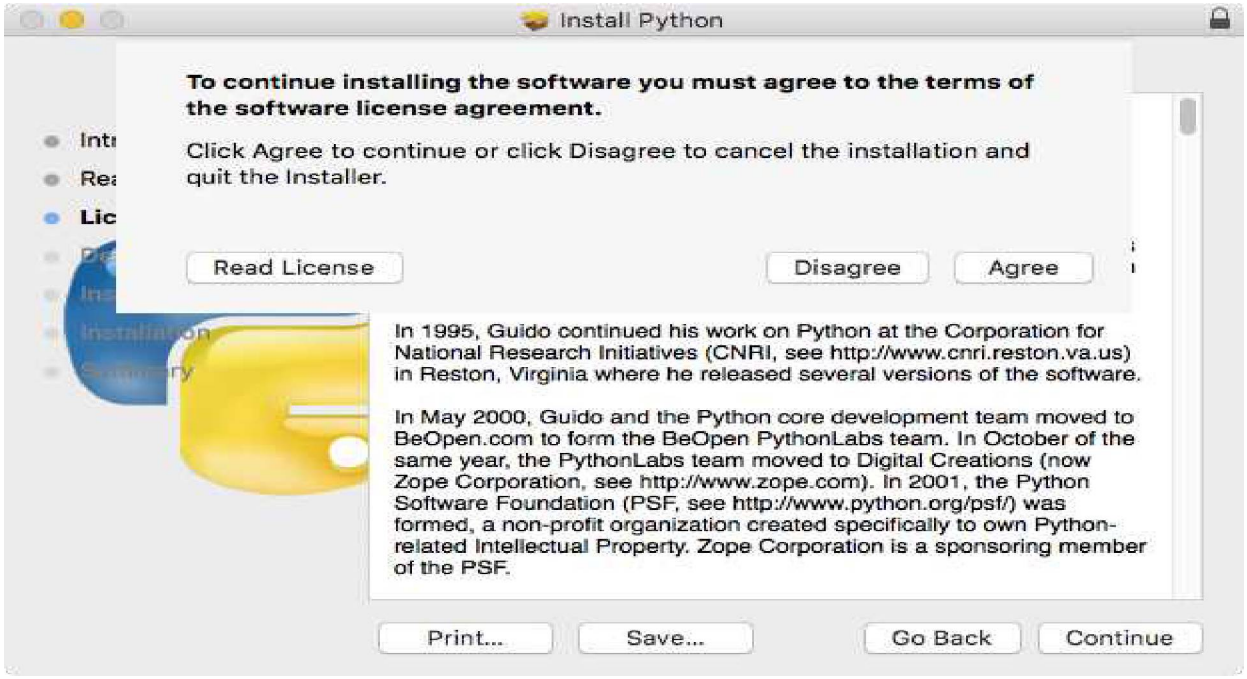
- Star "Install wizard"

- Important Information



❖ Setup Python 2.7

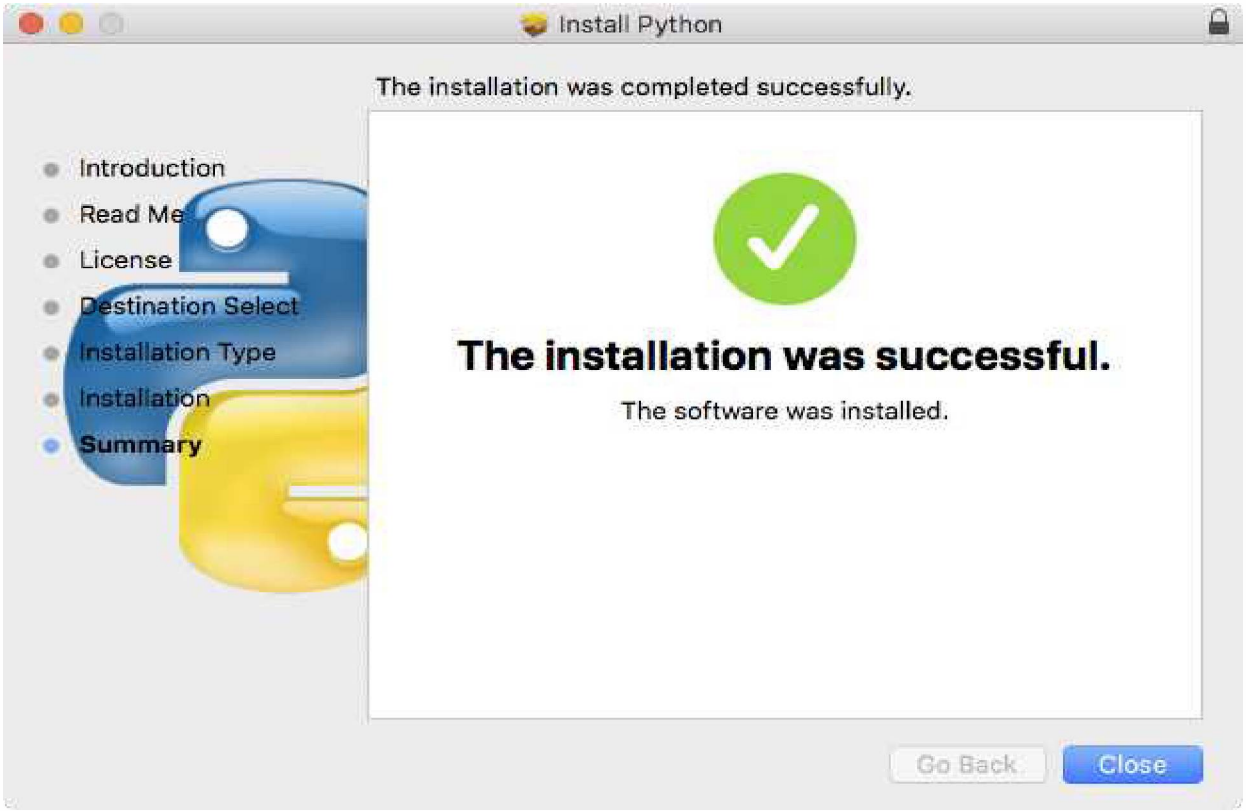
- License Agreement



- Standard Install

❖ Setup Python 2.7

- Installation Completed



❖ Check up Python 2

- python --version

❖ Check Pip Version

- pip --version

```
rainer@swMBA:~$ python --version
Python 2.7.14
rainer@swMBA:~$ pip --version
pip 9.0.1 from /Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/s
ite-packages (python 2.7)
```

❖ Setup Dependent Library

- Numpy Installation
  - pip install numpy
  - pip3 install numpy

```
rainer@swMBA:~$ pip install numpy
Collecting numpy
  Downloading numpy-1.14.0-cp27-cp27m-macosx_10_6_intel.macosx_10_9_intel.macosx
_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (4.7MB)
    100% |████████████████████████████████| 4.7MB 233kB/s
Installing collected packages: numpy
Successfully installed numpy-1.14.0
```

```
rainer@swMBA:~$ pip3 install numpy
Collecting numpy
  Downloading numpy-1.14.0-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx
_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (4.7MB)
    100% |████████████████████████████████| 4.7MB 68kB/s
Installing collected packages: numpy
Successfully installed numpy-1.14.0
```

- Numpy Check in Python

```
import numpy as np
np.__version__
```

```
rainer@swMBA:~$ python
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 12:01:12)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__version__
'1.14.0'
```

```
rainer@swMBA:~$ python3
Python 3.6.4 (v3.6.4:d48eceba05, Dec 18 2017, 21:07:28)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__version__
'1.14.0'
```

❖ Setup Dependent Library

- Matplotlib Installation
  - pip install matplotlib
  - pip3 install matplotlib

```
rainer@swMBA:~$ pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-2.1.2-cp27-cp27m-macosx_10_6_intel.macosx_10_9_intel.ma
cosx_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (13.2MB)
    100% |████████████████████████████████| 13.2MB 60kB/s
Requirement already satisfied: numpy>=1.7.1 in /Library/Frameworks/Python.framew
```

```
rainer@swMBA:~$ pip3 install matplotlib
Collecting matplotlib
  Downloading matplotlib-2.1.2-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.ma
cosx_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (13.2MB)
    100% |████████████████████████████████| 13.2MB 95kB/s
Requirement already satisfied: pytz in /Library/Frameworks/Python.framework/Vers
```

- Matplotlib Check in Python
  - import matplotlib
  - matplotlib.__version__

```
rainer@swMBA:~$ python
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 12:01:12)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib
>>> matplotlib.__version__
'2.1.2'
```

```
rainer@swMBA:~$ python3
Python 3.6.4 (v3.6.4:d48ecebad5, Dec 18 2017, 21:07:28)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib
>>> matplotlib.__version__
'2.1.2'
```

❖ OpenCV-Python Installation - PyPi

- Unofficial OpenCV packages for Python
- OpenCV-Python(Latest Version)
  - https://pypi.python.org/pypi/opencv-python
  - Core modules without contrib modules
  - Install on OpenCV-Python
    - pip install opencv-python
    - pip3 install opencv-python

```
rainer@swMBA:~$ pip install opencv-python
Collecting opencv-python
  Downloading opencv_python-3.4.0.12-cp27-cp27m-macosx_10_6_intel.macosx_10_9_in
tel.macosx_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (41.2MB)
    100% |████████████████████████████████| 41.2MB 25kB/s
Requirement already satisfied: numpy>=1.11.1 in /Library/Frameworks/Python.frame
work/Versions/2.7/lib/python2.7/site-packages (from opencv-python)
Installing collected packages: opencv-python
Successfully installed opencv-python-3.4.0.12
```

```
rainer@swMBA:~$ pip3 install opencv-python
Collecting opencv-python
  Downloading opencv_python-3.4.0.12-cp36-cp36m-macosx_10_6_intel.macosx_10_9_in
tel.macosx_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (41.2MB)
    100% |████████████████████████████████| 41.2MB 24kB/s
Requirement already satisfied: numpy>=1.11.1 in /Library/Frameworks/Python.frame
work/Versions/3.6/lib/python3.6/site-packages (from opencv-python)
Installing collected packages: opencv-python
Successfully installed opencv-python-3.4.0.12
```

- Unofficial OpenCV packages for Python
- OpenCV-Contrib-Python(Latest Version)
  - https://pypi.python.org/pypi/opencv-contrib-python
  - contains contrib modules
  - Install OpenCV-Contrib-Python
    - pip install opencv-contrib-python
    - pip3 install opencv-contrib-python

```
rainer@swMBA:~$ pip install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-3.4.0.12-cp27-cp27m-macosx_10_6_intel.macosx
_10_9_intel.macosx_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (46.4M
B)
    100% |████████████████████████████████| 46.4MB 25kB/s
Requirement already satisfied: numpy>=1.11.1 in /Library/Frameworks/Python.frame
work/Versions/2.7/lib/python2.7/site-packages (from opencv-contrib-python)
Installing collected packages: opencv-contrib-python
Successfully installed opencv-contrib-python-3.4.0.12
```

```
rainer@swMBA:~$ pip3 install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-3.4.0.12-cp36-cp36m-macosx_10_6_intel.macosx
_10_9_intel.macosx_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (46.4M
B)
    100% |████████████████████████████████| 46.4MB 27kB/s
Requirement already satisfied: numpy>=1.11.1 in /Library/Frameworks/Python.frame
work/Versions/3.6/lib/python3.6/site-packages (from opencv-contrib-python)
Installing collected packages: opencv-contrib-python
Successfully installed opencv-contrib-python-3.4.0.12
```

❖ OpenCV-Python Installation - PyPi

- Version Specific Installation
  - List up OpenCV Version (3.x only)
    - pip install opencv-python==?
  - Install specific version (example)
    - pip install opencv-python=3.1.x.x

```
rainer@swMBA:~$ pip install opencv-python==?
Collecting opencv-python==?
  Could not find a version that satisfies the requirement opencv-python==? (from
 versions: 3.1.0.0, 3.1.0.1, 3.1.0.2, 3.1.0.3, 3.1.0.4, 3.1.0.5, 3.2.0.6, 3.2.0.
7, 3.2.0.8, 3.3.0.9, 3.3.0.10, 3.3.1.11, 3.4.0.12)
No matching distribution found for opencv-python==?
```

❖ OpenCV-Python Installation - PyPi

- Check out Installed Version
  - import cv2
  - cv2.__version__

```
rainer@swMBA:~$ python
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 12:01:12)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.0'
```

```
rainer@swMBA:~$ python3
Python 3.6.4 (v3.6.4:d48eceb0d5, Dec 18 2017, 21:07:28)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.0'
```

❖ Test Code

```
import cv2

img_file = "./img/girl.jpg" #
Path to image
img = cv2.imread(img_file)

if not img is None:
cv2.imshow('IMG', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
else:
print('No image file.')
```

Opencv with Python

1. OpenCV3 Intro
2. Setup on Windows
3. Setup on Mac
4. **Setup on Ubuntu**
5. Setup on Raspberry-Pi

Setup Python

- Ubuntu 16.04 LTS
- Check Already installed
  - python --version
  - python3 --version
- If Python Not Installed
  - Install Python2
    - sudo apt-get install python
  - Install Python 3
    - sudo apt-get install python3
- Pip install & check
  - pip for python
    - sudo apt-get install python-pip
    - pip --version

```
lee@vbox-opencv:~$ python --version
Python 2.7.12
lee@vbox-opencv:~$ python3 --version
Python 3.5.2
lee@vbox-opencv:~$
```

```
lee@vbox-opencv:~$ sudo apt-get install python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libllvm4.0 libpython3-dev libpython3.5-dev python3-dev python3-wheel
  python3.5-dev
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  python-pip
```

```
lee@vbox-opencv:~$ pip --version
pip 8.1.1 from /usr/lib/python2.7/dist-packages (python 2.7)
```

❖ Setup Python

- Pip install & check
  - pip3 for python3
    - sudo apt-get install python3-pip
    - pip3 --version

```
lee@vbox-opencv:~$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no l
  libllvm4.0
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  python3-pip
```

```
lee@vbox-opencv:~$ pip3 --version
pip 8.1.1 from /usr/lib/python3/dist-packages (python 3.5)
```

❖ Setup Dependent Library

- Numpy Installation
  - `pip install numpy`
  - `pip3 install numpy`

```
lee@vbox-opencv:~$ pip install numpy
Collecting numpy
  Using cached numpy-1.14.0-cp27-cp27mu-manylinux1_x86_64.whl
Installing collected packages: numpy
Successfully installed numpy-1.14.0
```

```
lee@vbox-opencv:~$ pip3 install numpy
Collecting numpy
  Using cached numpy-1.14.0-cp35-cp35m-manylinux1_x86_64.whl
Installing collected packages: numpy
Successfully installed numpy-1.14.0
```

- Numpy Check in Python
  - import numpy
  - numpy.__version__

```
lee@vbox-opencv:~$ python
Python 2.7.12 (default, Nov 20 2017, 18:23:56)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information
>>> import numpy
>>> numpy.__version__
'1.14.0'
```

```
lee@vbox-opencv:~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__version__
'1.14.0'
```

❖ Setup Dependent Library

- Matplotlib Installation
  - pip install matplotlib
  - pip3 install matplotlib

```
lee@vbox-opencv:~$ pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-2.1.2-cp27-cp27mu-manylinux1_x86_64.whl (15.0MB)
    100% |                                | 15.0MB 64kB/s
Collecting numpy>=1.7.1 (from matplotlib)
  Using cached numpy-1.14.0-cp27-cp27mu-manylinux1_x86_64.whl
```

```
lee@vbox-opencv:~$ pip3 install matplotlib
Collecting matplotlib
  Downloading matplotlib-2.1.2-cp35-cp35m-manylinux1_x86_64.whl (15.0MB)
    100% |                                | 15.0MB 85kB/s
Collecting six>=1.10 (from matplotlib)
  Using cached six-1.11.0-py2.py3-none-any.whl
```

- Matplotlib Check in Python
  - import matplotlib
  - matplotlib.__version__

```
lee@vbox-opencv:~$ python
Python 2.7.12 (default, Nov 20 2017, 18:23:56)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib
>>> matplotlib.__version__
'2.1.2'
```

```
lee@vbox-opencv:~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib
>>> matplotlib.__version__
'2.1.2'
```

❖ OpenCV-Python Installation - PyPi

- Unofficial OpenCV packages for Python
- OpenCV-Python(Latest Version)
  - https://pypi.python.org/pypi/opencv-python
  - Core modules without contrib modules
  - Install OpenCV-Python
    - pip install opencv-python
    - pip3 install opencv-python

```
lee@vbox-opencv:~$ pip install opencv-python
Collecting opencv-python
  Using cached opencv_python-3.4.0.12-cp27-cp27mu-manylinux1_x86_64.whl
Collecting numpy>=1.11.1 (from opencv-python)
  Using cached numpy-1.14.0-cp27-cp27mu-manylinux1_x86_64.whl
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.14.0 opencv-python-3.4.0.12
```

```
lee@vbox-opencv:~$ pip3 install opencv-python
Collecting opencv-python
  Downloading opencv_python-3.4.0.12-cp35-cp35m-manylinux1_x86_64.whl (24.9MB)
    100% |████████████████████████████████| 24.9MB 58kB/s
Collecting numpy>=1.11.1 (from opencv-python)
  Using cached numpy-1.14.0-cp35-cp35m-manylinux1_x86_64.whl
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.14.0 opencv-python-3.4.0.12
```

- Unofficial OpenCV packages for Python
- OpenCV-Contrib-Python(Latest Version)
  - https://pypi.python.org/pypi/opencv-contrib-python
  - contains contrib modules
  - Install OpenCV-Contrib-Python
    - pip install opencv-contrib-python
    - pip3 install opencv-contrib-python

```
lee@vbox-opencv:~$ pip install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-3.4.0.12-cp27-cp27mu-manylinux1_
30.5MB)
    100% |████████████████████████████████| 30.5MB 34kB/s
Collecting numpy>=1.11.1 (from opencv-contrib-python)
  Using cached numpy-1.14.0-cp27-cp27mu-manylinux1_x86_64.whl
Installing collected packages: numpy, opencv-contrib-python
Successfully installed numpy-1.14.0 opencv-contrib-python-3.3.1.11
```

```
lee@vbox-opencv:~$ pip3 install opencv-contrib-python
Collecting opencv-contrib-python
  Using cached opencv_contrib_python-3.4.0.12-cp35-cp35m-manylinux1
Collecting numpy>=1.11.1 (from opencv-contrib-python)
  Using cached numpy-1.14.0-cp35-cp35m-manylinux1_x86_64.whl
Installing collected packages: numpy, opencv-contrib-python
Successfully installed numpy-1.14.0 opencv-contrib-python-3.4.0.12
```

❖ OpenCV-Python Installation - PyPi

- Version Specific Installation
  - List up OpenCV Version (3.x only)
    - pip install opencv-python==?
  - Install specific version (example)
    - pip install opencv-python=3.1.x.x

```
lee@vbox-opencv:~$ pip3 install opencv-contrib-python==?
Collecting opencv-contrib-python==?
  Could not find a version that satisfies the requirement opencv-contrib-python==?
(from versions: 3.2.0.7, 3.2.0.8, 3.3.0.9, 3.3.0.10, 3.3.1.11, 3.4.0.12)
No matching distribution found for opencv-contrib-python==?
```

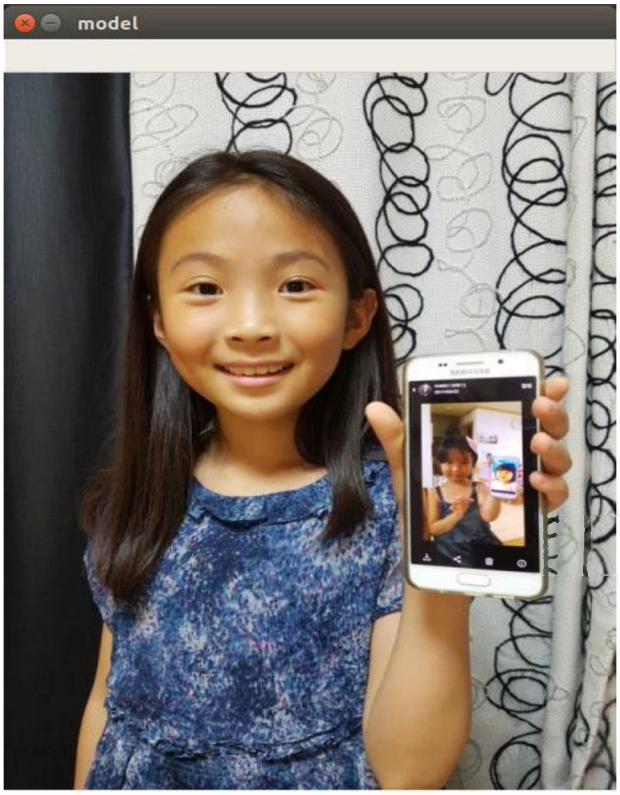
- Check out Installed Version
  - import cv2
  - cv2.__version__

```
lee@vbox-opencv:~$ python
Python 2.7.12 (default, Nov 20 2017, 18:23:56)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more inform
>>> import cv2
>>> cv2.__version__
'3.4.0'
```

```
lee@vbox-opencv:~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more infor
>>> import cv2
>>> cv2.__version__
'3.4.0'
```

❖ Test Code

```
import cv2



img_file = "img/model.jpg"
img = cv2.imread(img_file)



if img:
    cv2.imshow('model', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```



Opencv with Python

1. OpenCV3 Intro
2. Setup on Windows
3. Setup on Mac
4. Setup on Ubuntu
5. **Setup on Raspberry-Pi**

❖ Setup Python & pip

- Raspbian Stretch version
- Check Default Installed
  - python --version
  - python3 --version
  - pip --version
  - pip3 --version

```
pi@raspberry:~ $ python --version
Python 2.7.13
pi@raspberry:~ $ python3 --version
Python 3.5.3
pi@raspberry:~ $ pip --version
pip 9.0.1 from /usr/lib/python2.7/dist-packages (python 2.7)
pi@raspberry:~ $ pip3 --version
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.5)
```

❖ Installing Numpy

- pip not support numpy for raspberry-pi
- Install using APT
  - sudo apt-get install python-numpy
  - sudo apt-get install python3-numpy

```
pi@raspberrypi:~ $ sudo apt-get install python-numpy
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```
pi@raspberrypi:~ $ sudo apt-get install python3-numpy
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

❖ Installing Numpy

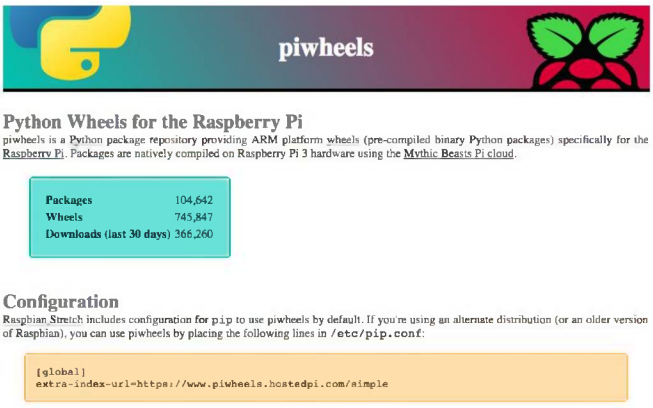
- check numpy version
  - import numpy
  - numpy.__version__

```
pi@raspberrypi:~ $ python
Python 2.7.13 (default, Jan 19 2017, 14:48:08)
[GCC 6.3.0 20170124] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__version__
'1.12.1'
```

```
pi@raspberrypi:~ $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__version__
'1.12.1'
```

❖ Installing OpenCV

- pip not support OpenCV-Python for RaspberryPi
  - pip default repository
    - https://pypi.python.org/
    - pypi not support OpenCV-Python for armv7l
  - piwheels repository
    - https://www.piwheels.hostedpi.com/
    - repository for raspberry-pi
    - /etc/pip.conf (rasbian stretch)
    - not work properly (Jan. 2018)
- 3 Ways to install OpenCV-Python
  - Installing from APT repository
  - Installing from Building source
  - Installing with Pre-built Debian Package
    - Provided by this course



❖ Installing OpenCV from APT repository

- support only Python2, OpenCV 2.4
- Easiest way to install OpenCV-Python on Raspberry Pi
- Not Support Python3, OpenCV 3+
- Installing
  - sudo apt-get install python-opencv

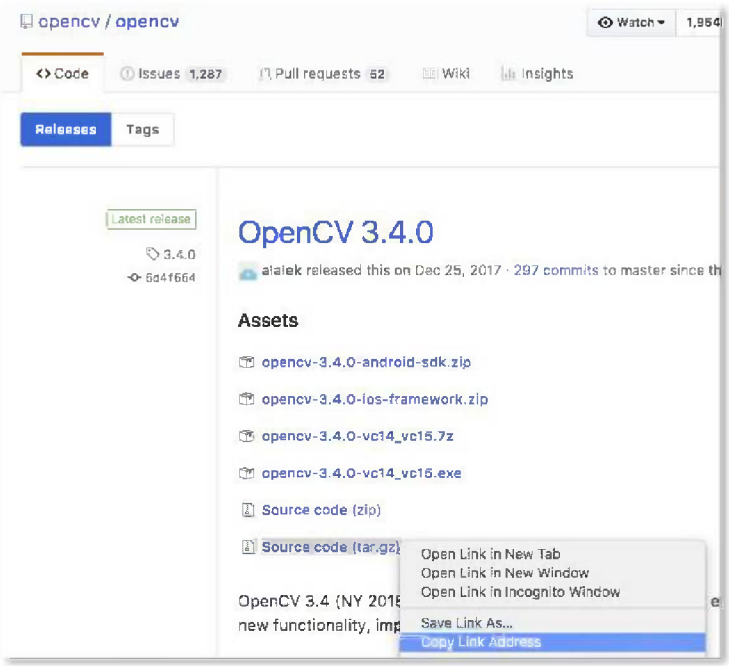
```
pi@raspberrypi:~ $ sudo apt-get install python-opencv
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libopencv-contrib2.4v5 libopencv-legacy2.4v5 libopencv-ml2.4v5
  libopencv-photo2.4v5
The following NEW packages will be installed:
  libopencv-contrib2.4v5 libopencv-legacy2.4v5 libopencv-ml2.4v5
  libopencv-photo2.4v5 python-opencv
```

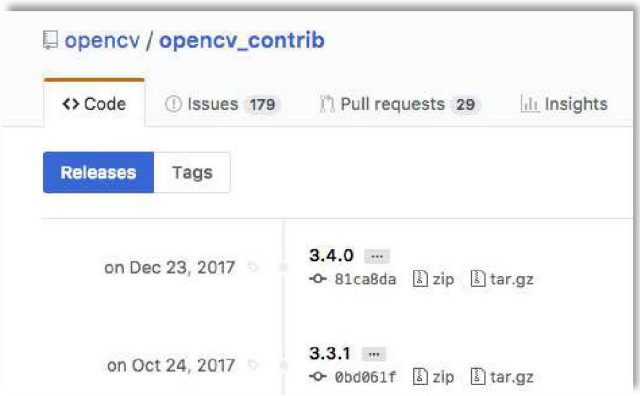
- check version
  - import cv2
  - cv2.__version__

```
pi@raspberrypi:~ $ python
Python 2.7.13 (default, Jan 19 2017, 14:48:08)
[GCC 6.3.0 20170124] on linux2
Type "help", "copyright", "credits" or "license" for more information.
import cv2
>>> import cv2
cv2.__version__
>>> cv2.__version__
'2.4.9.1'
```

❖ Installing OpenCV from Building Source

- Install tools for buildding process
  - sudo apt install build-essential cmake pkg-config
- Install dependant libraries
  - Image Format Library
    - sudo apt-get install -y libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
  - Video Codec libraries
    - sudo apt-get install -y libavcodec-dev libavformat-dev libswscale-dev libxvidcore-dev libx264-dev libxine2-dev
  - Linux Standard Video Device Libraries
    - sudo apt-get install -y libv4l-dev v4l-utils
  - Video Stream libraries
    - sudo apt-get install -y libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
  - GUI Window Library
    - sudo apt-get install libqt4-dev
- Install dependant libraries (Continue)
  - OpenGL Libraries
    - sudo apt-get install –y mesa-utils libgl1-mesa-dri libqt4-opengl-dev libatlas-base-dev gfortran libeigen3-dev
  - Python Libraries
    - sudo apt-get install -y python-dev python3-dev
  - Python Numpy
    - sudo apt-get install –y python-numpy python3-numpy
- Make a directory "~/opence"
  - ~ $ mkdir opencv
  - ~ $ cd opencv
- Download OpenCV Source Code
  - https://github.com/opencv/opencv/releases
  - https://github.com/opencv/opencv_contrib/releases





❖ Installing OpenCV from Building Source

- Download OpenCV Source Code
  - ~/opencv$ wget –O opencv-3.4.tar.gz https://github.com/opencv/opencv/archive/3.4.0.tar.gz

```
pi@raspberrypi:~/opencv $ wget -O opencv-3.4.tar.gz https://github.com/opencv/op
encv/archive/3.4.0.tar.gz
--2018-02-03 08:50:44--  https://github.com/opencv/opencv/archive/3.4.0.tar.gz
Resolving github.com (github.com)... 192.30.255.112, 192.30.255.113
Connecting to github.com (github.com)|192.30.255.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/opencv/opencv/tar.gz/3.4.0 [following]
--2018-02-03 08:50:45--  https://codeload.github.com/opencv/opencv/tar.gz/3.4.0
Resolving codeload.github.com (codeload.github.com)... 192.30.255.120, 192.30.25
5.121
Connecting to codeload.github.com (codeload.github.com)|192.30.255.120|:443... c
onnected.
HTTP request sent, awaiting response... 200 OK
Length: 87169544 (83M) [application/x-gzip]
Saving to: 'opencv-3.4.tar.gz'

opencv-3.4.tar.gz   100%[===================>]  83.13M  2.37MB/s    in 91s
```

❖ Installing OpenCV from Building Source

- Download OpenCV Source Code
  - ~/opencv$ wget –O opencv-3.4-contrib.tar.gz https://github.com/opencv/opencv_contrib/archive/3.4.0.tar.gz

```
pi@raspberrypi:~/opencv $ wget -O opencv-contrib-3.4.tar.gz https://github.com/o
pencv/opencv_contrib/archive/3.4.0.tar.gz
--2018-02-03 09:03:32--  https://github.com/opencv/opencv_contrib/archive/3.4.0.
tar.gz
Resolving github.com (github.com)... 192.30.255.113, 192.30.255.112
Connecting to github.com (github.com)|192.30.255.113|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/opencv/opencv_contrib/tar.gz/3.4.0 [follow
ing]
--2018-02-03 09:03:33--  https://codeload.github.com/opencv/opencv_contrib/tar.g
z/3.4.0
Resolving codeload.github.com (codeload.github.com)... 192.30.255.121, 192.30.25
5.120
Connecting to codeload.github.com (codeload.github.com)|192.30.255.121|:443... c
onnected.
HTTP request sent, awaiting response... 200 OK
Length: 55978680 (53M) [application/x-gzip]
Saving to: 'opencv-contrib-3.4.tar.gz'

opencv-contrib-3.4. 100%[===================>]  53.38M  2.64MB/s    in 52s
```

❖ Installing OpenCV from Building Source

- Unarchive
  - ~/opencv $ tar –xvf opencv-3.4.tar.gz
  - ~/opencv $ tar –xvf opencv-contrib-3.4.targ.gz
- Make Build Directory
  - ~/opencv $ mkdir -p /opencv/build
  - ~/opencv $ cd ./opencv/build

```
pi@raspberrypi:~/opencv $ ls
opencv-3.4.0       opencv-contrib-3.4.tar.gz
opencv-3.4.tar.gz  opencv_contrib-3.4.0
pi@raspberrypi:~/opencv $ mkdir -p ./opencv-3.4.0/build
pi@raspberrypi:~/opencv $ cd ./opencv-3.4.0/build/
pi@raspberrypi:~/opencv/opencv-3.4.0/build $
```

❖ Installing OpenCV from Building Source

- CMake

```
~/opencv/opencv3.4.0/build $ cmake -D
CMAKE_BUILD_TYPE=RELEASE \
 -D CMAKE_INSTALL_PREFIX=/usr/local \
 -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-
3.4.0/modules \
 -D BUILD_DOCS=ON \
 -D BUILD_EXAMPLES=ON \
 -D ENABLE_NEON=ON \
 -D WITH_QT=ON \
 -D WITH_OPENGL=ON \
 -D WITH_XINE=ON ../
```

- swap memory
  - Memory check
    - free –h
  - Swap check
    - swapon –s
  - Creating swap file 1GB
    - sudo fallocate -l 1024M /var/swap_temp
  - Making swap
    - sudo mkswap /var/swap_temp
  - Applying swap
    - sudo swapon /var/swap_temp
  - Remove swap
    - sudo swapoff -v /var/swap_temp
  - Remove swap file
    - rm –f /var/swap_temp

❖ Installing OpenCV from Building Source

- make

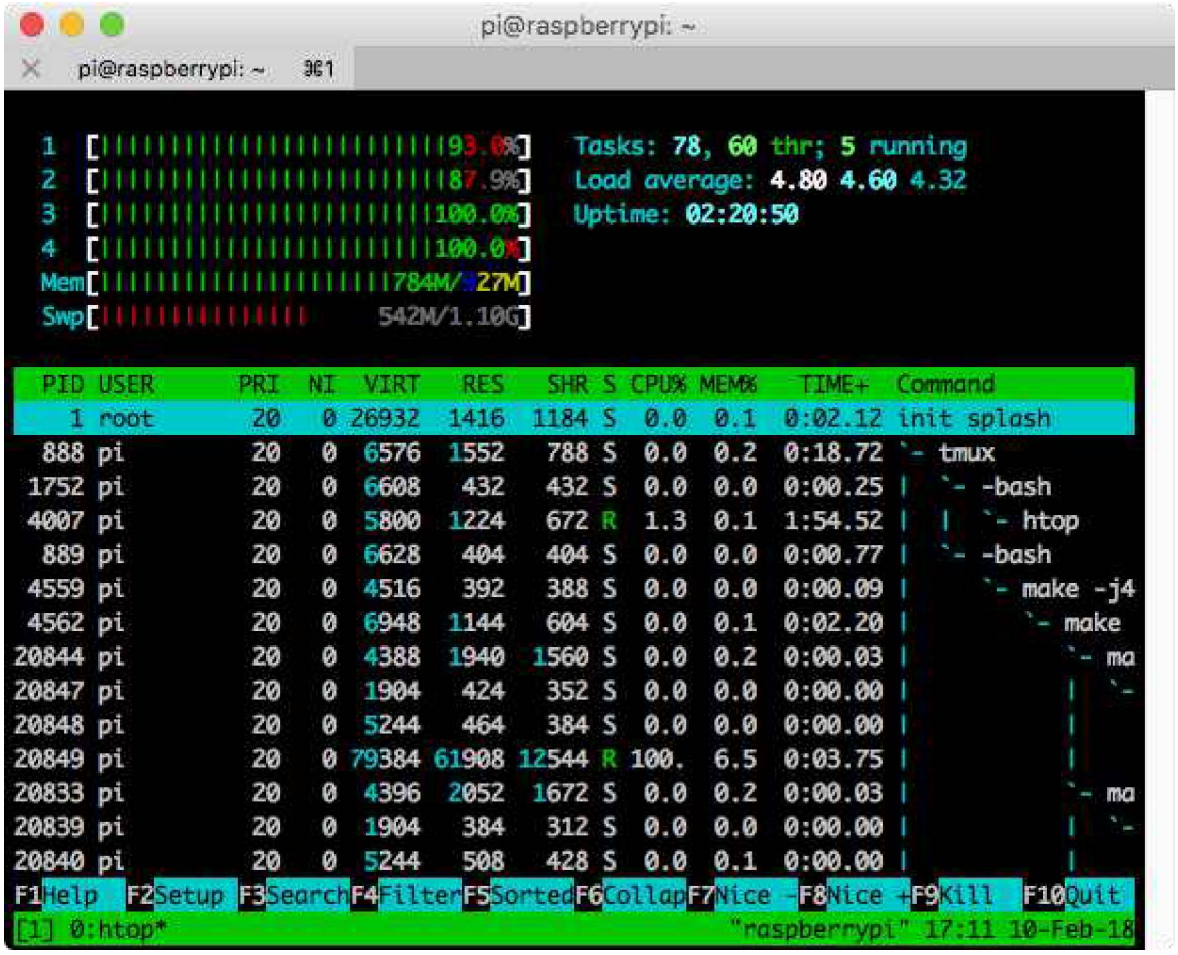
```
~/opencv/opencv3.4.0/build $ make
```

```
~/opencv/opencv3.4.0/build $ time make -j4
```

```
[ 99%] Building CXX object modules/optflow/CMakeFiles/opencv_optflow.dir/src/pcaflow.cpp.o
[ 99%] Building CXX object modules/optflow/CMakeFiles/opencv_optflow.dir/src/simpleflow.cpp.o
[100%] Building CXX object modules/optflow/CMakeFiles/opencv_optflow.dir/src/sparse_matching_gpc.cpp.o
[100%] Building CXX object modules/optflow/CMakeFiles/opencv_optflow.dir/src/sparsetodenseflow.cpp.o
[100%] Building CXX object modules/optflow/CMakeFiles/opencv_optflow.dir/src/variational_refinement.cpp.o
[100%] Building CXX object modules/optflow/CMakeFiles/opencv_optflow.dir/opencl_kernels_optflow.cpp.o
[100%] Linking CXX shared library ../../lib/libopencv_optflow.so
[100%] Built target opencv_optflow
[100%] Generating pyopencv_generated_include.h, pyopencv_generated_funcs.h, pyopencv_generated_types.h, pyope
ncv_generated_type_reg.h, pyopencv_generated_ns_reg.h
[100%] Generating pyopencv_generated_include.h, pyopencv_generated_funcs.h, pyopencv_generated_types.h, pyope
ncv_generated_type_reg.h, pyopencv_generated_ns_reg.h
Scanning dependencies of target opencv_python3
Scanning dependencies of target opencv_python2
[100%] Building CXX object modules/python2/CMakeFiles/opencv_python2.dir/__/src2/cv2.cpp.o
[100%] Building CXX object modules/python3/CMakeFiles/opencv_python3.dir/__/src2/cv2.cpp.o
[100%] Linking CXX shared module ../../lib/python3/cv2.cpython-34m.so
[100%] Linking CXX shared module ../../lib/cv2.so
[100%] Built target opencv_python2
[100%] Built target opencv_python3

real    45m16.204s
user    152m25.580s
sys     3m26.660s
```

❖ Installing OpenCV from Building Source

- htop



- make install

```
~/opencv/opencv3.4.0/build $ sudo make install
```

- check version

```
~ $ pkg-config -modversion opencv
```

```
pi@raspberrypi:~/opencv $ pkg-config --modversion opencv
3.4.0
```

❖ Installing OpenCV from Building Source

- Check Python 2, 3 cv2 module version

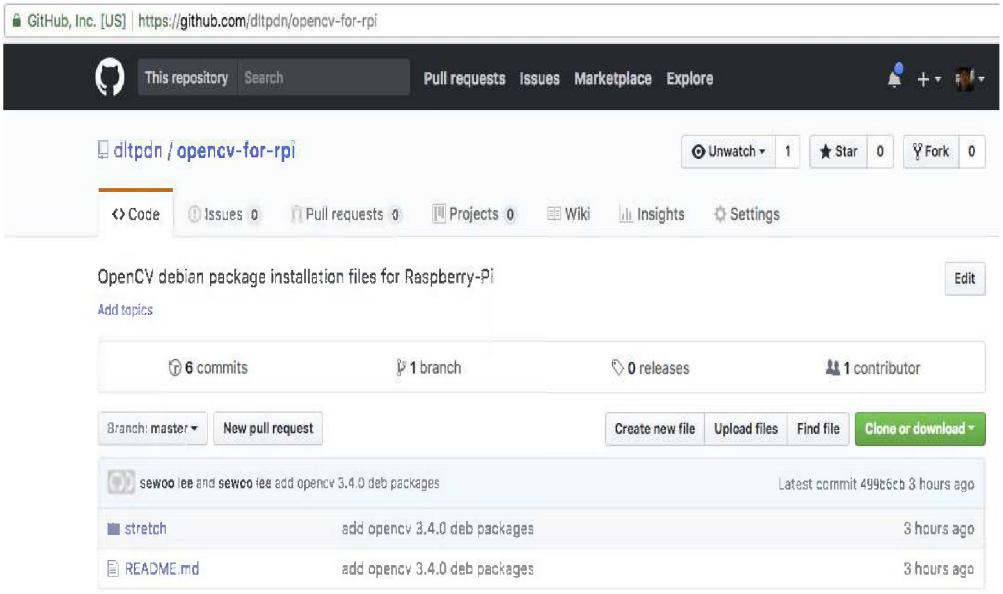
```
import cv2
cv2.  version
```

```
pi@raspberrypi:~ $ python
Python 2.7.13 (default, Jan 19 2017, 14:48:08)
[GCC 6.3.0 20170124] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.0'
```

```
pi@raspberrypi:~ $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.0'
```

❖ Installing OpenCV with Pre-built Debian Packages

- Provided from this course unofficially
- Download deb package files from github
  - https://github.com/dltpdn/opencv-for-rpi
    - opencv-main
    - opencv-libs
    - opencv-dev
    - opencv-python



❖ Installing OpenCV with Pre-built Debian Packages

- git download

```
~ $ git clone https://github.com/dltpdn/opencv-for-rpi.git
~ $ cd ./opencv-for-rpi/stretch/3.4.0
```

- directory
  - <debian_version> / <opencv_version>

```
pi@raspberrypi:~ $ git clone https://github.com/dltpdn/opencv-for-rpi.git
Cloning into 'opencv-for-rpi'...
remote: Counting objects: 34, done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 34 (delta 0), reused 11 (delta 0), pack-reused 22
Unpacking objects: 100% (34/34), done.
pi@raspberrypi:~ $ cd opencv-for-rpi/
pi@raspberrypi:~/opencv-for-rpi $ cd stretch/
pi@raspberrypi:~/opencv-for-rpi/stretch $ cd 3.4.0/
pi@raspberrypi:~/opencv-for-rpi/stretch/3.4.0 $ ls
OpenCV-unknown-armv7l-dev.deb     OpenCV-unknown-armv7l.sh
OpenCV-unknown-armv7l-libs.deb    OpenCV-unknown-armv7l.tar.Z
OpenCV-unknown-armv7l-main.deb    OpenCV-unknown-armv7l.tar.gz
OpenCV-unknown-armv7l-python.deb
```

- Installing

```
~/opencv-for-rpi/stretch/3.4.0 $ sudo apt -y install
./OpenCV*.dev
```

```
pi@raspberrypi:~/opencv-for-rpi/stretch/3.4.0 $ sudo apt -y install ./OpenCV*.de
b
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'opencv-dev' instead of './OpenCV-unknown-armv7l-dev.deb'
Note, selecting 'opencv-libs' instead of './OpenCV-unknown-armv7l-libs.deb'
Note, selecting 'opencv-main' instead of './OpenCV-unknown-armv7l-main.deb'
Note, selecting 'opencv-python' instead of './OpenCV-unknown-armv7l-python.deb'
The following additional packages will be installed:
  ghostscript libavcodec57 libavformat57 libavutil55 libgraphicsmagick-q16-3
  libgs9 libgs9-common libijs-0.35 libiso9660-8 libjasper1 libjbig2dec0
  libjpeg8 libmng1 libpaper-utils libpaper1 libqt4-dbus libqt4-opengl
  libqt4-test libqt4-xml libqtcore4 libqtdbus4 libqtgui4 libswresample2
  libswscale4 libvcdinfo0 libwmf0.2-7 libxine2 libxine2-bin libxine2-doc
  libxine2-ffmpeg libxine2-misc-plugins libxine2-plugins qdbus qt-at-spi
```

❖ Installing OpenCV with Pre-built Debian Packages

- Check result

```
~ $ pkg-config –modversion opencv
```

```
pi@raspberrypi:~/opencv-for-rpi/stretch/3.4.0 $ pkg-config --modversion opencv
3.4.0
```

❖ Installing python Matplotlib

- pip not support matplotlib for Raspberry Pi
- Installing from APT Repository

```
sudo apt-get install python-matplotlib
sudo apt-get install python3-matplotlib

sudo apt-get install python-gi-cairo
sudo apt-get install python3-gi-cairo

sudo apt-get install python-cairocffi
sudo apt-get install python3-cairocffi
```

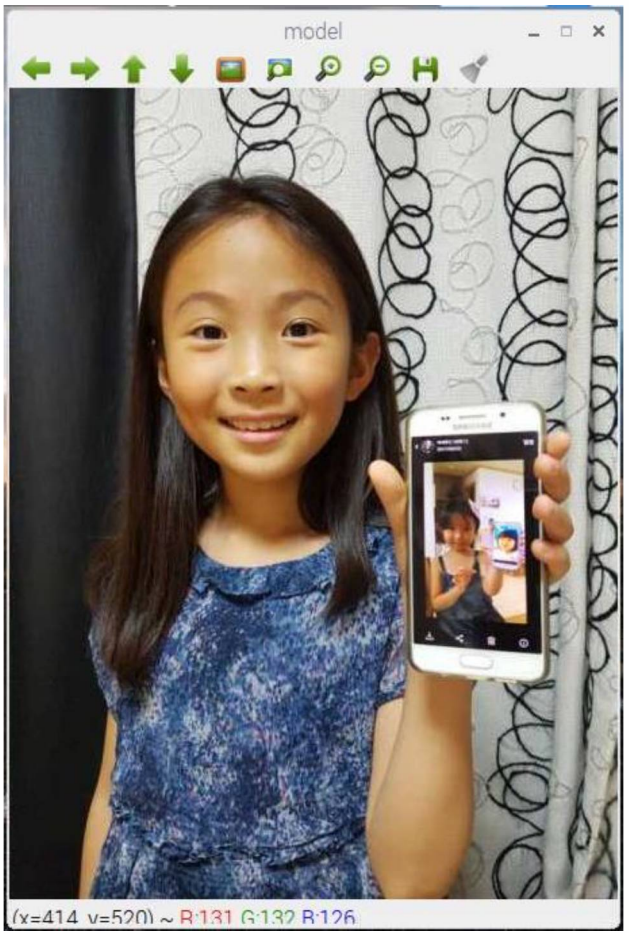
- Matplotlib Check in Python
  - import matplotlib
  - matplotlib.__version__

```
pi@raspberrypi:~ $ python
Python 2.7.13 (default, Jan 19 2017, 14:48:08)
[GCC 6.3.0 20170124] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib
>>> matplotlib.__version__
'2.1.2'
```

```
pi@raspberrypi:~ $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib
>>> matplotlib.__version__
'2.0.0'
```

❖ Test Code

```
import cv2


img_file = "img/model.jpg"
img = cv2.imread(img_file)


if img:
    cv2.imshow('model', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```



Basic Interface

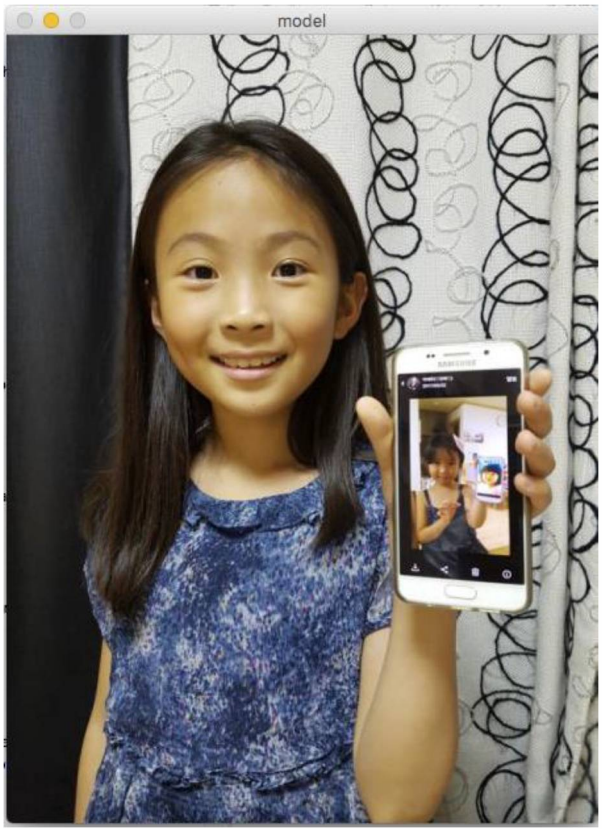
1. **Image and Video I/O**
2. Drawing
3. Window Management
4. Event Handling

Image show

- img = cv2.imread(file_name [,mod_flag]): Read image from file
  - file_name: Image path, string
  - mode_flag: Defining Read mode flag
    - cv2.IMREAD_UNCHANGED: Color (BGR) scale, base value -1
    - cv2.IMREAD_GRAYSCALE: Grey Scale, 0
    - cv2.IMREAD_COLOR: Scale saved in file, 1
  - img: reading image, NumPy ndarray
- cv2.imshow(title, img): Display on image
  - title : Name of the window, string
  - img : Image to show, NumPy ndarray
- key = cv2.waitKey([delay]): Keyboard input standby
  - delay : Standby time, ms
  - key : Inserted keyboard value, -1: timeout
- cv2.destroyAllWindows() : Close all open window

Image show

```
import cv2

img_file = ”../img/model.jpg"
img = cv2.imread(img_file)



if img:
    cv2.imshow('model', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```



❖ Read in GrayScale

```
import cv2

img_file = ”../img/model.jpg"
img = cv2.imread(img_file,
cv2.IMREAD_GRAYSCALE)

if not img is None:
    cv2.imshow(img_file, img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
else:
    print("no file:", img_file)
```

❖ Image Write

- cv2.imwrite(file_hame, img)
  - file_name: Name of file to save, string
  - img: Image to save (NumPy array)

```
import cv2

img_file = ”../img/model.jpg"
img = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE)
cv2.imshow("my darling daughter", img)

key = cv2.waitKey(0) & 0xFF
if key == 27: #esc
    cv2.destroyAllWindows()
elif key == ord('s'):
    print('saving file.');
    cv2.imwrite(”./gray.jpg", img)
    cv2.destroyAllWindows()
```

❖ Video Capture

- cap = cv2.VideoCapture(file_path or index): Video capture object generator
  - file_path: Image file path, str
  - index: Camera device number, int, Inclement gradually from 0 (i.e.:0,1,2,....)
- ret = cap.isOpened(): Check connection initialization
  - ret : Check for success, Boolean
- cap.set()/get() : Property Change and confirm
  - cv2.CAP_PROP_FRAME_WIDTH(3), cv2.CAP_PROP_FRAME_HEIGHT(4)
  - cv2.CAP_PROP_FPS: Number of frame per second
  - cv2.CAP_PROP_POS_MSEC: Frame position of video file (ms)
  - cv2.CAP_PROP_POS_AVI_RATIO: Position of video file (0-Start, 1-End)
  - cv2.CAP_PROP_FOURCC: Video file codec string
  - cv2.CAP_PROP_AUTOFOCUS: Camera auto focus
  - cv2.CAP_PROP_ZOOM: Camera Zoom
- ret, img = cap.read(): Read next frame
  - ret : Check for success, boolean
  - img: Image frame data (numpy array)

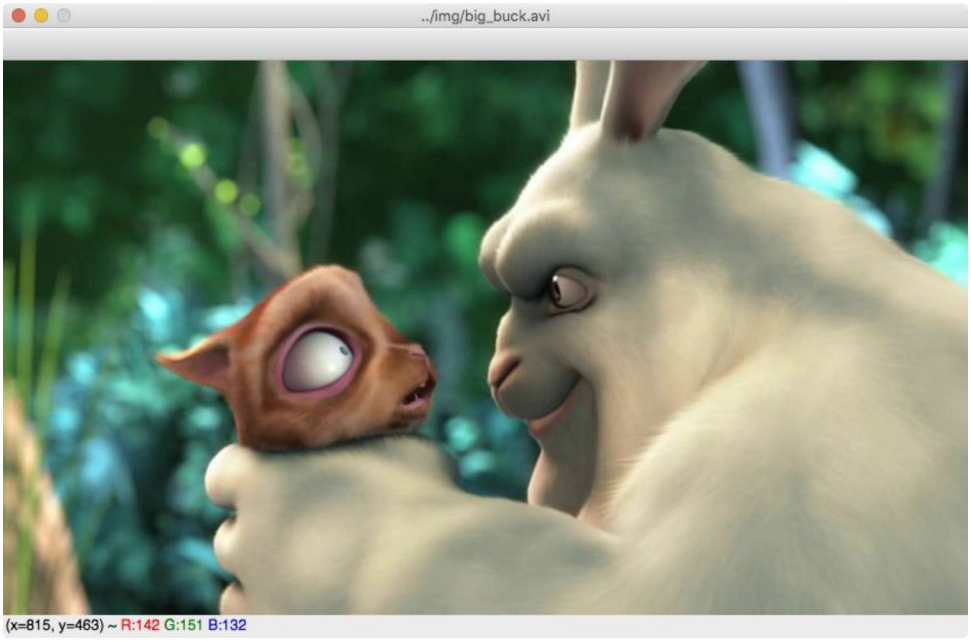
❖ Video File Play

```
import cv2
video_file = "../img/big_buck.avi"
cap = cv2.VideoCapture(video_file) #0 or -1
while cap.isOpened():
    ret, img = cap.read()
    if ret:
        cv2.imshow(video_file, img)
        if cv2.waitKey(1) & 0xFF == 27:
            break
    else:
        print('no more frame.')
        break
else:
    print('can't open video.')
cap.release()
cv2.destroyAllWindows()
```

Video File Play with FPS

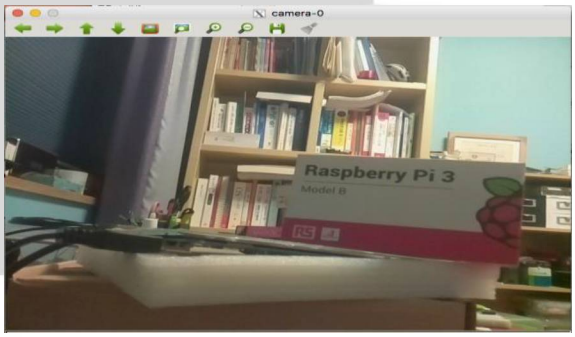
- delay = 1000/FPS

```
import cv2

video_file = "../img/big_buck.avi"
cap = cv2.VideoCapture(video_file) #0 or -1
fps = cap.get(cv2.CAP_PROP_FPS)
delay = int(1000/fps)
print("FPS: %f, Delay: %dms" %(fps, delay))

while cap.isOpened():
    ret, img = cap.read()
    if ret:
        cv2.imshow(video_file, img)
        if cv2.waitKey(delay) & 0xFF == 27:
            break
    else:
        print('no more frame.')
        break
else:
    print('can't open video.')
cap.release()
cv2.destroyAllWindows()
```



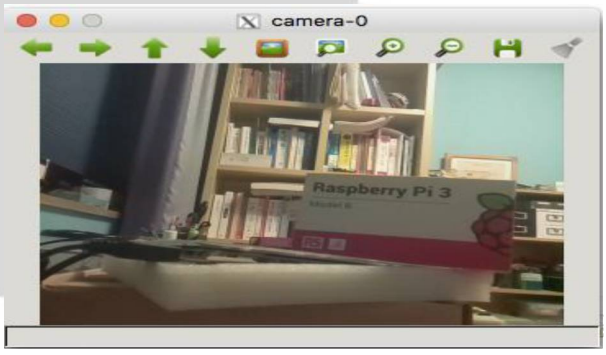
❖ Camera Frame

```
import cv2
cap = cv2.VideoCapture(0) #0 or -1
while cap.isOpened():
    ret, img = cap.read()
    if ret:
        cv2.imshow('camera-0', img)
        if cv2.waitKey(1) & 0xFF == 27: #esc
            break
    else:
        print('no camera!')
        break
cap.release()
cv2.destroyAllWindows()
```



❖ Camera Frame Resize

```
import cv2

cap = cv2.VideoCapture(0) #0 or -1
print("width: %d, height:%d" % (cap.get(3),
cap.get(4)) )
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
print("resized width: %d, height:%d" % (cap.get(3),
cap.get(4)) )
while cap.isOpened():
    ret, img = cap.read()
    if ret:
        cv2.imshow('camera-0', img)
        if cv2.waitKey(1) & 0xFF == 27: #esc
            break
    else:
        print('no camera!')
        break
cap.release()
cv2.destroyAllWindows()
```

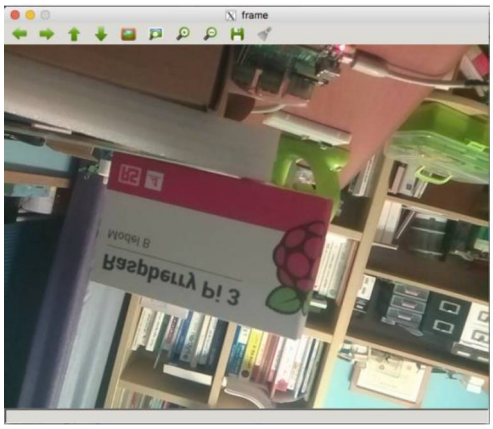


❖ Camera Frame Flip

- cv2.flip(img, flipCode)
  - Flip image
  - flipcode
    - 0 : x-axis (rows)
    - 1 : y-axis (cols)
    - -1 : x,y axis (rows, cols)

$$dst_{ij} = \begin{cases} src_{src.rows-i-1,j} & if\ flipCode = 0 \\ src_{i,src.cols-j-1} & if\ flipCode > 0 \\ src_{src.rows-i-1,src.cols-j-1} & if\ flipCode < 0 \end{cases}$$

```
import cv2
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if ret==True:
        frame = cv2.flip(frame,1)
        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == 27: #esc
            break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```



❖ Save a Camera Frame

```
import cv2

cap = cv2.VideoCapture(0)                          # Connect camera #0
if cap.isOpened() :
    while True:
        ret, frame = cap.read()                    # Read camera frame
        if ret:
            cv2.imshow('camera',frame)             # Display frame display
            if cv2.waitKey(1) != -1 :              # Select any key to
                cv2.imwrite('photo.jpg', frame) # Save frame as 'photo.jpg'
                break
        else:
            print('no frame!')
            break
else:
    print('no camera!')
cap.release()
cv2.destroyAllWindows()
```

❖ Video Recording with Camera

- writer = cv2.VideoWriter(file_path, fourcc, fps, (width, height))
  - file_path : Path to save
  - fourcc : Format to save, 4 letters
    - cv2.VideoWriter_fourcc(*'ABCD')
      - ABCD : 'MJPG' , 'DIVX'
    - ord('D') + (ord('I') <<8) + (ord('V') <<16) + (ord('X') <<24)
  - fps : Frame per Sec, float
  - (width, height) : Frame width, height, turple
- writer.write(frame)
  - frame : numpy array
- writer.set()/get()
  - Property change and check

- cv2.CAP_PROP_*
  - FRAME_WIDTH(3),
  - FRAME_HEIGHT(4)

```
import cv2

cap = cv2.VideoCapture(0)    # Connect #0 camera
file_path = './record.avi'
fps = 40                     # FPS, Frame per sec
fourcc = cv2.VideoWriter_fourcc(*'DIVX') # Encoding format letter
size = (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), \

     int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))          # Frame Size
out = cv2.VideoWriter(file_path, fourcc, fps, size) # Generate
VideoWriter Objects
while True:
     ret, frame = cap.read()
     if ret:
         cv2.imshow('camera-recording',frame)
         out.write(frame)                          # Save File
         if cv2.waitKey(1) != -1:    break
     else: break
else:
    print("can't open camera!")
out.release()                                     # Close File
cap.release()
cv2.destroyAllWindows()
```

Basic Interface

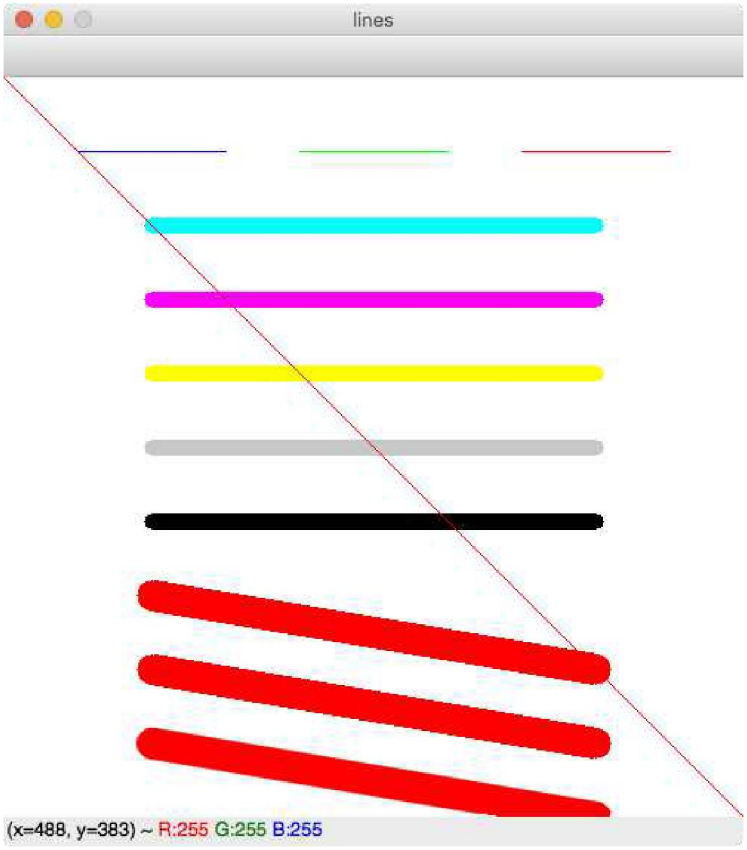
1. Image and Video I/O
2. **Drawing**
3. Window Management
4. Event Handling

❖ Line

- cv2.line(img, start, end, color [, thickness, lineType]): 직선을 그리기
  - img : 그림 그릴 대상 이미지, Numpy array
  - start : 선 시작 지점 좌표, 튜플(x,y)
  - end : 선 끝 지점 좌표, 튜플(x, y)
  - color : 선 색상, 튜플(Blue, Green, Red), 0~255
  - thickness : 선 두께
  - lineType : 선 그리기 형식,
    - cv2.LINE_4 : 4 연결 선 알고리즘
    - cv2.LINE_8 : 8 연결 선 알고리즘
    - cv2.LINE_AA : 안티에일리어싱(Antialiasing,계단현상없는 선)

```
import cv2

img = cv2.imread('./img/blank_500.jpg')

cv2.line(img, (50, 50), (150, 50), (255,0,0)) # Blue 1 pixel line
cv2.line(img, (200, 50), (300, 50), (0,255,0)) # Green 1 pixel line
cv2.line(img, (350, 50), (450, 50), (0,0,255)) # Red 1 pixel line
cv2.line(img, (100, 100), (400, 100), (255,255,0), 10)
cv2.line(img, (100, 150), (400, 150), (255,0,255), 10)
cv2.line(img, (100, 200), (400, 200), (0,255,255), 10)
cv2.line(img, (100, 250), (400, 250), (200,200,200), 10)
cv2.line(img, (100, 300), (400, 300), (0,0,0), 10)
cv2.line(img, (100, 350), (400, 350), (0,0,255), 20, cv2.LINE_4 )
cv2.line(img, (100, 400), (400, 400), (0,0,255), 20, cv2.LINE_8)
cv2.line(img, (100, 450), (400, 450), (0,0,255), 20, cv2.LINE_AA)
cv2.line(img, (0,0), (500,500), (0,0,255))   # Diagonal line across
the image

cv2.imshow('lines', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Line

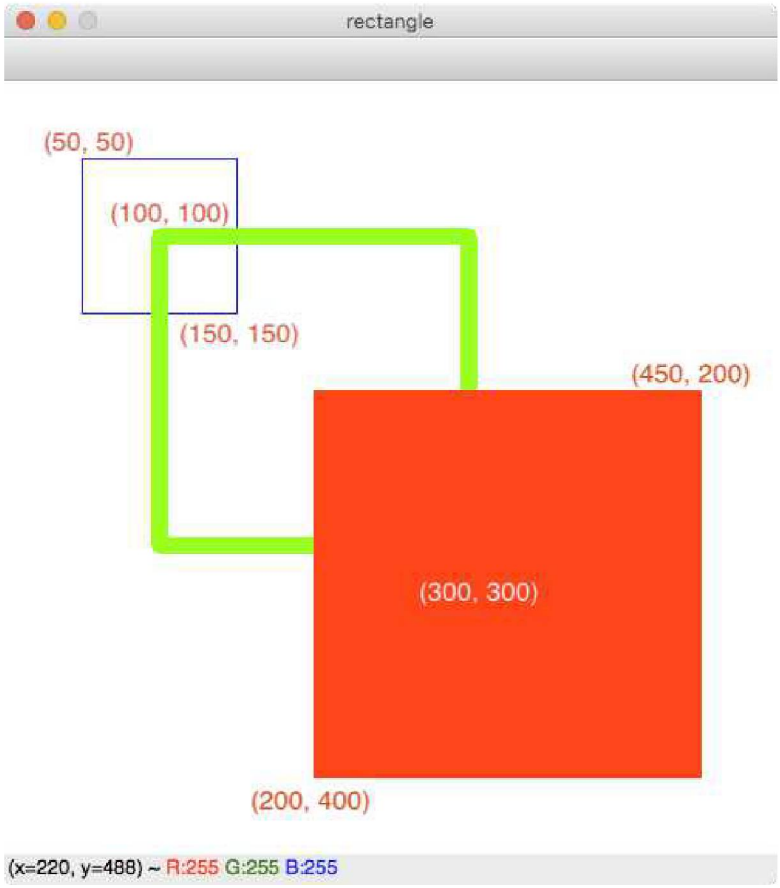


❖ Rectangle

- cv2.rectangle(img, start, end, color[, thickness, lineType]): Drawing quadrangle
  - img: Image to draw on
  - start: Vertex of quadrangle beginning, tuple (x, y)
  - end: Vertex of quadrangle end, tuple (x, y)
  - color: Color, tuple (Blue, Green, Red)
  - thickness: Line Thickness, -1: Fill
  - lineType: Line type,
    - cv2.LINE_4
    - cv2.LINE_8
    - cv2.LINE_AA

❖ Rectangle

```
import cv2

img = cv2.imread('./img/blank_500.jpg')

cv2.rectangle(img, (50, 50), (150, 150), (255,0,0) )
cv2.rectangle(img, (300, 300), (100, 100), (0,255,0), 10 )
cv2.rectangle(img, (450, 200), (200, 450), (0,0,255), -1 )

cv2.imshow('rectangle', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

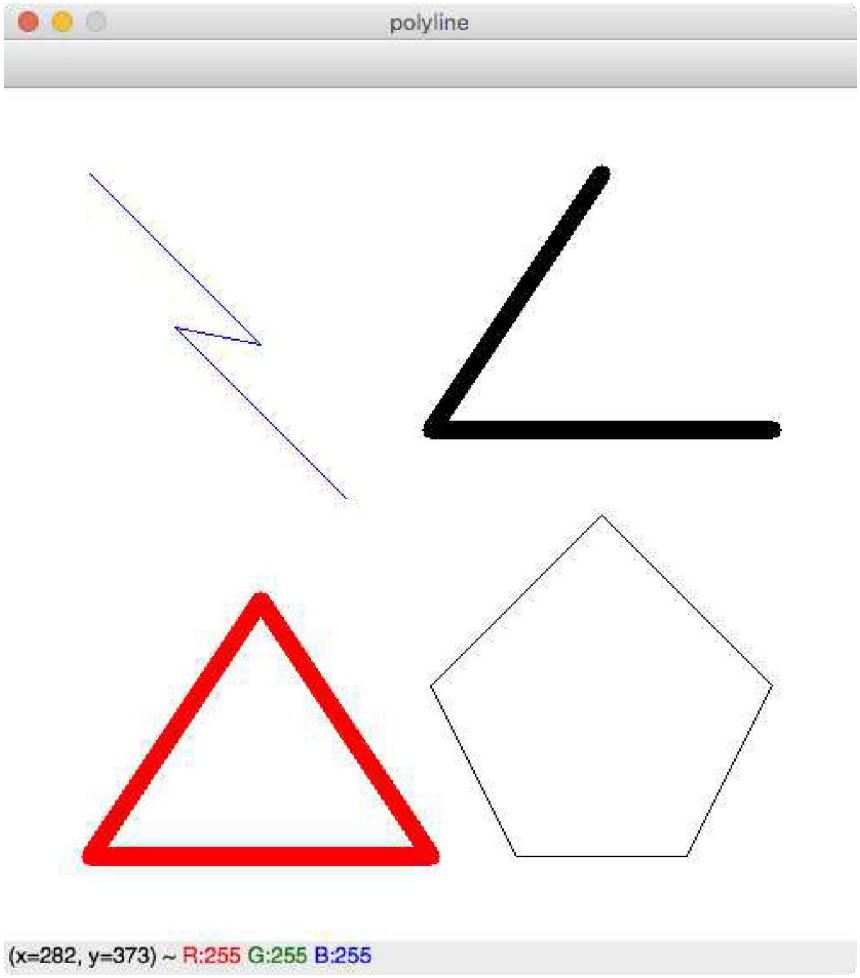


❖ Polylines

- cv2.polylines(img, points, isClosed, color[, thickness, lineType]): Draw polygon
  - img: Image to draw on
  - points: Vertex coordination, Numpy array's list
  - isClosed: Figure closed or not, Boolean
  - color: color, tuple (Blue, Green, Red)
  - thickness: Line thickness
  - lineType : Line type
    - cv2.LINE_4
    - cv2.LINE_8
    - cv2.LINE_AA

```
import cv2
import numpy as np

img = cv2.imread('./img/blank_500.jpg')

pts1 = np.array([[50,50], [150,150], [100,140],[200,240]],
dtype=np.int32)
pts2 = np.array([[350,50], [250,200], [450,200]], dtype=np.int32)
pts3 = np.array([[150,300], [50,450], [250,450]], dtype=np.int32)
pts4 = np.array([[350,250], [450,350], [400,450],[300,450],
[250,350]], \
              dtype=np.int32)

cv2.polylines(img, [pts1], False, (255,0,0))
cv2.polylines(img, [pts2], False, (0,0,0), 10)
cv2.polylines(img, [pts3], True, (0,0,255), 10)
cv2.polylines(img, [pts4], True, (0,0,0))

cv2.imshow('polyline', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Polylines



❖ Circle, Ellipse

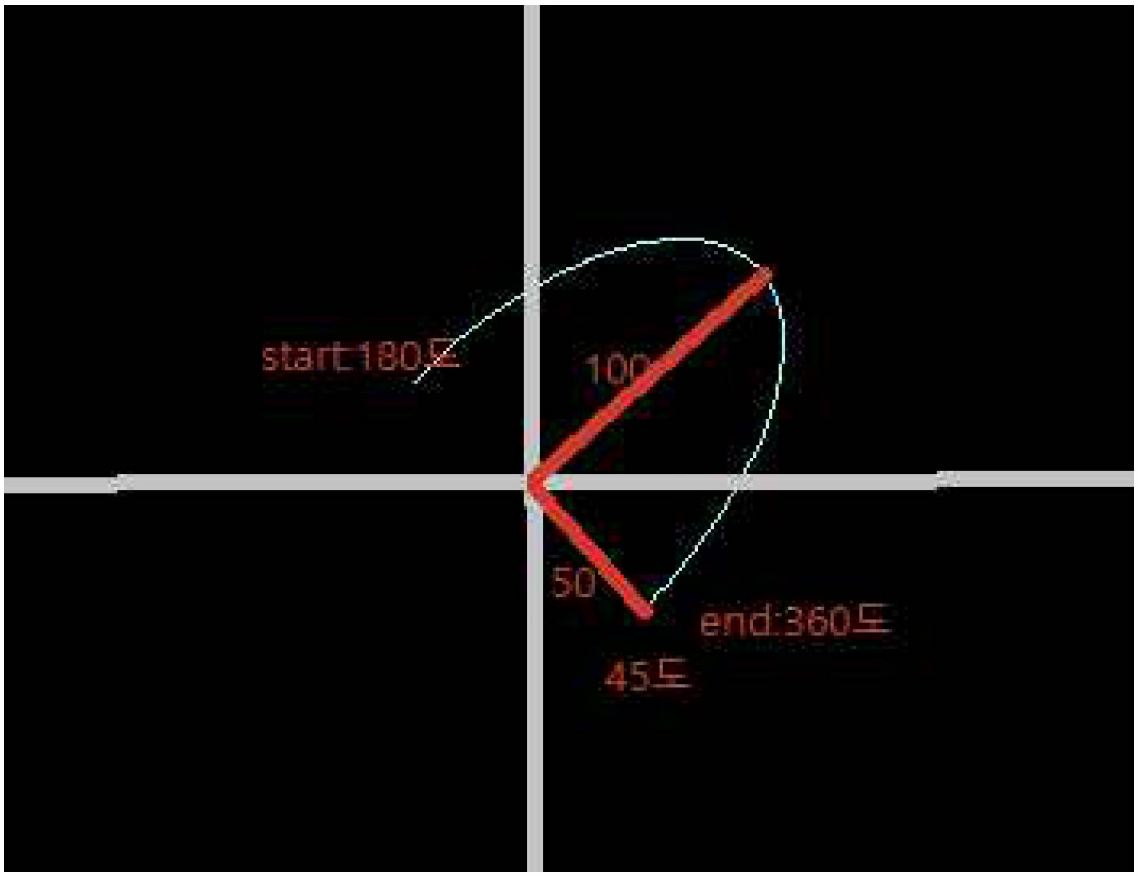
- cv2.circle(img, center, radius, color [, thickness, lineType]): Circle drawing function
  - img : Image to draw on
  - center: Original point, tuple (x, y)
  - radius
  - color : tuple (Blue, Green, Red)
  - thickness : Line Thickness, -1: fill
  - lineType : Line Type, cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA

- cv2.ellipse(img, center, axes, angle, from, to, color[, thickness, lineType])
  - img: Image to draw on
  - center : Original point, tuple (x, y)
  - axes: Length of base-axis
  - angle: Angle of base-axis
  - from, to: Angle of beginning and end point of the line
  - color: tuple (Blue, Green, Red)
  - thickness : Line Thickness, -1: fill
  - lineType: Line Type, cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA

❖ Ellipse Example

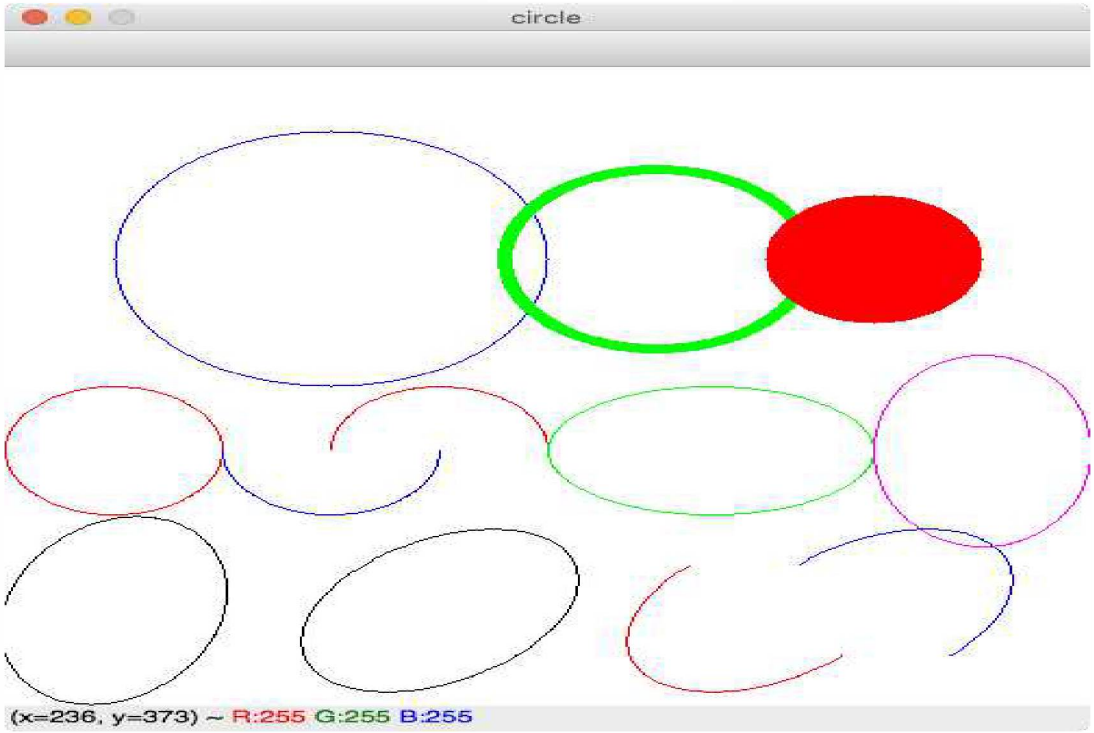
- cv2.ellipse(img, center, axes, angle, from, to, color[, thickness, lineType])

```
ellipse(img, (256,256), (50,100), 45, 180,360, 1)
```



❖ Circle, Ellipse

```
import cv2
img = cv2.imread('./img/blank_500.jpg')

cv2.circle(img, (150, 150), 100, (255,0,0))
cv2.circle(img, (300, 150), 70, (0,255,0), 5)
cv2.circle(img, (400, 150), 50, (0,0,255), -1)

cv2.ellipse(img, (50, 300), (50, 50), 0, 0, 360, (0,0,255))
cv2.ellipse(img, (150, 300), (50, 50), 0, 0, 180, (255,0,0))
cv2.ellipse(img, (200, 300), (50, 50), 0, 181, 360, (0,0,255))
cv2.ellipse(img, (325, 300), (75, 50), 0, 0, 360, (0,255,0))
cv2.ellipse(img, (450, 300), (50, 75), 0, 0, 360, (255,0,255))
cv2.ellipse(img, (50, 425), (50, 75), 15, 0, 360, (0,0,0))
cv2.ellipse(img, (200, 425), (50, 75), 45, 0, 360, (0,0,0))
cv2.ellipse(img, (350, 425), (50, 75), 45, 0, 180, (0,0,255))
cv2.ellipse(img, (400, 425), (50, 75), 45, 181, 360, (255,0,0))

cv2.imshow('circle', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



❖ Text

- cv2.putText(img, text, point, fontFace, fontSize, color [, thickness, lineType])
- img: Image to show letters
- text: String to show
- point: Coordination to show letter (Left bottom base point), tuple (x,y),
- fontFace : Font
  - cv2.FONT_HERSHEY_PLAIN: San sarif small font
  - cv2.FONT_HERSHEY_SIMPLEX: San sarif regular fond
  - cv2.FONT_HERSHEY_DUPLEX: San sarif bold font
  - cv2.FONT_HERSHEY_COMPLEX_SMALL: Sarif small font
  - cv2.FONT_HERSHEY_COMPLEX: Sarif regular font
  - cv2.FONT_HERSHEY_TRIPLEX: Sarif bold font
  - cv2.FONT_HERSHEY_SCRIPT_SIMPLEX: Cursive San Sarif Font
  - cv2.FONT_HERSHEY_SCRIPT_COMPLEX: Cursive San Sarif font
  - cv2.FONT_ITALIC: Italic Flag
- fontSize : Font Size
- color : tuple (Blue, Green, Red)
- thickness : Line Thickness, -1: Fill
- lineType : Line Type, cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA

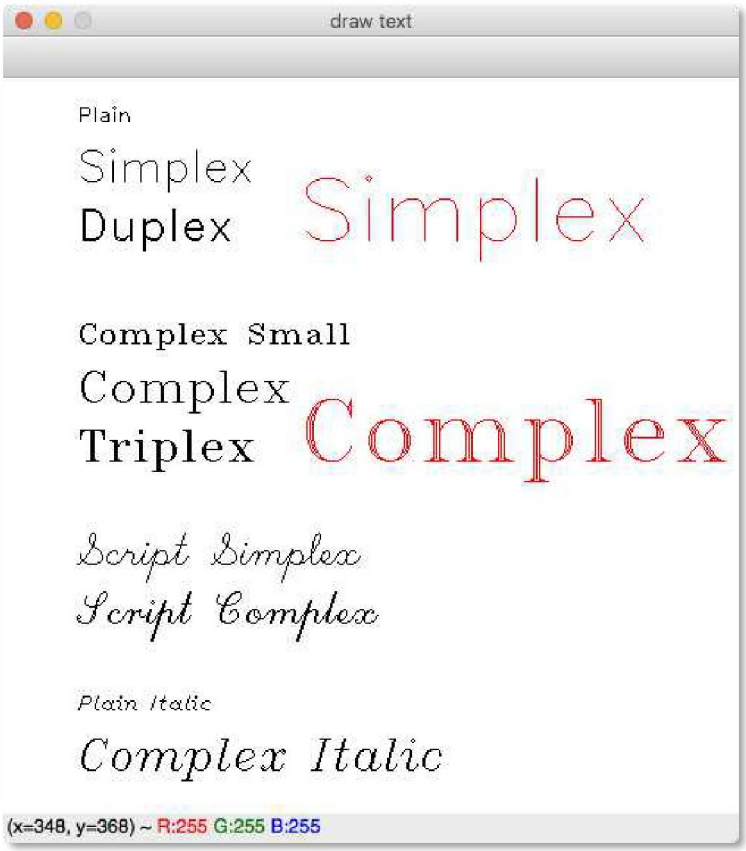
❖ Text (1/2)

```
import cv2

img = cv2.imread('./img/blank_500.jpg')

cv2.putText(img, "Plain", (50, 30), cv2.FONT_HERSHEY_PLAIN, 1, (0, 0,0))
cv2.putText(img, "Simplex", (50, 70), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,0))
cv2.putText(img, "Duplex", (50, 110), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0,0))
cv2.putText(img, "Simplex", (200, 110), cv2.FONT_HERSHEY_SIMPLEX, 2, \

      (0,0,250))
cv2.putText(img, "Complex Small", (50, 180), cv2.FONT_HERSHEY_COMPLEX_SMALL,\
                                                                    1, (0, 0,0))
```

❖ Text (2/2)

```
cv2.putText(img, "Complex", (50, 220), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0,0))
cv2.putText(img, "Triplex", (50, 260), cv2.FONT_HERSHEY_TRIPLEX, 1, (0, 0,0))
cv2.putText(img, "Complex", (200, 260), cv2.FONT_HERSHEY_TRIPLEX, 2,\
                                                                  (0,0,255))

cv2.putText(img, "Script Simplex", (50, 330), \
                                  cv2.FONT_HERSHEY_SCRIPT_SIMPLEX, 1, (0, 0,0))
cv2.putText(img, "Script Complex", (50, 370), \
                                  cv2.FONT_HERSHEY_SCRIPT_COMPLEX, 1, (0, 0,0))

cv2.putText(img, "Plain Italic", (50, 430), cv2.FONT_HERSHEY_PLAIN | \
                                                  cv2.FONT_ITALIC, 1, (0, 0,0))
cv2.putText(img, "Complex Italic", (50, 470), cv2.FONT_HERSHEY_COMPLEX | \
                                                      cv2.FONT_ITALIC, 1, (0,
0,0))

cv2.imshow('draw text', img)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Text



Basic Interface

1. Image and Video I/O
2. Drawing
3. **Window Management**
4. Event Handling

❖ Window Management

- cv2.namedWindow(title [, option]) : Open widow with title
  - title: Window name, show in subject line
  - option: Window Option, Start with "cv2.WINDOW_"
    - cv2.WINDOW_NORMAL : Temporary size, able to adjust window size
    - cv2.WINDOW_AUTOSIZE: size same as window, unable to readjust window size
- cv2.moveWindow(title, x, y) : Move window position
  - title: Window title to change position
  - x, y: Window position to move
- cv2.resizeWindow(title, width, height): Change window size
  - title: Name of window to change size
  - width, height: Width and height of window to change
- cv2.destroyWindow(title) : Close window
  - title : Title of window to close
- cv2.destroyAllWindows() : Close all open window

```
import cv2

file_path = './img/girl.jpg'
img = cv2.imread(file_path)
img_gray = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
cv2.namedWindow('origin', cv2.WINDOW_AUTOSIZE)
cv2.namedWindow('gray', cv2.WINDOW_NORMAL)

cv2.imshow('origin', img)
cv2.imshow('gray', img_gray)
cv2.moveWindow('origin', 0, 0)
cv2.moveWindow('gray', 100, 100)
cv2.waitKey(0)
cv2.resizeWindow('origin', 200, 200)
cv2.resizeWindow('gray', 100, 100)
cv2.waitKey(0)
cv2.destroyWindow("gray")
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Window Management

Basic Interface

1. Image and Video I/O
2. Drawing
3. Window Management
4. **Event Handling**

❖ Keyboard Event

- val = cv2.waitKey([delay]): Keyboard input standby, move only activated window
  - delay=0 : Standby time, ms
    - delay <= 0 : Wait infinitely
  - val : code of pressed key
    - -1 : time out
- Compare string of key code, Use with python internalized function
  - ord(c) : Turnaround mapped code to c string
    - ex : ord('a') #97
  - chr(code) : Turnaround mapped string to code
    - ex : chr(97) # 'a'
  - cv2.waitKey() == ord('a')
- ASCII(8bit) code overflow
  - Turnaround 32 bit integer at some 64bit environment, Need to initialize bit more than 8 bit
  - key = cv2.waitKey(0) & 0xFF

```
import cv2
img_file = "./img/girl.jpg"
img = cv2.imread(img_file)
title = 'IMG'                   # Window Title
x, y = 100, 100                 # Initial Coordination
while True:
    cv2.imshow(title, img)
    cv2.moveWindow(title, x, y)
    key = cv2.waitKey(0) & 0xFF # Wait infinitely for keyboard input, 8 bit
mask work
    print(key, chr(key))        # Keyboard input value, print string value
    if key == ord('h'):         # Move left with 'h'
        x -= 10
    elif key == ord('j'):       # Move down with 'j'
        y += 10
    elif key == ord('k'):       # Move up with 'k'
        y -= 10
    elif key == ord('l'):       # Move right with 'l'
        x += 10
    elif key == ord('q') or key == 27: # End if 'q' or 'esc'
        break
        cv2.destroyAllWindows()
    cv2.moveWindow(title, x, y )
```

❖ Mouse Event

- cv2.setMouseCallback(win_name, onMouse [, param]): Register onMouse function
  - win_name: Window title to register event
  - onMouse: Callback function object declared in advance for the event
  - param: Parameter to be delivered to onMouse function as needed
- MouseCallback(event, x, y, flags, param): Callback function declaration
- event: Mouse event type, All 12 types integer beginning with cv2.EVENT_
  - x, y: Mouse coordination
  - flags: Extra state happened with mouse move, All 6 types integer beginning with cv2.EVENT_FLAG_
  - param : cv2.setMouseCallback() Parameter delivered from function

```
def onMouse(event, x, y, flags, param):
  #Draft work fit to the mouse event.
  pass

cv2.setMouseCallback('title', onMouse)
```

❖ Mouse Event and Flags Type

- cv2.EVENT_*

```
import cv2

events = [i for i in dir(cv2) if 'EVENT_' in i]
print( len(events) ,"events" )
print( events )
```

```
18 events
['EVENT_FLAG_ALTKEY', 'EVENT_FLAG_CTRLKEY', 'EVENT_FLAG_LBUTTON',
'EVENT_FLAG_MBUTTON', 'EVENT_FLAG_RBUTTON', 'EVENT_FLAG_SHIFTKEY',
'EVENT_LBUTTONDBLCLK', 'EVENT_LBUTTONDOWN', 'EVENT_LBUTTONUP',
'EVENT_MBUTTONDBLCLK', 'EVENT_MBUTTONDOWN', 'EVENT_MBUTTONUP',
'EVENT_MOUSEHWHEEL', 'EVENT_MOUSEMOVE', 'EVENT_MOUSEWHEEL',
'EVENT_RBUTTONDBLCLK', 'EVENT_RBUTTONDOWN', 'EVENT_RBUTTONUP']
```

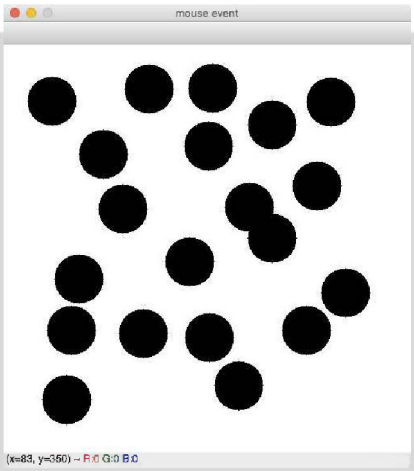
❖ Mouse Event Type

- cv2.EVENT_MOSEMOVE : Move mouse
- cv2.EVENT_LBUTTONDOWN : Push left button
- cv2.EVENT_RBUTTONDOWN : Push right button
- cv2.EVENT_MBUTTONDOWN : Push center button
- cv2.EVENT_LBUTTONUP : Release left button
- cv2.EVENT_RBUTTONUP : Release right button
- cv2.EVENT_MBUTTONUP : Release center button
- cv2.EVENT_LBUTTONDBLCLK: Left button double click
- cv2.EVENT_RBUTTONDBLCLK: Right button double click
- cv2.EVENT_MBUTTONDBLCLK: Center button double click
- cv2.EVENT_MOUSEWHEEL: Scroll Wheel
- cv2.EVENT_MOUSEHWHEEL: Scroll wheel horizontally

❖ Mouse Event Flag Type

- cv2.EVENT_FLAG_LBUTTON (1): Push left button
- cv2.EVENT_FLAG_RBUTTON (2): Push right button
- cv2.EVENT_FLAG_MBUTTON (4): Push center button
- cv2.EVENT_FLAG_CTRLKEY (8): Push Ctrl
- cv2.EVENT_FLAG_SHIFTKEY (16): Push Shift
- cv2.EVENT_FLAG_ALTKEY (32): Push Alt

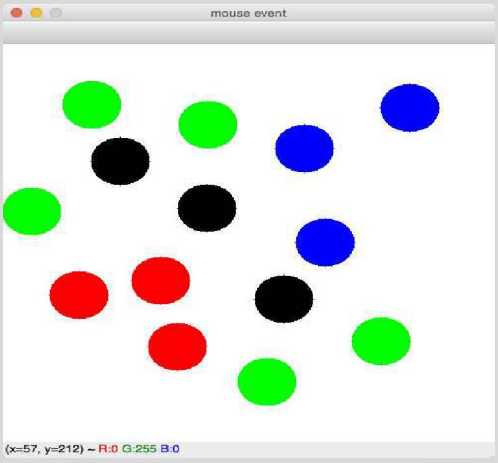
❖ Mouse Event




```
import cv2

title = 'mouse event'
img = cv2.imread('./img/blank_500.jpg')
cv2.imshow(title, img)

def onMouse(event, x, y, flags, param):
    print(event, x, y, )
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.circle(img, (x,y), 30, (0,0,0), -1)
        cv2.imshow(title, img)
cv2.setMouseCallback(title, onMouse)
while True:
    if cv2.waitKey(0) & 0xFF == 27:     # End with esc
        break
cv2.destroyAllWindows()
```

❖ Mouse Event Flag Example

```
import cv2
title = 'mouse event'
img = cv2.imread('./img/blank_500.jpg')
cv2.imshow(title, img)
colors = {'black':(0,0,0),'red': (0,0,255),'blue':(255,0,0), 'green':
(0,255,0) }

def onMouse(event, x, y, flags, param):
    print(event, x, y, flags)
    color = colors['black']
    if event == cv2.EVENT_LBUTTONDOWN:
        if flags & cv2.EVENT_FLAG_CTRLKEY and flags & cv2.EVENT_FLAG_SHIFTKEY
:
            color = colors['green']
        elif flags & cv2.EVENT_FLAG_SHIFTKEY :
            color = colors['blue']
        elif flags & cv2.EVENT_FLAG_CTRLKEY :
            color = colors['red']
        cv2.circle(img, (x,y), 30, color, -1)
        cv2.imshow(title, img)
cv2.setMouseCallback(title, onMouse)
while True:
    if cv2.waitKey(0) & 0xFF == 27:
        break
cv2.destroyAllWindows()
```



❖ Mouse Event

- Draw with Mouse and Key Event (1/3)

```
import cv2
import numpy as np

drawing = False # true if mouse is pressed
color = (0,0,255) # red as default
ix,iy = -1,-1

def draw_circle(event,x,y,flags,param):
    global ix,iy,drawing,mode
    if event == cv2.EVENT_LBUTTONDOWN :
        drawing = True
        ix,iy = x,y
        if event == cv2.EVENT_LBUTTONDOWN:
            mode = True
        else:
            mode = False
    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing:
            if flags == cv2.EVENT_FLAG_LBUTTON:
                cv2.circle(img, (x,y), 5, color, -1)
            elif flags == (cv2.EVENT_FLAG_SHIFTKEY \
                                                    +
cv2.EVENT_FLAG_LBUTTON):
                cv2.circle(img,(x,y),5, (0,0,0),-1)
    elif event == cv2.EVENT_LBUTTONUP or event == cv2.EVENT_RBUTTONUP:
        drawing = False
img = np.zeros((512,512,3), np.uint8)
cv2.namedWindow('paint')
cv2.setMouseCallback('paint',draw_circle)

while(1):
    cv2.imshow('paint',img)
    k = cv2.waitKey(1) & 0xFF
    if k == ord('r'):
        color = (0,0,255)
    elif k == ord('g'):
        color = (0,255,0)
    elif k == ord('b'):
        color = (255,0,0)
    elif k == 27:
        break
cv2.destroyAllWindows()
```

❖ Mouse Event

- Draw with Mouse and Key Event

❖ Trackbar

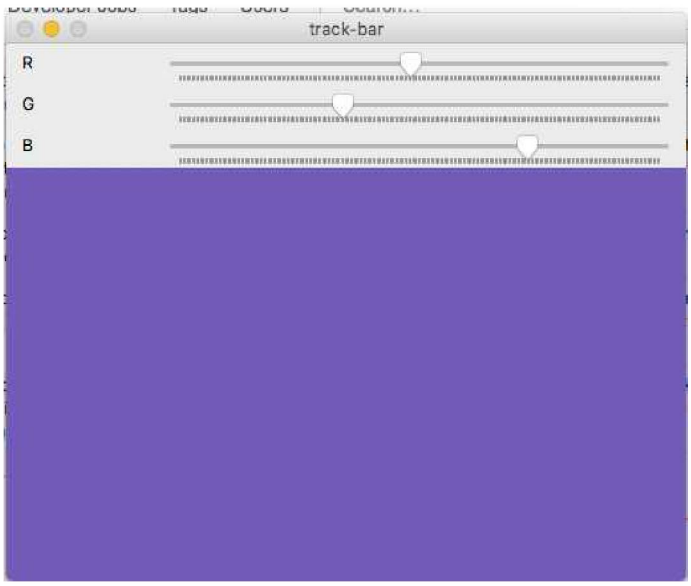
- cv2.createTrackbar(name, win_name, value, count, onChange): Generate Track bar
  - name: Name of track bar
  - win_name: Name of window to display trackbar
  - value: Initial value to have when trackbar first appear, b/w 0 ~ count
  - count: # of scale in trackbar, max value of trackbar
  - onChange : TrackbarCallback, Trackbar event handler function
- TrackbarCallback(value): Trackbar event callback function
  - value: Value of position trackbar moved

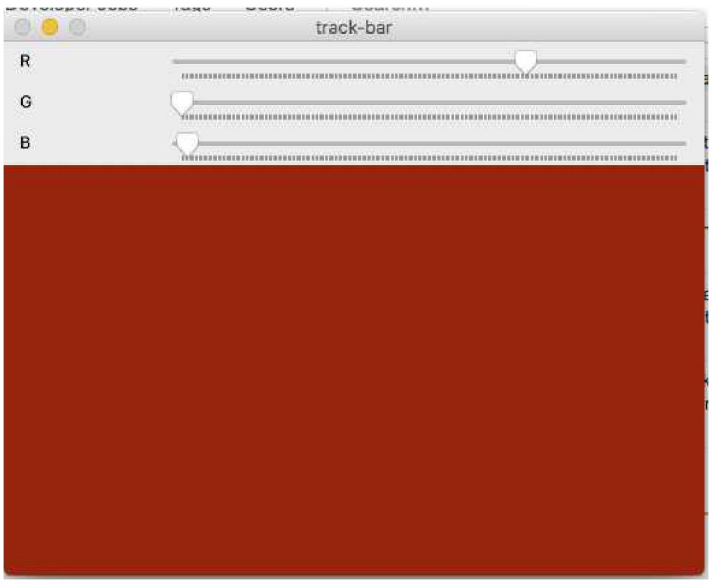
- pos = cv2.getTrackbarPos(name, win_name)
  - name: Name of trackbar to search
  - win_name: Name of window with trackbar
  - pos: Trackbar position value

❖ Trackbar

- Adjust color with trackbar

```
import cv2 , numpy as np

win_name = 'Trackbar'
img = cv2.imread('./img/blank_500.jpg')
cv2.imshow(win_name,img)

def onChange(x):
      r = cv2.getTrackbarPos('R',win_name)
    g = cv2.getTrackbarPos('G',win_name)
    b = cv2.getTrackbarPos('B',win_name)
    print(x, r, g, b)
    img[:] = [b,g,r]
    cv2.imshow(win_name, img)
cv2.createTrackbar('R', win_name, 255, 255, onChange)
cv2.createTrackbar('G', win_name, 255, 255, onChange)
cv2.createTrackbar('B', win_name, 255, 255, onChange)
while True:
    if cv2.waitKey(1) & 0xFF == 27:
        break
cv2.destroyAllWindows()
```



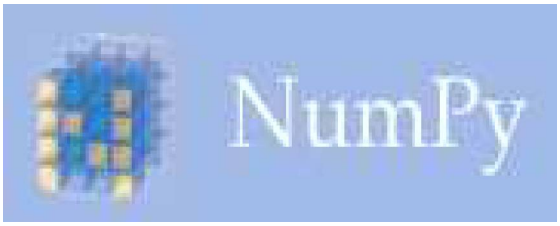


Numpy and Matplotlib

1. **Numpy**
2. Matplolib

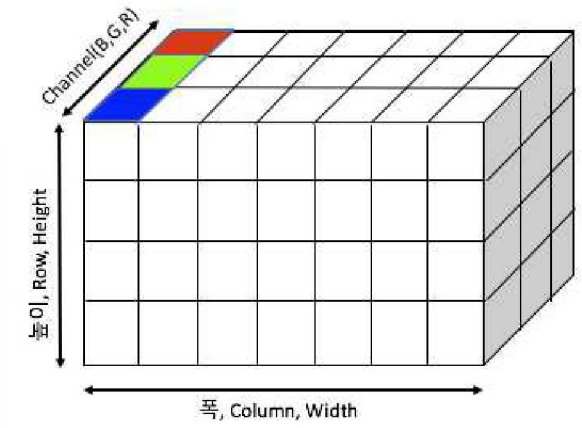
❖ Numpy

- http://www.numpy.org/
- https://docs.scipy.org/doc/numpy/genindex.html
- Fundamental package for scientific computing with Python
  - Strong N-th array object
  - Precise broadcasting function
  - C/C++, Fortran Combined tool
  - Useful linear algebra, Fourier transform, random number generating function
- OpenCV-Python ver 2.3+
  - Previously it used self internal data structure
  - Now it uses NumPy as image data structure
  - Change to cv2 package
  - import cv2
  - import numpy as np



❖ NumPy and Image

- ndarray
  - ndim: n dimension (axes)
  - shape: Size of each dimension, tuple
  - size : # of all item, Multiple of shape
  - dtype : data type of item
  - itemsize : Bite size of each item
  - data : Buffer having item, Not use in real




```
import numpy as np
import cv2
img = cv2.imread('../img/blank_500.jpg')
type(img)# <class 'numpy.ndarray'>
print(img.ndim)  #3
print(img.shape) #(500,500,3,)
print(img.size) #750000
print(img.dtype) # 'uint8'
print(a.itemsize) # 1, Size of each item
```

❖ Numpy

- Producing
  - Produce as value
    - array()
  - Generate as initial value
    - empty(), zeros(), ones(), full()
  - Generate as exiting array
    - empty_like(), zeros_like(), ones_like(), full_like()
  - Generate as consecutive value
    - arange()
  - Generate as random number
    - random.rand(), random.randn()
- Generate as value
  - numpy.array(list [, dtype]) : Generate NumPy array with designated values
    - list: Python list objects with value to be used in array generation
    - dtype: Data type, if omitted, it is auto decided by the value
      - int8, int16, int32, int64 : Integer with sign
      - uint8, uint16, uint32, uint64 : Integer without sign
      - float16, float32, float64, float128 : Real-number with floating decimal
      - complex64, complex128, complex256 : Complex number with floating decimal
      - bool : Boolian
- Generate as value

```
import numpy as np
a = np.array([1,2,3,4])
b = np.array([[1,2,3,4],
              [5,6,7,8]])
c = np.array([1,2,3.14, 4])
d = np.array([1,2,3,4], dtype=np.float64)

print(a, a.dtype, a.shape) # [1 2 3 4] int64 (4,)
print(b, b.dtype, b.shape) # [[1 2 3 4] [5 6 7 8]] int64 (2, 4)
print(c, c.dtype, c.shape) # [ 1. 2. 3.14 4.] float64 (4,)
print(d, d.dtype, d.shape) # [ 1.  2.  3.  4.] float64 (4,)
```

❖ Numpy

- Generate as size and initial value
  - np.empty (tuple [, dtype])
    - All factors not initialized, tuple size
    - np.empty((2,3))
    - np.empty((2,3), dtype=np.int16)
  - np.zeros(tuple [, dtype])
    - All factors are 0, tuple size
    - np.zeros((3,4))
    - np.zeros((2,3,4) , dtype=np.int16)
  - np.ones (tuple [, dtype])
    - All factors are 1, tuple size
    - np.ones((3,4))
    - np.ones((2,3,4) , dtype=np.int16)
  - np.full(shape, fill_value [, dtype])
    - Generate array initialized with fill_value
- Generate as size

```
import numpy as np

a = np.zeros( (2,3))
b = np.zeros( (2,3,4), dtype=np.uint8)

c = np.ones( (2,3), dtype=np.float32)
d = np.empty( (2,3))  #not initialized, garbage value
e = np.empty( (2,3), dtype=np.int16)
f = np.fill(255)

print(a, a.dtype, a.shape)
print(b, b.dtype, b.shape)
print(c, c.dtype, c.shape)
print(d, d.dtype, d.shape)
print(f, f.dtype, f.shape)
```

❖ Numpy

- Generate as exiting array
  - np.empty_like(array [, dtype])
    - Generate non-initialized dtype and same shape array
  - zeros_like(array [, dtype])
    - Generate 0(zero) initialized dtype and same shape array
  - ones_like(array [, dtype])
    - Generate 1 initialized dtype and same shape array
  - ones_like(array, fill_value [, dtype])
    - Generate fill_value initialized dtype and same shape array
- Generate as exiting array

```
import numpy as np
img = cv2.imread('../img/girl.jpg')

a = np.empty_like(img)
b = np.zeros_like(img)
c = np.ones_like(img)
d = np.full_like(img, 255)

print(a, a.dtype, a.shape)
print(b, b.dtype, b.shape)
print(c, c.dtype, c.shape)
print(d, d.dtype, d.shape)
```

- Generate as Sequence and Random Number
  - numpy.arange([start=0,] stop [, step=1, dtype=float64])
  - start : start value
  - stop : stop value, Number included in the scope are stop-1
  - step : Increasing value
  - dtype : data type
- numpy.random.rand([d0 [, d1 [..., dn]]]) :
  - d0, d1..dn : shape, if omitted, return a random number
  - Random number b/w 0 and 1
- numpy.random.randn([d0 [, d1 [..., dn]]]) :

- Mean=0, Distribution=1, Random number following standard distribution

❖ Numpy

- Generate as Sequence and Random Number

```
import numpy as np

a = np.arange(5)
b = np.arange(5.0)
c = np.arange(3,9,2)

e = np.random.rand()
f = np.random.randn()
g = np.random.rand(2,3)
h = np.random.randn(2,3)
```

- Change Data Type
  - a.astype('typename') , a.astype(np.dtype)
  - np.uintXX(), np.intXX(), np.floatXX(), np.complexXX()

```
a = np.arange(10)
print(a, a.dtype)

b = a.astype('float32')
print(b, b.dtype)

c = np.uint8(b)
print(c, c.dtype)

d = c.astype(np.float64)
print(d, d.dtype)
```

❖ Numpy

- Change Dimension
  - np.reshape(array, newshape [, order])
  - arr.reshape(newshape [, order])
    - array : Array the objects to rearrange
    - newshape : Size of new array, tuple
      - -1 : The column size is not decided
    - order : {'C', 'F', 'A'} C:C-like, F:Fotran-like, A:F or C
  - np.ravel(array)
    - Change to 1 dimension
  - arr.T : Transposed array
- Change Dimension

```
a=np.arange(6) #[0 1 2 3 4 5] (6,)
b=a.reshape(2,3)  #[[0 1 2] [3 4 5]] (2, 3)
c= np.arange(24).reshape(2,3,4)
#[[[ 0  1  2  3]…
 [[12 13 14 15]…
  [20 21 22 23]]] (2, 3, 4)

d = np.arange(100).reshape(-1, 5)
# [[0 1 2 3 4]
   [5 6 7 8 9]
    ...
   [95 96 97 98 99]]

e = np.arange(100).reshape(2, -1)
#[[0 1 2 3 ... 49]
  [50 51 52 ...99]]

f = np.ravel(c) #[0 1 2 3 4 5 6 7 8 .... 21 22 23]
```

❖ Numpy

- Broadcasting Calculation
  - Scaler and Calculation
  - Array and calculation, two arrays should have same shape

```
a = np.array([10,20,30,40])
b = np.arange(1,5)

c = a + b
d = a - b
e = a * b
f = a / b

g = a + 5
h = a * 2
i = b ** 2
```

```
[10 20 30 40]
[1 2 3 4]

[11 22 33 44]
[ 9 18 27 36]
[ 10  40  90 160]
[ 10.  10.  10.  10.]

[15 25 35 45]
[20 40 60 80]
[ 1  4  9 16]
```

- Matrix
  - Matrix Addition : +
  - Matrix Multiply: np.dot(a, b), a.dot(b)
  - Transposed Matrix: a.T , a.transpose()

```
a = np.arange(6).reshape(2,3)     #[[0 1 2]
                                  # [3 4 5]]
b = np.array([1,2,3])             #[1 2 3]
c = a + b                         #[[1 3 5]
                                  #[4 6 8]]
e = np.dot(a , b)                 #[ 8 26]
f = a.T                           #[[0 3]
                                  #[1 4]
                                  #[2 5]]
a.dot(b)                          #[ 8 26]
```

❖ Numpy

- Indexing/Slicing
  - 1 Dimension
    - a[0] : Certain Parameter
    - a[1 : 5] : 1~4 Parameter
    - a[5:] : 5 to end
    - a[:5] : Beginning to 4
    - a[:] : All parameter
  - 2 Dimension
    - a[2, 3] : 2 column 3 row
    - a[2:4, 3] : 2~3 column 3 row
    - a[2, 1:4] : 2 column 1~3 row
    - a[2, :] :2 column and all row
    - a[:, 2] : All column and 2 row
    - a[1:3, :] : 1~2 column and all row
- Indexing/Slicing

```
a = np.arange(10) # [0 1 2 3 4 5 6 7 8 9]

print(a[5])  # 5
print(a[1:5])# [1 2 3 4]
print(a[5:]) # [5 6 7 8 9]
print(a[:5]) # [0 1 2 3 4]
print(a[:]). # [0 1 2 3 4 5 6 7 8 9]

b = np.arange(16).reshape(4,-1)

print(b)
print(b[2, 2])
print(b[2, :])
print(b[:, 2])
print(b[1:3, :])
```

❖ Numpy

- Fancy Indexing
  - Generate data fit to condition or initialize
  - Condition calculation: > , >=, <, <=, ==, !=
  - Bool type (True/False)
  - a[a > 4]
  - a[a > 4] = 0
- Fency Indexing

```
a = np.arange(10) #[0 1 2 3 4 5 6 7 8 9]
b = a > 4
print(b) #[False False False False False  True  True  True  True  True]
print(a[b]) #[5 6 7 8 9]
print(a[a>4]) #[5 6 7 8 9]

a[a>4] = 0
print(a) # [0 1 2 3 4 0 0 0 0 0]

names = np.array(['John', 'Tom', 'Lee', 'Tom'])
scores = np.random.rand(4,4) * 50 + 50
print(scores)
print(names=='Tom')
print(scores[names=='Tom',  :]) #Tom의 점수만 추출
```

- Merge
  - numpy.hstack(arrays) : Merge arrays horizontally
  - numpy.vstack(arrays) : Merge arrays vertically
  - numpy.concatenate(arrays, axis=0) : Merge arrays as designated axis
  - numpy.stack(arrays, axis=0) : Merge arrays as new axis
    - arrays : Merge objects, tuple

❖ Numpy

- Merge

```
>>> a = np.arange(4).reshape(2,2)
>>> a
array([[0, 1],
       [2, 3]])
>>> b = np.arange(10, 14).reshape(2,2)
>>> b
array([[10, 11],
       [12, 13]])
>>> np.vstack( (a,b) )
array([[ 0,  1],
       [ 2,  3],
       [10, 11],
       [12, 13]])
>>> np.hstack( (a,b) )
array([[ 0,  1, 10, 11],
      [ 2,  3, 12, 13]])
>>> np.concatenate((a,b), 0)
array([[ 0,  1],
      [ 2,  3],
      [10, 11],
      [12, 13]])
>>> np.concatenate((a,b), 1)
array([[ 0,  1, 10, 11],
      [ 2,  3, 12, 13]])
>>> a = np.arange(12).reshape(4,3)
>>> b = np.arange(10, 130, 10).reshape(4,3)
>>> a
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
>>> b
array([[ 10,  20,  30],
       [ 40,  50,  60],
       [ 70,  80,  90],
       [100, 110, 120]])
>>> c = np.stack( (a,b), 0)
>>> c.shape
(2, 4, 3)
>>> c
array([[[  0,   1,   2],
        [  3,   4,   5],
        [  6,   7,   8],
        [  9,  10,  11]],
       [[ 10,  20,  30],
        [ 40,  50,  60],
        [ 70,  80,  90],
        [100, 110, 120]]])
```

❖ Numpy

- splitting
  - np.hsplit(array, indice) = np.split(axis=1)
  - np.vsplit(array, indice) = np.split(axis=0)
  - np.split(array, indice, axis=0)

```
a = np.arange(12) # [ 0  1  2  3  4  5  6  7  8  9 10 11]

b = np.hsplit(a, 3)
# [array([0, 1, 2, 3]), array([4, 5, 6, 7]), array([ 8,  9, 10, 11])]

c = np.hsplit(a, (3,6))
# [array([0, 1, 2]), array([3, 4, 5]), array([ 6,  7,  8,  9, 10, 11])]
```

❖ Numpy Search

- numpy.where(condition [, t, f]) : Finding items fit to condition
  - return : Tuple, array filled with index or changed value fit to the search condition
  - condition : Condition that will be used in search
  - t, f : Value or array set according to condition, if it is array, same shame as condition
    - t : Value or array fit to condition
    - f : Value or array unfit to condition
- numpy.nonzero(array) : Turnaround index that is non-0 among array parameter, tuple
- numpy.all(array [, axis]) : Search true for all array parameter
  - array : Array of search subject
  - axis : Axis to search, if omitted, search all axis, if defined, turnaround result as the number of axis
- numpy.any(array [, axis]) : Search for true in any array

Numpy

- Search

```
>>> a = np.arange(10, 20)
>>> a
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
>>> np.where(a > 15)
(array([6, 7, 8, 9]),)
>>> np.where(a > 15, 1, 0)
array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1])
>>> a
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
>>> np.where(a>15, 99, a)
array([10, 11, 12, 13, 14, 15, 99, 99, 99, 99])
>>> np.where(a>15, a, 0)
array([ 0,  0,  0,  0,  0,  0, 16, 17, 18, 19])
>>> a
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
>>> b = np.arange(12).reshape(3,4)
>>> b
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> coords = np.where(b>6)
>>> coords
(array([1, 2, 2, 2, 2]), array([3, 0, 1, 2, 3]))
>>> np.stack( (coords[0], coords[1]), -1)
array([[1, 3],
       [2, 0],
       [2, 1],
       [2, 2],
       [2, 3]])
>>> z = np.array([0,1,2,0,1,2])
>>> np.nonzero(z)
(array([1, 2, 4, 5]),)
```

Numpy

- Search

```
>>> a = np.arange(10)
>>> b = np.arange(10)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> b
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> a==b
array([ True,  True,  True,  True,  True,  True,  True,
True,  True,
        True])
>>> np.all(a==b)
True
>>> b[5] = -1
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> b
array([ 0,  1,  2,  3,  4, -1,  6,  7,  8,  9])
>>> np.all(a==b)
False
>>> np.where(a==b)
(array([0, 1, 2, 3, 4, 6, 7, 8, 9]),)
>>> np.where(a!=b)
(array([5]),)
>>>
```

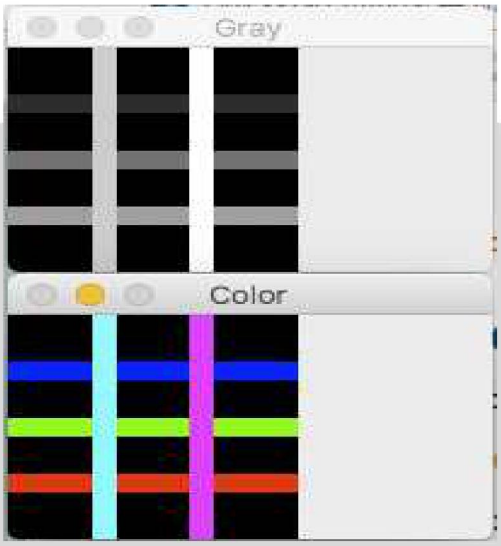
❖ Numpy

- Basic Statistics Function
  - numpy.sum(array [, axis]) : add array
  - numpy.mean(array [, axis]) : Average of array
  - numpy.amin(array [, axis]) : Min. value of array
  - numpy.min(array [, axis]) : Same as numpy.amin()
  - numpy.amax(array [, axis]) : Max value of array
  - numpy.max(array [, axis]) : Same as numpy.amax()
    - array : Arry of calculation objects
    - axis : Base axis to calculate, if it is not defined, all parameter is in subject to calculation

```
a = np.arange(12).reshape(-1,4)

print(np.sum(a)) #np.sum(-1)
print(np.sum(a, 0))
print(np.sum(a, 1))
print(np.sum(a, (0,)))

print(np.min(a))
print(np.min(a, 0))
print(np.min(a, 1))

print(np.max(a))
print(np.max(a, 0))
print(np.max(a, 1))

print(np.mean(a))
print(np.mean(a, 0))
print(np.mean(a, 1))
```

❖ Numpy

- Creating Gray Image
  - n x m
  - n : height, rows
  - m : width, columns
  - value : 0~255 (dtype=np.uint8)
- Creating Color Image
  - n x m x 3
  - n : height, rows
  - m : width, columns
  - 3 : color channels, Default = [Blue, Green, Red]
  - value : 0~255 (dtype=np.uint8)
- Numpy Image



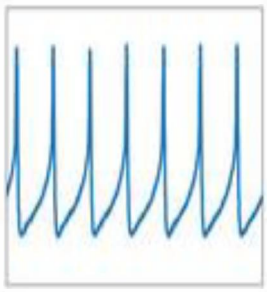
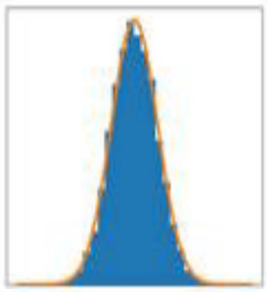
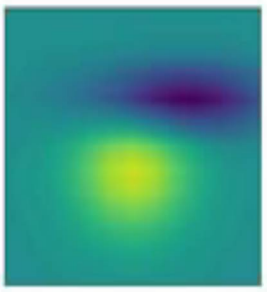

```
img = np.zeros( (120,120), dtype=np.uint8)
img[25:35, :] = 45
img[55:65, :] = 115
img[85:95, :] = 160
img[:, 35:45] = 205
img[:, 75:85] = 255
cv2.imshow('Gray', img)

img2 = np.zeros( (120,120,3), dtype=np.uint8)
img2[25:35, : ] = [255,0,0]
img2[55:65, : ] = [0,255,0]
img2[85:95, : ] = [0,0,255]
img2[ : , 35:45] = [255,255,0]
img2[ : , 75:85] = [255,0,255]
cv2.imshow('Color', img2)
```

Numpy and Matplotlib

1. Numpy
2. **Matplolib**

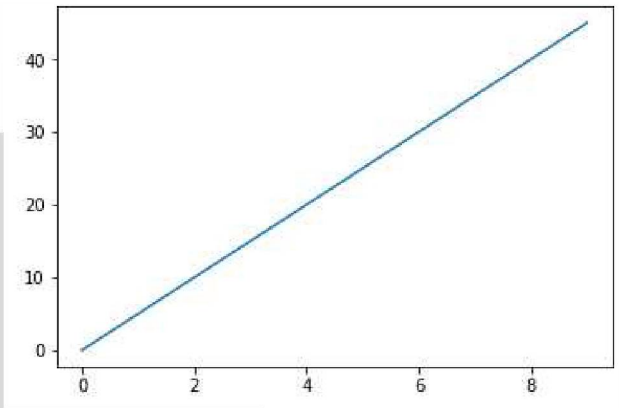
❖ Matplotlib

- https://matplotlib.org/
- Python 2D plotting library
- Mark various graph, map
- Show multiple image at one display
- Installation
  - pip install matplotlib 또는 pip3 install matplotlib
  - sudo apt-get install python3-matplotlib
- checking
  - import matplotlib
  - matplotlib.__version__
- import pyplot
  - import matplotlib.pyplot as plt


- plt.plt(x, y)
- plt.show()

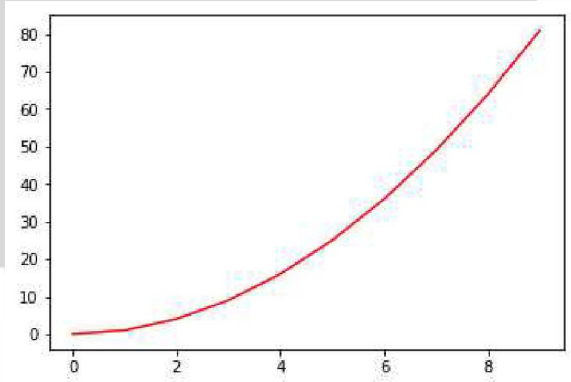
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x * 5
x, y
plt.plot(x,y)
plt.show()
```



❖ Matplotlib

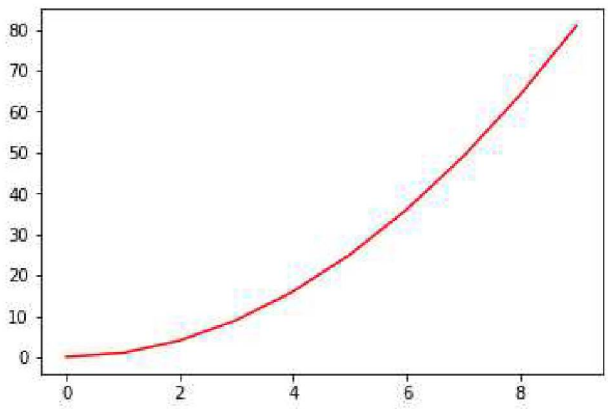
- color : plt.plot(x, y, 'r')
  - b : blue
  - g : green
  - r : red
  - c : cyan
  - m : magenta
  - y : yellow
  - k : black
  - w : white
- style : plt.plot(x, y, '--g')
-

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x**2
plt.plot(x,y,'r')
plt.show()
```



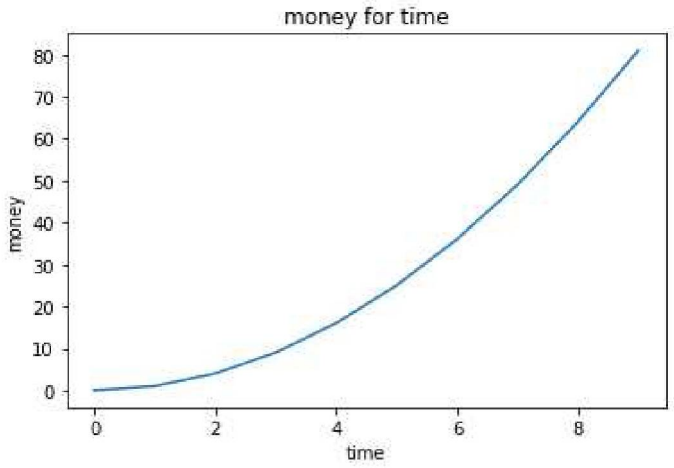
char	style	char	style
-	solid(*)	--	dashed
-.	dash-dot	:	dotted
.	point	,	pixel
o	circle	v	triangle dn
^	trangle up	<	traingle left
>	triangle right	1	small tri dn
2	small tri up	3	small tri left
4	small tri rigt	s	square
p	pentagon	*	star
h	hexagon	+	plus
D	diamond	x	ex

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x**2
plt.plot(x,y,'--g')
plt.show()
```



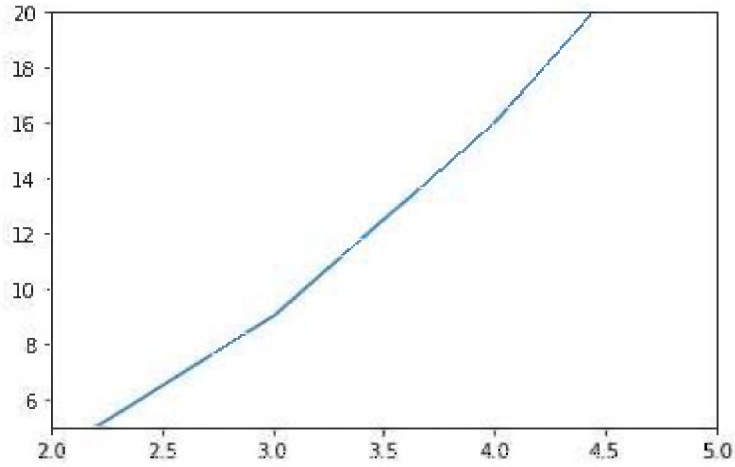
❖ Matplotlib

- label
  - plt.xlabel('text')
  - plt.ylabel('text')
  - plt.title('text')




```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x **2

plt.plot(x,y)
plt.xlabel('time')
plt.ylabel('money')
```

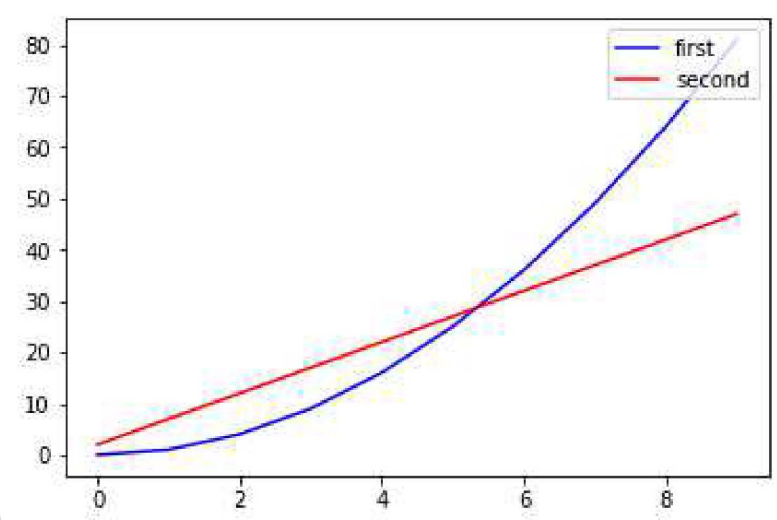
- limit
  - Mark only specific area of all data
  - plt.xlim(x,y)
  - plt.ylim(x,y)




```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x **2
plt.plot(x,y)
plt.xlim(2,5)
plt.ylim(5,20)
plt.show()
```

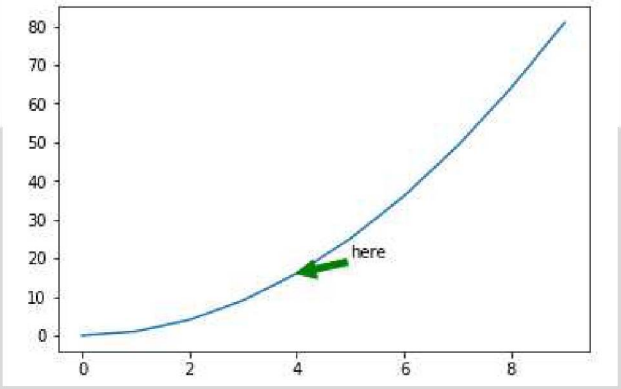
❖ Matplotlib

- legend
  - plt.plot(x, y, 'b', label='text')
  - plt.legend(loc='upper right')
    - upper left
    - upper center
    - upper right
    - center left
    - center
    - center right
    - lower left
    - lower center
    - lower right
    - best
    - right




```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x **2
y2 = x*5 + 2

plt.plot(x,y, 'b', label='first')
plt.plot(x,y2, 'r', label='second')
plt.legend(loc='upper right')
plt.show()
```

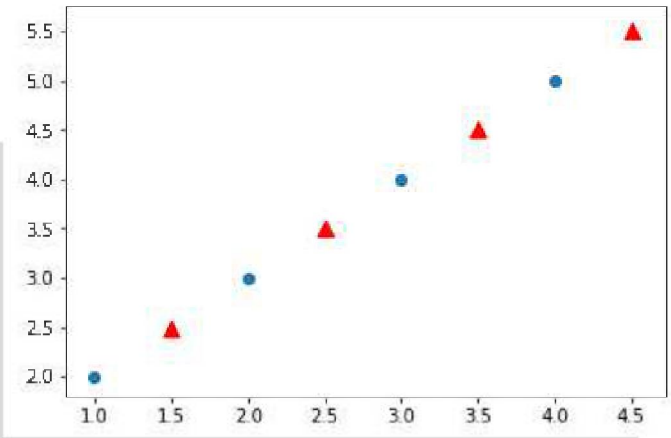
- annotate
  - plt.annotate('text')




```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)
y = x**2

plt.plot(x,y)
plt.annotate('here', xy=(4,16), xytext=(5,20),
arrowprops={'color':'green'})
plt.show()
```

❖ Matplotlib

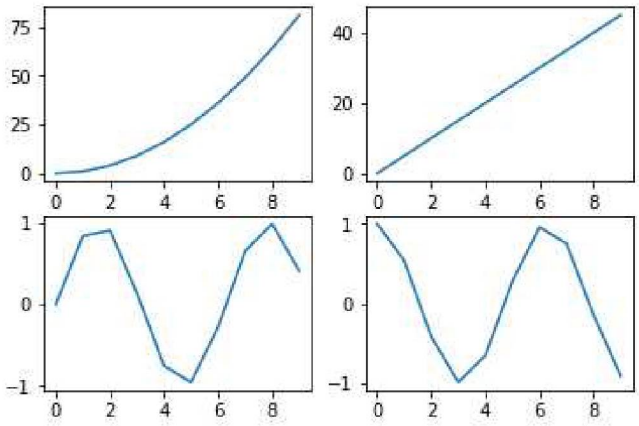
- scatter
  - plt.scatter(x, y, s=None, c=None, market=None)




```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(1,5)
y = np.arange(2, 6)
x2 = np.arange(1.5,5.5)
y2 = np.arange(2.5,6.5)
plt.scatter(x, y)
plt.scatter(x2, y2, s=80, c='r', marker='^')
plt.show()
```

- subplot
  - plt.subplot(r,c,idx)
  - plt.subplot(3digit)

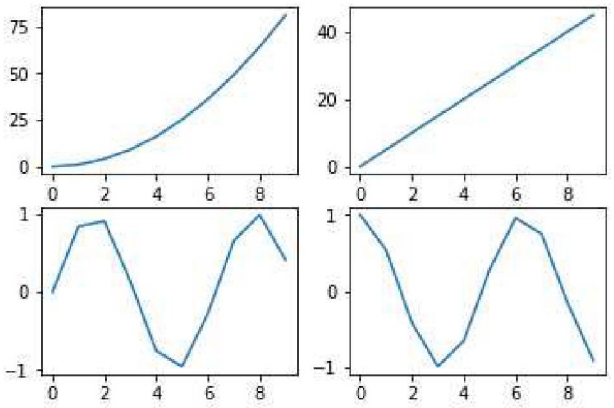
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)

plt.subplot(2,2,1)
plt.plot(x,x**2)
plt.subplot(2,2,2)
plt.plot(x,x*5)
plt.subplot(223)
plt.plot(x, np.sin(x))
plt.subplot(224)
plt.plot(x,np.cos(x))
plt.show()
```



❖ Matplotlib

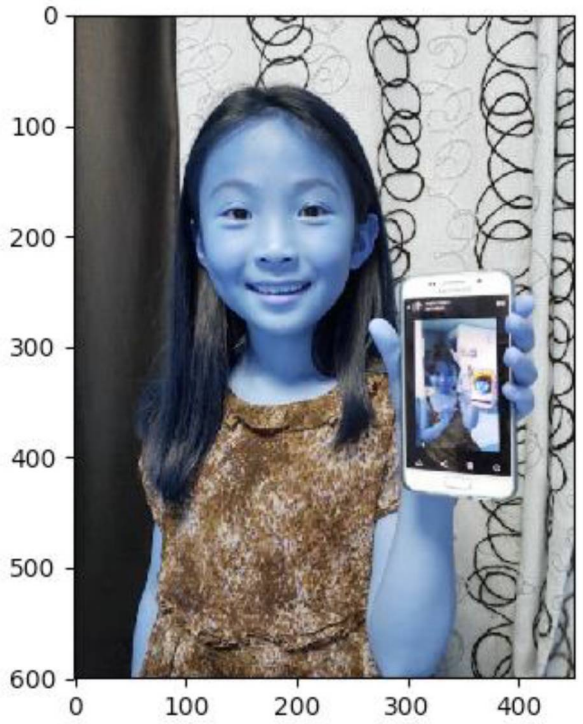
- subplot
  - plt.subplot(r,c,idx)
  - plt.subplot(3digit)

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(10)

plt.subplot(2,2,1)
plt.plot(x,x**2)
plt.subplot(2,2,2)
plt.plot(x,x*5)
plt.subplot(223)
plt.plot(x, np.sin(x))
plt.subplot(224)
plt.plot(x,np.cos(x))
plt.show()
```



- OpenCV Image
  - plt.imshow(img)
  - OpenCV = GBR, Plot = RGB

```
import cv2
from matplotlib import pyplot as
plt

img =
cv2.imread('../img/model.jpg')
plt.imshow(img)
#plt.xticks([])
#plt.yticks([])
plt.show()
```



❖ Matplotlib

- OpenCV Image
  - change BGR to RGB
    - numpy slicing
    - cv2.split(), cv2.merge()
    - cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

```
img1 = cv2.imread('../img/model.jpg')
img2 = cv2.imread('../img/model2.jpg')
img3 = cv2.imread('../img/model3.jpg')

img1_rgb = np.zeros(img1.shape, dtype=np.uint8)
img1_rgb[:,:,2],img1_rgb[:,:,1], img1_rgb[:,:,0] = img1[:,:,0], \

     img1[:,:,1], img1[:,:,2]
#img1_rgb = img1[:,:, ::-1] or img1_rgb = img1[:,:,(2,1,0)]

b,g,r = cv2.split(img2)
img2_rgb = cv2.merge([r,g,b])
```

- OpenCV Image
  - change BGR to RGB

❖ Matplotlib

- Histogram
  - plt.hist(img, bins, range)
  - https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.hist.html#matplotlib.pyplot.hist

```
import matplotlib.pylab as plt
import cv2

img = cv2.imread('../img/model2.jpg', cv2.IMREAD_GRAYSCALE)
d1 = img.ravel()

plt.subplot(1,2,1)
plt.title('img')
plt.imshow(img, cmap='gray')
plt.subplot(122)
plt.title('hist')
plt.hist(d1, 255)
plt.show()
```

- Histogram

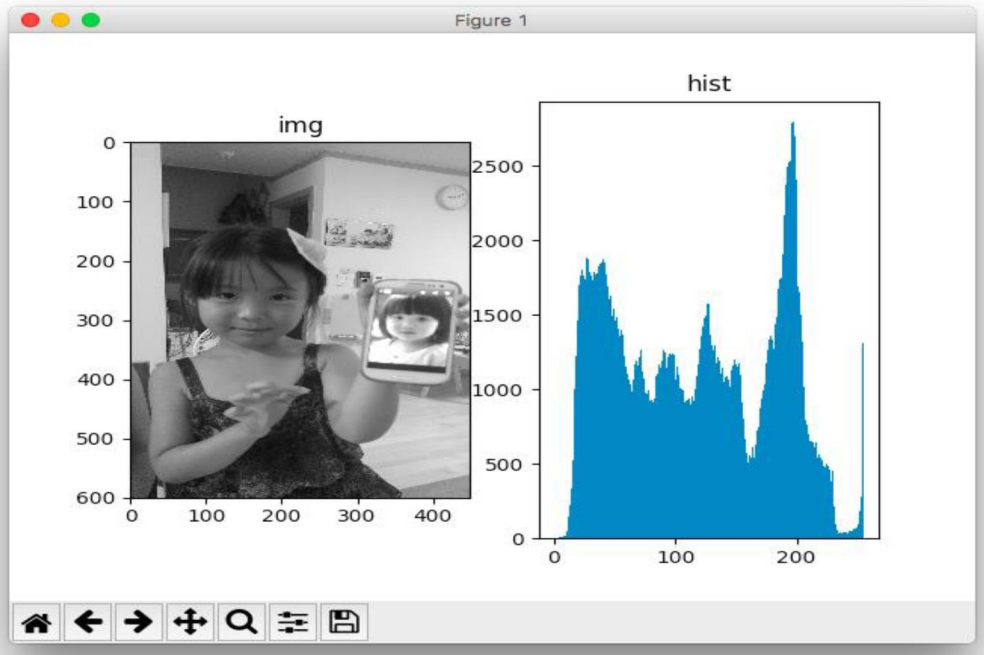


Image Processing Basic

1. **ROI(Region Of Interest)**
2. Color Space
3. Threshold
4. Image Arithmetic
5. Histogram
6. Workshop

❖ ROI

- Region Of Interest
  - Interested region
  - Subject to image edit Region, No All Pixel
  - Improve calculation speed
- Set certain area as Numpy slicing
  - img[h:h+y, w:w+x]
    - Slicing is effective for area pixel approach
  - img.item(h, w, c), img.itemset((h, w, c), val)
    - Item() function is effective for individual pixel approach
  - Turnaround original reference
    - Different as Python list.
    - If ROI is edited, original changes too.
    - If original is edited, ROI is changed too.
    - If you do not want to edit, copy to use (img.copy())

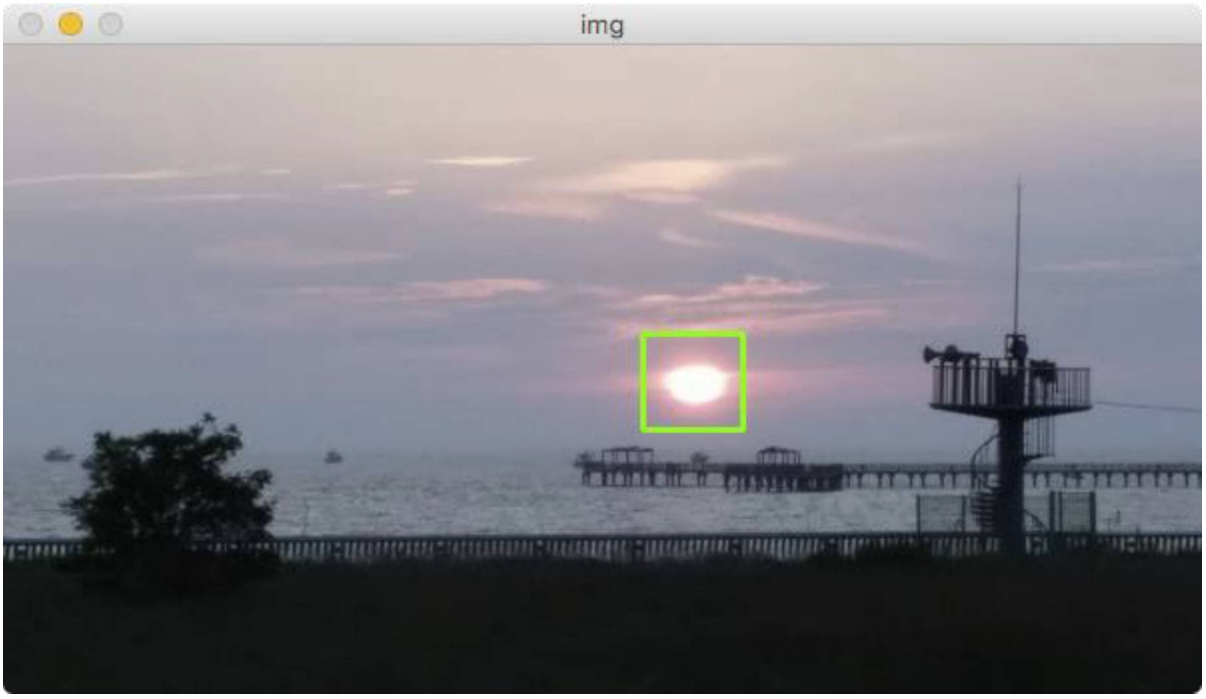
❖ ROI (Region Of Image)

- Set ROI

```
import cv2
import numpy as np

img = cv2.imread('./img/sunset.jpg')

x=320; y=150; w=50; h=50        # roi Coordination
roi = img[y:y+h, x:x+w]         # set ROI

print(roi.shape)                # roi shape, (50,50,3)
cv2.rectangle(roi, (0,0), (h-1, w-1), (0,255,0))
cv2.imshow("img", img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ ROI(Region Of Image)

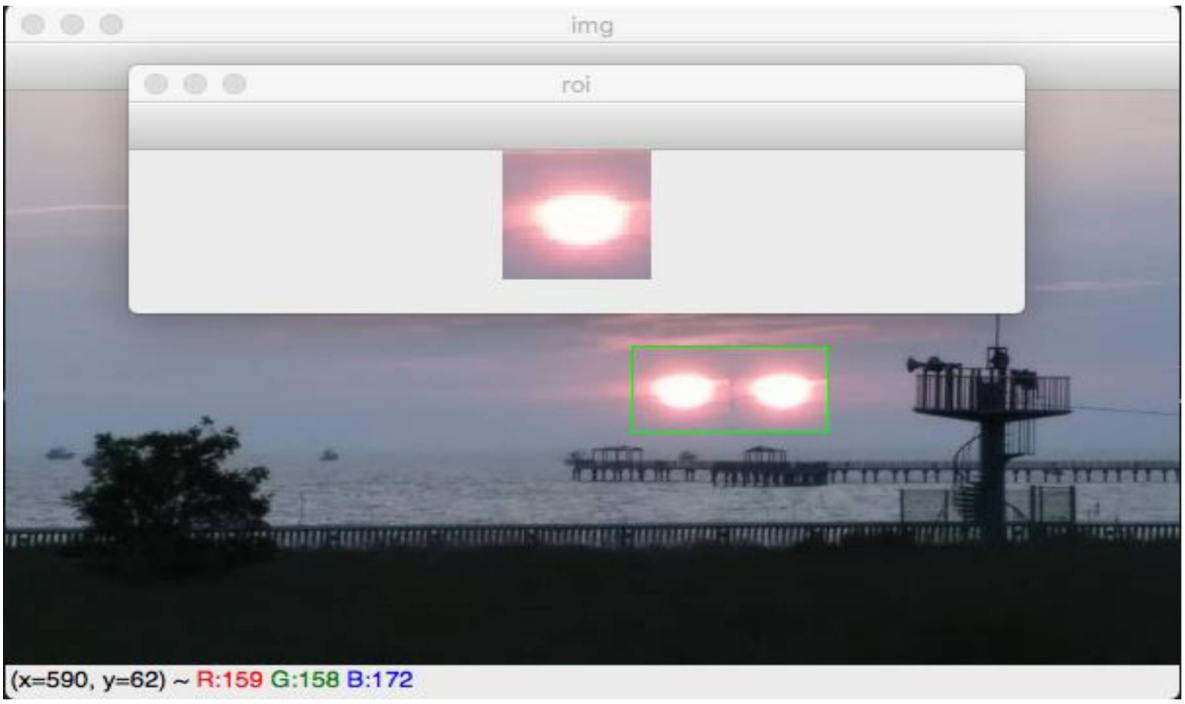
- Set ROI



❖ ROI Copy and Open New Window

```
import cv2
import numpy as np

img = cv2.imread('./img/sunset.jpg')

x=320; y=150; w=50; h=50
roi = img[y:y+h, x:x+w]     # roi Set
img2 = roi.copy()           # roi Array Copy---①

img[y:y+h, x+w:x+w+w] = roi
cv2.rectangle(img, (x,y), (x+w+w, y+h), (0,255,0))

cv2.imshow("img", img)      # Print original image
cv2.imshow("roi", img2)     # Print roi only

cv2.waitKey(0)
cv2.destroyAllWindows()
```

ROI Copy and Open New Window



Select ROI with Mouse (1/2)

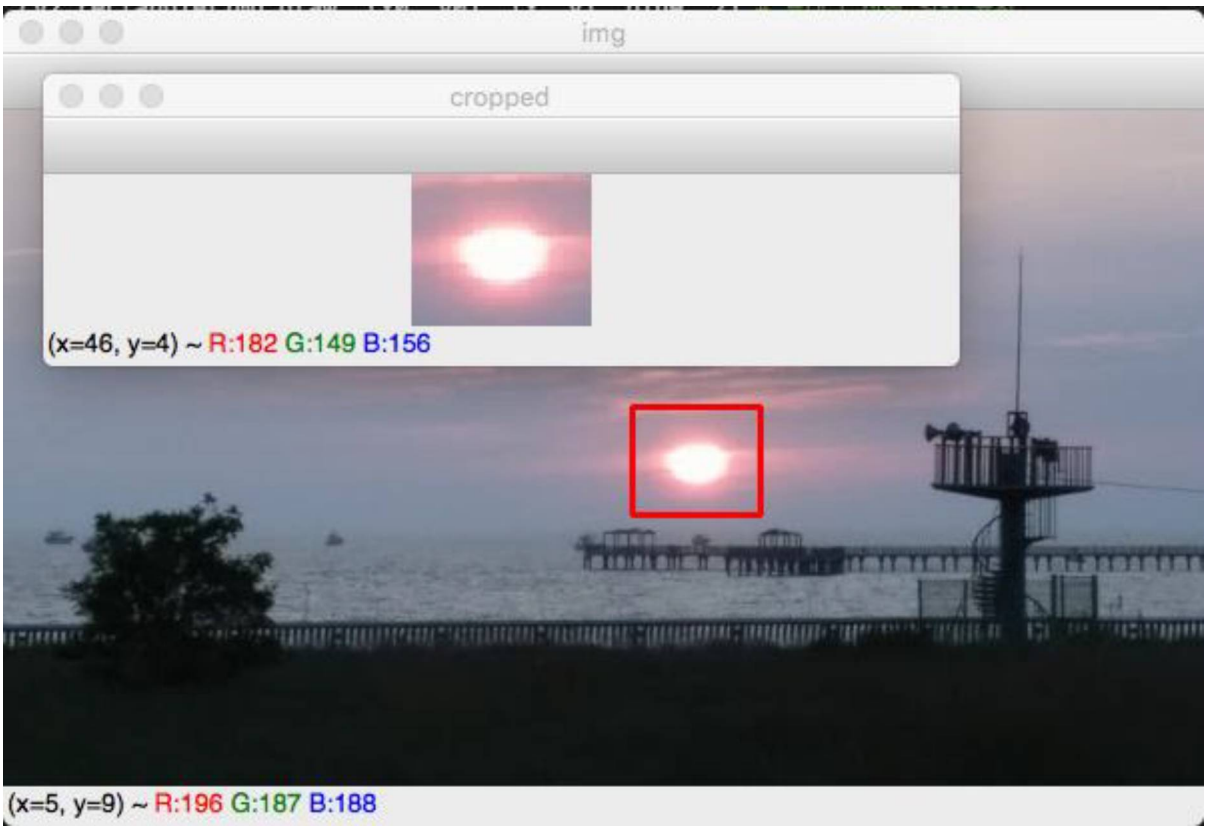
```
import cv2
import numpy as np

isDragging = False
x0, y0, w, h = -1,-1,-1,-1
blue, red = (255,0,0),(0,0,255)

def onMouse(event,x,y,flags,param):
    global isDragging, x0, y0, img
    if event == cv2.EVENT_LBUTTONDOWN:
        isDragging = True
        x0 = x
        y0 = y
    elif event == cv2.EVENT_MOUSEMOVE:
        if isDragging:
            img_draw = img.copy()
            cv2.rectangle(img_draw, (x0, y0), (x, y), blue, 2)
            cv2.imshow('img', img_draw)
```

Select ROI with Mouse (2/2)

```
elif event == cv2.EVENT_LBUTTONUP:
        if isDragging:
            isDragging = False
            w = x - x0
            h = y - y0
            print("x:%d, y:%d, w:%d, h:%d" % (x0, y0, w, h))
            if w > 0 and h > 0:
                img_draw = img.copy()
                cv2.rectangle(img_draw, (x0, y0), (x, y), red, 2)
                cv2.imshow('img', img_draw)
                roi = img[y0:y0+h, x0:x0+w]
                cv2.imshow('cropped', roi)
                cv2.moveWindow('cropped', 0, 0)
                cv2.imwrite('./cropped.jpg', roi)
                print("croped.")
            else:
                cv2.imshow('img', img)
                print(“Drag area from left top to right bottom.")
img = cv2.imread('./img/sunset.jpg')
cv2.imshow('img', img)
cv2.setMouseCallback('img', onMouse)
cv2.waitKey()
cv2.destroyAllWindows()
```

Select ROI with Mouse

```
x:193, y:174, w:-56, h:-77
Drag area from left top to right bottom
x:314, y:148, w:64, h:54
croped.
```



❖ Select ROI

- Easy Mouse ROI selection
- ret=cv2.selectROI([win_name,] img[, showCrossHair, fromCenter])
  - win_name : Name of the window for ROI selection, str
  - img : Image for ROI selection, NumPy ndarray
  - showCrossHair=True : Show or not show cross at the center of region
  - fromCenter=False : Set around mouse starting point as a center of area
  - ret : Size and coordination of selected area (x, y, w, h), When selection is canceled, all 0
- Space or Enter : End Selection
- key board 'C' : Cancel Selection

❖ Select ROI

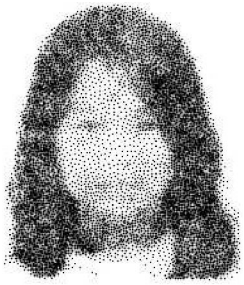
```
import cv2,  numpy as np

img = cv2.imread('../img/sunset.jpg')

x,y,w,h  =    cv2.selectROI('img', img, False)
if w and h:
    roi = img[y:y+h, x:x+w]
    cv2.imshow('cropped', roi)
    cv2.moveWindow('cropped', 0, 0)
    cv2.imwrite('./cropped2.jpg', roi)

cv2.imshow('img', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

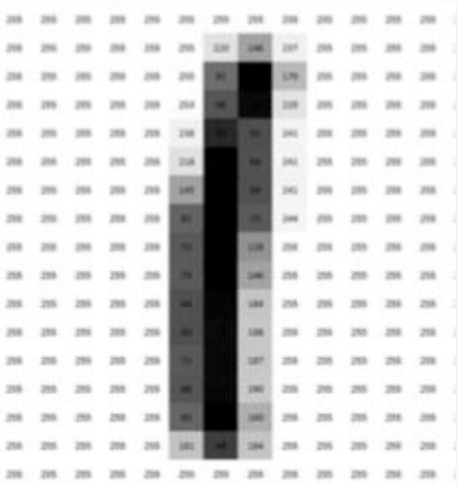
Image Processing Basic

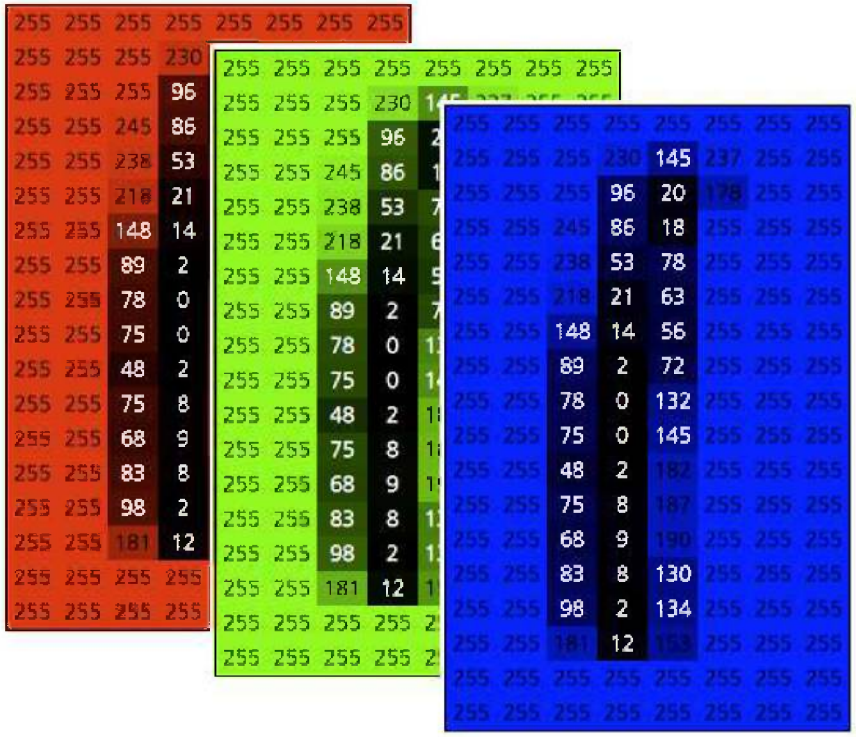
1. ROI(Region Of Interest)
2. **Color Space**
3. Threshold
4. Image Arithmetic
5. Histogram
6. Workshop

❖ Digital Image

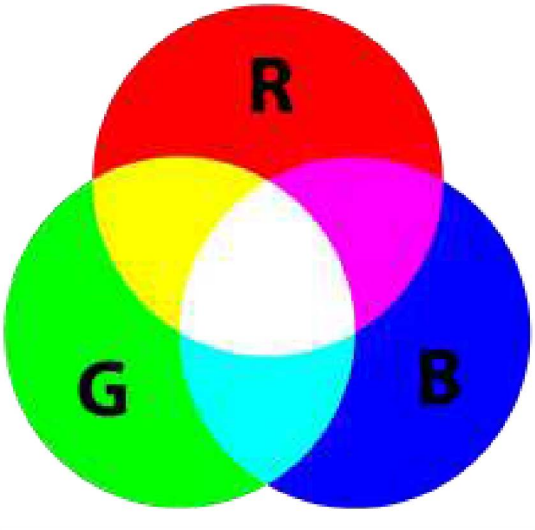
- Binary Image
  - 1px per 1bit (white or black),
  - Show light and shade with density of dots
- Grayscale Image
  - 1px per 1byte=8bit
  - Display 225 light & shade
- Color Image
  - 1px per 3byte=24bit
  - 16,777,216 Colors
  - Various ways to show







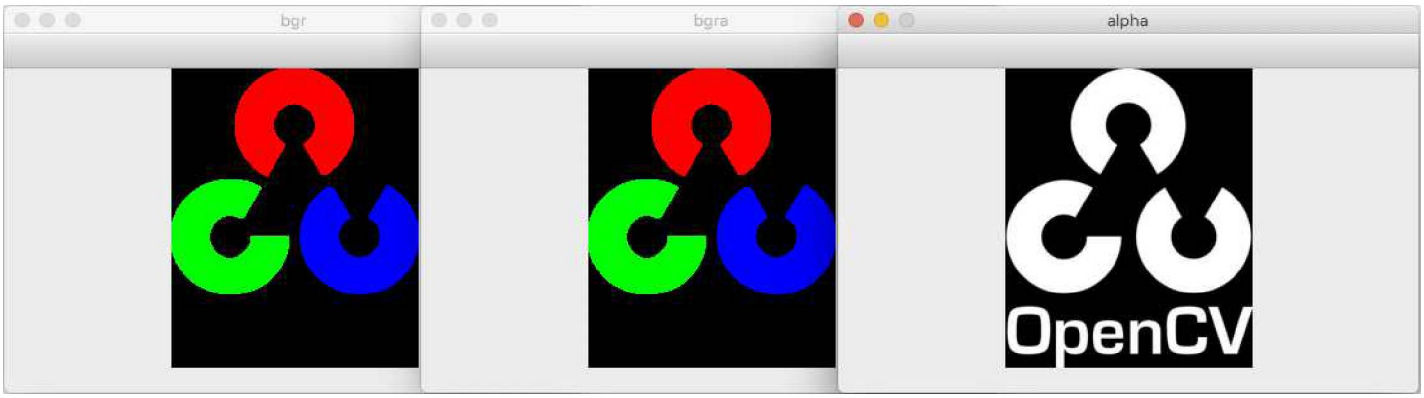
❖ RGB, BGR, RGBA



- RGB
  - Most commonly used Color Space
  - row x column x channel
  - Each color is from 0 ~ 255
- BGR
  - Have reverse RGV number in OpenCV
- RGBA
  - RGB + Alpha
  - Alpha : Clarity
  - File is RGBA, cv2.IMREAD_UNCHANGED flag is used => BGRA

❖ BGR, BGRA, Alpha

```
import cv2
import numpy as np

img = cv2.imread('../img/opencv_logo.png')
bgr = cv2.imread('../img/opencv_logo.png', cv2.IMREAD_COLOR)
bgra = cv2.imread('../img/opencv_logo.png', cv2.IMREAD_UNCHANGED)

print("default:", img.shape, "color:", bgr.shape, "unchanged:",
bgra.shape)
cv2.imshow('bgr', bgr)
cv2.imshow('bgra', bgra)
cv2.imshow('alpha', bgra[:,:,3])
cv2.waitKey(0)
cv2.destroyAllWindows()
```



❖ Color Change Function

- out = cv2.cvtColor(img, flag)img : Numpy array, Image to transfer
  - flag : Color space to transfer, Name starting with cv2.COLOR_ , 274
    - cv2.COLOR_BGR2GRAY : Change BGR color image to grey scale
    - cv2.COLOR_GRAY2BGR : Change grey scale image to BGR color image
    - cv2.COLOR_BGR2RGB : Change BGR color image to RGB image
    - cv2.COLOR_BGR2HSV : Change BGR color image to HSV image
    - cv2.COLOR_HSV2BGR : Change HSV color image to BGR image
    - cv2.COLOR_BGR2YUV : Change BGR color image to YUV
    - cv2.COLOR_YUV2BGR : Change YUV color image to BGR
    - [i for i in dir(cv2) if i.startswith('COLOR_')]
  - out : Changed result (NumPy array)

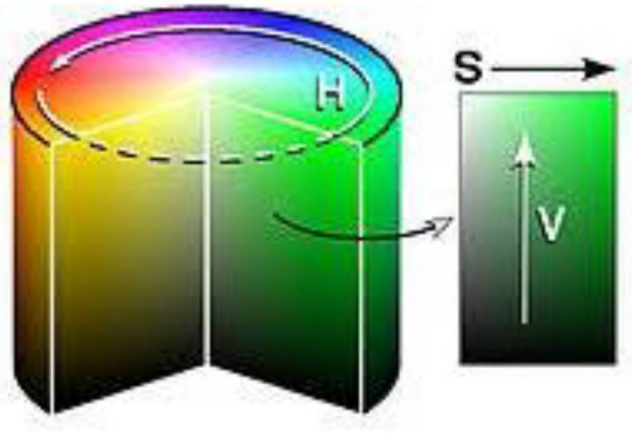
❖ BGR to Gray

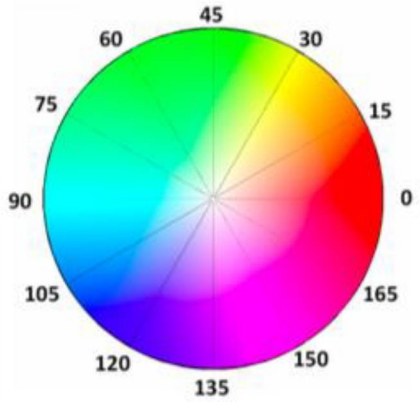
- cv2.COLOR_BGR2GRAY

```
import cv2
import numpy as np

img = cv2.imread('../img/girl.jpg')
img2 = img.astype(np.uint16)
b,g,r = cv2.split(img2)
gray1 = ((b + g + r)/3).astype(np.uint8)
gray2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow('original', img)
cv2.imshow('gray1', gray1)
cv2.imshow('gray2', gray2)
cv2.waitKey(0)
cv2.destrovAllWindows()
```

❖ HSV, HSI, HSL

- Distinguish 3 traits of color
- Show it as cylindrical system channel
- Hue : 0~179
- Saturation :0~255
- Value : 0~255
- Color Range
  - RED : 165~15, 0 ~ 15
  - Green : 45~75
  - Blue : 90~120
- cv2.COLOR_BGR2HSV





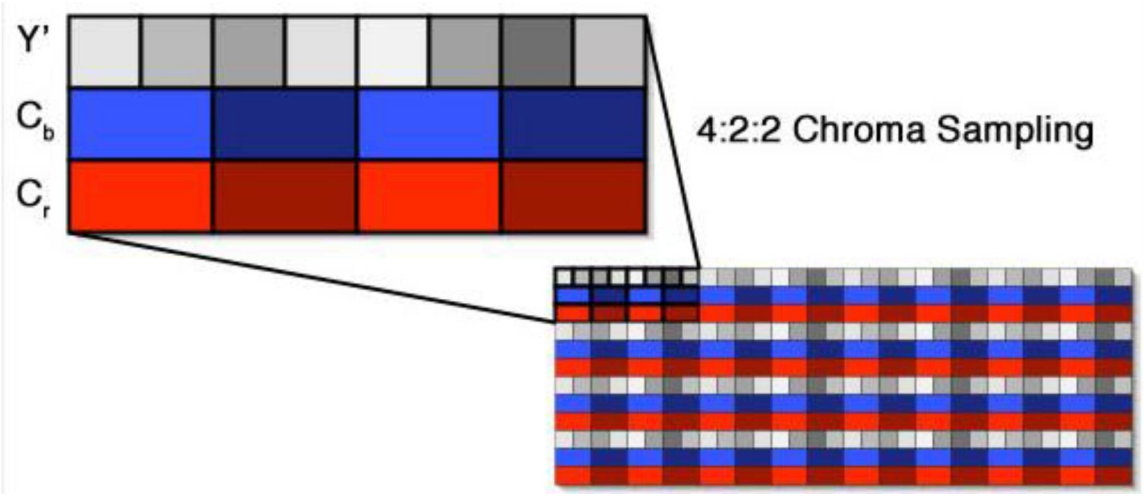
❖ BGR to HSV

```
import cv2
import numpy as np

red_bgr = np.array([[[0,0,255]]], dtype=np.uint8)
green_bgr = np.array([[[0,255,0]]], dtype=np.uint8)
blue_bgr = np.array([[[255,0,0]]], dtype=np.uint8)
yellow_bgr = np.array([[[0,255,255]]], dtype=np.uint8)

red_hsv = cv2.cvtColor(red_bgr, cv2.COLOR_BGR2HSV);
green_hsv = cv2.cvtColor(green_bgr, cv2.COLOR_BGR2HSV);
blue_hsv = cv2.cvtColor(blue_bgr, cv2.COLOR_BGR2HSV);
yellow_hsv = cv2.cvtColor(yellow_bgr, cv2.COLOR_BGR2HSV);

print("red:",red_hsv)
print("green:", green_hsv)
print("blue", blue_hsv)
print("yellow", yellow_hsv)
```

```
red: [[[  0 255 255]]]
green: [[[ 60 255 255]]]
blue [[[120 255 255]]]
```

❖ YUV, YCbCr

- People are sensitive to lights, and less sensitive to color
- Y : light
- U(chroma Blue, Cb) : Color difference between light and blue
- V(Chroma Red, Cr) : Color difference between light and red
- Data compression effect
  - Allocate many bit to Y and less bit to UV
- cv2.COLOR_BGR2YUV
- cv2.COLOR_BGR2YCrCb



❖ BGR to YUV

```
import cv2
import numpy as np

dark = np.array([[[0,0,0]]], dtype=np.uint8)
middle = np.array([[[127,127,127]]], dtype=np.uint8)
bright = np.array([[[255,255,255]]], dtype=np.uint8)
dark_yuv = cv2.cvtColor(dark, cv2.COLOR_BGR2YUV)
middle_yuv = cv2.cvtColor(middle, cv2.COLOR_BGR2YUV)
bright_yuv = cv2.cvtColor(bright, cv2.COLOR_BGR2YUV)
print("dark:",dark_yuv)
print("middle:", middle_yuv)
print("bright", bright_yuv)
```

```
dark: [[[  0 128 128]]]
middle: [[[127 128 128]]]
bright [[[255 128 128]]]
```

❖ CMYK

- Cyan, Magenta, Yellow, Key(Black)
- Subtractive Mixture: Primary color + Black
- It is difficult to show pure color in reality
- Used I printing field

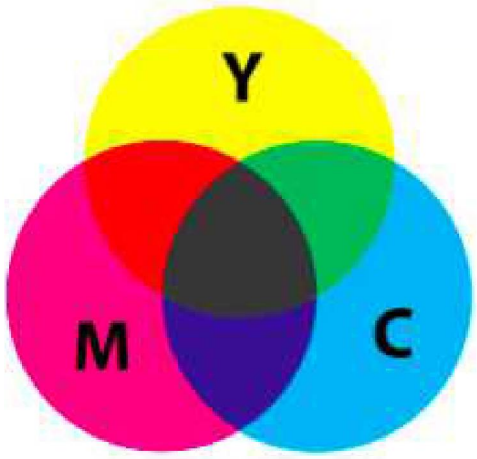


Image Processing Basic

1. ROI(Region Of Interest)
2. Color Space
3. **Threshold**
4. Image Arithmetic
5. Histogram
6. Workshop

❖ Binarization

- Two Class Classification
- Letters and paper
- Background and Front View

❖ Global fixed thresholding

- Use one threshold to one image
- Deciding Optimal Threshold
- Simple Thresholding : Trial and Error Method
- Outsu's Method

❖ Locally Adaptive Thresholding

- Use different threshold for each pixel
- When each area has different light level

❖ Simple Global fixed threshold

- Set temporary threshold (boundary) value
- numpy masking
  - dst[src > thresh] =255
- ret, dst = cv.threshold(src, thresh, maxval, flag)
  - src : input image
  - thresh : threshold value
  - value : Value to be applied to the pixel fit to the threshold value
  - flag : cv2.THRESH_*
    - BINARY : px > thresh ? value : 0
    - BINARY_INV : px > thresh ? 0 : value
    - TRUNC : px > thresh ? value : px
    - TOZERO : px > thresh ? px : 0
    - TOZERO_INV : px > thresh ? 0 : px

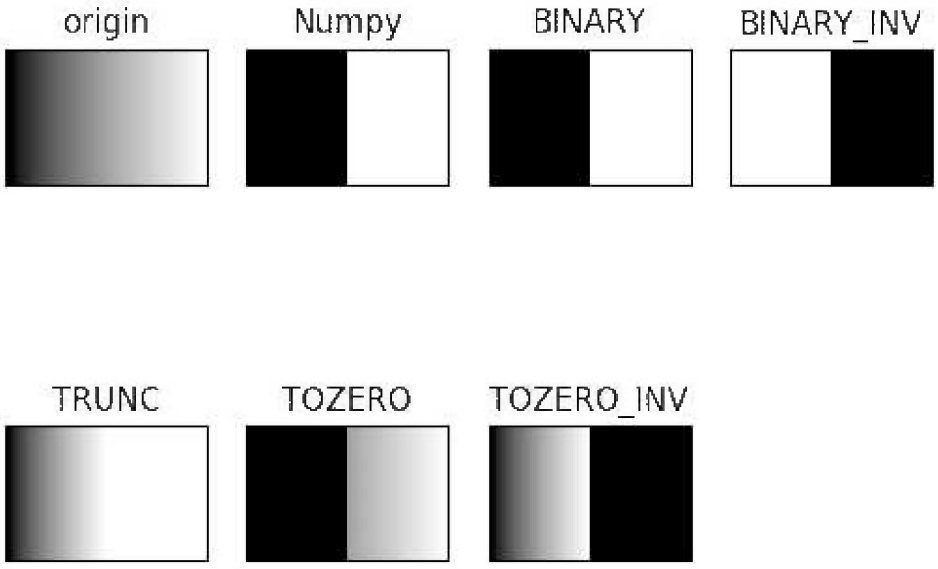
❖ Simple Global fixed threshold

- example

```
import cv2
import numpy as np
import matplotlib.pylab as plt

img = cv2.imread('../img/gray-gradient.jpg', cv2.IMREAD_GRAYSCALE)

t_np = np.zeros_like(img)
t_np[ img > 127] = 255

ret, t_bin = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
ret, t_bininv = cv2.threshold(img, 127, 255,
cv2.THRESH_BINARY_INV)
ret, t_truc = cv2.threshold(img, 127, 255, cv2.THRESH_TRUNC)
ret, t_2zr = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO)
ret, t_2zrinv = cv2.threshold(img, 127, 255,
cv2.THRESH_TOZERO_INV)

titles = ['origin','Numpy', 'BINARY', 'BINARY_INV', \
                                    'TRUNC', 'TOZERO', 'TOZERO_INV']
imgs = [img, t_np, t_bin, t_bininv, t_truc, t_2zr, t_2zrinv]

for i in range(len(imgs)):
    plt.subplot(2,4, i+1)
    plt.imshow(imgs[i], cmap='gray')
    plt.title(titles[i])
    plt.xticks([])
    plt.yticks([])

plt.show()
```

❖ Simple Global fixed threshold

- example < Result>

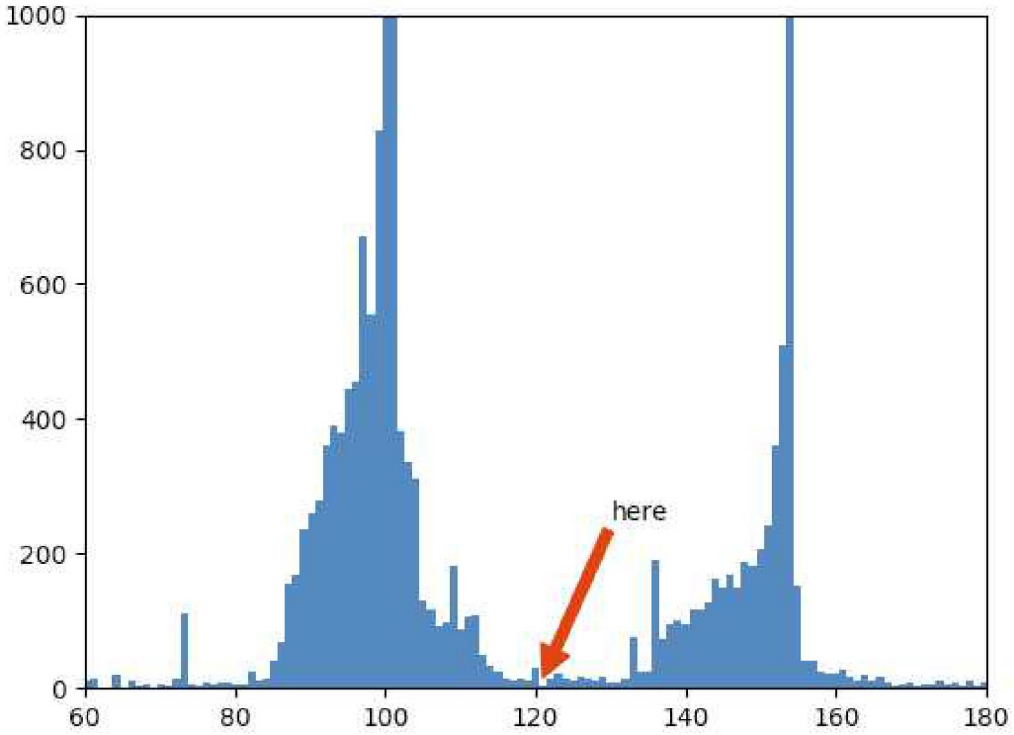


❖ Otsu's Binarization Method

- Finding Optimal Threshold

❖ Otsu's Binarization Method

- Nobuyuki Otsu, 1979
- Divided ad pixels into two classes according to light level
- Middle point as threshold
- How to
  - Divice image pixels into two classes with threshold value t
    - t: 0~255
    - Ration of each class: ω_1, ω_2,
    - Mean of each class light: μ1, μ2
    - Distribution of each class: σ_1^2, σ_2^2
    - Intra-class variance: $\sigma_\omega^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$
      - Minimun t Value
      - Maximum t Value
    - Inter-Class Variance: $\sigma_b^2(t) = \omega_1(t)\omega_2(t)\,[\mu_1(t) - \mu_2(t)]^2$
  - cv2.threshold
    - Flag: cv2.THRESH_OTSU, Turnaround: T value
    - Effective after applying Gaussian filter



❖ Otsu's Binarization Method

```
import cv2
import numpy as np
import matplotlib.pylab as plt

img = cv2.imread('../img/scaned_paper.jpg', cv2.IMREAD_GRAYSCALE)
_, t_130 = cv2.threshold(img, 130, 255, cv2.THRESH_BINARY)
t, t_otsu = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY |
cv2.THRESH_OTSU)
print('otsu threshold:', t)

imgs = {'Original': img, 't:130':t_130, 'otsu:%d'%t: t_otsu}
for i , (key, value) in enumerate(imgs.items()):
    plt.subplot(1, 3, i+1)
    plt.title(key)
    plt.imshow(value, cmap='gray')
    plt.xticks([]); plt.yticks([])

plt.show()
```

❖ Adaptive Threshold

- Set threshold according to surrounding resolution value
- Adopt and react to surrounding situation
- cv2.adaptiveThreshold(src, maxVal, method, type, blockSize, C)
  - src : gray scaled image
  - maxVal : Value applied to pixels when it meets threshold value
  - method : Deciding threshold
    - cv2.ADPTIVE_THRESH_*
    - MEAN_C : Decide by mean of neighboring pixel
    - GAUSSIAN_C: Decide by addition of weight by Gaussian distribution
  - type : threshold type
  - blockSize : Size of neighboring area
  - C : Decide threshold by adding subtracting to the calculated result

$$T(x,y)=\frac{1}{n^2}\sum_{x_i}\sum_{y_j} I(x+x_i,y+y_j) \; - \; C$$

- sudoku scan example < 1/2 >

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

file_name = '../img/sudoku.png'
img = cv2.imread(file_name, cv2.IMREAD_GRAYSCALE)
print(img.shape)
ret, th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY,11,7)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY,11,7)
```

❖ Adaptive Threshold

- sudoku scan example < 2/2 >

```
titles = ['Original','Global','Mean','Gaussian']
images = [img,th1,th2,th3]

for i in range(4):
  plt.subplot(2,2,i+1)
    plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

- sudoku scan example < Result >

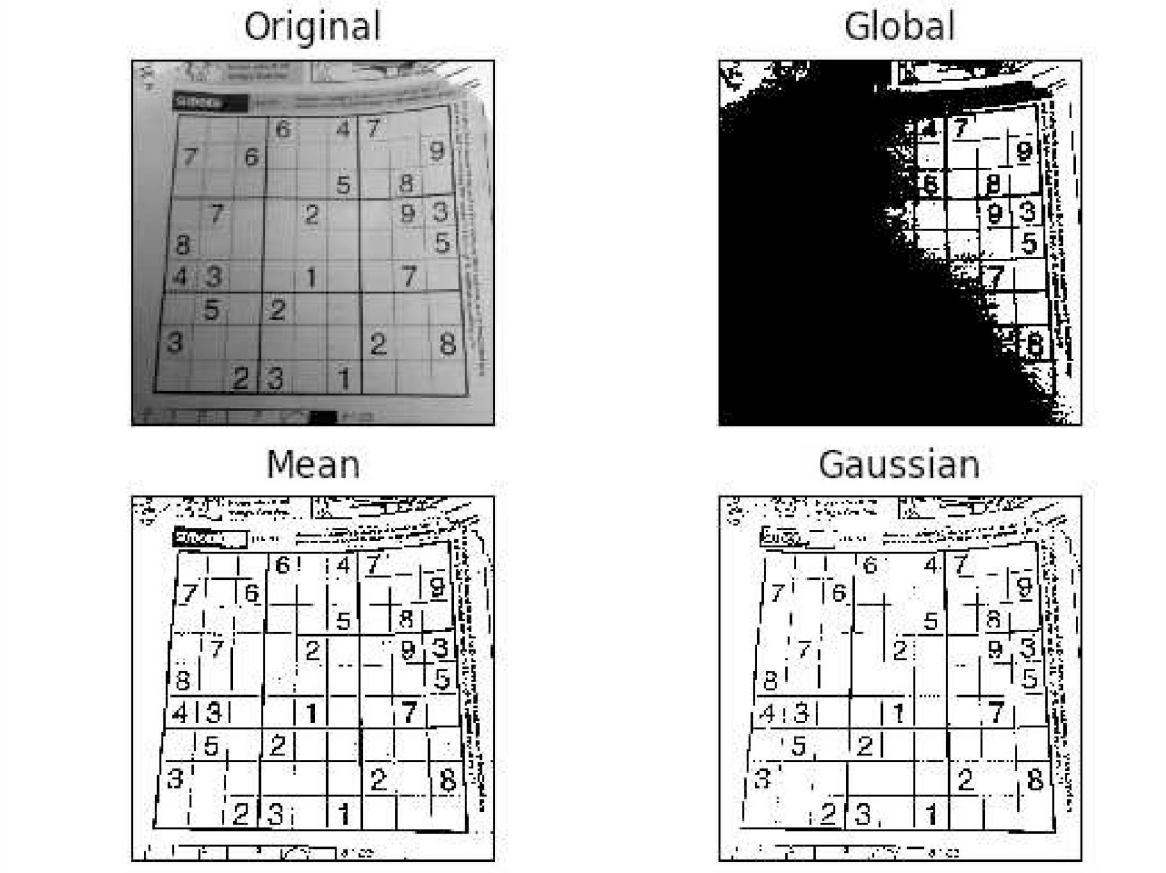


Image Processing Basic

1. ROI(Region Of Interest)
2. Color Space
3. Threshold
4. **Image Arithmetic**
5. Histogram
6. Workshop

❖ Image Arithmetic

- NumPy Broadcasting (+ , - , *, /)
  - 0 > Calculation Result > 255
- OpenCV Function
  - cv2.add(src1, src2[, dest, mask, dtype])
    - src1 : Input image 1 or number of
    - src2 : Input image 2 or number of
    - out : Print image
    - mask : Calculate pixels that are not 0
    - dtype : Print dtype
  - cv2.substract(src1, src2[, dest, mask, dtype])
    - Same as cv2.add()
  - cv2.multiply(src1, src2[, dest, scale, dtype])
    - scale : Value to add to the calculation result
  - cv2.divide(src1, src2[, dest, scale, dtype])
    - Same as cv2.multiply()

❖ Image Addition

```
import cv2
import numpy as np
a = np.uint8([[200, 50]])
b = np.uint8([[100, 100]])
add1 = a + b
sub1 = a - b
mult1 = a * 2
div1 = a / 3
add2 = cv2.add(a, b)
sub2 = cv2.subtract(a, b)
mult2 = cv2.multiply(a , 2)
div2 = cv2.divide(a, 3)
print(add1, add2)
print(sub1, sub2)
print(mult1, mult2)
print(div1, div2)
```

```
[ 44 150]] [[255 150]]
[[100 206]] [[100   0]]
[[144 100]] [[255 100]]
[[66.66666667 16.66666667]] [[67 17]]
```

❖ Image Addition with Mask

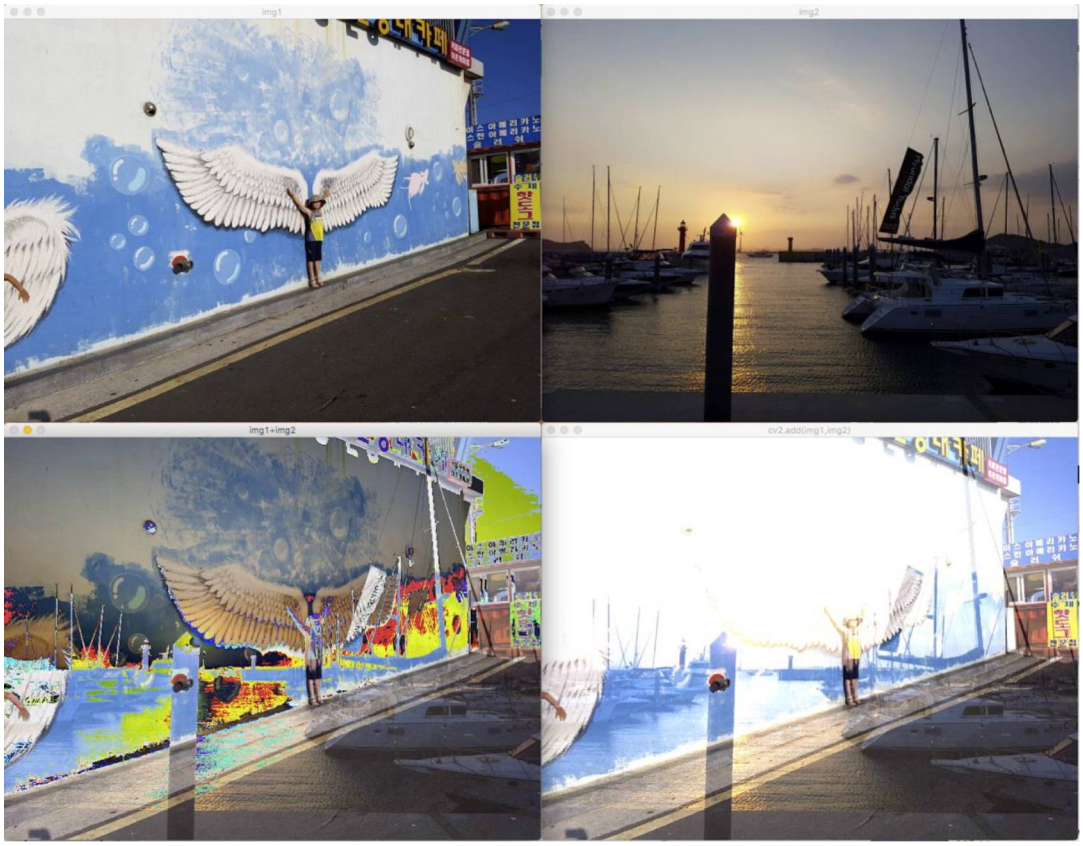
```
import cv2
import numpy as np

#---① 연산에 사용할 배열 생성
a = np.array([[1, 2]], dtype=np.uint8)
b = np.array([[10, 20]], dtype=np.uint8)
#---② 2번째 요소가 0인 마스크 배열 생성
mask = np.array([[1, 0]], dtype=np.uint8)

#---③ 누적 할당과의 비교 연산
c1 = cv2.add( a, b , None, mask)
print(c1)
c2 = cv2.add( a, b , b, mask)
print(c2)
```

```
[[11  0]]
[[11 20]]
```

❖ Image Blending

```
a = np.array([250], dtype=np.uint8)
b = np.array([10], dtype=np.uint8)
c = a + b
print(a, b)
print( a+b )  # 250 + 10 = 260, 260 - 256 = 4
print( cv2.add(a, b)) #over 255

img1 = cv2.imread('../img/wingwall.jpg')
img2 = cv2.imread('../img/yate.jpg')

cv2.imshow('img1', img1)
cv2.imshow('img2', img2)
cv2.imshow('img1+img2', img1+img2)
cv2.imshow('cv2.add(img1,img2)', cv2.add(img1,img2))

cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Image Blending



❖ Alpha Blending

- Mix 2 images
- Decide weights, alpha : beta , alpha + beta = 1
- a*alpha + b*beta <= 255
- Realize with numpy

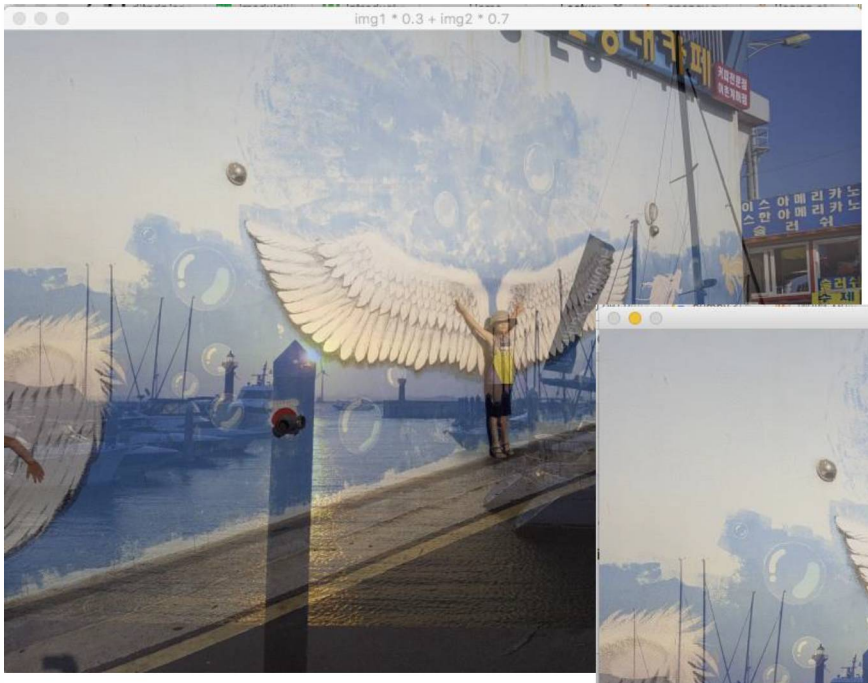
$$g(x) = (1 - \alpha) f_0(x) + \alpha f_1(x)$$

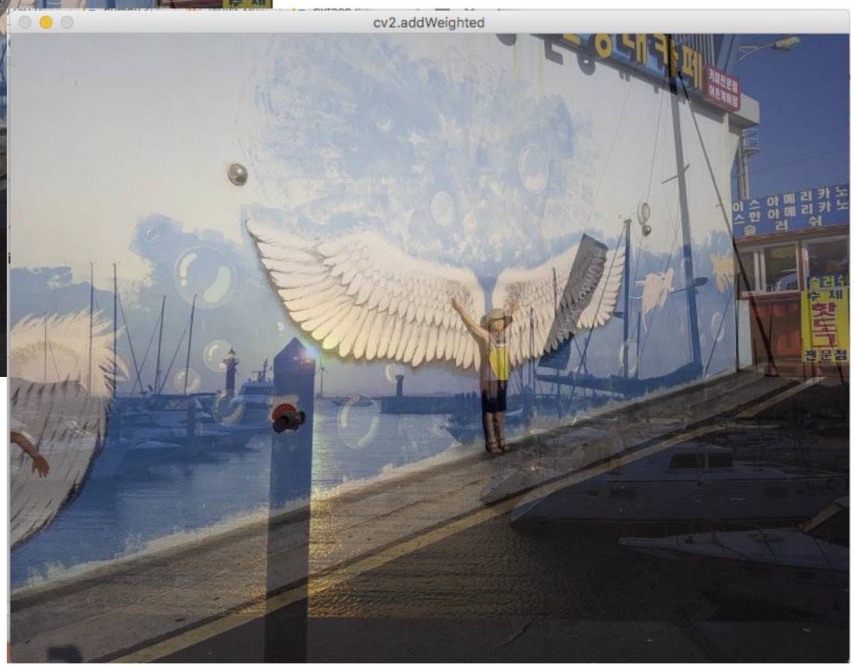
- cv2.addWeighted(img1, alpha, img2, beta, gamma)
  - img1, img2 : two images to mix
  - alpha : Weight on img1 (alpha value)
  - beta : weight on img2, usually (1- alpha)
  - gamma : integer that can be added to calculation result, usually 0(zero)

$$dst = \alpha \cdot img1 + \beta \cdot img2 + \gamma$$

❖ Alpha Blending

```
alpha = 0.3

img1 = cv2.imread('../img/wingwall.jpg')
img2 = cv2.imread('../img/yate.jpg')

blended = img1 *alpha + img2 * (1-alpha)
blended = blended.astype(np.uint8)
cv2.imshow('img1 * 0.3 + img2 * 0.7', blended)

dst = cv2.addWeighted(img1, alpha, img2, (1-alpha), 0)
cv2.imshow('cv2.addWeighted', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





❖ Image Blending – Track bar

```
import cv2
import numpy as np

win_name = 'Alpha blending'      # 창 이름
trackbar_name = 'fade'           # 트렉바 이름

def onChange(x):
    alpha = x/100
    dst = cv2.addWeighted(img1, 1-alpha, img2, alpha, 0)
    cv2.imshow(win_name, dst)

img1 = cv2.imread('../img/man_face.jpg')
img2 = cv2.imread('../img/lion_face.jpg')
cv2.imshow(win_name, img1)
cv2.createTrackbar(trackbar_name, win_name, 0, 100, onChange)

cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Bitwise Operation

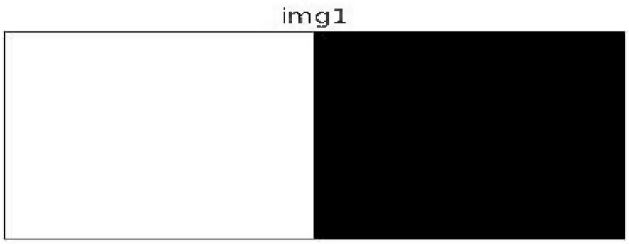
- Calculate bitwise for each pixel of image 2
- Use for deleting background, separating background and Front View, and blending
  - cv2.bitwise_and(img1, img2, mask=None)
  - cv2.bitwise_or(img1, img2, mask=None)
  - cv2.bitwise_xor(img1, img2, mask=None)
  - cv2.bitwise_not(img1, mask=None)
    - mask (optional)
      - Calculate only pixel that mask value is not 0, Use mask value for 0
      - binary image(Black image)
    - Rows and columns of img1, img2 of mask should be same

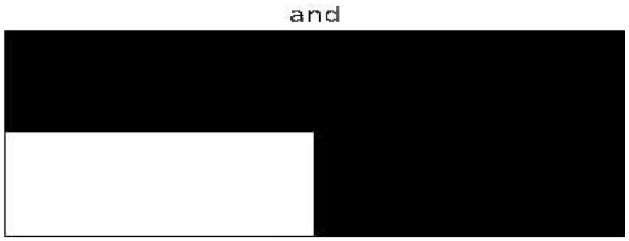
❖ Image Bitwise

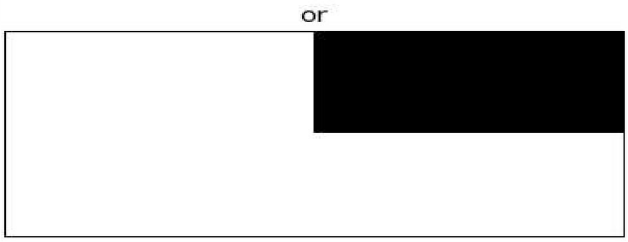
```
import numpy as np, cv2
import matplotlib.pylab as plt
img1 = np.zeros( ( 200,400), dtype=np.uint8)
img2 = np.zeros( ( 200,400), dtype=np.uint8)
img1[:, :200] = 255         # 왼쪽은 검정색(0), 오른쪽은 흰색(255)
img2[100:200, :] = 255      # 위쪽은 검정색(0), 아래쪽은 흰색(255)

bitAnd = cv2.bitwise_and(img1, img2)
bitOr = cv2.bitwise_or(img1, img2)
bitXor = cv2.bitwise_xor(img1, img2)
bitNot = cv2.bitwise_not(img1)

imgs = {'img1':img1, 'img2':img2, 'and':bitAnd,
          'or':bitOr, 'xor':bitXor, 'not(img1)':bitNot}
for i, (title, img) in enumerate(imgs.items()):
    plt.subplot(3,2,i+1)
    plt.title(title)
    plt.imshow(img, 'gray')
    plt.xticks([]); plt.yticks([])

plt.show()
```

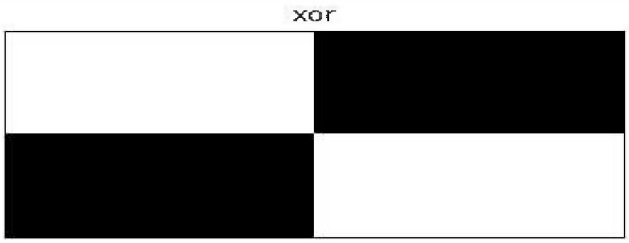
❖ Image Bitwise

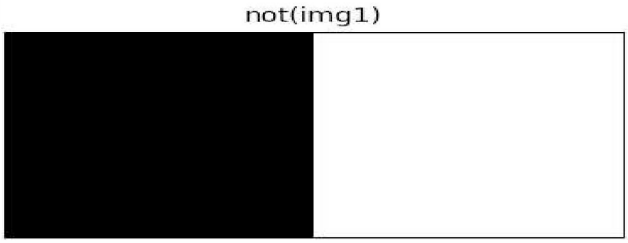
- bitwise example







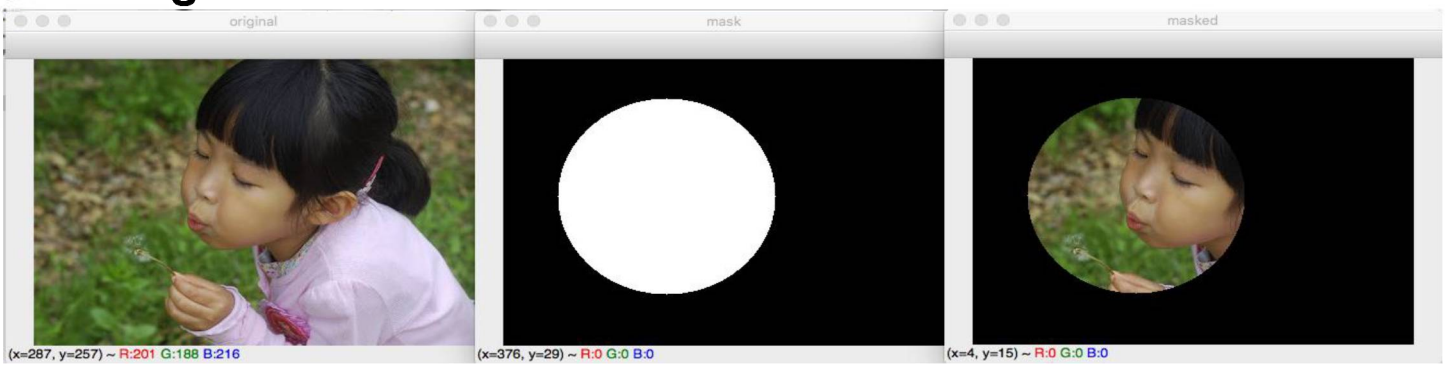




❖ Masking with Bitwise

```
import numpy as np, cv2
import matplotlib.pylab as plt
img = cv2.imread('../img/girl.jpg')
mask = np.zeros_like(img)
cv2.circle(mask, (150,140), 100, (255,255,255), -1)
masked = cv2.bitwise_and(img, mask)

cv2.imshow('original', img)
cv2.imshow('mask', mask)
cv2.imshow('masked', masked)
cv2.waitKey()
cv2.destroyAllWindows()
mask = np.zeros(img.shape[:2], dtype=np.uint8)
cv2.circle(mask, (150,140), 100, (255), -1)
masked = cv2.bitwise_and(img, img, mask=mask)
```

❖ Masking with Bitwise



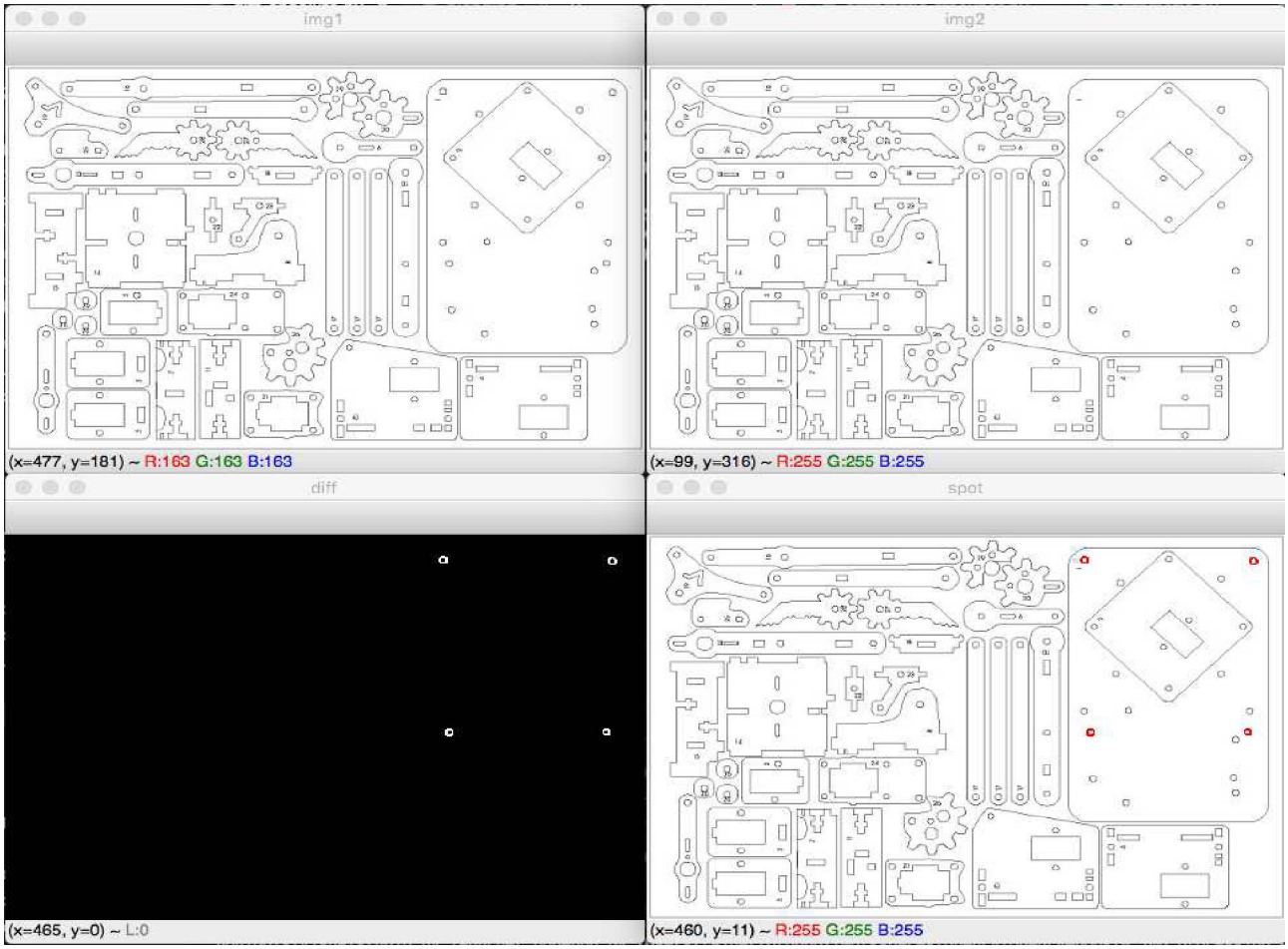
❖ Diff Image

- Taking out image out of image
- Looking for differences
- diff = cv2.absdiff(img1, img2)
  - img1, img2 : input image
  - diff : Turnaround of difference between two images

❖ Diff Image (Finding Map Differences)

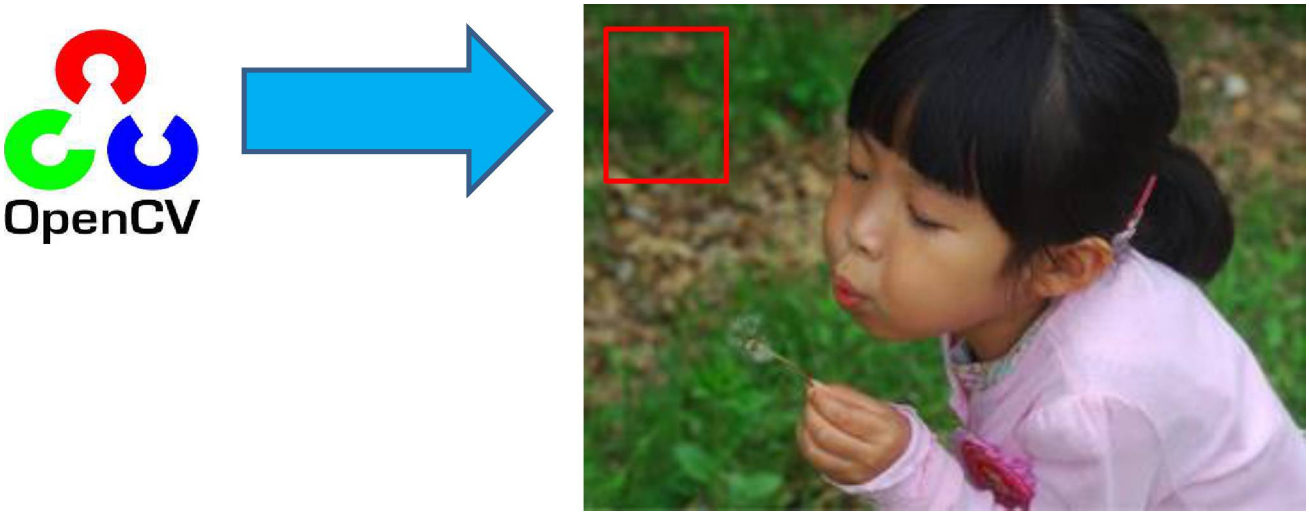
```
import numpy as np, cv2

img1 = cv2.imread('../img/robot_arm1.jpg')
img2 = cv2.imread('../img/robot_arm2.jpg')
img1_gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
img2_gray = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
diff = cv2.absdiff(img1_gray, img2_gray)
_, diff = cv2.threshold(diff, 1, 255, cv2.THRESH_BINARY)
diff_red = cv2.cvtColor(diff, cv2.COLOR_GRAY2BGR)
diff_red[:,:,2] = 0
spot = cv2.bitwise_xor(img2, diff_red)
cv2.imshow('img1', img1)
cv2.imshow('img2', img2)
cv2.imshow('diff', diff)
cv2.imshow('spot', spot)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Diff Image (Finding Map Differences)



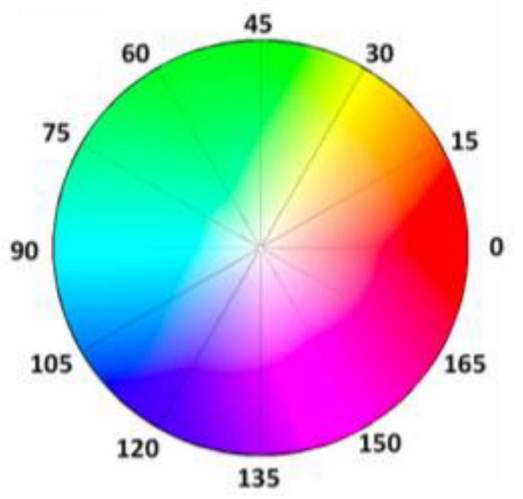
❖ Blending with Alpha Channel(RGBA PNG)

- Blending logo to image



❖ HSV Color Masking

- Color mask using HSV color space
- dst=cv2.inRange(src, lowerb, upperb)
  - src : Original organization
  - Low and upper : Zone
  - dst : Mark zero(0) for factors out of the original zone



- Distinguishing by color <1/2>

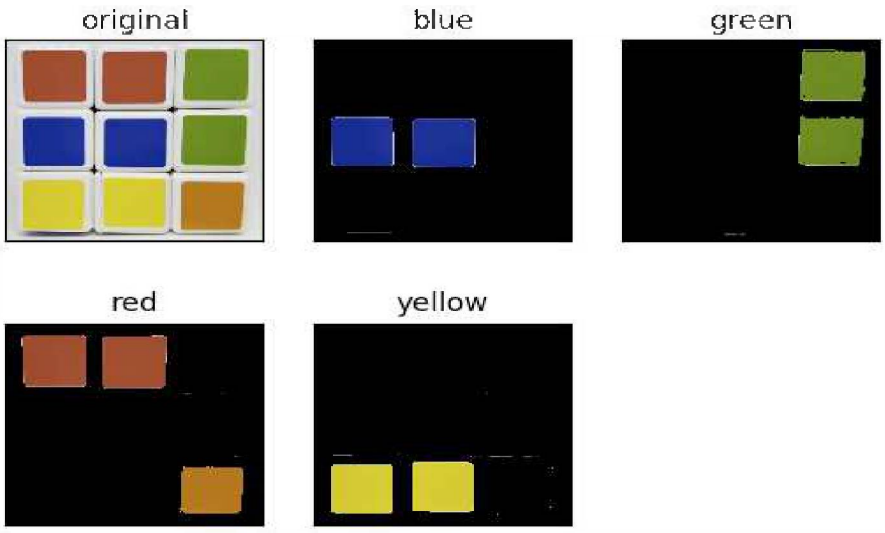
```
import cv2
import numpy as np
import matplotlib.pylab as plt
img = cv2.imread("../img/cube.jpg")
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
blue1 = np.array([90, 50, 50])
blue2 = np.array([120, 255,255])
green1 = np.array([45, 50,50])
green2 = np.array([75, 255,255])
red1 = np.array([0, 50,50])
red2 = np.array([15, 255,255])
red3 = np.array([165, 50,50])
red4 = np.array([180, 255,255])
yellow1 = np.array([20, 50,50])
yellow2 = np.array([35, 255,255])
```

❖ HSV Color Masking

- Distinguishing by color <2/2>

```
mask_blue = cv2.inRange(hsv, blue1, blue2)
mask_green = cv2.inRange(hsv, green1, green2)
mask_red = cv2.inRange(hsv, red1, red2)
mask_red2 = cv2.inRange(hsv, red3, red4)
mask_yellow = cv2.inRange(hsv, yellow1, yellow2)
res_blue = cv2.bitwise_and(img, img, mask=mask_blue)
res_green = cv2.bitwise_and(img, img, mask=mask_green)
res_red1 = cv2.bitwise_and(img, img, mask=mask_red)
res_red2 = cv2.bitwise_and(img, img, mask=mask_red2)
res_red = cv2.bitwise_or(res_red1, res_red2)
res_yellow = cv2.bitwise_and(img, img, mask=mask_yellow)
imgs = {'original': img, 'blue':res_blue, 'green':res_green,
                            'red':res_red, 'yellow':res_yellow}
for i, (k, v) in enumerate(imgs.items()):
    plt.subplot(2,3, i+1)
    plt.title(k)
    plt.imshow(v[:,:,::-1])
    plt.xticks([]); plt.yticks([])
plt.show()
```

- Distinguishing by color <Result>



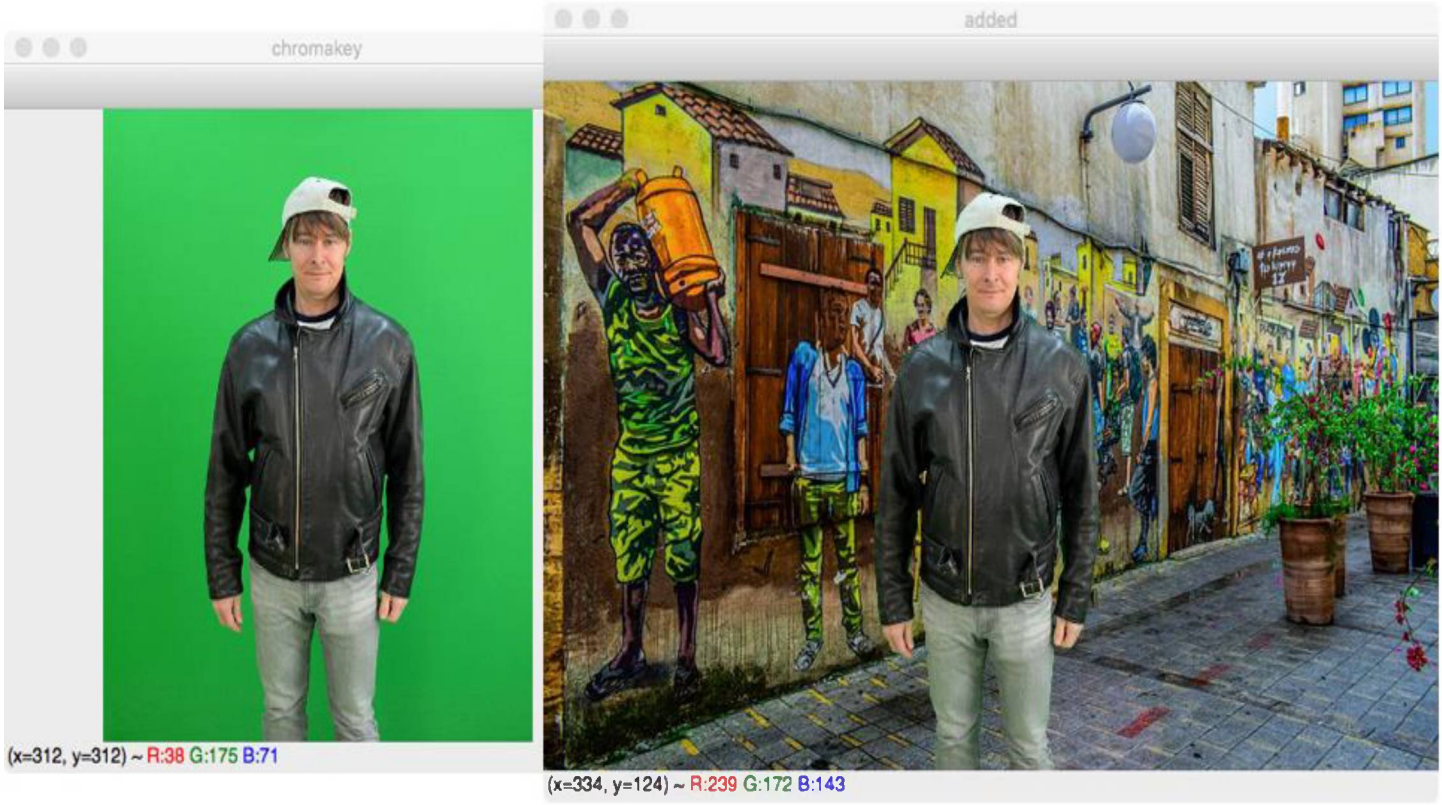
❖ Chromakey Masking and Blending

- Chroma-key blending

```
import cv2
import numpy as np
import matplotlib.pylab as plt

img1 = cv2.imread('../img/man_chromakey.jpg')
img2 = cv2.imread('../img/street.jpg')

height1, width1 = img1.shape[:2]
height2, width2 = img2.shape[:2]
x = (width2 - width1)//2
y = height2 - height1
w = x + width1
h = y + height1

chromakey = img1[:10, :10, :]
offset = 20

hsv_chroma = cv2.cvtColor(chromakey, cv2.COLOR_BGR2HSV)
hsv_img = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
chroma_h = hsv_chroma[0]
lower = np.array([chroma_h.min()-offset, 100, 100])
upper = np.array([chroma_h.max()+offset, 255, 255])
mask = cv2.inRange(hsv_img, lower, upper)
mask_inv = cv2.bitwise_not(mask)
roi = img2[y:h, x:w]
fg = cv2.bitwise_and(img1, img1, mask=mask_inv)
bg = cv2.bitwise_and(roi, roi, mask=mask)
img2[y:h, x:w] = fg + bg
cv2.imshow('chromakey', img1)
cv2.imshow('added', img2)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Chromakey Masking and Blending

- Chroma-key blending <Result>



❖ SeamlessClone

- dst = cv2.seamlessClone(src, dst, mask, coords, flags[, output])
  - src : Input image, normally Front View
  - dst : Image subject, normally background
  - mask : Mask, 225 for zone to blend in src, remainder is 0
  - coodrs : DST coordination (Center) wishing to locate src
  - flags : Blending Type
    - cv2.NORMAL_CLONE : Maintain original input
    - cv2.MIXED_CLONE : Blend input and subject
  - output : Blending Result
  - dst : Blending Reusult

❖ seamless clone blending

```
import cv2
import numpy as np
import matplotlib.pylab as plt
img1 = cv2.imread("../img/drawing.jpg")
img2= cv2.imread("../img/my_hand.jpg")
mask = np.full_like(img1, 255)
height, width = img2.shape[:2]
center = (width//2, height//2)
normal = cv2.seamlessClone(img1, img2, mask, center, cv2.NORMAL_CLONE)
mixed = cv2.seamlessClone(img1, img2, mask, center, cv2.MIXED_CLONE)
cv2.imshow('normal', normal)
cv2.imshow('mixed', mixed)
cv2.waitKey()
cv2.destroyAllWindows()
```

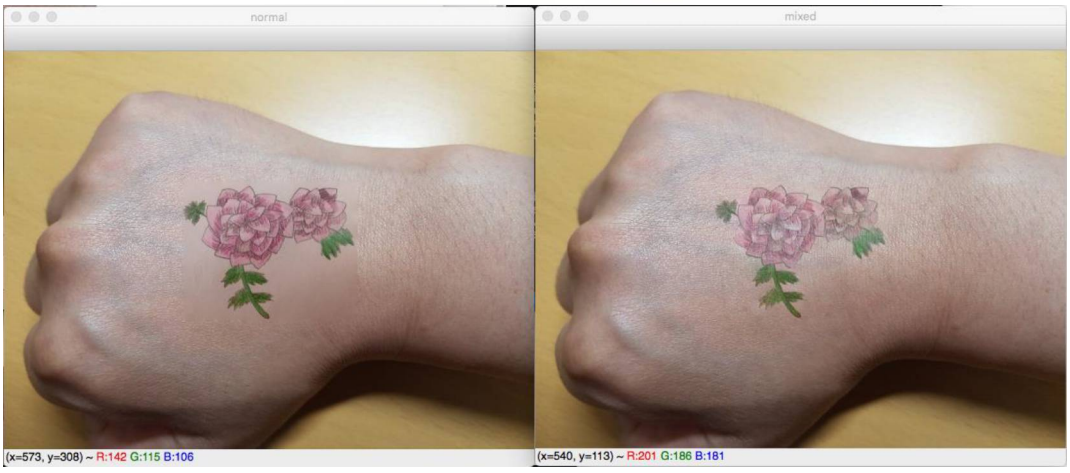
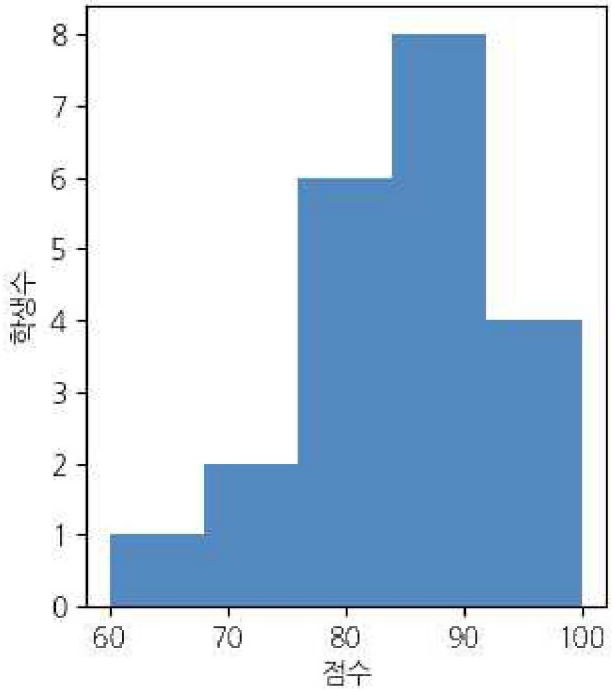


Image Processing Basic

1. ROI(Region Of Interest)
2. Color Space
3. Threshold
4. Image Arithmetic
5. **Histogram**
6. Workshop

❖ Histogram

- Show frequency of color value of pixel
  - 1Channel : Gray Color
  - 2Channel : Cross two values
  - 3Channel : Difficult to show
- Contrast Equalization, Various application like background deleting
- Able to support all OpenCV, Numpy, MatPlotlib
  - cv2.calcHist()
  - np.histogram()
  - plt.hist()



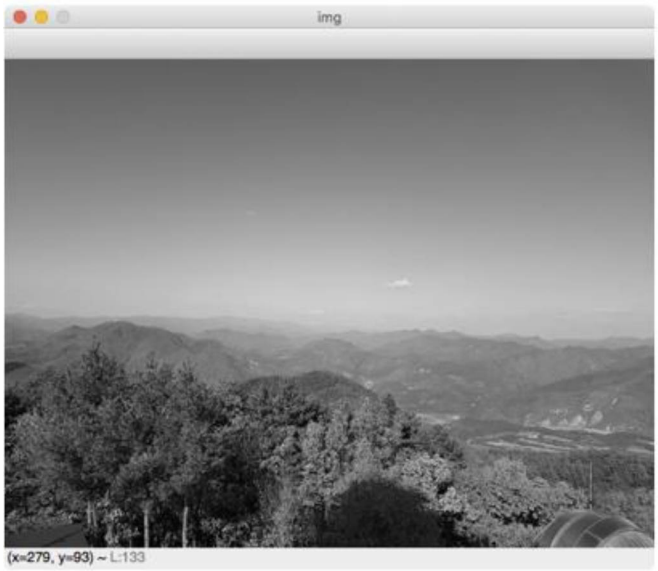
- cv2.calcHist(img, channel, mask, histSize, ranges)
  - img : input image, bind in list like [img]
  - channel : Channel to work on bind in list
    - 1channel: [0], 2channel:[0,1], 3channel:[0,1,2]
  - mask : Calculate histogram for only the pixels set at mask
  - histSize : Number of level (bin), show in list according to the number of channels
    - 1channel:[256], 2channel:[256, 256], 3channel:[256,256,256]
  - ranges : Range of values that each pixel can have, for RGB [0, 256]

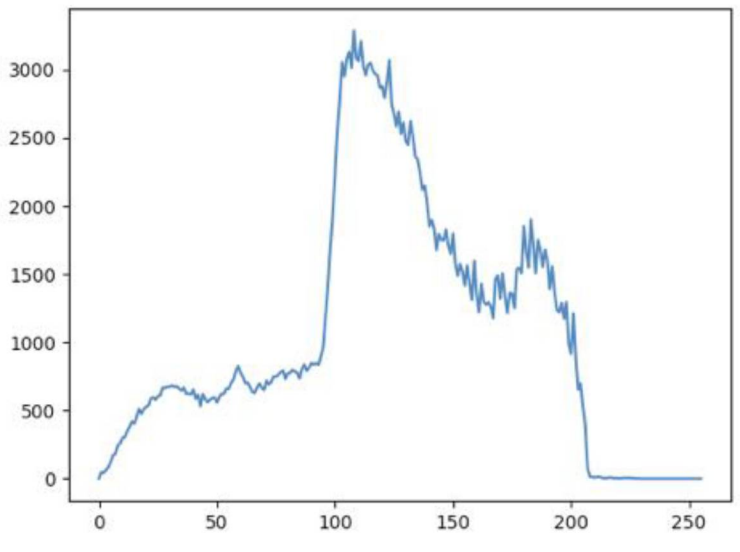
❖ Gray Scale 1 Channel Histogram

```
import cv2
import numpy as np
import matplotlib.pylab as plt

img = cv2.imread('../img/mountain.jpg', cv2.IMREAD_GRAYSCALE)
cv2.imshow('img', img)

hist = cv2.calcHist([img], [0], None, [256], [0,256])
plt.plot(hist)

print(hist.shape)
print(hist.sum(), img.shape)
plt.show()
```

❖ Gray Scale 1Channel Histogram

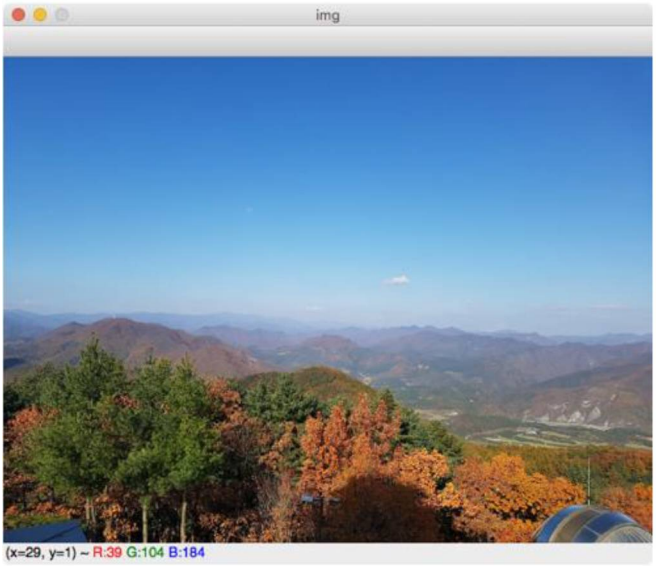
```
hist.shape: (256, 1)
hist.sum(): 270000.0 img.shape: (450, 600)
```

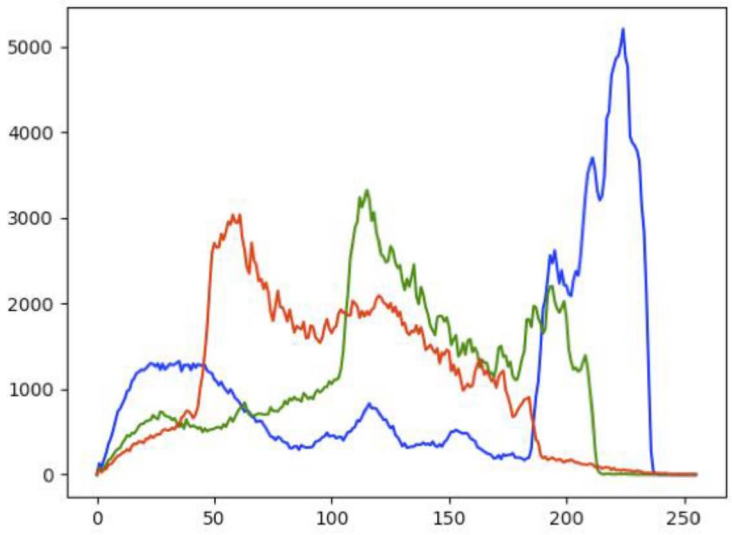




❖ Color Scale Histogram

```
import cv2
import numpy as np
import matplotlib.pylab as plt

img = cv2.imread('../img/mountain.jpg')
cv2.imshow('img', img)

channels = cv2.split(img)
colors = ('b', 'g', 'r')
for (ch, color) in zip (channels, colors):
    hist = cv2.calcHist([ch], [0], None, [256], [0, 256])
    plt.plot(hist, color = color)
plt.show()
```





❖ Mormalize

- Evenly distribute pixel
- If distribution of pixel is centered on certain area, quality bad.
  - Improve image quality

$$I_N = (I - Min)\frac{newMax - newMin}{Max - Min} + newMin$$

- I : 정규화 이전 값
- Min, Max : 정규화전 범위 최소 값, 최대 값
- $newMin, newMax$: 정규화 후 최소 값, 최대 값
- I_N: 정규화 이후 값

95	96	97	98	99	100

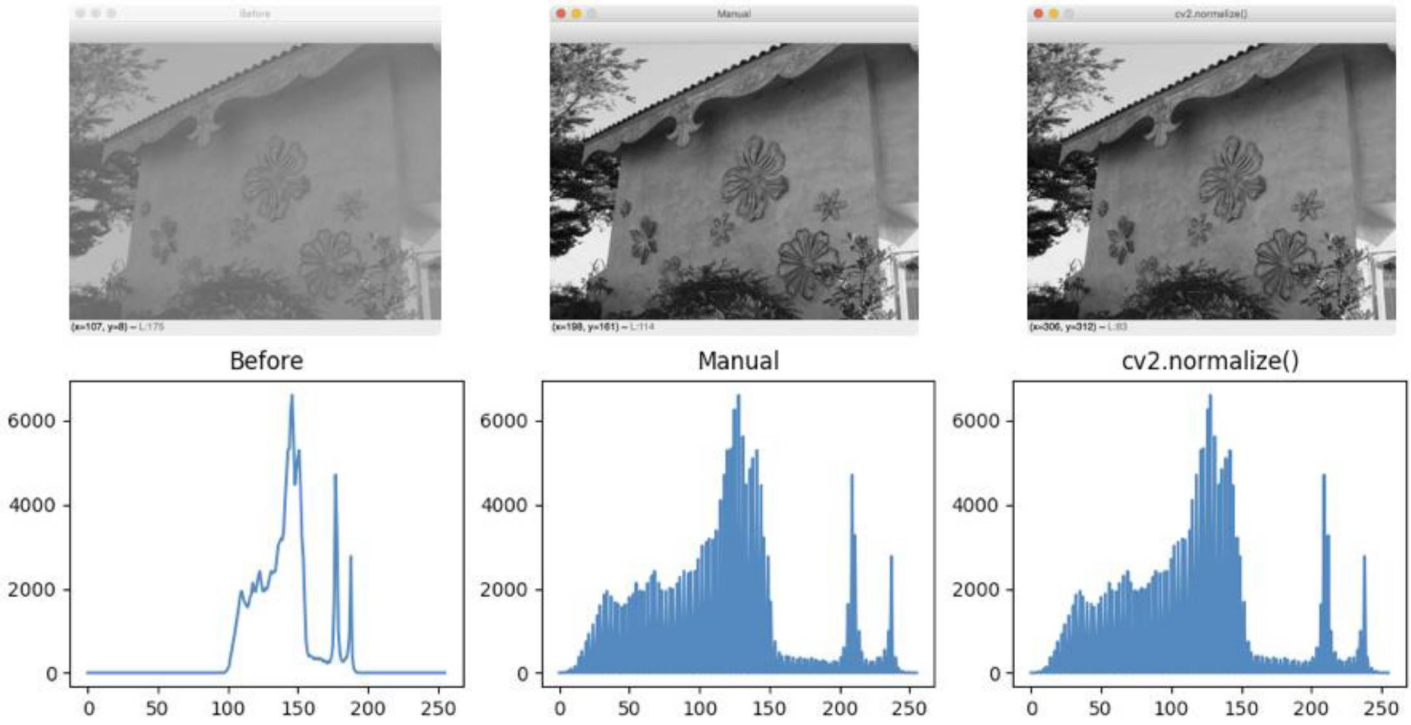
70	76	82	88	94	100

- dst = cv2.normalize(src, dst, alpha, beta, type_flag)
  - src : Previous data before mormalize
  - dst : Data after mormalize
  - alpha : Mormalizing area 1
  - beta : Mormalizing area 2, Not in use when it is not sectional mormalization
  - type_flag : Flag integer selected for mormalization algorithm
    - cv2.NORM_MINMAX : Mormalize by alpha & beta area
    - cv2.NORM_L1 : Divide by total addition, alpha = Addition of all mormalization
    - cv2.NORM_L2 :Mormalize unit vector
    - cv2.NORM_INF : Divide by maximum number

- Histogram Normalize

```
import cv2
import numpy as np
import matplotlib.pylab as plt

img = cv2.imread('../img/abnormal.jpg', cv2.IMREAD_GRAYSCALE)

img_f = img.astype(np.float32)
img_norm = ((img_f - img_f.min()) * (255) / (img_f.max() -
img_f.min())).astype(np.uint8)

img_norm2 = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)

hist = cv2.calcHist([img], [0], None, [256], [0, 256])
hist_norm = cv2.calcHist([img_norm], [0], None, [256], [0, 256])
hist_norm2 = cv2.calcHist([img_norm2], [0], None, [256], [0, 256])

cv2.imshow('Before', img)
cv2.imshow('Manual', img_norm)
cv2.imshow('cv2.normalize()', img_norm2)

hists = {'Before' : hist, 'Manual':hist_norm,
'cv2.normalize()':hist_norm2}
for i, (k, v) in enumerate(hists.items()):
    plt.subplot(1,3,i+1)
    plt.title(k)
    plt.plot(v)
plt.show()
```

❖ Histogram Normalize



❖ Equalize

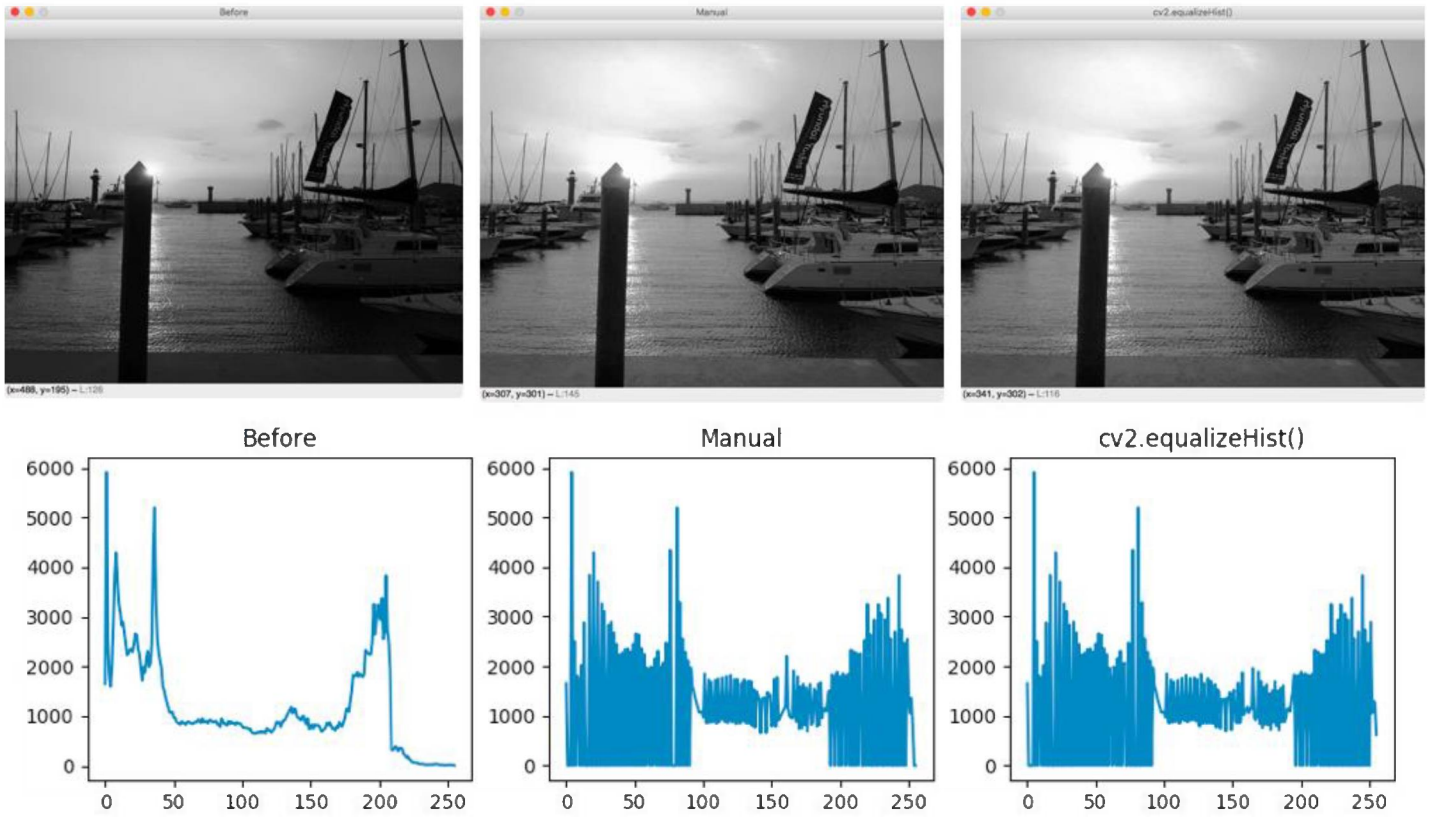
- Control height of pixel distribution not width
- Effective for lighting
- dst = cv.equalizeHist(src[, dst])
  - src : Image subject, 8 bit 1 channel
  - dst : Result image

- $H'(v) = round\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$
  - $cdf(v)$: 히스토그램 누적 함수
  - cdf_{min} : 누적 최소 값, 1
  - $M \times N$: 픽셀 수, 폭 x 높이
  - L : 분포 영역, 256
  - $round(v)$: 반올림
  - $H'(v)$: 아퀄라이즈된 히스토그램 값

❖ Equalize Gray Scale

```
import cv2
import numpy as np
import matplotlib.pylab as plt

img = cv2.imread('../img/yate.jpg', cv2.IMREAD_GRAYSCALE)
rows, cols = img.shape[:2]

hist = cv2.calcHist([img], [0], None, [256], [0, 256])
cdf = hist.cumsum() cdf_m = np.ma.masked_equal(cdf, 0) cdf_m =
(cdf_m - cdf_m.min()) /(rows * cols) * 255
cdf = np.ma.filled(cdf_m,0).astype('uint8')
img2 = cdf[img]
img3 = cv2.equalizeHist(img)

hist2 = cv2.calcHist([img2], [0], None, [256], [0, 256])
hist3 = cv2.calcHist([img3], [0], None, [256], [0, 256])

cv2.imshow('Before', img)
cv2.imshow('Manual', img2)
cv2.imshow('cv2.equalizeHist()', img3)
hists = {'Before':hist, 'Manual':hist2,
'cv2.equalizeHist()':hist3}
for i, (k, v) in enumerate(hists.items()):
    plt.subplot(1,3,i+1)
    plt.title(k)
    plt.plot(v)
plt.show()
```

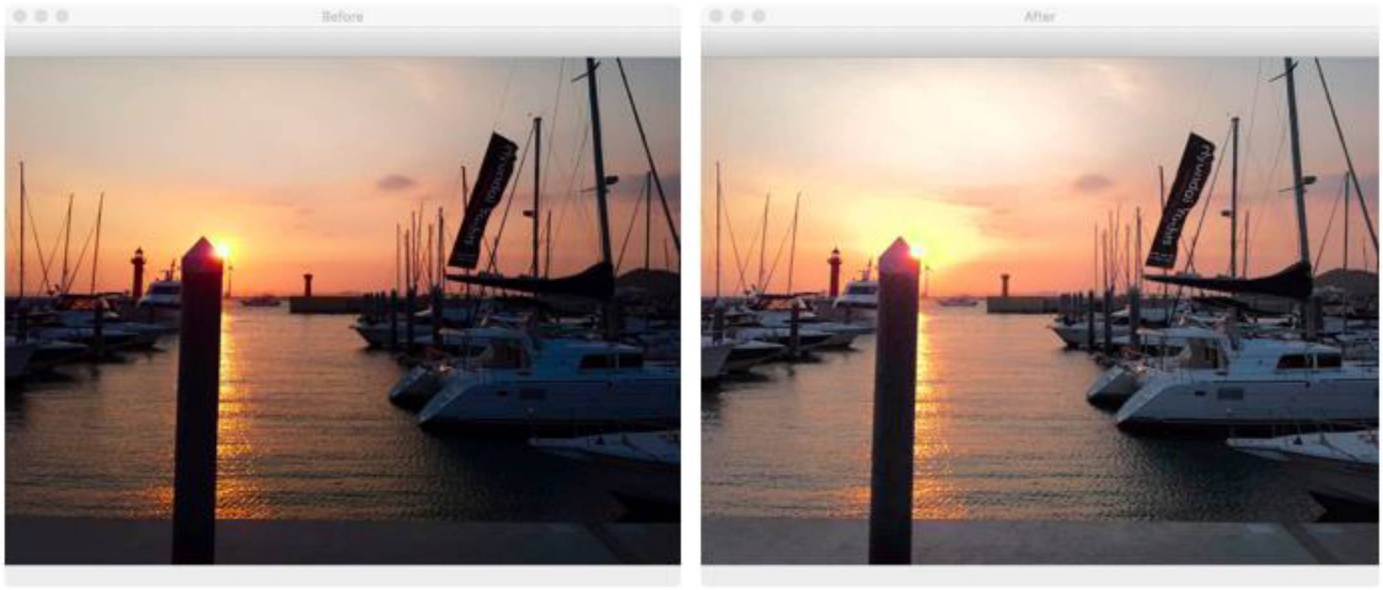
❖ Equalize Gray Scale



❖ Equalize Color(YUV) Scale

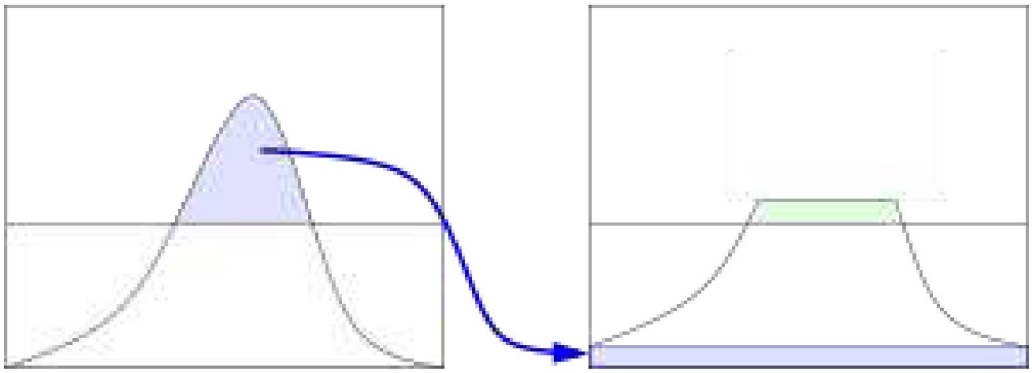
```
import numpy as np, cv2

img = cv2.imread('../img/yate.jpg')

img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)

img_yuv[:,:,0] = cv2.equalizeHist(img_yuv[:,:,0])

img2 = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
cv2.imshow('Before', img)
cv2.imshow('After', img2)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Equalize Color(YUV) Scale



❖ CLAHE

- Contrast Limiting Adaptive Histogram Equalization
- When applying equalizer, prevent light area as much as possible
- If it goes over limit value of histogram, distribute to other scale
- cv2.createCLAHE(clipLimit, tileGridSize) :generate CLAHE
  - clipLimit : Contrast Limit boundary value, basic 40.0
  - tileGridSize : area size, basic 8x8
- CLAHE : CLAHE algorithm object
  - apply(src) : apply CLAHE
    - src : input image

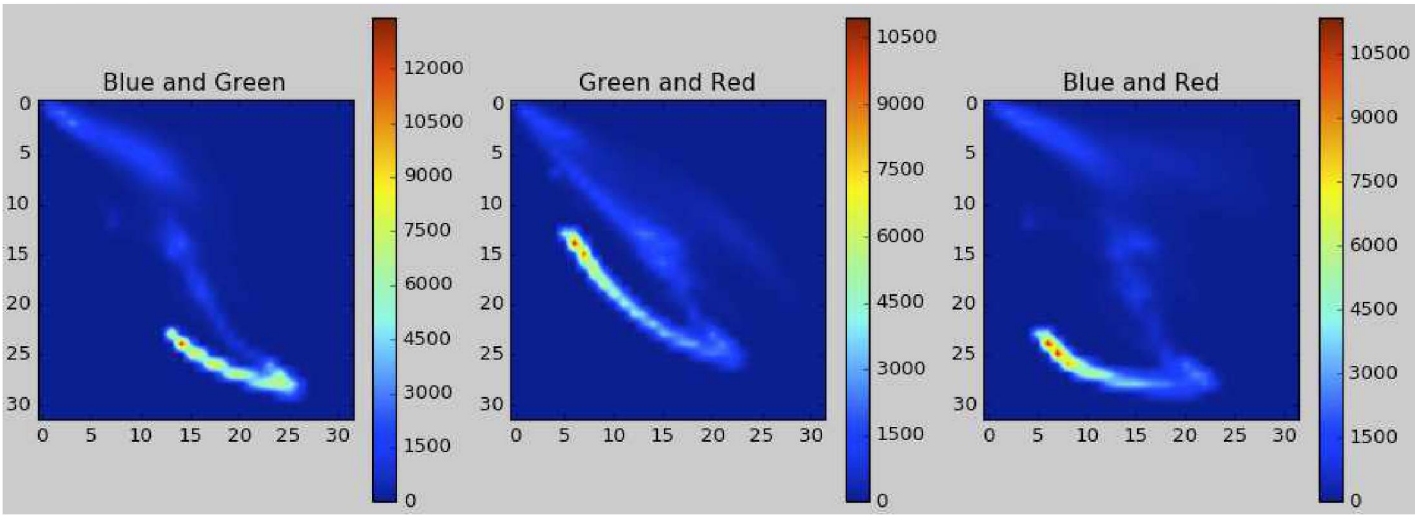


❖ Improve Quality of image with CLAHE

```
import cv2
import numpy as np
import matplotlib.pylab as plt

img = cv2.imread('../img/bright.jpg')
img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)

img_eq = img_yuv.copy()
img_eq[:,:,0] = cv2.equalizeHist(img_eq[:,:,0])
img_eq = cv2.cvtColor(img_eq, cv2.COLOR_YUV2BGR)

img_clahe = img_yuv.copy()
clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
#generate CLAHE
img_clahe[:,:,0] = clahe.apply(img_clahe[:,:,0])           #apply
CLAHE
img_clahe = cv2.cvtColor(img_clahe, cv2.COLOR_YUV2BGR)

cv2.imshow('Before', img)
cv2.imshow('CLAHE', img_clahe)
cv2.imshow('equalizeHist', img_eq)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ 2D Histogram

```
import cv2, matplotlib.pylab as plt

plt.style.use('classic')            # Use 1.x style for color
style
img = cv2.imread('../img/mountain.jpg')
plt.subplot(131)
hist = cv2.calcHist([img], [0,1], None, [32,32], [0,256,0,256])
p = plt.imshow(hist)
plt.title('Blue and Green')
plt.colorbar(p)

plt.subplot(132)
hist = cv2.calcHist([img], [1,2], None, [32,32], [0,256,0,256])
p = plt.imshow(hist)
plt.title('Green and Red')
plt.colorbar(p)

plt.subplot(133)
hist = cv2.calcHist([img], [0,2], None, [32,32], [0,256,0,256])
p = plt.imshow(hist)
plt.title('Blue and Red')
plt.colorbar(p)
plt.show()
```



❖ Back Projection

- Histogram of HS at HSV
- ROI/all ratio
- Ration for non-selected area is mostly 0
- Map original pixel with ratio
- Pixel excluding selected color is 0
- Fit to use as mask
- cv2.calcBackProject(img, channel, hist, ranges, scale)
  - img : input image, Bind as list like [img]
  - channel : Channel to work on, Bind as list like
    - 1channel: [0], 2channel:[0,1], 3channel:[0,1,2]
  - hist : histogram used for reverse projection
  - ranges : Range of value for each pixel
  - scale : Coefficient of scale that will be applied to result
- Masking by HSV reverse projection <1/2>

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

win_name = 'back_projection'
img = cv2.imread('../img/pump_horse.jpg')
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
draw = img.copy()

def masking(bp, win_name):
    disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
    cv2.filter2D(bp,-1,disc,bp)
    _, mask = cv2.threshold(bp, 1, 255, cv2.THRESH_BINARY)
    result = cv2.bitwise_and(img, img, mask=mask)
    cv2.imshow(win_name, result)
def backProject_manual(hist_roi):
    hist_img = cv2.calcHist([hsv_img], [0,1], None,[180,256],
[0,180,0,256])
```

❖ Back Projection

- Masking by HSV reverse projection <2/2>

```
   hist_rate = hist_roi/ (hist_img + 1)
    h,s,v = cv2.split(hsv_img)
    bp = hist_rate[h.ravel(), s.ravel()]
    bp = np.minimum(bp, 1)
    bp = bp.reshape(hsv_img.shape[:2])
    cv2.normalize(bp,bp, 0, 255, cv2.NORM_MINMAX)
    bp = bp.astype(np.uint8)
    masking(bp,'result_manual')
def backProject_cv(hist_roi):
     bp = cv2.calcBackProject([hsv_img], [0, 1], hist_roi,  [0,
180, 0, 256], 1)
    masking(bp,'result_cv')
(x,y,w,h) = cv2.selectROI(win_name, img, False)
if w > 0 and h > 0:
    roi = draw[y:y+h, x:x+w]
    cv2.rectangle(draw, (x, y), (x+w, y+h), (0,0,255), 2)
    hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
    hist_roi = cv2.calcHist([hsv_roi],[0, 1], None, [180, 256],
[0, 180, 0, 256] )
    backProject_manual(hist_roi)
    backProject_cv(hist_roi)
cv2.imshow(win_name, draw)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Compare Histogram

- Compare histogram to figure out similarity among images
- cv2.compareHist(hist1, hist2, method)
  - hist1, hist2 : two histogram to compare, size and dimension should be same.
  - method : Flag integer for comparing algorithm
    - cv2.HISTCMP_CORREL : correlation
      - 1 : Complete match, -1 : Max un-match, 0: no relation
    - cv2.HISTCMP_CHISQR : Chi square
      - 0 : Complete match, Big value (unknown): Max un-match
      - cv2.HISTCMP_INTERSECT : Crossing
      - 1 : Complete match, 0 : Max un-match (When mormalized as 1)
    - cv2.HISTCMP_BHATTACHARYYA : 바타차야
      - 0: Complete match, 1 : Max un-match
      - cv2.HISTCMP_HELLINGER : Same as HISTCMP_BHATTACHARYYA

❖ Compare Histogram <1/2>

```
import cv2, numpy as np
import matplotlib.pylab as plt
img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/taekwonv2.jpg')
img3 = cv2.imread('../img/taekwonv3.jpg')
img4 = cv2.imread('../img/dr_ochanomizu.jpg')
cv2.imshow('query', img1)
imgs = [img1, img2, img3, img4]
hists = []
for i, img in enumerate(imgs) :
    plt.subplot(1,len(imgs),i+1)
    plt.title('img%d'% (i+1))
    plt.axis('off')
    plt.imshow(img[:,:,::-1])
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    hist = cv2.calcHist([hsv], [0,1], None, [180,256], [0,180,0,
256])
```

❖ Compare Histogram <1/2>

```
cv2.normalize(hist, hist, 0, 1, cv2.NORM_MINMAX)
    hists.append(hist)

query = hists[0]
methods = {'CORREL' :cv2.HISTCMP_CORREL,
'CHISQR':cv2.HISTCMP_CHISQR,
           'INTERSECT':cv2.HISTCMP_INTERSECT,
'BHATTACHARYYA':cv2.HISTCMP_BHATTACHARYYA}
for j, (name, flag) in enumerate(methods.items()):
    print('%-10s'%name, end='\t')
    for i, (hist, img) in enumerate(zip(hists, imgs)):
        ret = cv2.compareHist(query, hist, flag)
        if flag == cv2.HISTCMP_INTERSECT:
            ret = ret/np.sum(query)
        print("img%d:%7.2f"% (i+1 , ret), end='\t')
    print()
plt.show()
```


```
CORREL          img1:   1.00    img2:   0.70    img3:   0.56    img4:   0.23
CHISQR          img1:   0.00    img2:  67.33    img3:  35.71    img4:1129.49
INTERSECT       img1:   1.00    img2:   0.54    img3:   0.40    img4:   0.18
BHATTACHARYYA   img1:   0.00    img2:   0.48    img3:   0.47    img4:   0.79
```

Image Processing Basic

1. ROI(Region Of Interest)
2. Color Space
3. Threshold
4. Image Arithmetic
5. Histogram
6. **Workshop**

❖ Monster Face Synthesis

- Blend two faces into one looking naturally.
- File to use:
  - img/man_face.jpg
  - img/skull.jpg
- Result Example



- Hint
  - 0%~50%, 50%~0% Alpha blending for matching areas of two faces

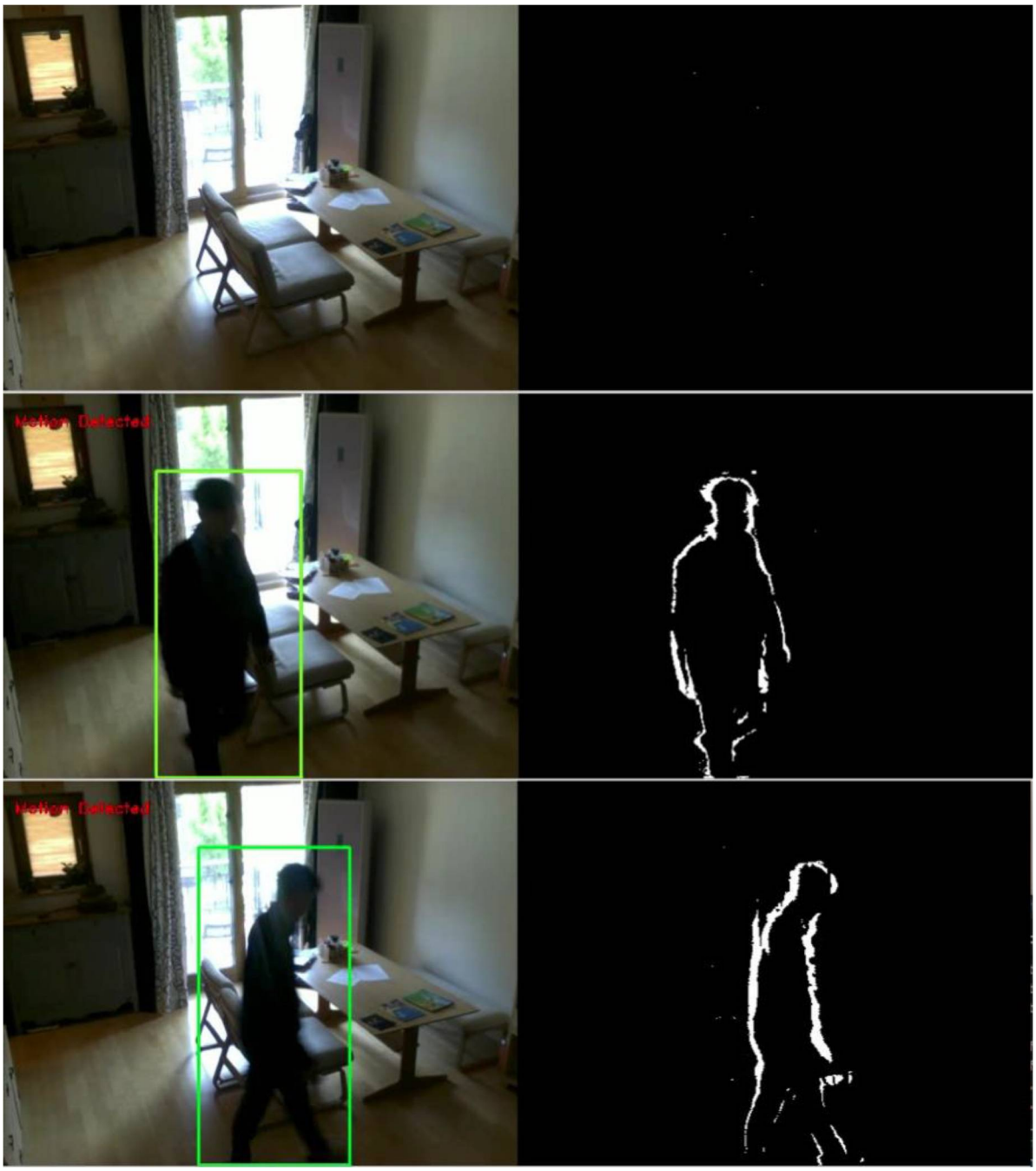
❖ Motion Detecting CCTV

- Make camera detecting motion .
- Result example

❖ Motion Detecting CCTV

- Detect motion object or human with camera and mark the area.
- When motion is detected, print "Motion Detected " message in red
- Divide one screen into two, left and right and show screen with motion area marked on one screen and for the other area show only pixel accordingly; show them in one screen.
- Hint
  - Get difference image to find differences.

- It cannot be solved with before/current image differences.
- It will define it as motion when A and B has differences and B and C has differences
- Two factors that can be set:
  - Size of pixel differences
  - Number of pixel differences

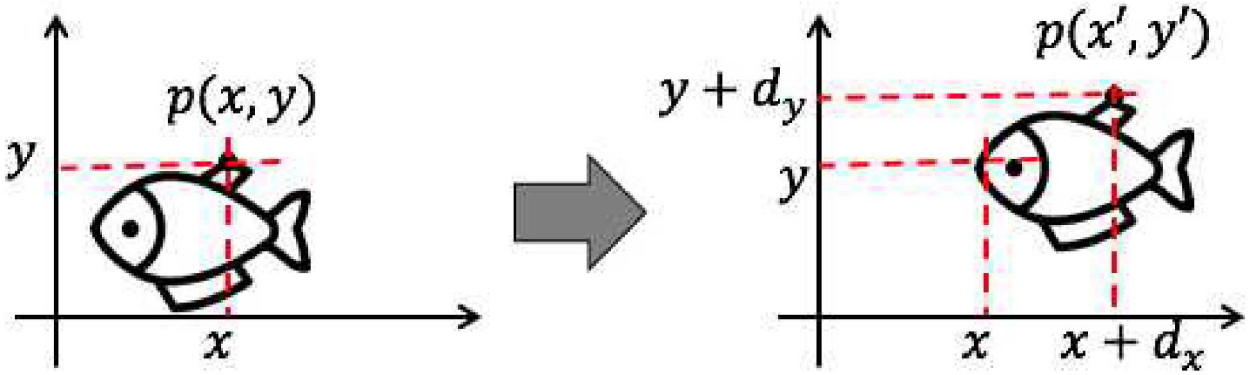


Geometric Transform

1. **Translate, Scaling, Rotate**
2. Warping
3. Lens Distortion
4. Workshop

❖ Translate

- Move Image
  - Coordination to move = Original coordination + Distance to move
- Equation
  - $x' = x + d_x$
  - $y' = y + d_y$
- Determinant
  - $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \end{bmatrix} = \begin{bmatrix} 1x + 0y + 1d_x \\ 0x + 1y + 1d_y \end{bmatrix}$



❖ Translate

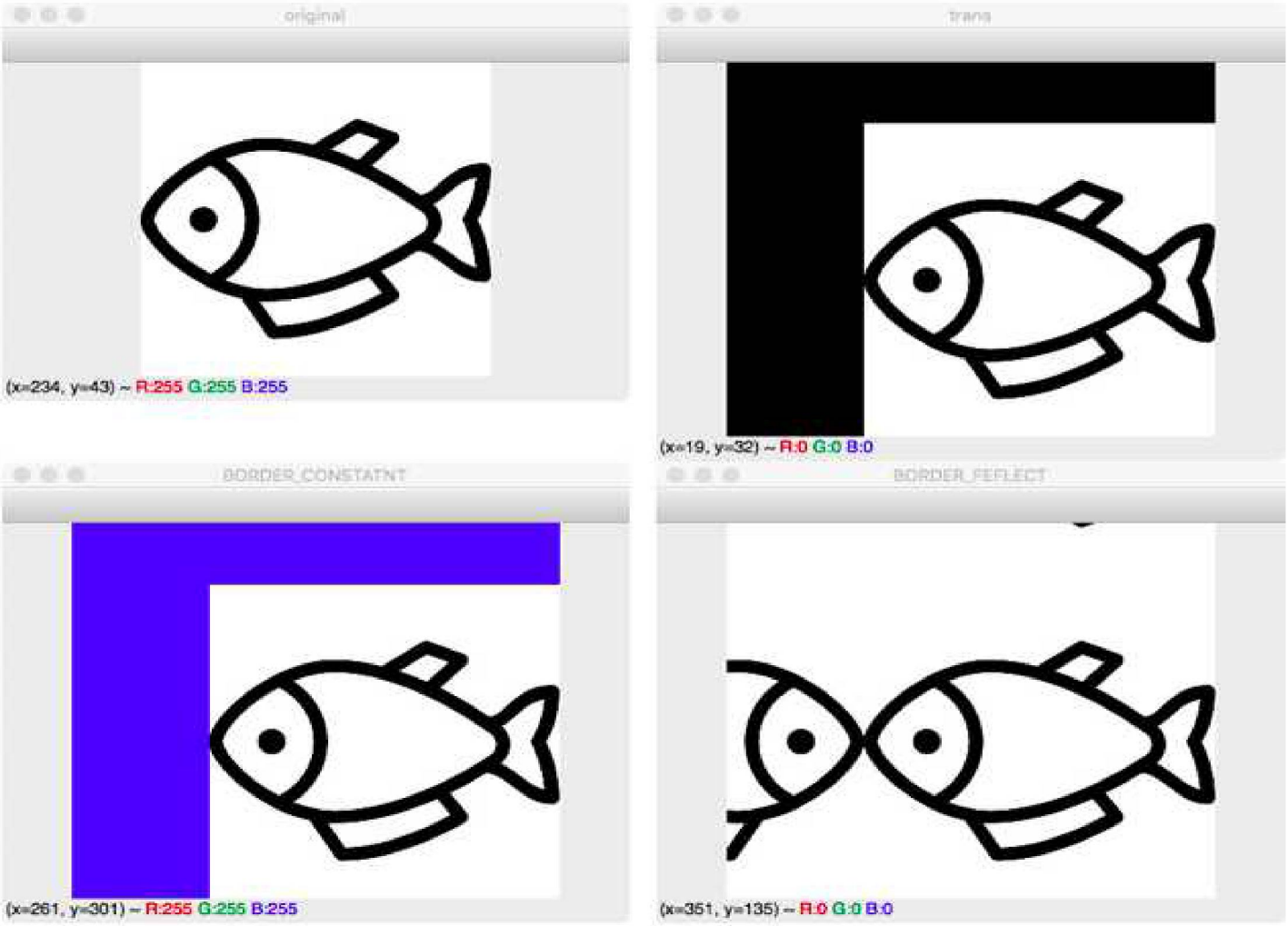
- dst = cv2.warpAffine(src, mtrx, dsize [, dst, flags, borderMode, borderValue])
  - src: Original image, NumPy array
  - mtrx: 2 x 3 matrix of change, NumPy array, dtype=float32
  - dsize: Result image size, tuple(width, height)
  - flags: Interpolation algorithm selection flag
    - cv2.INTER_LINEAR: base value, use distance weight for near 4 pixels
    - cv2.INTER_NEAREST: Use closest pixel value
    - cv2.INTER_AREA: Re-sampling using pixel area relationship
    - cv2.INTER_CUBIC: use distance weight for near 16 pixels
    - cv2.INTER_LANCZOS4: Lanco ‘s algorithm using near 8 pixels

- borderMode : Border area revision flag
  - cv2.BORDER_CONSTANT : Fixed color value (999|12345|999)
  - cv2.BORDER_REPLICATE : Copy boarder (111|12345|555)
  - cv2.BORDER_WRAP : repeat (345|12345|123)
  - cv2.BORDER_REFLECT : reflect (321|12345|543)
- borderValue : Color value used for cv2.BORDER_CONSTANT, base value = 0
- dst : Result image, NumPy array

❖ Translate

```
import cv2
import numpy as np
img = cv2.imread('../img/fish.jpg')
rows,cols = img.shape[0:2]  # Size of image

dx, dy = 100, 50            # Moving distance of pixel

mtrx = np.float32([[1, 0, dx],
                   [0, 1, dy]])
dst = cv2.warpAffine(img, mtrx, (cols+dx, rows+dy))
dst2 = cv2.warpAffine(img, mtrx, (cols+dx, rows+dy), None,
cv2.INTER_LINEAR, cv2.BORDER_CONSTANT, (255,0,0) )
dst3 = cv2.warpAffine(img, mtrx, (cols+dx, rows+dy), None,
cv2.INTER_LINEAR, cv2.BORDER_REFLECT)

cv2.imshow('original', img)
cv2.imshow('trans',dst)
cv2.imshow('BORDER_CONSTATNT', dst2)
cv2.imshow('BORDER_FEFLECT', dst3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

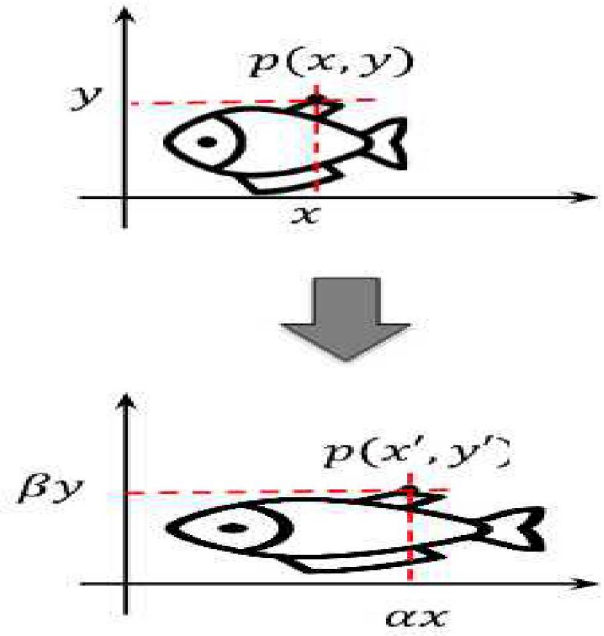
❖ Translate



❖ Scaling

- Zooming: α · β Change Matrix
  - $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- Dst=cv2.resize(src, dsize, fx, fy, interpolation)
  - Src: Input image
  - Dsize : Size of print image (w,h)
  - Fx, fy : Ratio, if omitted apply dsize

- Interpolation
  - cv2.INTER_LINEAR :default
  - cv2.INTEAR_NEAREST
  - cv2.INTEAR_AREA
  - cv2.INTEAR_CUBIC
  - cv2.INTEAR_LANCZOS4
- dst : Result img



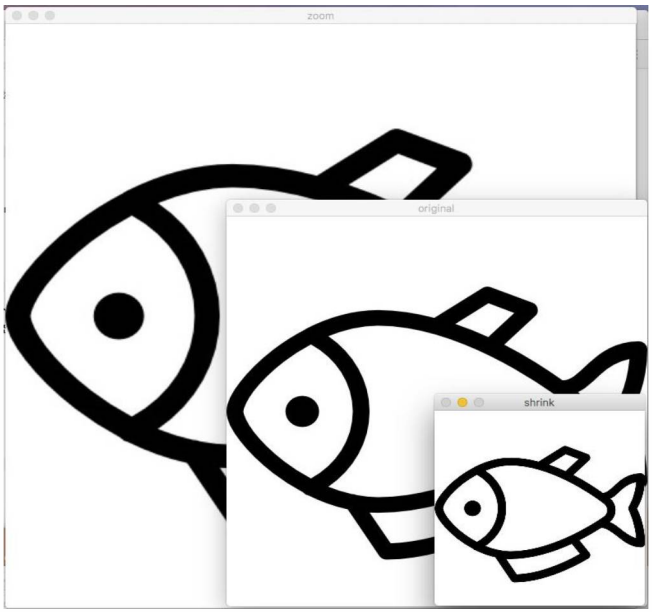
❖ Scaling

- Change Matrix Example

```
img = cv2.imread('../img/fish.jpg')
height, width = img.shape[:2]
m_shrink = np.float32([[0.5, 0, 0],
                       [0, 0.5, 0]])  # matrix
m_zoom = np.float32([[1.5, 0, 0],
                     [0, 1.5, 0]])  # matrix

shrink = cv2.warpAffine(img, m_shrink, (int(width*0.5),
int(height*0.5)) )
zoom = cv2.warpAffine(img, m_zoom, (int(width*1.5),
int(height*1.5)) )

cv2.imshow("original", img)
cv2.imshow("shrink", shrink)
cv2.imshow("zoom", zoom)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Scaling

- Change Matrix Example <Result>



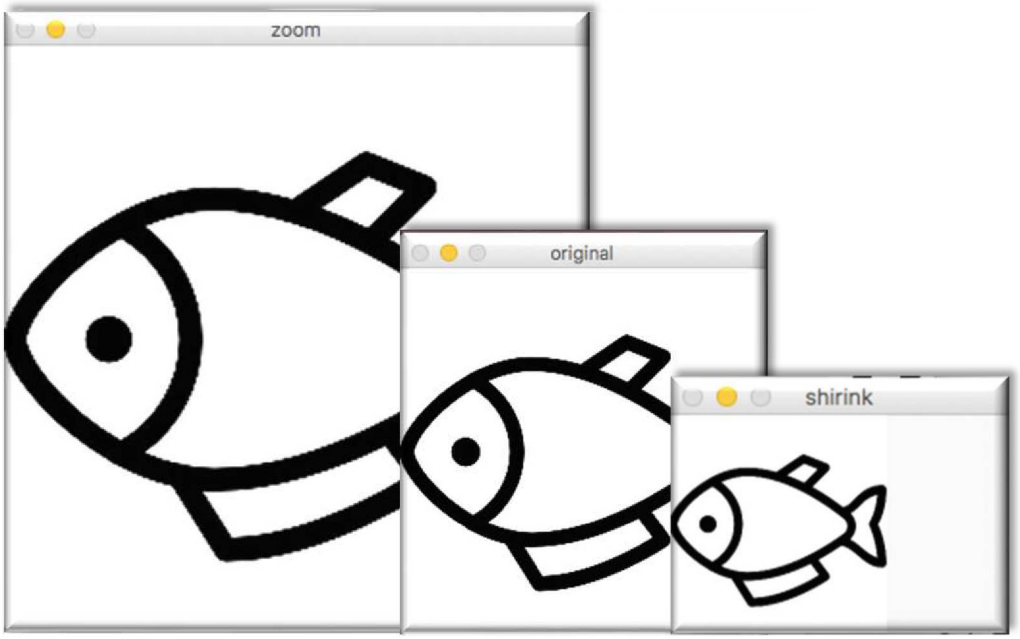
Scaling

- cv2.resize function Example

```
import cv2
import numpy as np
img = cv2.imread('../img/fish.jpg')
height, width = img.shape[:2]
shrink = cv2.resize(img,None,fx=0.5, fy=0.5,
                                    interpolation =
cv2.INTER_AREA)
zoom = cv2.resize(img,(int(1.5*width), int(1.5*height)),
                                        interpolation =
cv2.INTER_CUBIC)
cv2.imshow("original", img)
cv2.imshow("shirink", shrink)
cv2.imshow("zoom", zoom)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

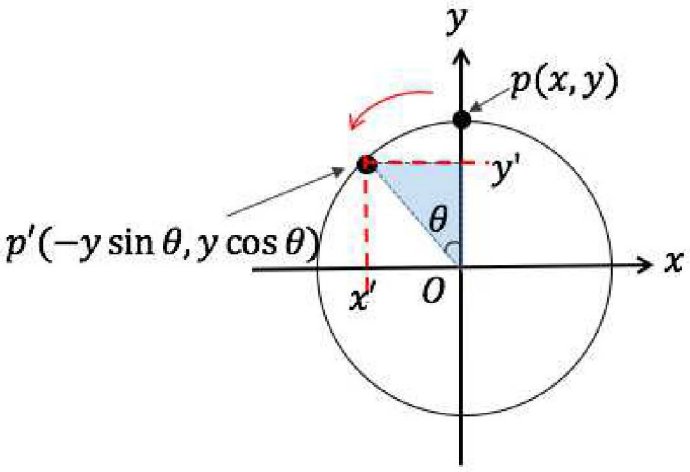
❖ Scaling

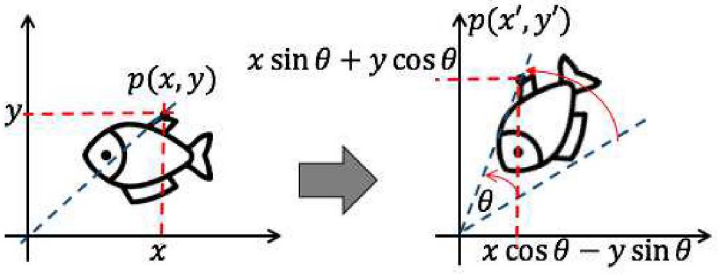
- cv2.resize function Example <Result>

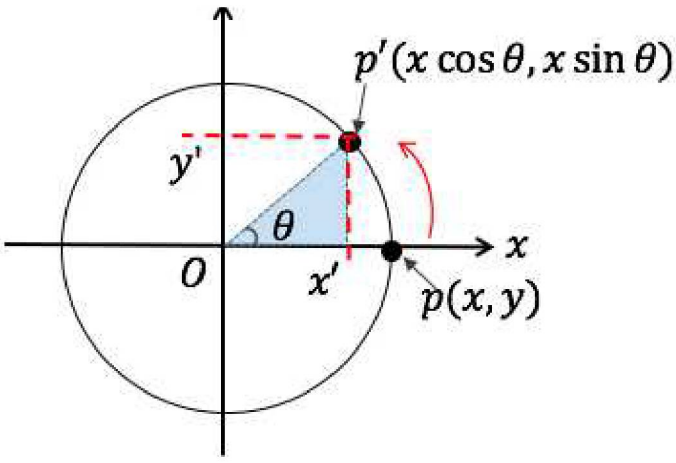


- Rotate
  - Rotation Change Matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$







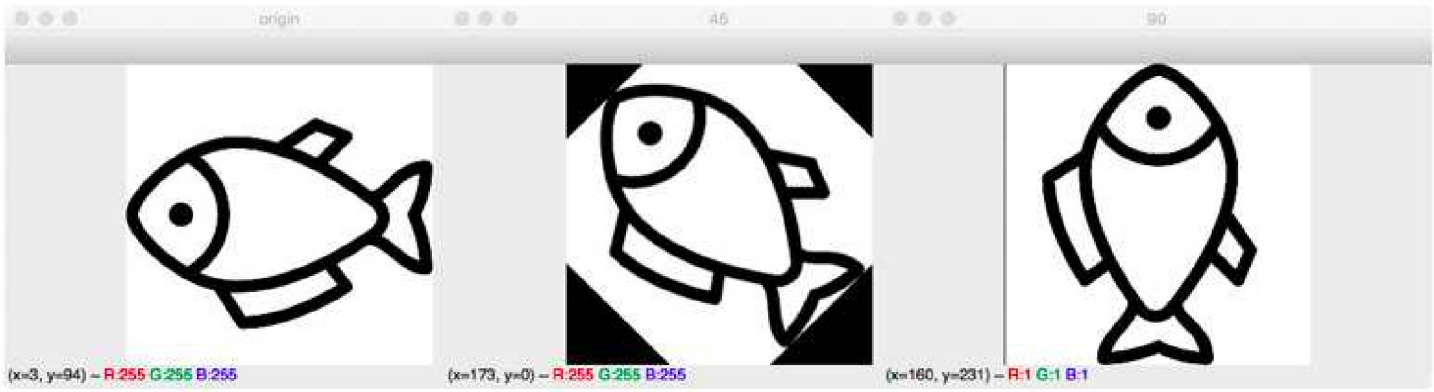
❖ Rotate

- Change Matrix

```
import cv2, numpy as np

img = cv2.imread('../img/fish.jpg')
rows,cols = img.shape[0:2]

d45 = 45.0 * np.pi / 180    # 45도
d90 = 90.0 * np.pi / 180    # 90도
m45 = np.float32( [[ np.cos(d45), -1* np.sin(d45), rows//2],
                   [np.sin(d45), np.cos(d45),    -1*cols//4]])
m90 = np.float32( [[ np.cos(d90), -1* np.sin(d90), rows],
                   [np.sin(d90), np.cos(d90), 0]])

r45 = cv2.warpAffine(img,m45,(cols,rows))
r90 = cv2.warpAffine(img,m90,(rows,cols))
cv2.imshow("origin", img)
cv2.imshow("45", r45)
cv2.imshow("90", r90)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Rotate

- Change Matrix



❖ Rotate

- Turnaround changed matrix for rotation
- Able to set rotating function, rotating axis, scale
- mtrx = cv2.getRotationMatrix2D(center, angle, scale)
  - center : center of rotating axis, coordination, tuple (x, y)
  - angle : rotating angle, sexagesimal system
  - scale : Zooming scale

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

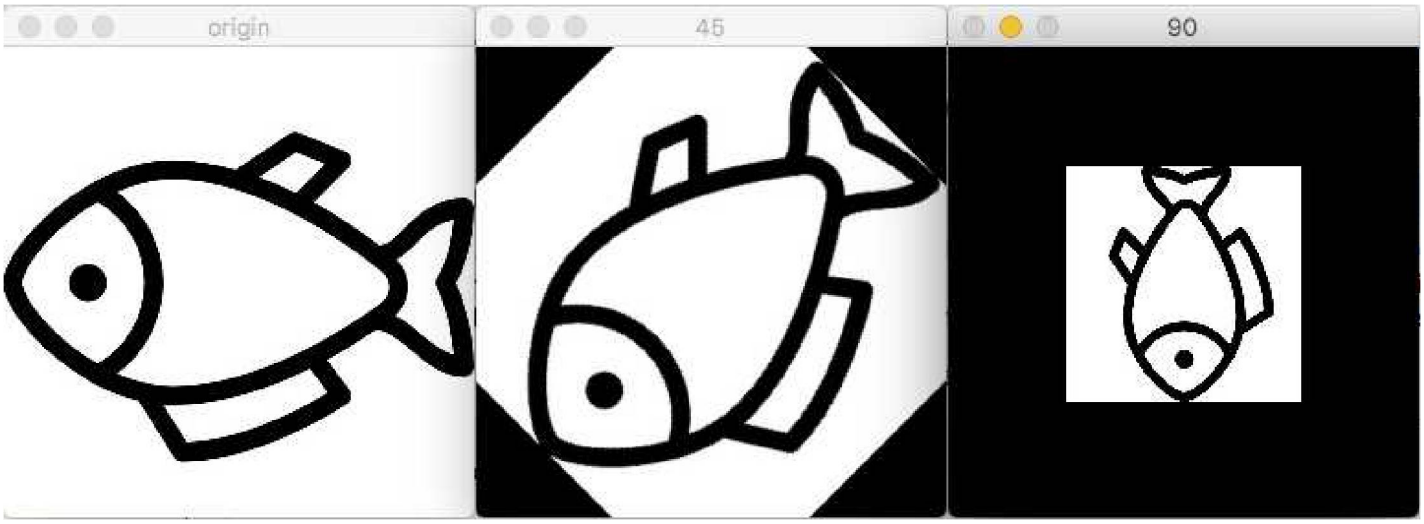
$$\alpha = scale \cdot \cos\theta,$$
$$\beta = scale \cdot \sin\theta$$

- Rotating Function Example

```
img = cv2.imread('../img/fish.jpg')
rows,cols = img.shape[0:2]

matrix_45 = cv2.getRotationMatrix2D((cols/2,rows/2),45,1)
matrix_90 = cv2.getRotationMatrix2D((cols/2,rows/2),90,0.5)

print(matrix_45)
r45 = cv2.warpAffine(img, matrix_45,(cols,rows))
r90 = cv2.warpAffine(img, matrix_90,(cols,rows))

cv2.imshow('origin',img)
cv2.imshow("45", r45)
cv2.imshow("90", r90)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Rotate

- Rotating Function Example <Result>

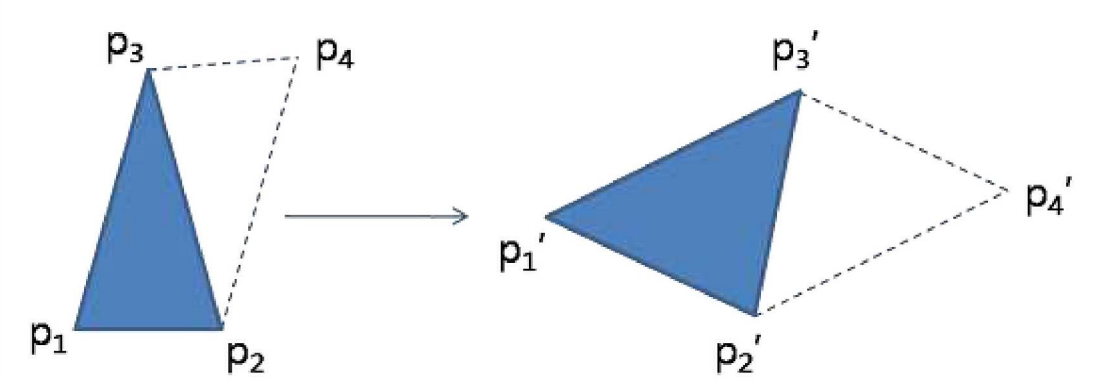


Geometric Transform

1. Translate, Scaling, Rotate
2. **Warping**
3. Lens Distortion
4. Workshop

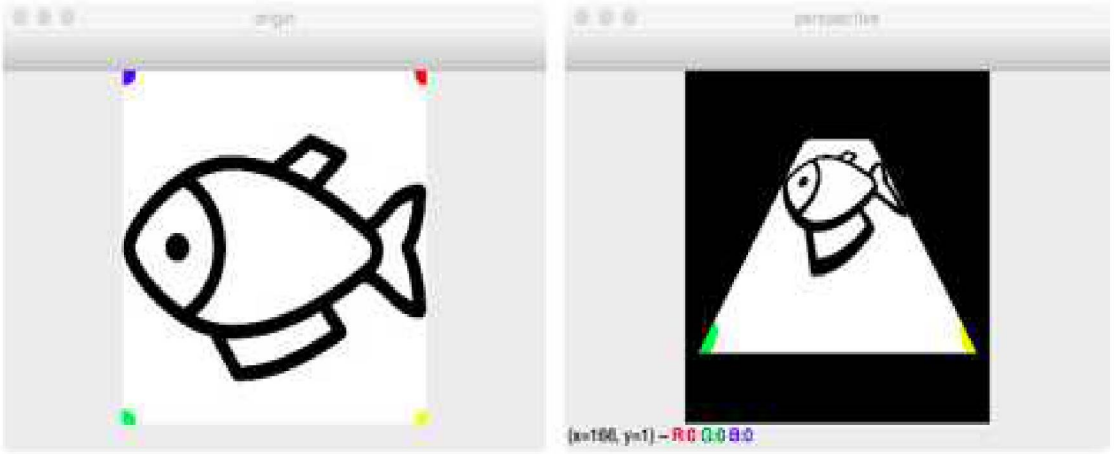
❖ Affine Transform

- Maintain parallelism of lines, image transform
- It includes all of move, enlarge, scale
- Transformation maintaining line, ration of length and parallelism
- Map 3 coordination within original image to result image
- matrix = cv2.getAffineTransform(pts1, pts2)
  - pts1: 3 coordination before transform , 3 x 2 NumPy array(float32)
  - pts2: 3 coordination after transform, Same as pts1
  - matrix: Turnaround transform array, 2 x 3 array




```
import cv2
import numpy as np
from matplotlib import pyplot as plt

file_name = '../img/fish.jpg'
img = cv2.imread(file_name)
rows, cols = img.shape[:2]
pts1 = np.float32([[100, 50], [200, 50], [100, 200]])
pts2 = np.float32([[80, 70], [210, 60], [250, 120]])
cv2.circle(img, (100,50), 5, (255,0), -1)
cv2.circle(img, (200,50), 5, (0,255,0), -1)
cv2.circle(img, (100,200), 5, (0,0,255), -1)
mtrx = cv2.getAffineTransform(pts1, pts2)
dst = cv2.warpAffine(img, mtrx, (int(cols*1.5), rows))
cv2.imshow('origin',img)
cv2.imshow('affin', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Perspective Transform



- Giving Scanning Effect < 1/2 >

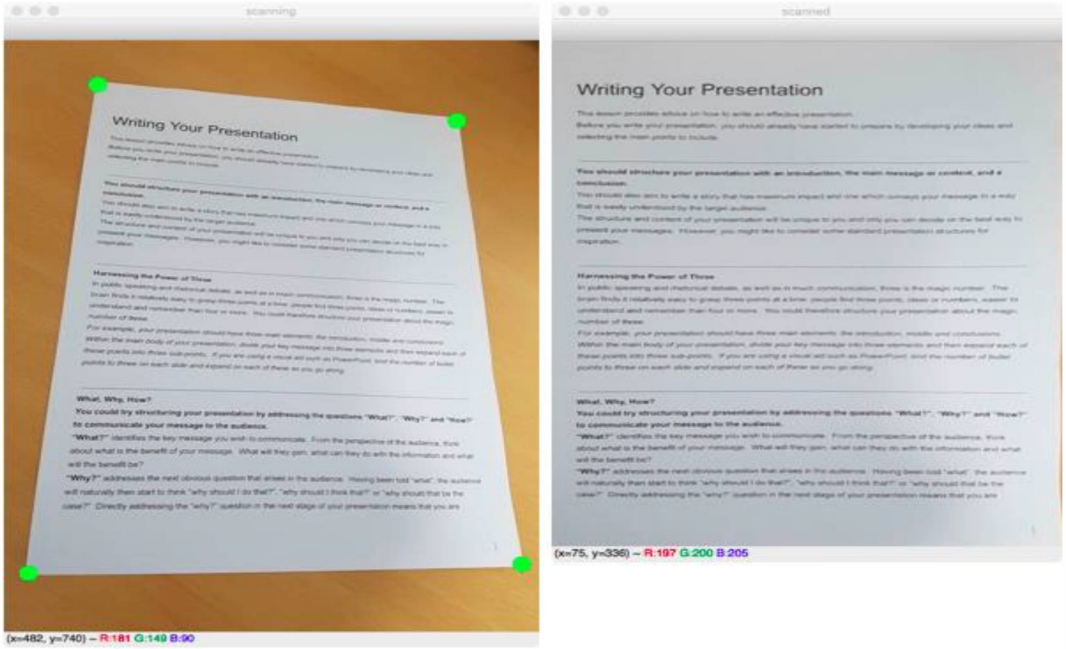
```
import cv2
import numpy as np

win_name = "scanning"
img = cv2.imread("../img/paper.jpg")
rows, cols = img.shape[:2]
draw = img.copy()
pts_cnt = 0
pts = np.zeros((4,2), dtype=np.float32)

def onMouse(event, x, y, flags, param):
    global  pts_cnt
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.circle(draw, (x,y), 10, (0,255,0), -1)
        cv2.imshow(win_name, draw)

        pts[pts_cnt] = [x,y]
        pts_cnt+=1
        if pts_cnt == 4:
            sm = pts.sum(axis=1)
            diff = np.diff(pts, axis = 1)
topLeft = pts[np.argmin(sm)]
bottomRight = pts[np.argmax(sm)]
topRight = pts[np.argmin(diff)]
```

❖ Perspective Transform

- Giving Scanning Effect < 2/2 >

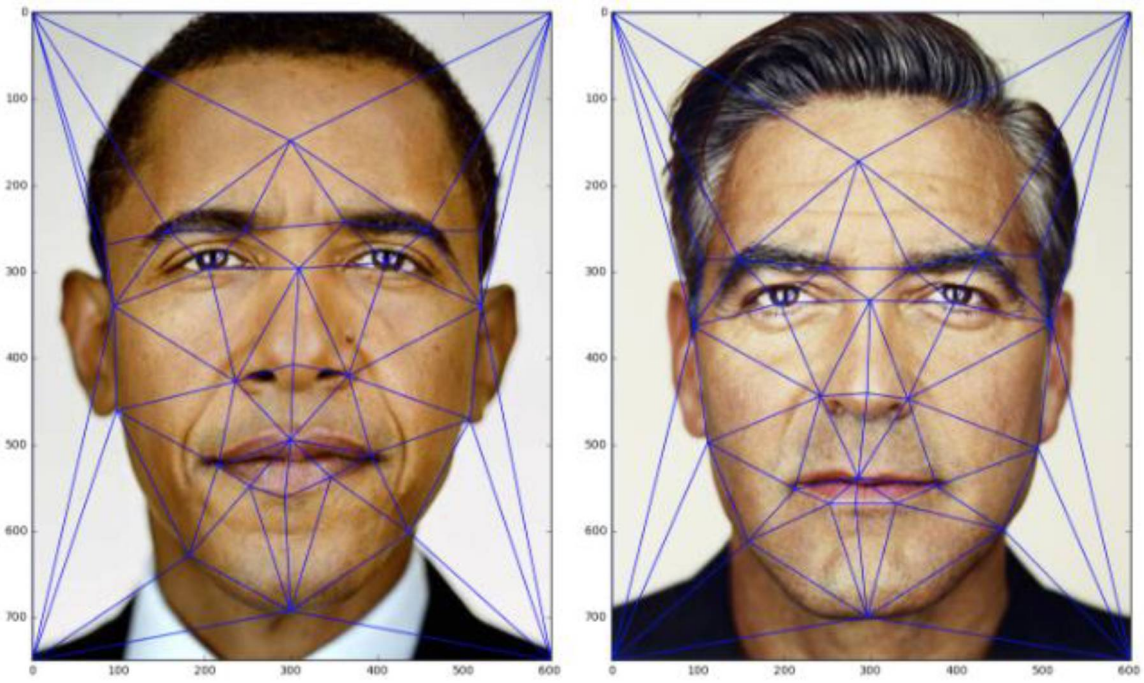
```
bottomLeft = pts[np.argmax(diff)]
pts1 = np.float32([topLeft, topRight, bottomRight , bottomLeft])
w1 = abs(bottomRight[0] - bottomLeft[0])   # 상단 좌우 좌표간의 거리
w2 = abs(topRight[0] - topLeft[0])         # 하당 좌우 좌표간의 거리
h1 = abs(topRight[1] - bottomRight[1])     # 우측 상하 좌표간의 거리
h2 = abs(topLeft[1] - bottomLeft[1])       # 좌측 상하 좌표간의 거리
     width = max([w1, w2])
     height = max([h1, h2])
pts2 = np.float32([[0,0], [width-1,0], [width-1,height-1],
[0,height-1]])
mtrx = cv2.getPerspectiveTransform(pts1, pts2)
result = cv2.warpPerspective(img, mtrx, (width, height))
cv2.imshow('scanned', result)
cv2.imshow(win_name, img)
cv2.setMouseCallback(win_name, onMouse)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Giving Scanning Effect < Result >



❖ Triangluar Affine Transform

- Image is in rectangle
- Need triangular affine transformation
  - Distort image
  - Image Morphing
  - Liquefy



❖ Triangluar Affine Transform

- No function offered by OpenCV
- Procedure for triangular affine transform
  1. Select 3 matching coordination of triangle before transformation
  2. Select 3 matching coordination of triangle after transformation
  3. Calculate border quadrangle coordination surrounding step 1 triangle coordination.
  4. Select quadrangle area of step 3 as interested area.
  5. Interested area as in step 4, calculate change matrix with step 1 and step 2 coordination and transform affine.

6. Mask only triangle coordination of step 2 from the transformed interested area in step 5.
7. Using mask in step 6, compose it with original or other images.

- x,y,w,h = cv2.boundingRect(pts)pts : polygon coordination, need step 3
  - x, y, w, h : Width and height of border quadrangle
- cv2.fillConvexPoly(img, points, color [, lineType])img : Input image, need step 6
  - points : Vertex of polygon
  - color : Color to use for fill
  - lineType : Selection flag for line drawing algorithm
- Triangluar Affine Transform < 1/2>

```
import cv2
import numpy as np

img = cv2.imread("../img/taekwonv1.jpg")
img2 = img.copy()
draw = img.copy()

pts1 = np.float32([[188,14], [85,202], [294,216]])
pts2 = np.float32([[128,40], [85,307], [306,167]])

x1,y1,w1,h1 = cv2.boundingRect(pts1)
x2,y2,w2,h2 = cv2.boundingRect(pts2)

roi1 = img[y1:y1+h1, x1:x1+w1]
roi2 = img2[y2:y2+h2, x2:x2+w2]

offset1 = np.zeros((3,2), dtype=np.float32)
offset2 = np.zeros((3,2), dtype=np.float32)
for i in range(3):
    offset1[i][0], offset1[i][1] = pts1[i][0]-x1, pts1[i][1]-y1
    offset2[i][0], offset2[i][1] = pts2[i][0]-x2, pts2[i][1]-y2
```

- Triangluar Affine Transform <2/2>

```
mtrx = cv2.getAffineTransform(offset1, offset2)
warped = cv2.warpAffine( roi1, mtrx, (w2, h2), None,
cv2.INTER_LINEAR, cv2.BORDER_REFLECT_101 )

mask = np.zeros((h2, w2), dtype = np.uint8)
cv2.fillConvexPoly(mask, np.int32(offset2), (255))

warped_masked = cv2.bitwise_and(warped, warped, mask=mask)
roi2_masked = cv2.bitwise_and(roi2, roi2,
mask=cv2.bitwise_not(mask))
roi2_masked = roi2_masked + warped_masked
img2[y2:y2+h2, x2:x2+w2] = roi2_masked

cv2.rectangle(draw, (x1, y1), (x1+w1, y1+h1), (0,255,0), 1)
cv2.polylines(draw, [pts1.astype(np.int32)], True, (255,0,0), 1)
cv2.rectangle(img2, (x2, y2), (x2+w2, y2+h2), (0,255,0), 1)

cv2.imshow('origin', draw)
cv2.imshow('warped triangle', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Geometric Transform

1. Translate, Scaling, Rotate
2. Warping
3. **Lens Distortion**
4. Workshop

❖ Remapping

- Non-linear turnaround that cannot be expressed in determinant
- dst = cv2.remap(src, mapx, mapy, interpolation [, dst, borderMode, borderValue])
  - src : input image
  - mapx, mapy : Coordination that will move to x and y axis (Index)
    - Same size as src, dtype=float32
  - Other factors are the same as cv2.warpAffine()
  - dst : Result image
- mapy, mapx = np.indices((rows, cols), dtype=np.float32)
  - Generate array with the index value
  - Use cv2.remap() function initial array

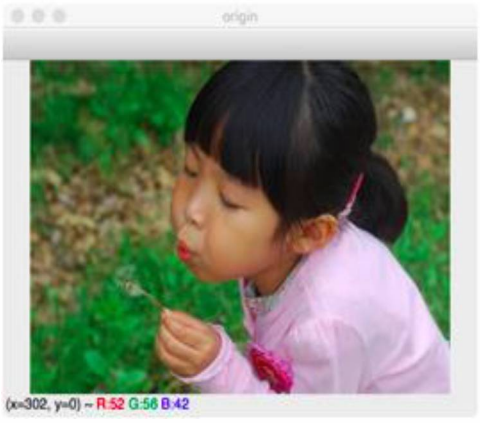
❖ Flip as Remapping

- Equation
  - $x' = \text{cols} - x - 1$
  - $y' = \text{rows} - y - 1$
- Determinant
  - $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 & \text{cols} - 1 \\ 0 & -1 & \text{rows} - 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- Linear change that can be shown in determinant is inefficient

- Flip as Remapping

```
import cv2
import numpy as np
import time

img = cv2.imread('../img/girl.jpg')
rows, cols = img.shape[:2]
st = time.time()
mflip = np.float32([ [-1, 0, cols-1],[0, -1, rows-1]])
fliped1 = cv2.warpAffine(img, mflip, (cols, rows))
print('matrix:', time.time()-st)

st2 = time.time()
mapy, mapx = np.indices((rows, cols),dtype=np.float32)
mapx = cols - mapx -1
mapy = rows - mapy -1
fliped2 = cv2.remap(img,mapx,mapy,cv2.INTER_LINEAR)
print('remap:', time.time()-st2)

cv2.imshow('origin', img)
cv2.imshow('fliped1',fliped1)
cv2.imshow('fliped2',fliped2)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
matrix: 0.0009329319000244141
remap: 0.00399327278137207
```



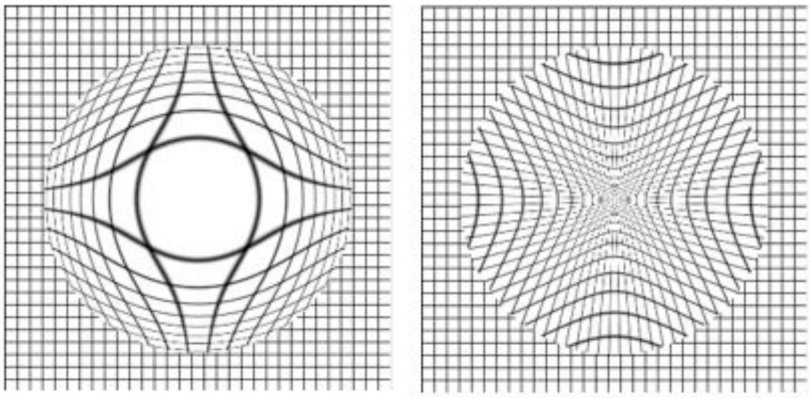
❖ Trigonometric function non-linear Remap

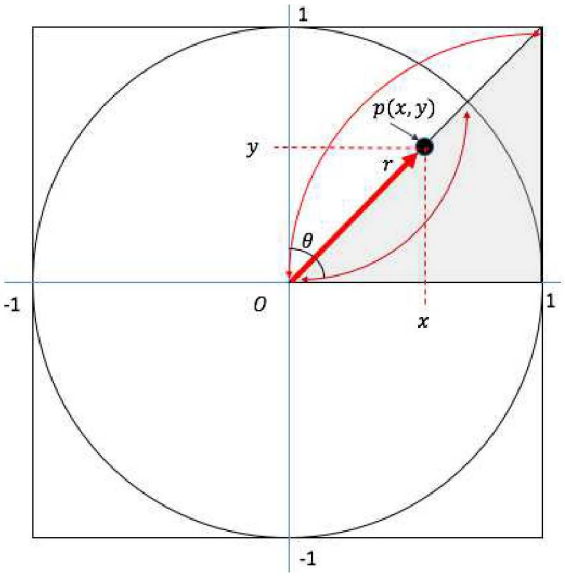
```
import cv2
import numpy as np

l = 20      # wave length
amp = 15    # amplitude
img = cv2.imread('../img/taekwonv1.jpg')
rows, cols = img.shape[:2]

mapy, mapx = np.indices((rows, cols),dtype=np.float32)
sinx = mapx + amp * np.sin(mapy/l)
cosy = mapy + amp * np.cos(mapx/l)

img_sinx=cv2.remap(img, sinx, mapy, cv2.INTER_LINEAR) # Apply sin
curve for only x
img_cosy=cv2.remap(img, mapx, cosy, cv2.INTER_LINEAR) # Apply cos
curve for only y
img_both=cv2.remap(img, sinx, cosy, cv2.INTER_LINEAR, None,
cv2.BORDER_REPLICATE)

cv2.imshow('origin', img)
cv2.imshow('sin x', img_sinx)
cv2.imshow('cos y', img_cosy)
cv2.imshow('sin cos', img_both)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Convex Concave Lens Distortion

- Coordsnate System
  - 직교좌표 → 극좌표 : $\theta = \arctan(x, y), \quad r = \sqrt{x^2 + y^2}$
  - 극좌표 → 직교좌표 : $x = r\cos(\theta), \quad y = r\sin(\theta)$

- r, theta = cv.cartToPolar(x, y) : Rectangular Coordination → Polar Coordination
- x, y = cv.polarToCart(r, theta): Polar Coordination → Rectangular Coordination
  - x, y : x, y array
  - r : Distance from zeron
  - theta : Angle value





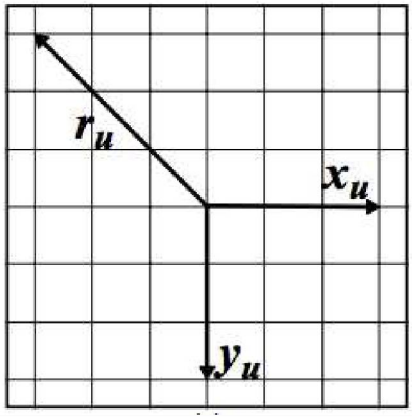
❖ Convex Concave Lens Distortion

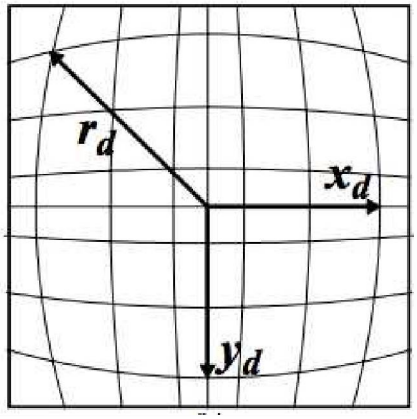
```
import cv2, numpy as np

img = cv2.imread('../img/taekwonv1.jpg')
rows, cols = img.shape[:2]
exp = 0.5       # Convex, Concave (concave : 0.1 ~ 1, Convex: 1.1~)
scale = 1       # Size of turnaround (0 ~ 1)

mapy, mapx = np.indices((rows, cols),dtype=np.float32)
mapx = (2*mapx - cols)/cols
mapy = (2*mapy - rows)/rows

r, theta = cv2.cartToPolar(mapx, mapy)
r[r< scale] = r[r<scale] **exp
mapx, mapy = cv2.polarToCart(r, theta)

mapx = ((mapx + 1)*cols)/2
mapy = ((mapy + 1)*rows)/2
distorted = cv2.remap(img,mapx,mapy,cv2.INTER_LINEAR)

cv2.imshow('origin', img)
cv2.imshow('distorted', distorted)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Radial Distortion

- Quadrangle image from circular camera lenses
- Barrel Distortion: Image distortion with convex edge
- Barrel distortion equation :
  - Depends on coefficient, pin cushion distortion, barrel distortion

- $r_d = r_u(1 + k_1 r_u^2 + k_2 r_u^4 + k_3 r_u^6)$
  - r_d : 왜곡 변형 후
  - r_u : 왜곡 변형 전
  - k_1, k_2, k_3 : 왜곡 계수






```
import cv2

import numpy as np
k1, k2, k3 = 0.5, 0.2, 0.0 # Barrel distortion
#k1, k2, k3 = -0.3, 0, 0    # Pin cushion distortion
img = cv2.imread('../img/girl.jpg')
rows, cols = img.shape[:2]
mapy, mapx = np.indices((rows, cols),dtype=np.float32)
mapx = (2*mapx - cols)/cols
mapy = (2*mapy - rows)/rows
r, theta = cv2.cartToPolar(mapx, mapy)
ru = r*(1+k1*(r**2) + k2*(r**4) + k3*(r**6))
mapx, mapy = cv2.polarToCart(ru, theta)
mapx = ((mapx + 1)*cols)/2
mapy = ((mapy + 1)*rows)/2
distored = cv2.remap(img,mapx,mapy,cv2.INTER_LINEAR)

cv2.imshow('original', img)
cv2.imshow('distored', distored)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Radial Distortion

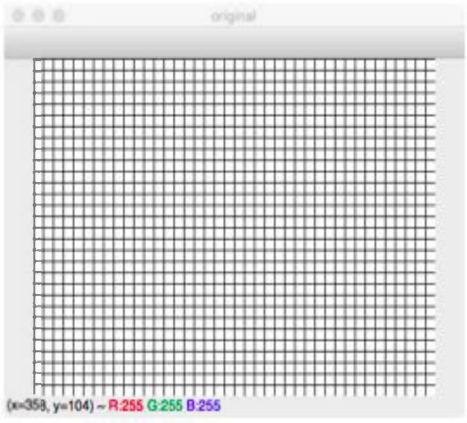
- Function eliminating barrel distortion OpenCV
- dst = cv2.undistort(src, cameraMtrix, distCoeffs)
  - src : Original input image
  - cameraMatrix : Camera matrix

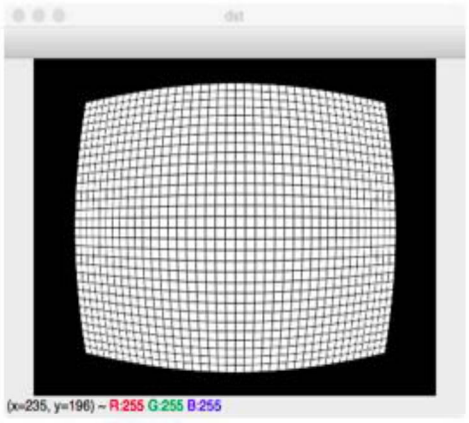
$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

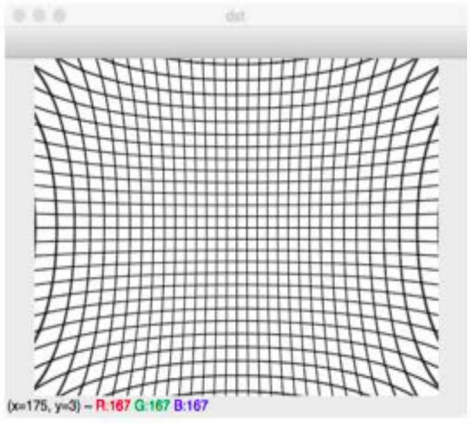
  - distCoeffs : Distortion coefficient. Min 4, or 5, 8, 12, 14
    - (k1, k2, p1, p2[,k3])

- cv2.undistort()

```
import numpy as np
import cv2

img = np.full((300,400,3), 255, np.uint8)
img[::10, :, :] = 0
img[:, ::10, :] = 0
width  = img.shape[1]
height = img.shape[0]

k1, k2, p1, p2 = 0.001, 0, 0, 0     # Barrel distortion
#k1, k2, p1, p2 = -0.0005, 0, 0, 0  # Pin cushion distortion
distCoeff = np.float64([k1, k2, p1, p2])

fx, fy = 10, 10
cx, cy = width/2, height/2
camMtx = np.float32([[fx,0, cx],
                     [0, fy, cy],
                     [0 ,0 ,1]])
dst = cv2.undistort(img,camMtx,distCoeff)
cv2.imshow('original', img)
cv2.imshow('dst',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```







Geometric Transform

1. Translate, Scaling, Rotate
2. Warping
3. Lens Distortion
4. **Workshop**

❖ Mosaic

- Blur selected area.
- Result example

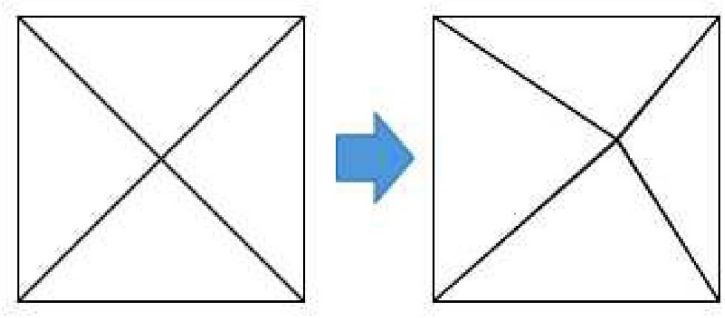
- Hint
  - If selected area is minimized and enlarged, clarity will get worse.
  - If cv2.INTER_AREA is selected during interpolation, check board time blur is generated.

❖ Liquefiy Tool

- Draft a program similar to Liquefiy tool in Photoshop.
- Edit only selected area.
- Result Example

❖ Liquefiy Tool

- Hint
  - Make same size quadrangle based on mouse coordination.
  - Make 4 triangles using the center of quadrangle.
  - Calculate center point moved from the original center point by mouse dragging.
  - Using the size of new triangles, affine transform the triangles.



❖ Distortion Camera

- Make distortion camera with magic mirror effects.
- 6 effects
  - Original, Bilateral symmetry, up an down symmetry
  - Wave, Convex, Concave
- Result Example

❖ Distortion Camera

- Hint
  - Refer to lens distortion example, combine 6 mapping result into one images
  - Make only one remapping coordination and apply the coordination for each frame.

Filter

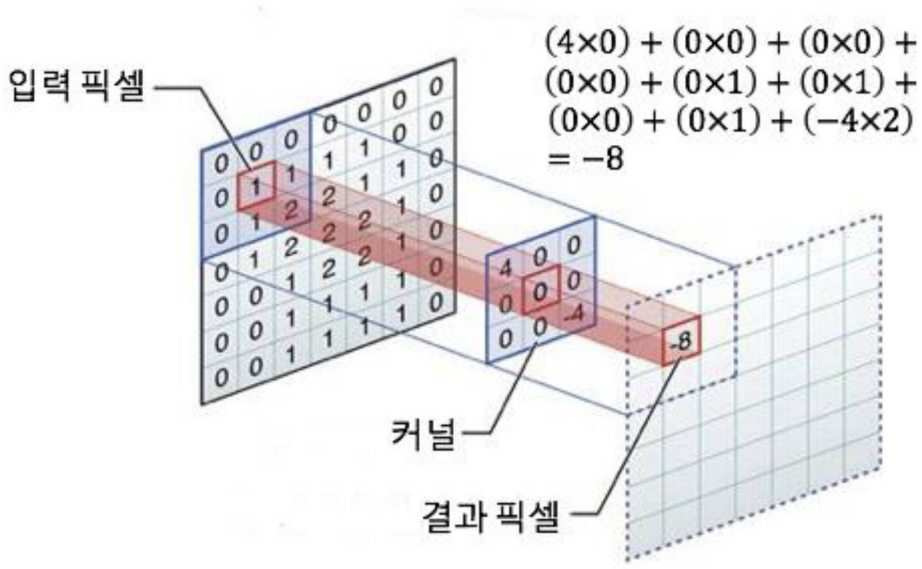
1. **Convolution Filter**
2. Blurring
3. Edge Detection
4. Morphology
5. Image Pyramids
6. Workshop

❖ Filter

- Filter out unwanted value and get only wanted value.
  - LPF(Low Pass Filter) : Blurring, Eliminate Noise
  - HPF(High Pass Filter) : Sharpening, Edge Detection
- Domain
  - Spatial Domain Filter
    - Calculate by using surrounding pixel value not just one pixel
    - Convolution Calculation
  - Frequency Domain Filter
    - Calculate by transferring pixel value differences to frequency
    - Fourier Transform

❖ Convolution

- Core calculation for spatial domain filter
- Multiply input pixel value matching to each factors of Kernel and add them all
- Kernel: Select surrounding pixels to be used in calculation
  - Name may vary like window, mask, filter etc.
  - Kernel Size: n x n
  - n : Generally odd number



❖ Convolution Function for Calculation

- dst = cv2.filter2D(src, ddepth, kernel[, dst, anchor, delta, borderType])
  - src : Input image, NumPy arrau
  - ddepth : dtype of output image,
    - -1 : Same is input image
    - CV_8U, CV16U/CV16S, CV_32F, CV_64F
  - kernel : Convolution kernel, float32
  - dst : Result image, NumPy array
  - anchor : Standard point of kernel, default: Center point(-1,-1)
  - delta : Value to be added in result value with filter
  - borderType : Designate border pixel correction method

Filter

1. Convolution Filter
2. **Blurring**
3. Edge Detection
4. Morphology
5. Image Pyramids
6. Workshop

❖ Blurring

- Blur image using Filter
- Superior for Noise removal
- Blur Types
  - Averaging Blur
    - cv2.blur, cv2.boxFilter
  - Gassian Blur
    - cv2.GaussianBlur
  - Median Blur
    - cv2.medianBlur
  - Bilateral Filter
    - cv2.bilateralFilter

❖ Averaging Blurring

- Change pixel value to average value of surrounding factors
- Decide domain of surrounding factors
  - Generate Kernel matrix
  - More unclear as Kernel gets large
- Method
  - cv2.filter2D()
  - cv2.boxFilter()
  - cv2.blur()
- Apply surrounding pixel to average pixels
- Average eigen value to give blur effects
- 5 X 5 average filter

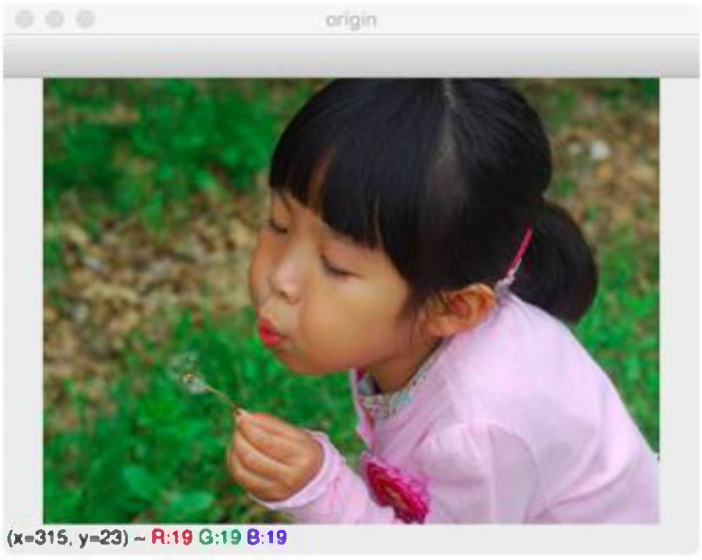
$$k = \begin{bmatrix} 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \end{bmatrix} \qquad k = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

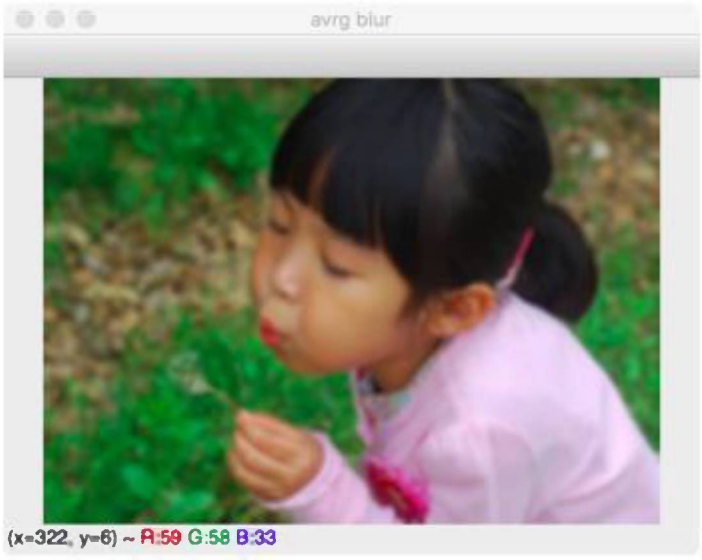
❖ Averaging Blurring

- cv2.filter2D Example

```
import cv2, numpy as np

img = cv2.imread('../img/girl.jpg')
'''
#5x5 평균 필터 커널 생성
kernel = np.array([[0.04, 0.04, 0.04, 0.04, 0.04],
                   [0.04, 0.04, 0.04, 0.04, 0.04],
                   [0.04, 0.04, 0.04, 0.04, 0.04],
                   [0.04, 0.04, 0.04, 0.04, 0.04],
                   [0.04, 0.04, 0.04, 0.04, 0.04]])
'''
kernel = np.ones((5,5))/5*5
blured = cv2.filter2D(img, -1, kernel)

cv2.imshow('origin', img)
cv2.imshow('avrg blur', blured)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Example <Result>





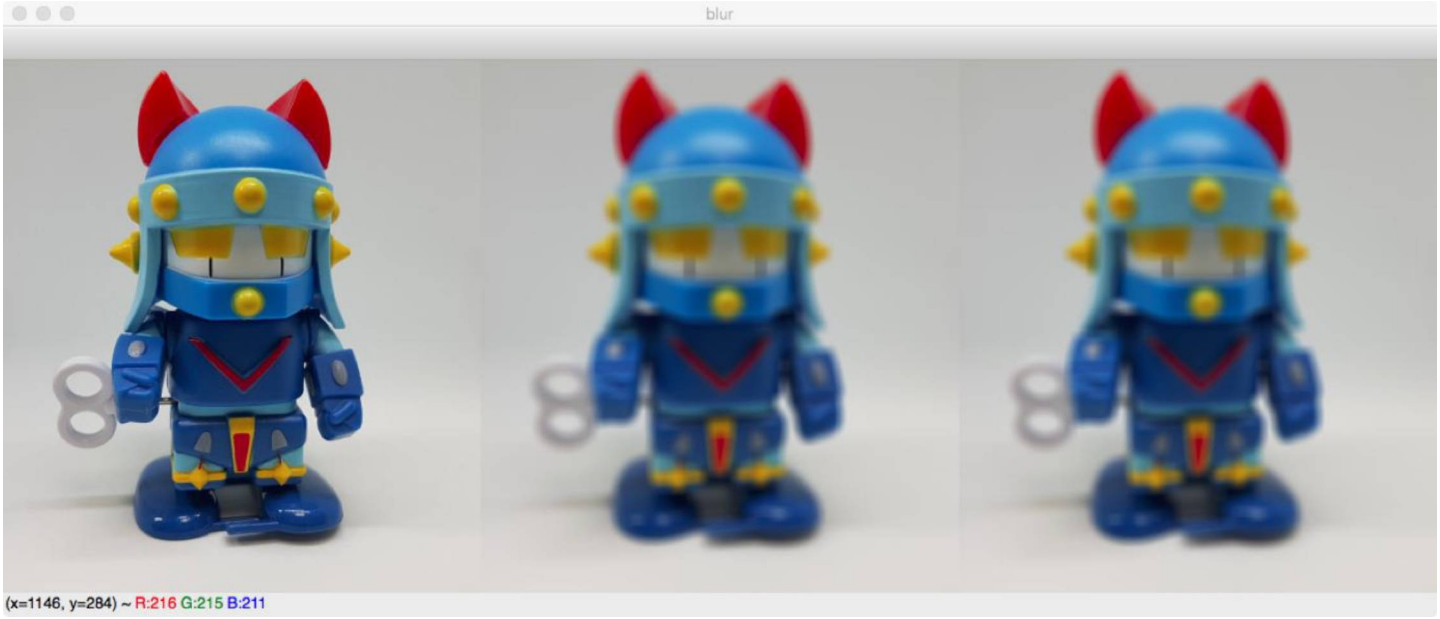
❖ Averaging Blurring

- Applying blur without defining kernel
- dst = cv2.blur(src, ksize[, dst, anchor, borderType])
  - src : Input image, NumPy array
  - ksize : Size of kernel
  - Other factors are same as cv2.filter2D()
- dst = cv2.boxFilter(src, ddepth, ksize[, dst, anchor, normalize, borderType])
  - src : Input image, NumPy array
  - ddepth : dtype of output image, -1 : Same as input image
  - normalize : rather to normalize by kernel size, boolean
  - Other factors are same as cv2.filter2D()
- Blur specific function Example

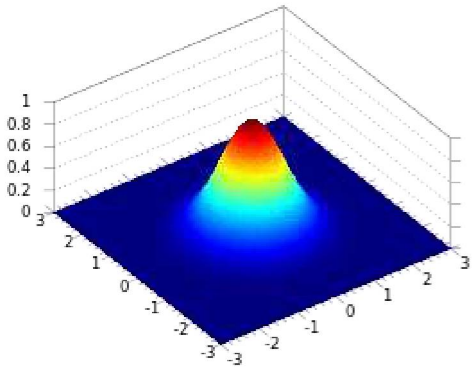
```
import cv2
import numpy as np

file_name = '../img/taekwonv1.jpg'
img = cv2.imread(file_name)

# blur()
blur1 = cv2.blur(img, (10,10))

# boxFilter()
blur2 = cv2.boxFilter(img, -1, (10,10))

merged = np.hstack( (img, blur1, blur2))
cv2.imshow('blur', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Averaging Blurring

- Blur specific function Result



❖ Gaussian Blurring



- Using Kernel applying Gaussian function
- Effective for white noise
- Cv2.Gaussian(src, ksize, sigmaX [, sigmaY, border])
  - Src : Input size
  - Ksize: kernel size
  - sigmaX: X direction standard deviation, 0 = auto
    - σ =0.3(ksize-1)0.5-1)+0.8
  - sigmaY: Y direction standard deviation, default = sigmaX
  - boder : Revising boder
- kernel = cv2.getGaussianKernel(ksize, sigma)
  - ksize : Kernel size
  - sigma : Standar Deviation
  - kernel*kernel.T

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

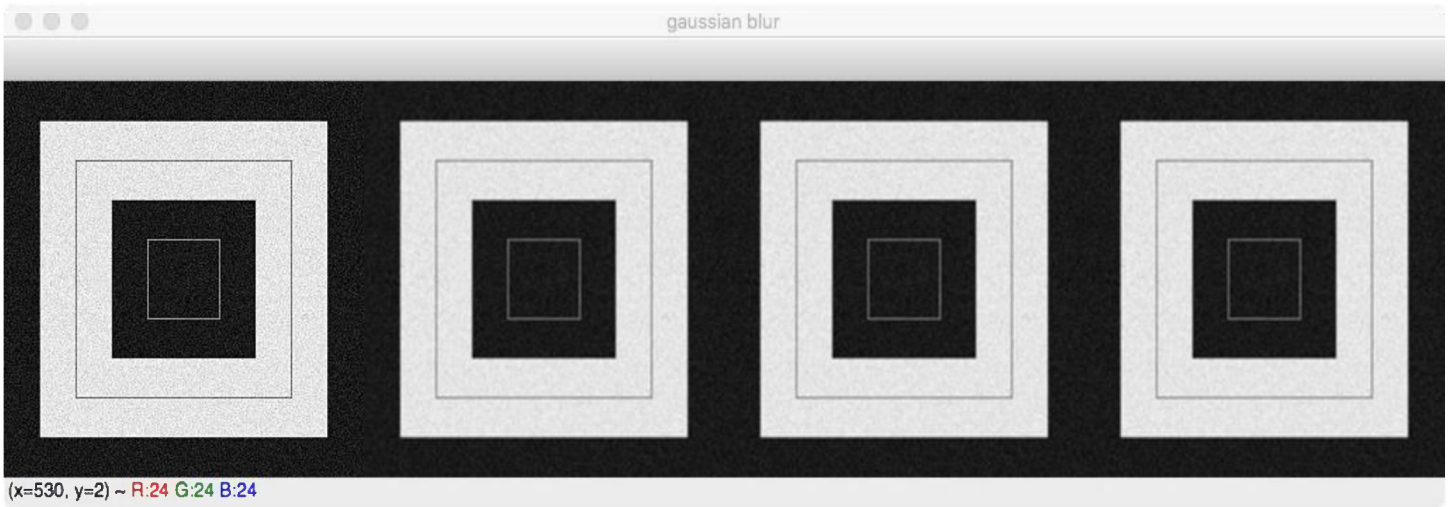
❖ Gaussian Blurring

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/gaussian_noise.jpg')

k1 = np.array([[1, 2, 1],
               [2, 4, 2],
               [1, 2, 1]]) *(1/16)
blur1 = cv2.filter2D(img, -1, k1)

k2 = cv2.getGaussianKernel(3, 0)
blur2 = cv2.filter2D(img, -1, k2*k2.T)

blur3 = cv2.GaussianBlur(img, (3, 3), 0)

print('k1:', k1)
print('k2:', k2*k2.T)
merged = np.hstack((img,blur1,blur2,blur3))
cv2.imshow('gaussian blur', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
k1: [[0.06250.125  0.0625]
 [0.125  0.25   0.125 ]
 [0.0625 0.125  0.0625]]
k2: [[0.06250.125  0.0625]
 [0.125  0.25   0.125 ]
 [0.0625 0.125  0.0625]]
```

- Example <Execution Result>



❖ Median Blurring

- Of the kernel domain pixel value, apply middle value
- Effective for salt-and-pepper noise
- Reuse existing pixel value not new value
- dst = cv2.medianBlur(src, ksize)
  - src : Input Image, NumPy array
  - ksize : Kernel Size

```
import cv2
import numpy as np

img = cv2.imread("../img/salt_pepper_noise.jpg")

blur = cv2.medianBlur(img, 5)

merged = np.hstack((img,blur))
cv2.imshow('media', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

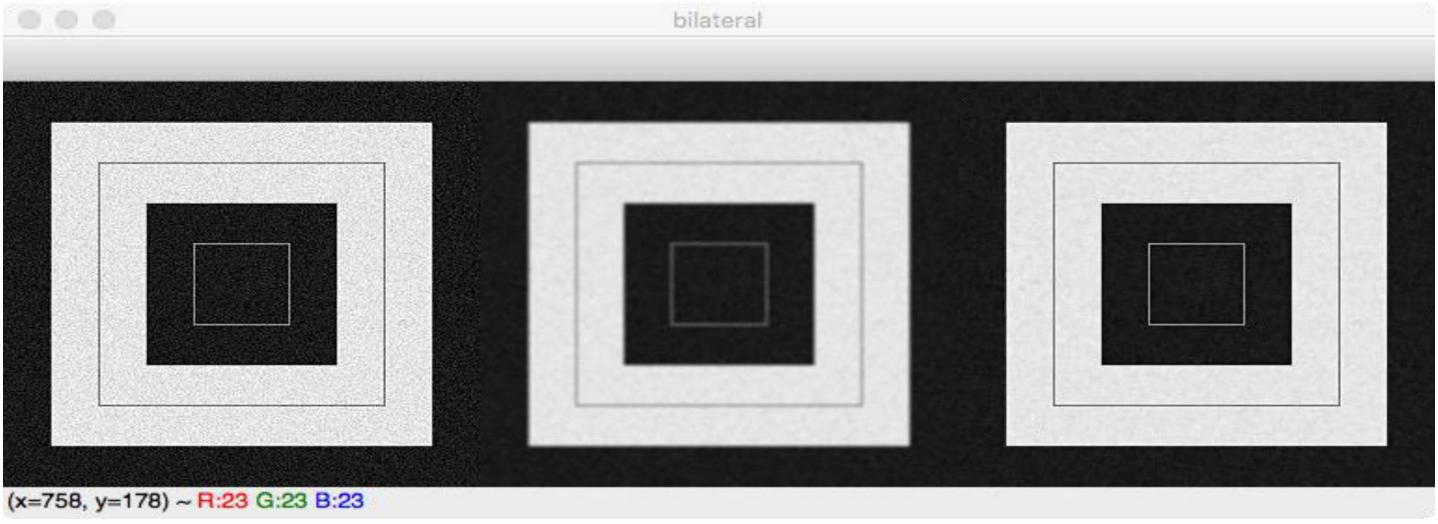
❖ Bilateral Filter Blurring

- Both way filter, use 2 filters
  - Gaussian filter + boarder filter
- Blur result blurs edge as well, it improves this issue
- Slow, due to long calculation time
- dst = cv2.bilateralFilter(src, d, sigmaColor, sigmaSpace[,dst, borderType])
  - src : Input image, NumPy array

- d : diameter of filter, If it is bigger than 5, it is very slow
- sigmaColor : Sigma value of color domain filter
- sigmaSpace : Sigma value of coordination domain
  - Suggest to use same value for Sigma Color and Sigma Space for simpler use
  - Suggest value in between 10 ~ 150

- Example

```
import cv2
import numpy as np

img = cv2.imread("../img/gaussian_noise.jpg")

blur1 = cv2.GaussianBlur(img, (5,5), 0)

blur2 = cv2.bilateralFilter(img, 5, 75, 75)

merged = np.hstack((img, blur1, blur2))
cv2.imshow('bilateral', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Filter

1. Convolution Filter
2. Blurring
3. **Edge Detection**
4. Morphology
5. Image Pyramids
6. Workshop

❖ Detecting Boarder

- Dividing front image and background image
- Sharping: Emphasizing boarder by detecting
- Boarder: Point with large pixel value
- Differentiation
  - Ration of resolution change
  - Slope of image

Continuous function:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{h \to 0} \frac{f(x+h,y) - f(x,y)}{h}$$

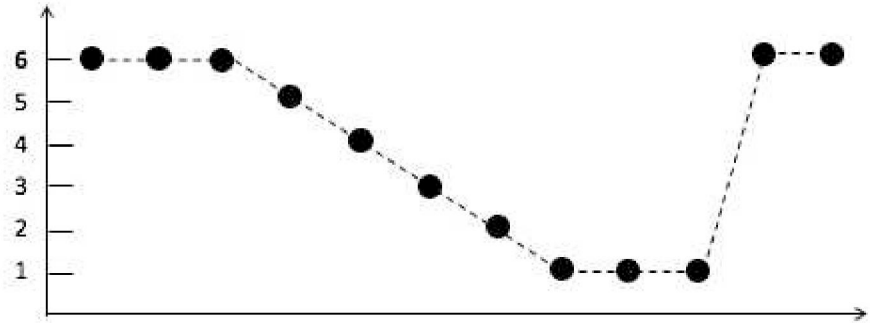
$$\frac{\partial f(x,y)}{\partial y} = \lim_{h \to 0} \frac{f(x,y+h) - f(x,y)}{h}$$

❖ Basic Differentiation Filter

- Simplify differentiation equation
  - Not sequential space
  - Differentiate to calculate approximate
  - Next pixel – current pixel
  - 2-way convolution kernel

- $Gx = \frac{\partial f(x,y)}{\partial x} \approx f_{x+1,y} - f_{x,y}$
- $Gy = \frac{\partial f(x,y)}{\partial y} \approx f_{x,y+1} - f_{x,y}$

- $Gx = \begin{bmatrix} -1 & 1 \end{bmatrix}$
- $Gy = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$



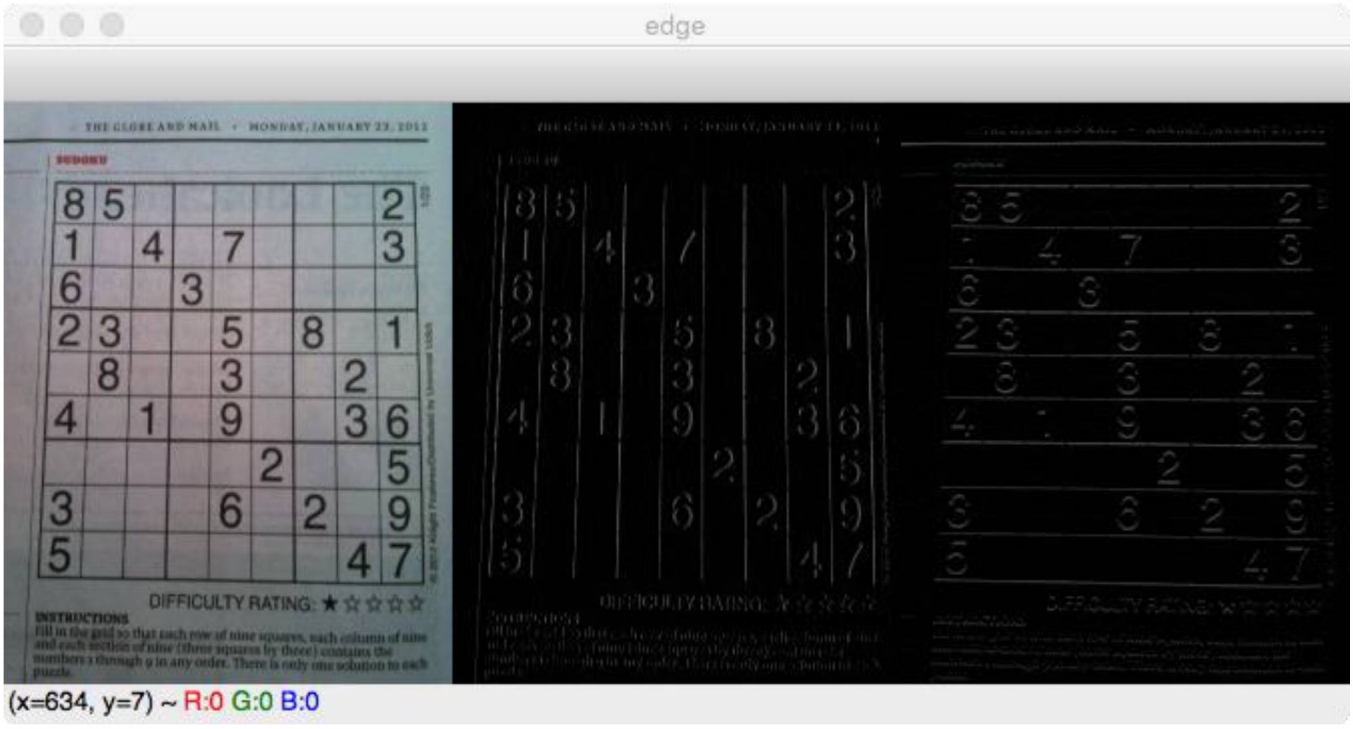
픽셀 값	6	6	6	5	4	3	2	1	1	1	6	6
1차 미분	0	0	0	-1	-1	-1	-1	0	0	5	0	
2차 미분	0	0	0	-1	0	0	1	0	5	-5	0	

❖ Basic Differentiation Filter

- kernel [-1, 1] Example

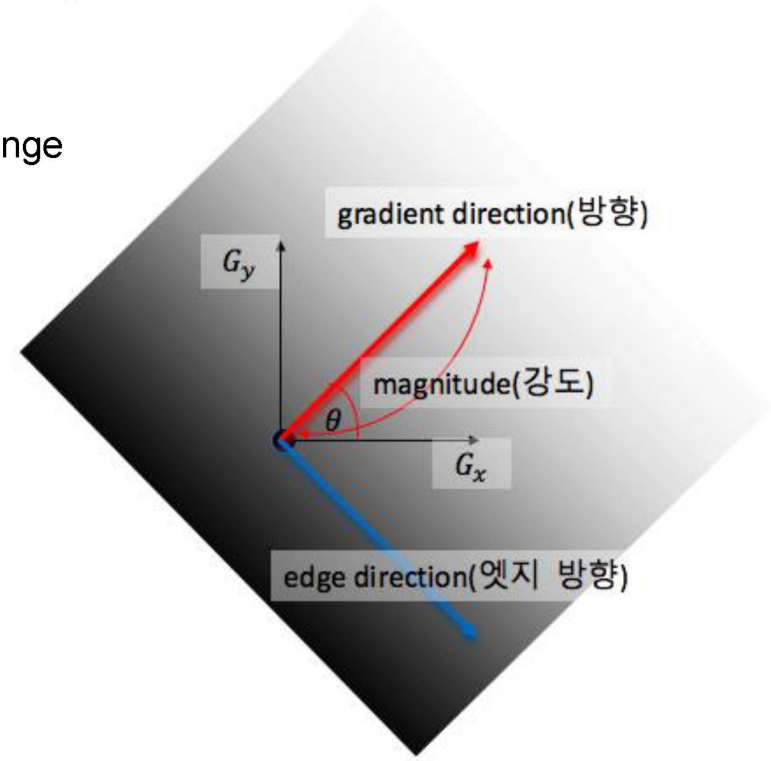
```
import cv2
import numpy as np

img = cv2.imread("../img/sudoku.jpg")

gx_kernel = np.array([[ -1, 1]])
gy_kernel = np.array([[ -1],[ 1]])

edge_gx = cv2.filter2D(img, -1, gx_kernel)
edge_gy = cv2.filter2D(img, -1, gy_kernel)

merged = np.hstack((img, edge_gx, edge_gy))
cv2.imshow('edge', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- kernel [-1, 1] Example < Result >



❖ Direction and magnitude of edge

- Gradient
  - Magnitude : Strength of change
  - Direction : Direction of value change
    - Perpendicular to edge
- Use for vectorize of image
  - $\text{magnitude} = \sqrt{G_x^2 + G_y^2}$
  - $\text{direction}(\theta) = \arctan(\frac{G_y}{G_x})$



❖ Roberts cross operator

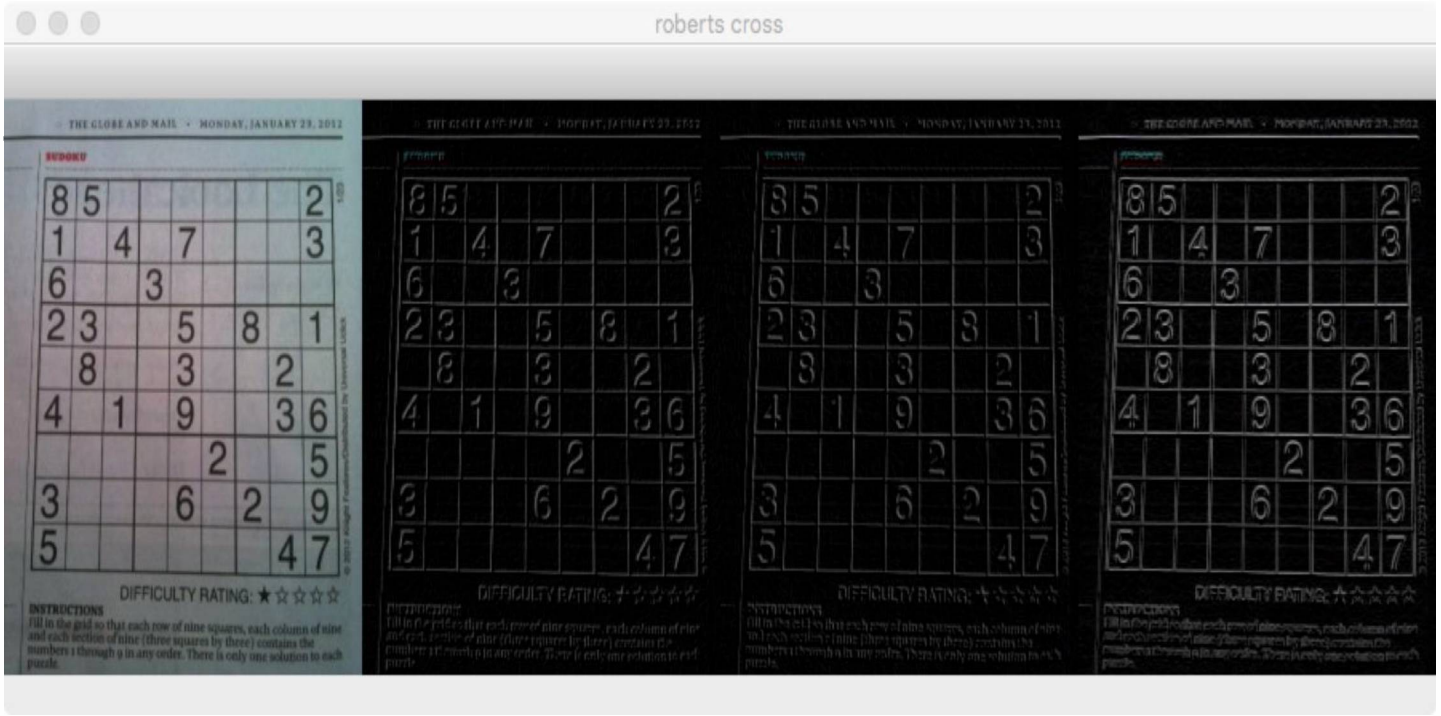
- Suggested by Lawrence Roberts (1963)
- Oldest one of all improved kernel differentiation
- Place 1 and -1 in diagonal direction
- Better diagonal line detecting effect compare to basic differentiation mask
- Sensitive to noise
- Weak edge magnitude
  - $Gx = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}$
  - $Gx = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}$

❖ Roberts cross operator

- Example

```
import cv2
import numpy as np

img = cv2.imread("../img/sudoku.jpg")

gx_kernel = np.array([[1,0], [0,-1]])
gy_kernel = np.array([[0, 1],[-1,0]])

edge_gx = cv2.filter2D(img, -1, gx_kernel)
edge_gy = cv2.filter2D(img, -1, gy_kernel)

merged = np.hstack((img, edge_gx, edge_gy, edge_gx+edge_gy))
cv2.imshow('roberts cross', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example < Result >



❖ Prewitt Operator

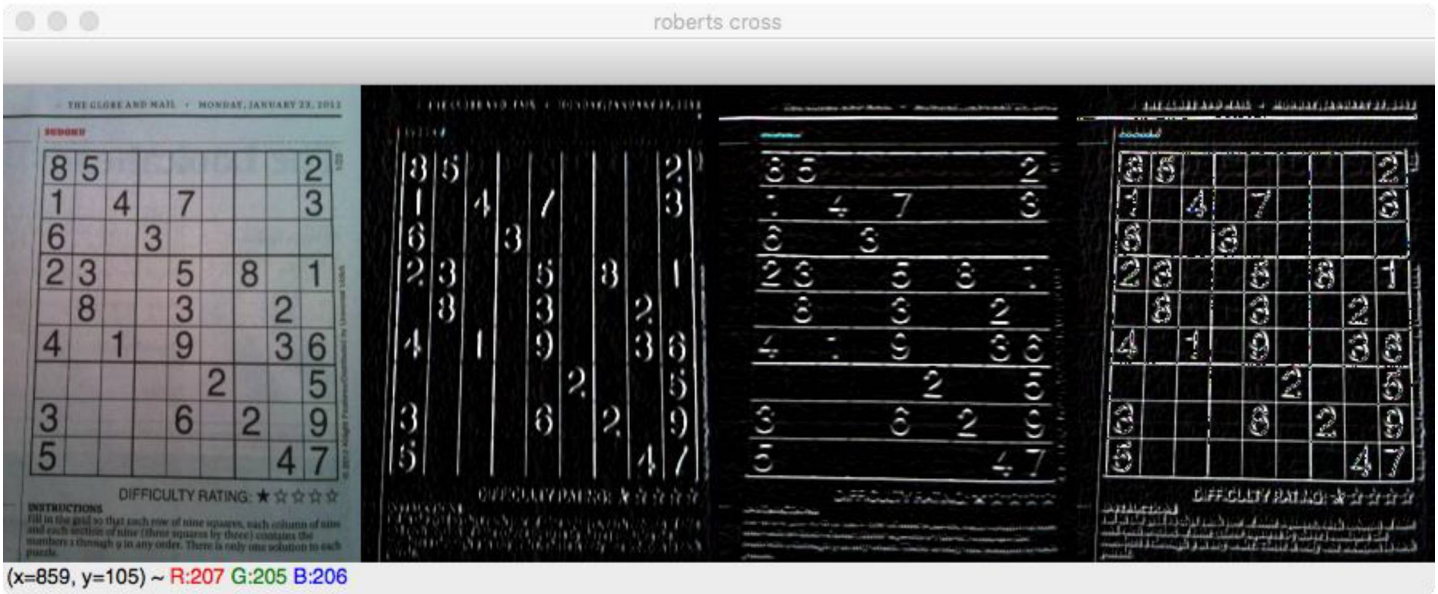
- Suggested by Judith M. S. Prewitt
- 3- difference of each direction
- Strong edge magnitude
- Same for Perpendicular and parallel edge
- Week diagonal detection
- Example

$$\bullet\ Gx = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$$

$$\bullet\ Gy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

```
import cv2
import numpy as np
file_name = "../img/sudoku.jpg"
img = cv2.imread(file_name)
gx_k = np.array([[-1,0,1], [-1,0,1],[-1,0,1]])
gy_k = np.array([[-1,-1,-1],[0,0,0], [1,1,1]])
edge_gx = cv2.filter2D(img, -1, gx_k)
edge_gy = cv2.filter2D(img, -1, gy_k)
merged = np.hstack((img, edge_gx, edge_gy, edge_gx+edge_gy))
cv2.imshow('roberts cross', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>



❖ Sobel Operator

- Suggested by Irwin Sobel (1968)
- Mot representative of primary differentiation operator
- Double the weight of center resolution different
- Strong for all parallel, perpendicular and diagonal edge

- $Gx = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$
- $Gx = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$

- dst = cv2.Sobel(src, ddepth, dx, dy[, dst, ksize, scale, delta, borderType])
  - src : Input image, NumPy array
  - ddepth : dtype of output image, -1 : Same as input image
  - dx, dy : # of differentiation, Select one of 0,1,2, cannot both be 0
  - ksize : Size of kernel, Select 1,3,5,7
  - scale : Coefficient to use in differentiation
  - delta : Value that will be added to calculation result
- Example

```
import cv2
import numpy as np

img = cv2.imread("../img/sudoku.jpg")

gx_k = np.array([[-1,0,1], [-2,0,2],[-1,0,1]])
gy_k = np.array([[-1,-2,-1],[0,0,0], [1,2,1]])

edge_gx = cv2.filter2D(img, -1, gx_k)
edge_gy = cv2.filter2D(img, -1, gy_k)

sobelx = cv2.Sobel(img, -1, 1, 0, ksize=3)
sobely = cv2.Sobel(img, -1, 0, 1, ksize=3)

merged1 = np.hstack((img, edge_gx, edge_gy, edge_gx+edge_gy))
merged2 = np.hstack((img, sobelx, sobely, sobelx+sobely))
merged = np.vstack((merged1, merged2))
cv2.imshow('sobel', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Sobel Operator

- Example <Result>

❖ Scharr Operator

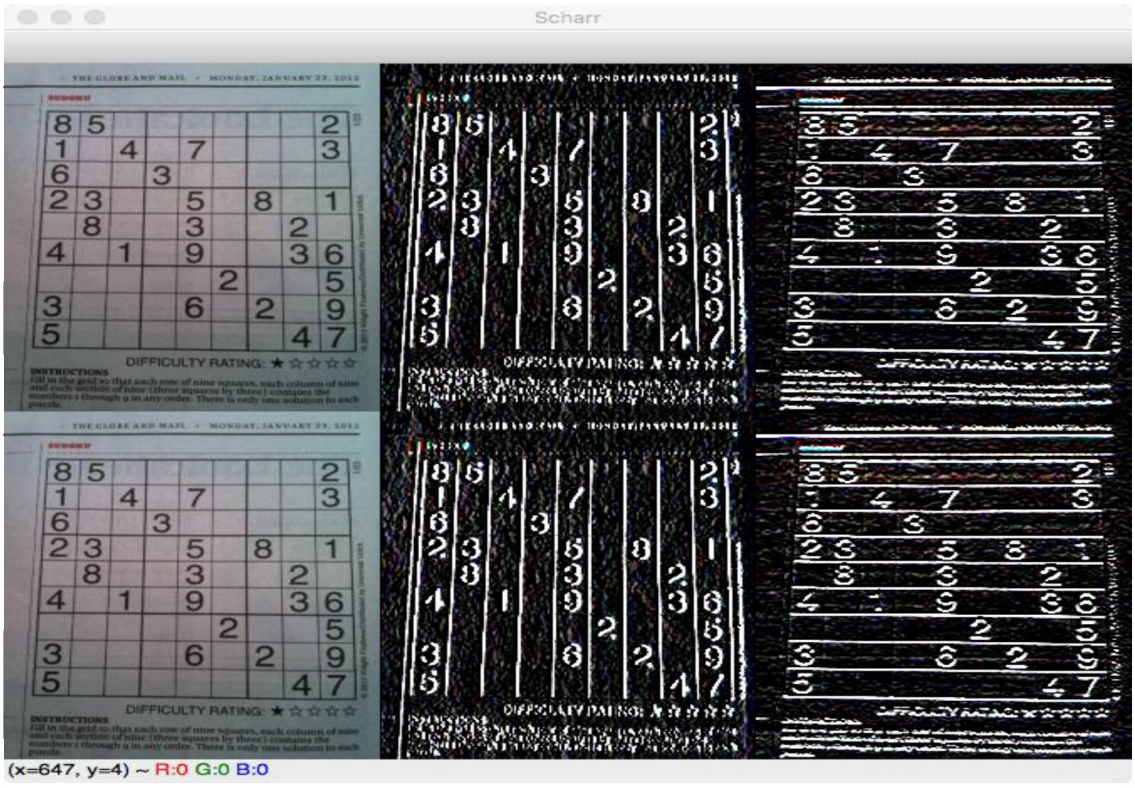
- Issues of Sobel
  - Reduces accuracy as it is distanced from center even though kernel size is large or small
- Improve Sobel
  - $Gx = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix}$
  - $Gx = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}$
- dst = cv2.Scharr(src, ddepth, dx, dy[, dst, scale, delta, borderType])
  - Function is same as cv2.Sobel() except it does not have ksize factor

❖ Scharr Operator

- Example

```
import cv2
import numpy as np

img = cv2.imread("../img/sudoku.jpg")
gx_k = np.array([[-3,0,3], [-10,0,10],[-3,0,3]])
gy_k = np.array([[-3,-10,-3],[0,0,0], [3,10,3]])
edge_gx = cv2.filter2D(img, -1, gx_k)
edge_gy = cv2.filter2D(img, -1, gy_k)
scharrx = cv2.Scharr(img, -1, 1, 0)
scharry = cv2.Scharr(img, -1, 0, 1)
merged1 = np.hstack((img, edge_gx, edge_gy))
merged2 = np.hstack((img, scharrx, scharry))
merged = np.vstack((merged1, merged2))
cv2.imshow('Scharr', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>

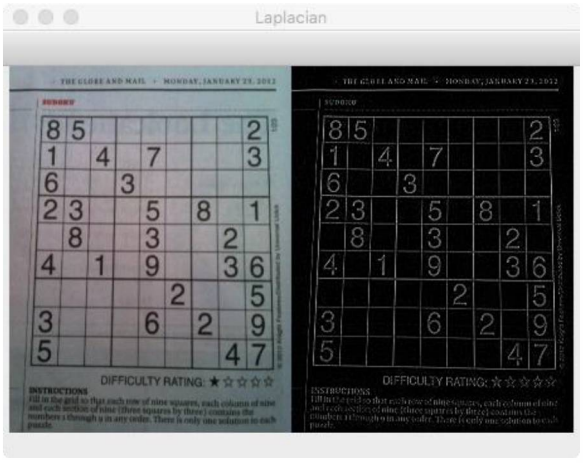


❖ Laplacian Operator

- Pierre-Simon de Laplace(1749-1827)
- Representative 2nd differentiation operator
- Solve gradual edge detecting issues in preliminary differentiation
  - $$\frac{d^2 f}{dy^2} = \frac{df(x,y+1)}{dy} - \frac{df(x,y+1)}{dy}$$
    $$= [f(x,y+1) - f(x,y)] - [f(x,y) - f(x,y-1)]$$
    $$= f(x,y+1) - 2f(x,y) + f(x,y-1)$$
  - $$kernel = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$
- cv2.Laplacian(src, ddepth)
  - Function factor is same as cv2.Sobel()
- Example

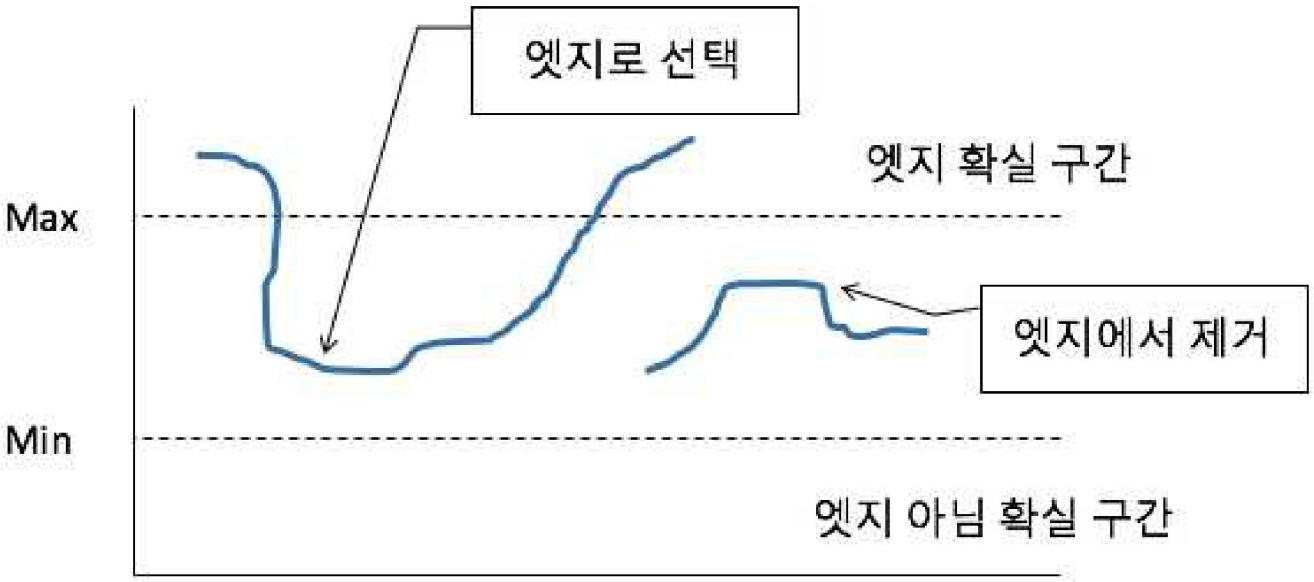
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("../img/sudoku.jpg")
edge = cv2.Laplacian(img, -1)
merged = np.hstack((img, edge))
cv2.imshow('Laplacian', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>



❖ Canny Edge Dectector

- John F Canny 제안(1986)
- 4 step algorithm, not one
  1. Noise Reduction
     - 5x5 Gaussian Filter
  2. Edge Gradient Detection
     - Calculate Gradient direction angle using Sobel
  3. Non-Maximum Suppression
     - Select biggest value from Gradient direction, drop the rest
  4. Hysteresis Thresholding
     - Set max and min threshold
     - Inspect threshold area edge
     - Connect to the value bigger than max, if there is no such value, drop it

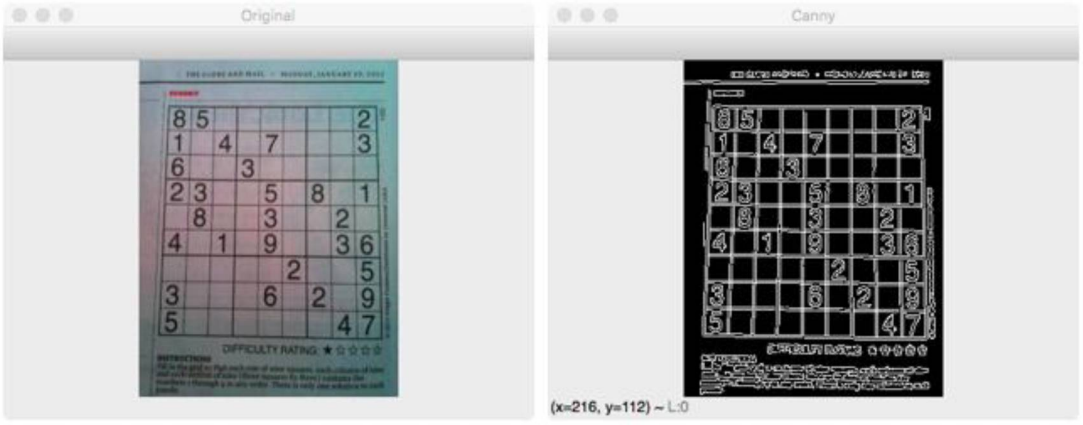


❖ Canny Edge Dectector

- edges = cv2.Canny(img, threshold1, threshold2 [,edges, apertureSize, L2gardient])
  - img : Input image, NumPy array
  - threshold1, threshold2 : Max and min value for thresholding
  - apertureSize : Kernel size to be used in Sobel mask
  - L2gradient : Designate flag to calculate gradient magnitude
    - True : $\sqrt{G_x^2 + G_y^2}$
    - False : $|G_x| + |G_y|$
  - egges : 2-dimensinal array with edge result value
- Example

```
import cv2
import numpy as np

img = cv2.imread("../img/sudoku.jpg")

edges = cv2.Canny(img,100,200)

cv2.imshow('Original', img)
cv2.imshow('Canny', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>



Filter

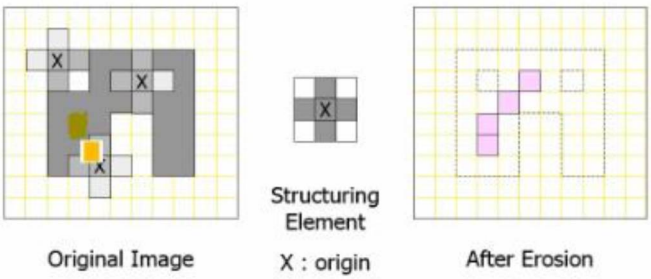
1. Convolution Filter
2. Blurring
3. Edge Detection
4. **Morphology**
5. Image Pyramids
6. Workshop

❖ Morphological Operation

- For Gray-Scaled image
- Eliminate noise, simplify, use for distanced part or filling holes
- Structuring Element
  - Kernel applied to original image
  - Quadrangle, Oval, Cross
  - np.ones( ) :Quadrangle
  - cv2.getStructuringElements(shape, ksize[, anchor]) : Other than Quadrangle
    - shape: cv2.MORPH_*
      - RET, ELLIPSE, CROSS
    - ksize : Kernel size
    - anchor : Standard point of structural factor, meaningless other than cross type
- Operation Type
  - Erosion
  - Dilation
  - Opening
  - Closing

❖ Erosion

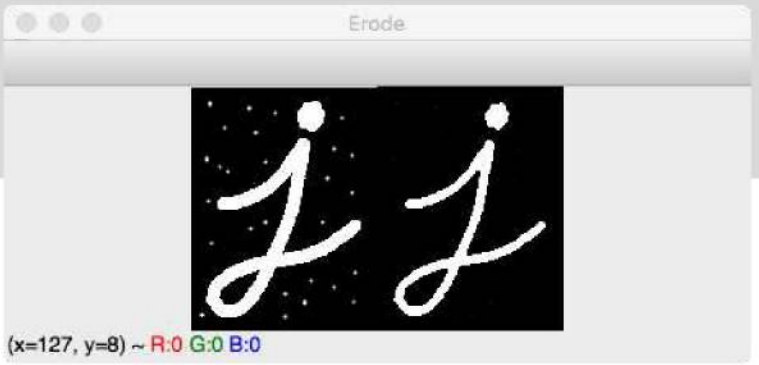
- Apply Structuring Element, if there is at least one pixel, delete
  - Delete if kernel cannot be placed
- Effective for eliminating small objects (mostly noise)
- cv2.erode(src, kernel [, anchor, iteration])
  - src : Input image
  - kernel : Structuring Element
  - anchor : Center point of Kernel, default(-1, -1)
  - iteration : # of repetition, default(1)



❖ Erosion

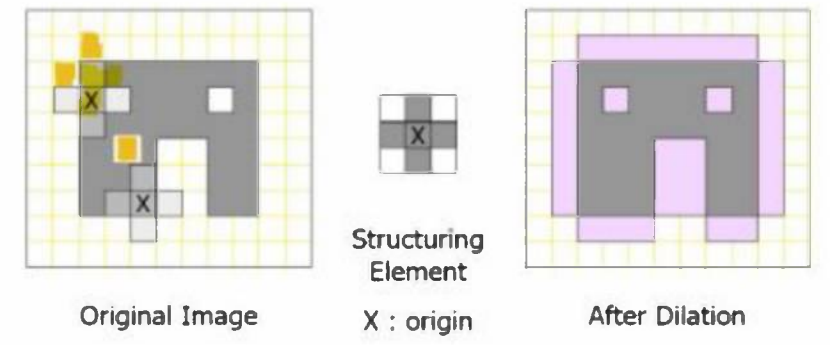
- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/morph_dot.png')

k = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))
erosion = cv2.erode(img, k)

merged = np.hstack((img, erosion))
cv2.imshow('Erode', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



❖ Dilation

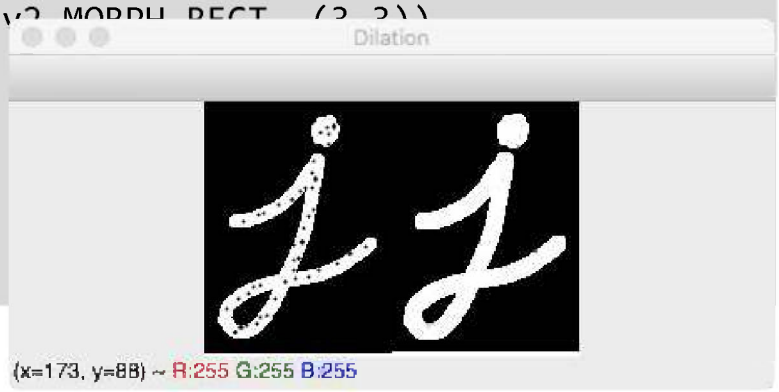
- Fill 0 pixel using Structuring Element.
  - Fill uncovered area after placing kernel
- Link disconnection, effective for hole filling
- cv2.dilation(src, kenel [, anchor, iteration)
  - src : Input image
  - kernel : Structuring Element
  - anchor : Center point of kernel, default(-1, -1)
  - iteration : # of repetition, default(1)



❖ Dilation

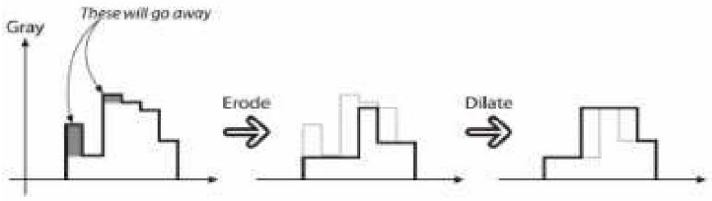
- Example

```
import cv2

import numpy as np
img = cv2.imread('../img/morph_hole.png')
k = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))
dst = cv2.dilate(img, k)
merged = np.hstack((img, dst))
cv2.imshow('Dilation', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

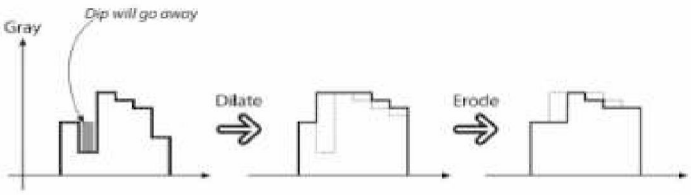


❖ Opening & Closing

- Combination of Erosion & Dilation
- Opening : Erosion + Dilation
  - Effective for eliminating lighter noise
  - Divide adjacent independent objects
- Closing : Dilation + Erosion
  - Effective for eliminating darker noise
  - Link disconnection
- cv2.morphologyEx(src, op, kernel)
- src : Input image
- op : Operation type
  - cv2.MORPH_OPEN
  - cv2.MORPH_CLOSE
  - cv2.MORPH_GRADIENT : Difference between dilation & erosion
  - cv2.MORPH_TOPHAT : Difference between opening and original
  - cv2.MORPH_BLACKHAT : Difference between closing and original



Morphological opening operation



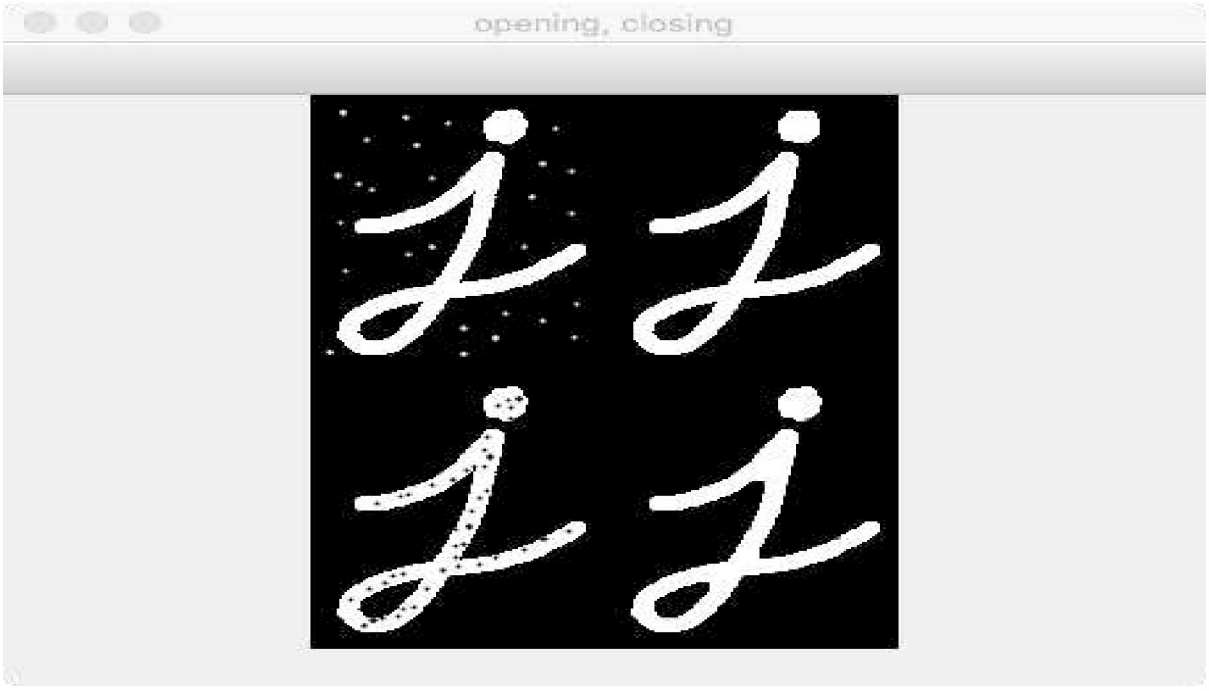
Morphological closing operation

❖ Opening & Closing

- Example

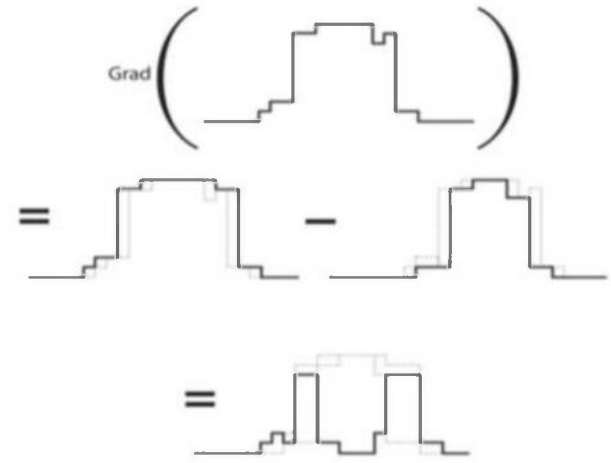
```
import cv2
import numpy as np

img1 = cv2.imread('../img/morph_dot.png', cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread('../img/morph_hole.png',
cv2.IMREAD_GRAYSCALE)

k = cv2.getStructuringElement(cv2.MORPH_RECT, (5,5))
opening = cv2.morphologyEx(img1, cv2.MORPH_OPEN, k)
closing = cv2.morphologyEx(img2, cv2.MORPH_CLOSE, k)

merged1 = np.hstack((img1, opening))
merged2 = np.hstack((img2, closing))
merged3 = np.vstack((merged1, merged2))
cv2.imshow('opening, closing', merged3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>



Gradient, Top-Hat, Black-Hat

- Gradient = dilate – erode
  - Leaving only boarder line
- TopHat = src – opening
  - Eliminate especially light value
- BlackHat = closing – src
  - Emphasize darker area



Morphological gradient operation

- Example

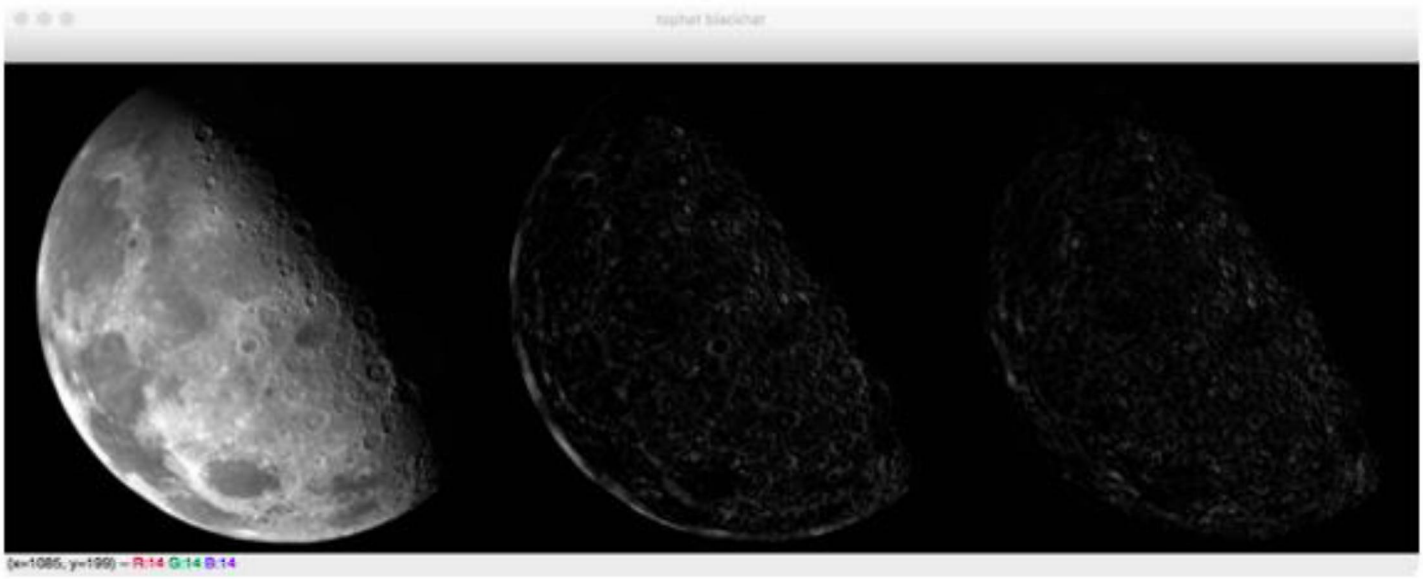
```
import cv2
import numpy as np

img = cv2.imread('../img/morphological.png')

k = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))
gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, k)

merged = np.hstack((img, gradient))
cv2.imshow('gradient', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Top-Hat, Black-Hat

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/moon_gray.jpg')

k = cv2.getStructuringElement(cv2.MORPH_RECT, (9,9))
tophat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, k)
blackhat = cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, k)

merged = np.hstack((img, tophat, blackhat))
cv2.imshow('tophat blackhat', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>

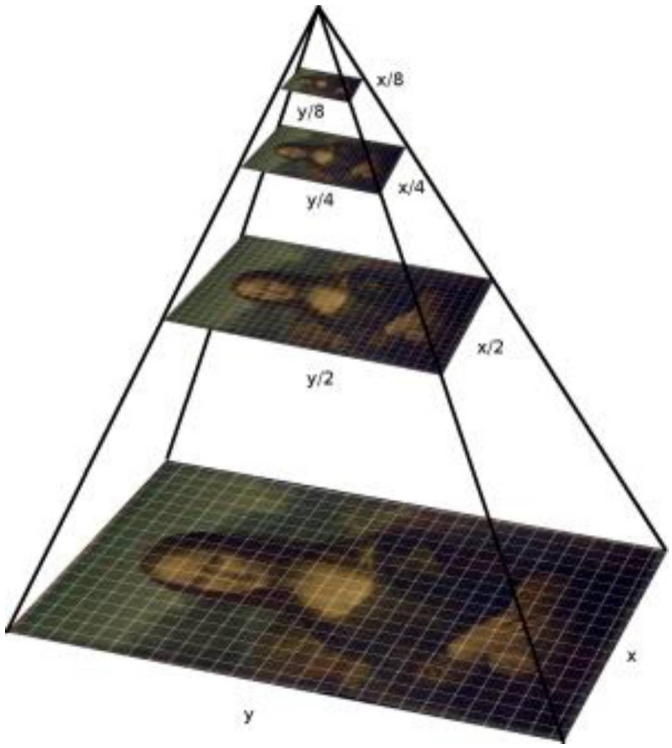


Filter

1. Convolution Filter
2. Blurring
3. Edge Detection
4. Morphology
5. **Image Pyramids**
6. Workshop

❖ Image Pyramids

- Stacking up like pyramid by reducing and enlarging image step-by-step
- Speed and accuracy
  - Analyze fast with small image, and have detail analysis for next level image
- Compensate result differences by size
- Gaussian Pyramids
  - Down Sampling/Scaling
  - Shrinking Images
- Laplacian Pyramids
  - Up Sampling/Scaling
  - Enlarging Images



❖ Gaussian Pyramids

- cv2.pyrDown(src [, dstsize, borderType])
  - src : Original Image
  - dtsize : Size of output image
  - Generate upper level image of pyramid
  - After applying Gaussian kernel, delete odd number column and row
  - Reduced to ¼ size
- cv2.pyrUp()
  - Generate lower level image of pyramid
  - Fill column and row with 0, convolution with designated filter and set
  - Enlarged to 4 times.
  - Blurred

- Example

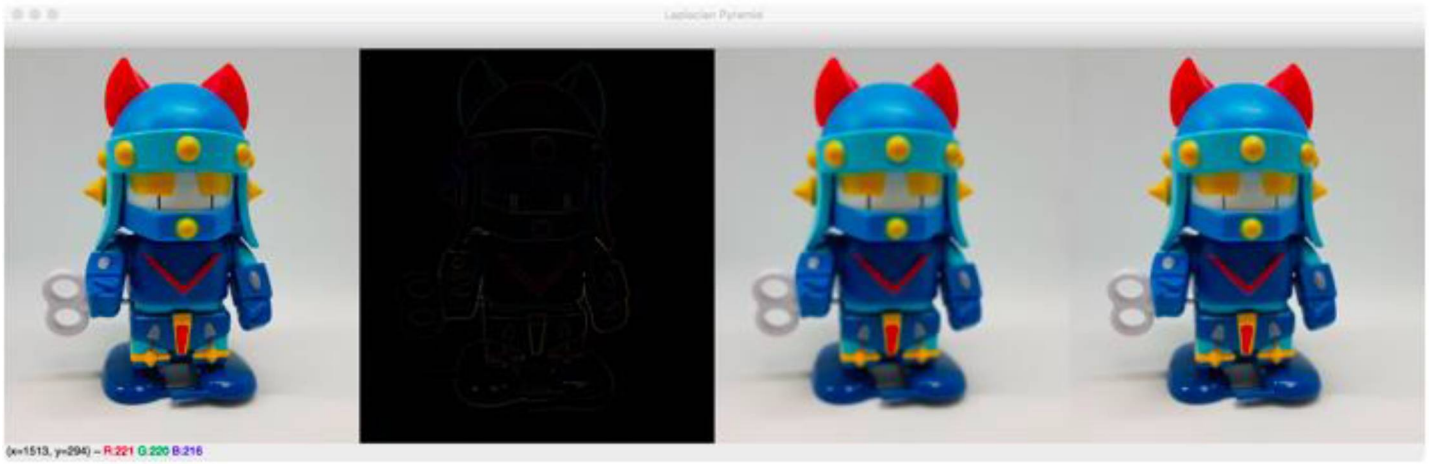
```
import cv2

img = cv2.imread('../img/girl.jpg')

smaller = cv2.pyrDown(img) # img x 1/4
bigger = cv2.pyrUp(img) # img x 4

cv2.imshow('img', img)
cv2.imshow('pyrDown', smaller)
cv2.imshow('pyrUp', bigger)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ LaplacianPyraminds

- Small level enlarging diffirence between original and Gaussian pyrmaid
- $L_i = G_i - \text{pyrUp}(G_{i+1})$
- Use restoring original from small pyramid
- Most pixel is 0(zero) except edge

```
import cv2
import numpy as np

img = cv2.imread('../img/taekwonv1.jpg')

smaller = cv2.pyrDown(img)
bigger = cv2.pyrUp(smaller)

laplacian = cv2.subtract(img, bigger)
restored = bigger + laplacian
merged = np.hstack((img, laplacian, bigger, restored))
cv2.imshow('Laplacian Pyramid', merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ LaplacianPyramids



Filter

1. Convolution Filter
2. Blurring
3. Edge Detection
4. Morphology
5. Image Pyramids
6. **Workshop**

❖ Mosaic2

- Select area to Mosaic
- Result Example

- Hint
  - Average blur for selected domain

❖ Cartoonizing Camera

- Give cartoon or water color effect to image
- Output sketchy image and watercolor image together.
- Result Example
- Hint
  - Sketch Image: Reverse to grey scale to get edge
    - cv2.Laplacian()
    - cv2.GauusianBlur()
  - Watercolor: Apply blur to color scale image and combine it with sketch image
    - cv2.blur()
    - cv2.bitwise_and()

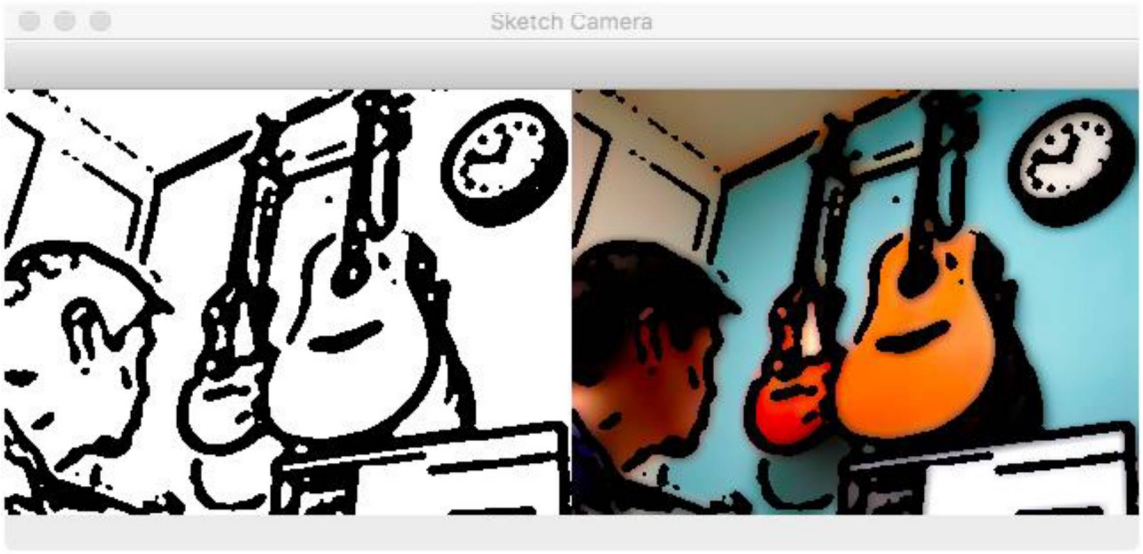
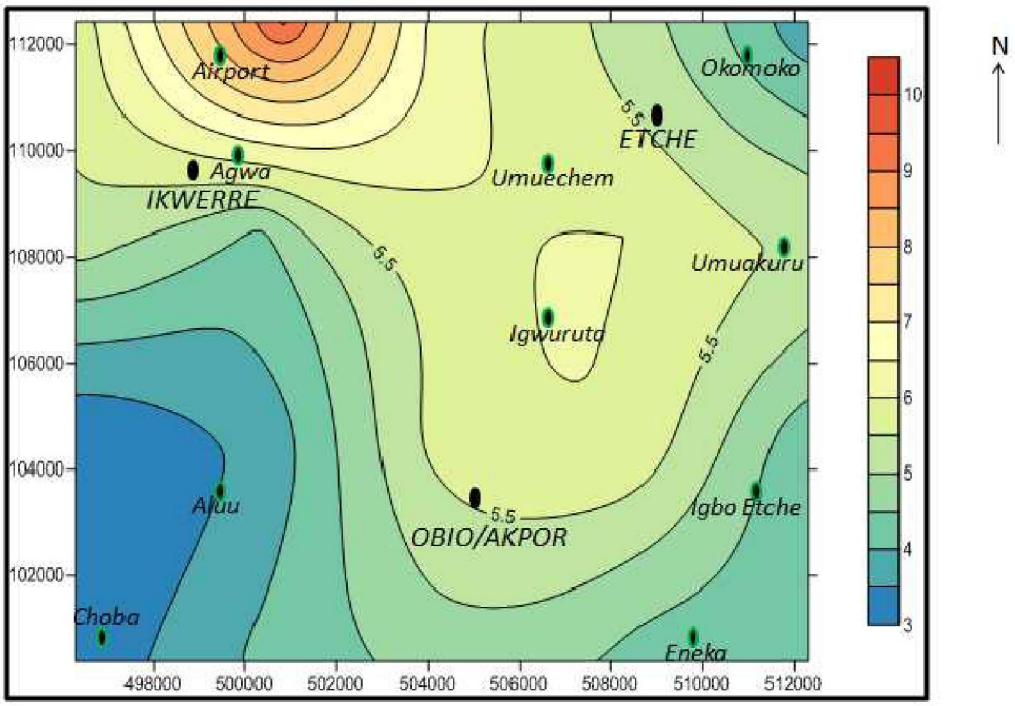


Image Segmentation

1. **Contour**
2. Hough Transform
3. Connected Component
4. Workshop

❖ Contours

- A curve connecting dots of same color or light
- Effective for figure analysis and object recognition
- Better to binaries for better accuracy
  - Threshold, Canny Edge
  - Finding white objects from black background



❖ Contours

- dst, contours, hierarchy = cv2.findContours(img, mode, method)
  - img : Input image
  - mode : Result contour Mode
    - cv2.RETR_EXTERNAL : External boarder line
    - cv2.RETR_LIST : All line without hierarchy
    - cv2.RETR_CCOMP : All line as 2 hierarchy
    - cv2.RETR_TREE : All line, all hierarchy info. In tree structure
  - method : approximation method
    - cv2.CHAIN_APPROX_NONE : Save all coordination

        - cv2.CHAIN_APPROX_SIMPLE : Save only vertex coordination
        - cv2.CHAIN_APPROX_TC89_L1 : Teh-Chin approximate algorithm
        - cv2.CHAIN_APPROX_TC89_KC0S : Teh-Chin approximate algorithm
        - dst : Edited image, OpenCV3.2+ (Original is damaged in previous version)
    - contours : Detected contour coordination, Python List
    - hierachy : Hierarchy info.
        - Next, Prev, FirstChild, Parent
        - -1 : Do not apply

❖ Contours

- Draw line following given contour (Regardless of #, show all lines)
- cv2.drawContours(img, contours, contourIdx, color, thickness)
    - img : Input image
    - contours : Python List
    - contourIdx : draw subject index, -1: all
    - color
    - thickness: line thickness, 0 : fill
- Example < 1/2 >

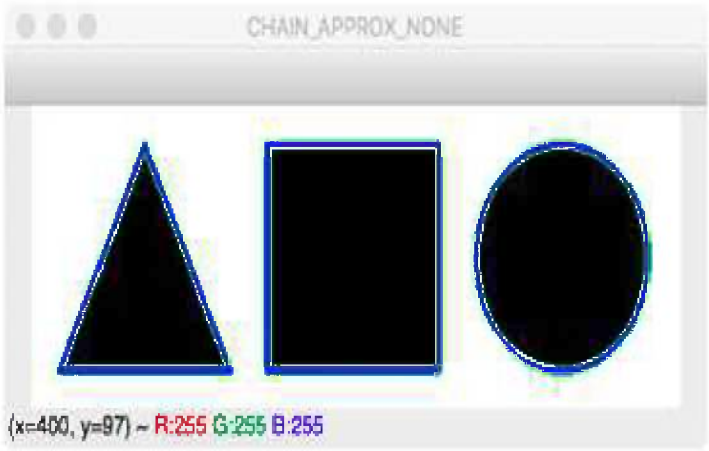
```
import cv2
import numpy as np

img = cv2.imread('../img/shapes.png')
img2 = img.copy()
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, imthres = cv2.threshold(imgray, 127, 255,
cv2.THRESH_BINARY_INV)

im2, contour, hierarchy = cv2.findContours(imthres,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
im2, contour2, hierarchy = cv2.findContours(imthres,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
print(‘# of figures: %d(%d)'% (len(contour), len(contour2)))
```

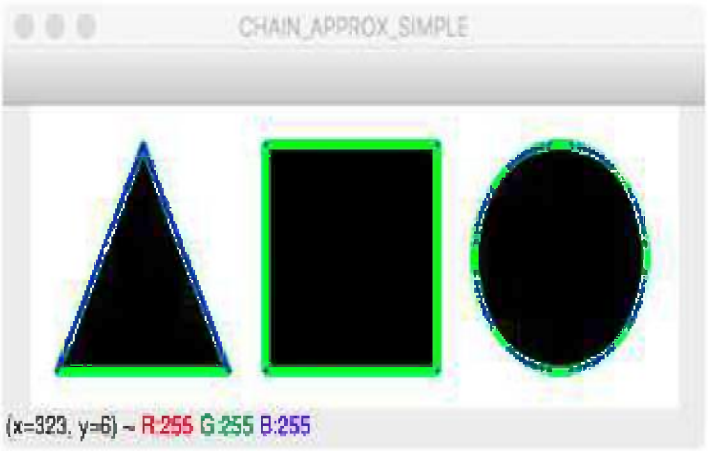
❖ Contours

- Example < 2/2 >

```
cv2.drawContours(img, contour, -1, (0,255,0), 4)
cv2.drawContours(img2, contour2, -1, (0,255,0), 4)

for i in contour:
    for j in i:
        cv2.circle(img, tuple(j[0]), 1, (255,0,0), -1)
for i in contour2:
    for j in i:
        cv2.circle(img2, tuple(j[0]), 1, (255,0,0), -1)

cv2.imshow('CHAIN_APPROX_NONE', img)
cv2.imshow('CHAIN_APPROX_SIMPLE', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>





Contour Hierarchy

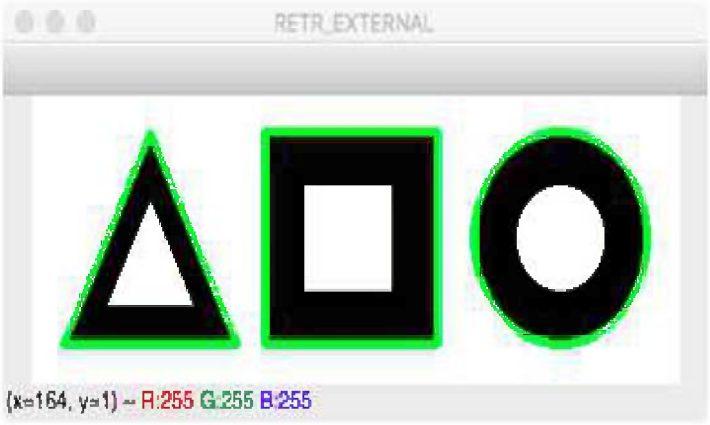
- Example

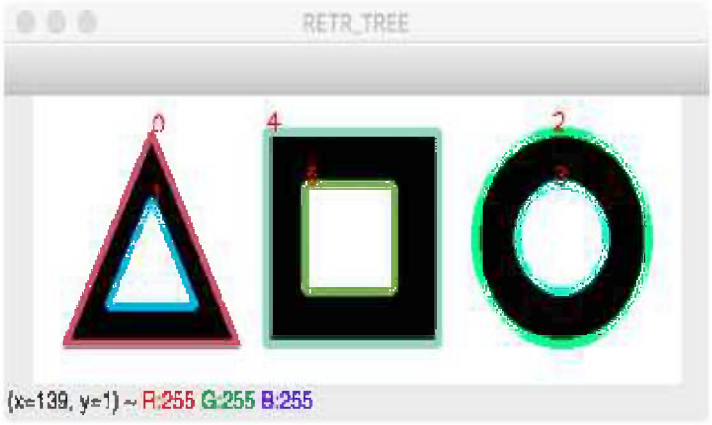
```
import cv2
import numpy as np

img = cv2.imread('../img/shapes_donut.png')
img2 = img.copy()
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, imthres = cv2.threshold(imgray, 127, 255,
cv2.THRESH_BINARY_INV)

im2, contour, hierarchy = cv2.findContours(imthres,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
print(len(contour), hierarchy)

im2, contour2, hierarchy = cv2.findContours(imthres, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
print(len(contour2), hierarchy)

cv2.drawContours(img, contour, -1, (0,255,0), 3)
for idx, cont in enumerate(contour2):
    color = [int(i) for i in np.random.randint(0,255, 3)]
    cv2.drawContours(img2, contour2, idx, color, 3)
    cv2.putText(img2, str(idx), tuple(cont[0][0]),
cv2.FONT_HERSHEY_PLAIN, 1, (0,0,255))

cv2.imshow('RETR_EXTERNAL', img)
cv2.imshow('RETR_TREE', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Contour Hierarchy

- Example <Result>

```
3 [[[ 1 -1 -1 -1]
  [ 2  0 -1 -1]
  [-1  1 -1 -1]]]
6 [[[ 2 -1  1 -1]
  [-1 -1 -1  0]
  [ 4  0  3 -1]
  [-1 -1 -1  2]
  [-1  2  5 -1]
  [-1 -1 -1  4]]]
```





❖ Image Moment

- Moment
  - Physics, Terminology for the quantity of force
- Image Moment
  - Show quantity of the object
  - $m_{p,q} = \sum_x \sum_y f(x,y) x^p y^q$
  - Area that contour is surrounding; addition of multiplication of x,y coordination pixel value and coordination index p,q
  - Binary image: If it is not 0, it is 1

  - p, q degree : Calculate by changing from 0 ~3, get meaningful info
- m00 : Area of contour
  - 0 degree of all number is 1, Add by multiplying by 1 to the number of area
- Center moment
  - $\mu_{p,q} = \sum_x \sum_y f(x,y)(x-\bar{x})^p (y-\bar{y})^q$
  - $\bar{x} = \frac{m_{10}}{m_{00}}$
  - $\bar{y} = \frac{m_{01}}{m_{00}}$

❖ Image Moment

- moment = cv2.moments(contour)
  - contour :Coordination of the calculation subject for moment
  - moment : Result Moment, Python dictionary
    - m00, m01, m10, m11, m02, m12, m20, m21, m03, m30 : Space moment
    - mu20, mu11, mu02, mu30, mu21, mu12, mu03 : Center moment
    - nu20, nu11, nu02, nu30, nu21, nu03 : Regular center moment
- retval = cv.contourArea(contour[, oriented=False]): Claculate area with contour
  - contour: Contour to calculate area
  - oriented: Contour orientation flag
    - True: Turnaround negative depends on contour orientation
    - False : Turnround absolute value
  - retval : Area of Contour
- retval = cv.arcLength(curve, closed) : Calculate circumference
  - curve : Contour to calculate
  - closed : Close or not flag
  - retval : Value of countour circumference

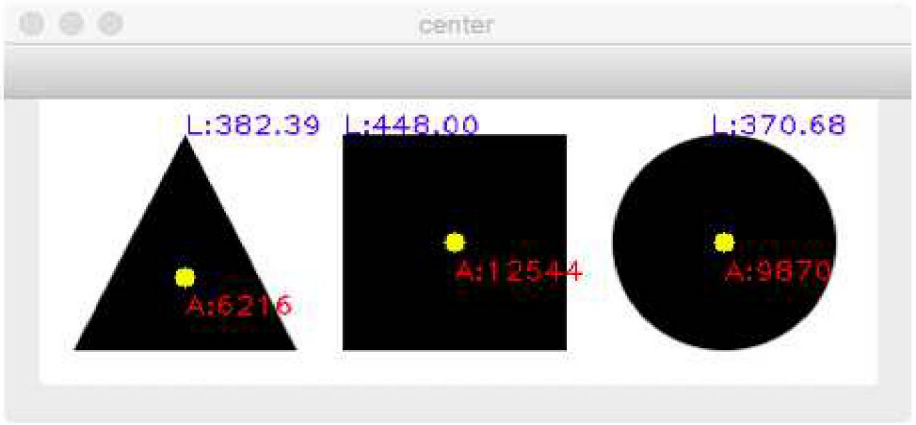
❖ Image Moment

- Center Point, Area, Circumference Example < 1/2 >

```
import cv2
import numpy as np
img = cv2.imread("../img/shapes.png")
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, th = cv2.threshold(imgray, 127, 255, cv2.THRESH_BINARY_INV)
img2, contours, hierachy = cv2.findContours(th, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
for c in contours:
    mmt = cv2.moments(c)
    cx = int(mmt['m10']/mmt['m00'])
    cy = int(mmt['m01']/mmt['m00'])
    a = mmt['m00']
    l = cv2.arcLength(c, True)
   cv2.circle(img, (cx, cy), 5, (0, 255, 255), -1)
    cv2.putText(img, "A:%.0f"%a, (cx, cy+20) ,
cv2.FONT_HERSHEY_PLAIN, 1, (0,0,255))
    cv2.putText(img, "L:%.2f"%l, tuple(c[0][0]),
cv2.FONT_HERSHEY_PLAIN, 1, (255,0,0))
    print("area:%.2f"%cv2.contourArea(c, False))
cv2.imshow('center', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Center Point, Area, Circumference Example <Result>

```
area:9870.00
area:12544.00
area:6216.00
```



❖ Bounding Shapes

- x,y,w,h = cv2.boundingRect(contour) : Calculate quadrangle surrounding coordination
  - x, y : Left top coordination of quadrangle
  - w, h : Width, Height
- rotateRect = cv2.minAreaRect(contour) : Minimum quadrangle surrounding coordination
  - rotateRect : Coordination of rotated quadrangle
    - center : Center point (x,y)
    - size : Size (w, h)
    - angle : Rotation angle, positive: clockwise, Negative: counter clockwise
- vertex = cv2.boxPoints(rotateRect): Vertex coordination from rotateRect
  - vertex : 4 Vertex, including decimal, need to change to integer
- center, radius = cv2.minEnclosingCircle(contour) :
  - Minimum circle surrounding coordination
  - center : Original coordination (x, y), Tuple
  - radius : Radius
- area, triangle = cv.minEnclosingTriangle(points) : Minimum triangle surrounding coordination
  - area : Area
  - triangle : Coordination of 3 vertex
- ellipse = cv.fitEllipse(points) : Coordination of minimum eclipse surrounding coordination
  - ellipse
    - center : Center coordination(x,y),Tuple
    - axes : :Length of axis(x, y), Tuple
    - angle : Rotation Angle
- line = cv.fitLine(points, distType, param, reps, aeps[, line]) : Calculate straight line passing center point
  - distType : Calculating distance

- cv2.DIST_L2, cv2.DIST_L1, cv2.DIST_L12, cv2.DIST_FAIR, cv2.DIST_WELSCH, cv2.DIST_HUBER
- param : Parameter to deliver to distType, 0= Select optimal value
- reps : Accuracy of radius, Distance between line and original coordination, Recommend 0.01
- aeps : Angle accuracy, Recommend 0.01
- line
  - vx, vy : Regular unit vector, Slope of straight line, Tuple
  - x0, y0 : Center point, coordination, Tuple

❖ Bounding Shapes

- Example < 1/2 >

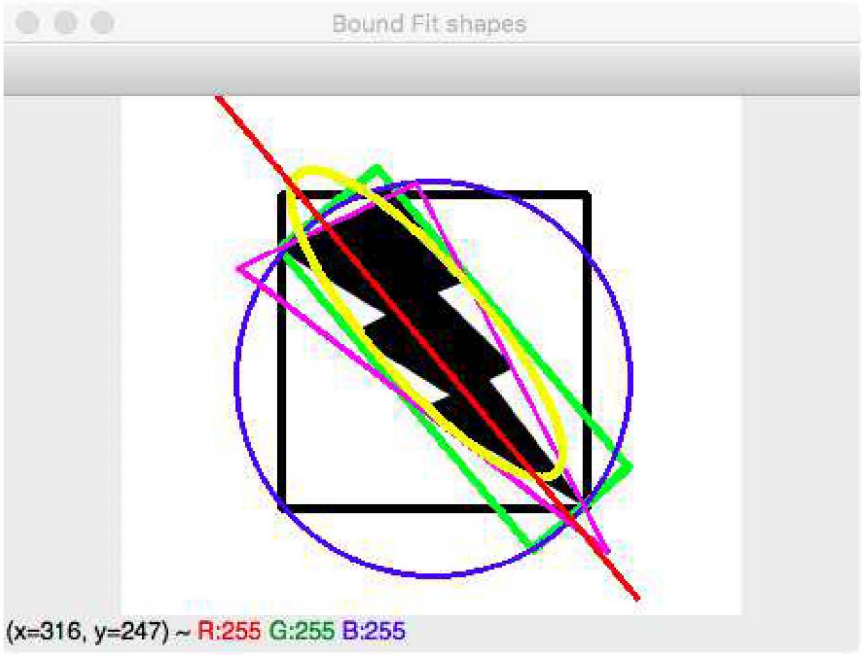
```
import cv2
import numpy as np

img = cv2.imread("../img/lightning.png")
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, th = cv2.threshold(imgray, 127,255,cv2.THRESH_BINARY_INV)

im, contours, hr = cv2.findContours(th, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
contr = contours[0]

x,y,w,h = cv2.boundingRect(contr)
cv2.rectangle(img, (x,y), (x+w, y+h), (0,0,0), 3)

rect = cv2.minAreaRect(contr)
box = cv2.boxPoints(rect)   # 중심점과 각도를 4개의 꼭지점 좌표로 변환
box = np.int0(box)          # 정수로 변환
cv2.drawContours(img, [box], -1, (0,255,0), 3)
```

❖ Bounding Shapes

- Example < 2/2 >

```
(x,y), radius = cv2.minEnclosingCircle(contr)
cv2.circle(img, (int(x), int(y)), int(radius), (255,0,0), 2)

ret, tri = cv2.minEnclosingTriangle(contr)
cv2.polylines(img, [np.int32(tri)], True, (255,0,255), 2)

ellipse = cv2.fitEllipse(contr)
cv2.ellipse(img, ellipse, (0,255,255), 3)

[vx,vy,x,y] = cv2.fitLine(contr, cv2.DIST_L2,0,0.01,0.01)
cols,rows = img.shape[:2]
cv2.line(img,(0, 0-x*(vy/vx) + y), (cols-1, (cols-x)*(vy/vx) +
y),(0,0,255),2)

cv2.imshow('Bound Fit shapes', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>



❖ Contour Approximation

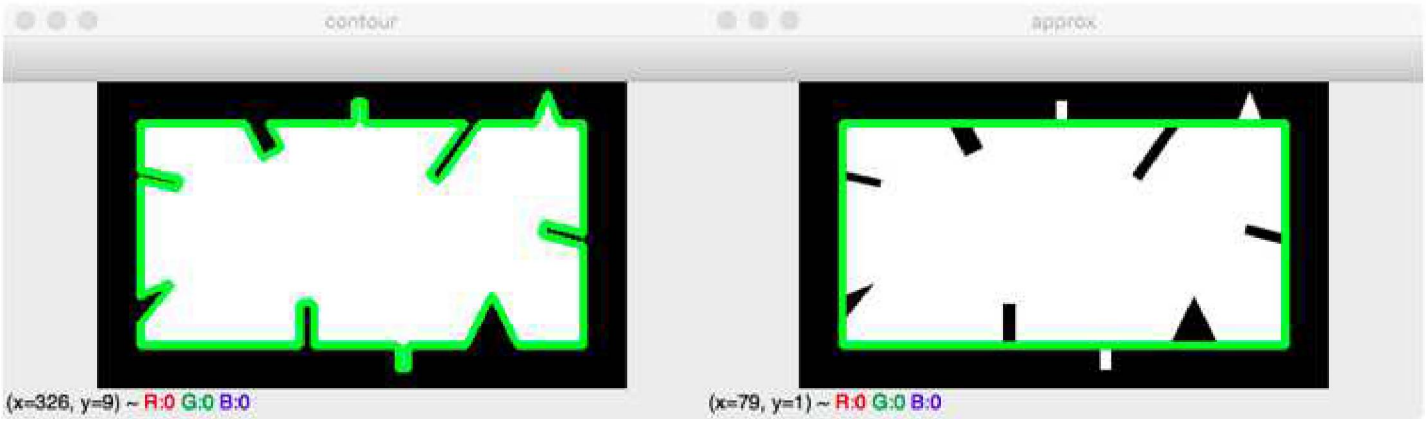
- When accurate contour is inaccurate
  - Noise, Erosion
- Douglas-Peucker Algorithm
  - Set epsilon Value: Max distance from contour to approximate contour
- approx = cv2.approxPolyDP(contour, epsilon, closed)
  - contour : Contour coordination
  - epsilon : Accuracy of approximate value, margin of error
  - closed : Rather contour is closed
  - approx : Coordination of approximate contour

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/bad_rect.png')
img2 = img.copy()
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, th = cv2.threshold(imgray, 127, 255, cv2.THRESH_BINARY)
temp, contours, hierachy = cv2.findContours(th, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
contour = contours[0]
epsilon = 0.05 * cv2.arcLength(contour, True)
approx = cv2.approxPolyDP(contour, epsilon, True)
cv2.drawContours(img, [contour], -1, (0,255,0), 3)
cv2.drawContours(img2, [approx], -1, (0,255,0), 3)
cv2.imshow('contour', img)
cv2.imshow('approx', img2)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Contour Approximation

- Example <Result>



❖ Convex Hull

- Convex Hull: without concaved part, object formed with convex curve
- Coordination of object boarder area
- Effective in finding actual area and center point of the object
- hull = cv2.convexHull(points, hull, clockwise, returnPoints)
  - points : contours
  - hull : output, avoid
  - clockwise : Set orientation, True/False
  - returnPoints :
    - True*: hull points turnaround coordination
    - False : Turnaround hull given index of contour
- ret = cv2.isContourConvex(contour) : True/False
  - ret : Check Convex
- defects = cv2.convexityDefects(contour, convexhull) : Finding defect of convex hull
  - contour : Input contour
  - convexhull : Contour index for convex object
  - defects: Array index of contour with convex hull defects, N x 1 x 4 array

- [start, end, farthest, distance]
  - start : Index of contour starting convex angle
  - end : Index of contour ending convex angle
  - farthest: Contour index of farthest from convex hull
  - distance : Distance from farthest to convex hull
    - 8 bit fixed decimal (distance/256.0)

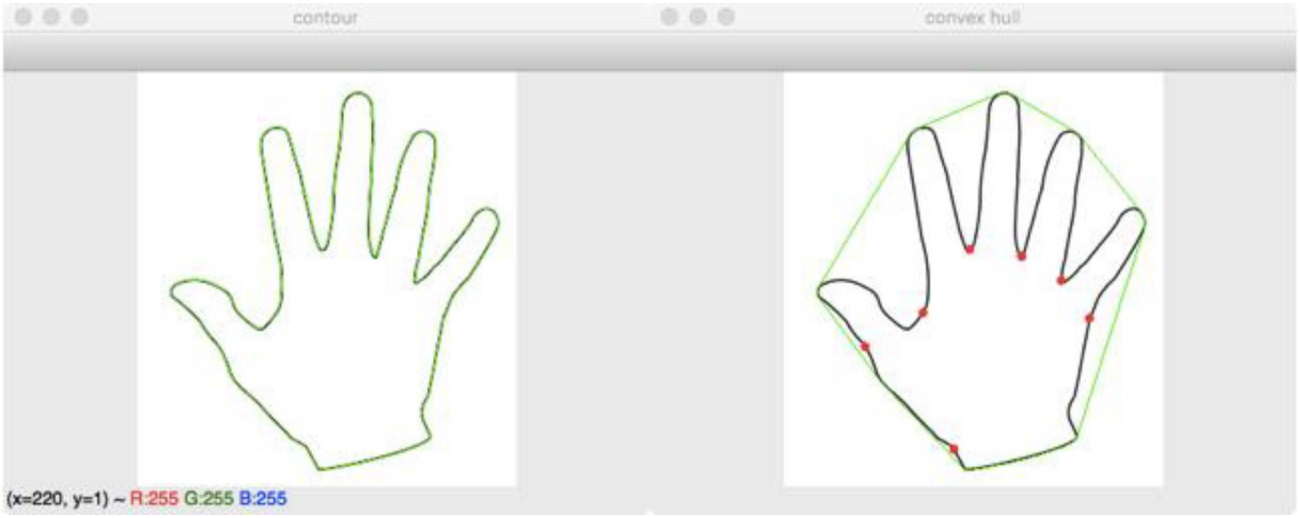
❖ Convex Hull

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/hand.jpg')
img2 = img.copy()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, th = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV)

temp, contours, heiarchy = cv2.findContours(th, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cntr = contours[0]
cv2.drawContours(img, [cntr], -1, (0, 255,0), 1)

hull = cv2.convexHull(cntr)
cv2.drawContours(img2, [hull], -1, (0,255,0), 1)
print(cv2.isContourConvex(cntr), cv2.isContourConvex(hull))
hull2 = cv2.convexHull(cntr, returnPoints=False)
defects = cv2.convexityDefects(cntr, hull2)
for i in range(defects.shape[0]):
    startP, endP, farthestP, distance = defects[i, 0]
    farthest = tuple(cntr[farthestP][0])
    dist = distance/256.0
    if dist > 1 :
        cv2.circle(img2, farthest, 3, (0,0,255), -1)
cv2.imshow('contour', img)
cv2.imshow('convex hull', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Convex Hull

- Example <Result>



❖ Point Polygon Test

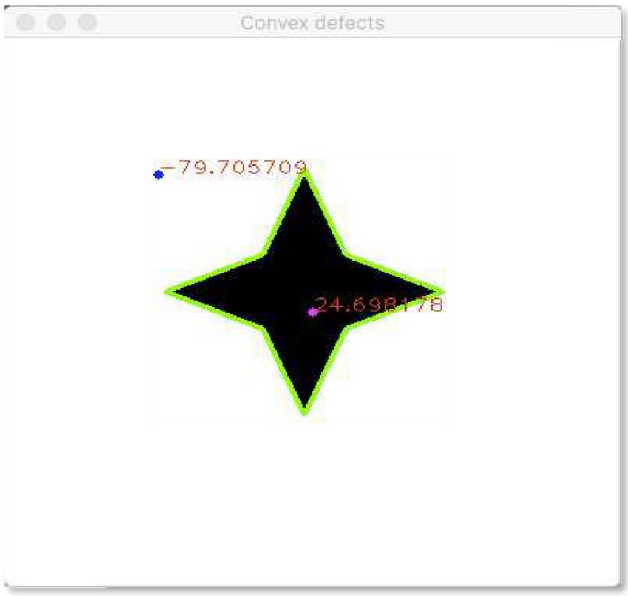
- Calculate distance between certain point of image to contour
  - cv2.pointPolygonTest(img, pt, measureDist)
    - img : input image
    - pt : (x,y) , Measured point
    - measureDist : True/False
      - True : Distance measure
      - False : Show only internal and external (-1,0,1)
    - Turnaround
      - Positive: Internal
      - Negative: External

❖ Point Polygon Test

- Example

```
img = cv2.imread('../img/4star.jpg')
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, th = cv2.threshold(imgray, 127,255, cv2.THRESH_BINARY_INV)
_, contours, _ = cv2.findContours(th, cv2.RETR_EXTERNAL, \
                                        cv2.CHAIN_APPROX_SIMPLE)
contour = contours[0]
cv2.drawContours(img, [contour], 0, (0,255,0),2)
p1 = (100,100)
p2 = (200,200)
cv2.circle(img, p1, 3, (255,0,0), -1)
cv2.circle(img, p2, 3, (255,0,255), -1)
dist1 = cv2.pointPolygonTest(contour, p1, True)
dist2 = cv2.pointPolygonTest(contour, p2, True)
cv2.putText(img, '%f'%dist1, p1, cv2.FONT_HERSHEY_PLAIN, 1,
(0,0,255), 1)
cv2.putText(img, '%f'%dist2, p2, cv2.FONT_HERSHEY_PLAIN, 1,
(0,0,255), 1)
cv2.imshow('Convex defects', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>



❖ Match Shape

- Compare two different contours
- ret = cv2.matchShape(cntr1, cntr2, method, param)
  - cntr1, cntr2 : contours to compare
  - method : Hu Moment comparing algorithms
    - cv2.CONTOURS_MATCH_I1
    - cv2.CONTOURS_MATCH_I2
    - cv2.CONTOURS_MATCH_I3
  - param : Parameter delivering algorithm, 0.0 (not supported)
  - ret:
    - Similarity: Smaller it is, similar it is, 0=Same
- Example <1/2>

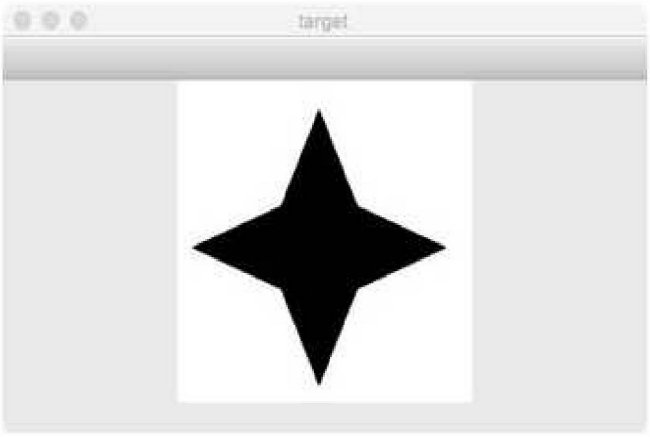
```
import cv2
import numpy as np

target = cv2.imread('../img/4star.jpg') # 매칭 대상
shapes = cv2.imread('../img/shapestomatch.jpg') # 여러 도형
targetGray = cv2.cvtColor(target, cv2.COLOR_BGR2GRAY)
shapesGray = cv2.cvtColor(shapes, cv2.COLOR_BGR2GRAY)
ret, targetTh = cv2.threshold(targetGray, 127, 255,
cv2.THRESH_BINARY_INV)
ret, shapesTh = cv2.threshold(shapesGray, 127, 255,
cv2.THRESH_BINARY_INV)

_, cntrs_target, _ = cv2.findContours(targetTh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
_, cntrs_shapes, _ = cv2.findContours(shapesTh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

❖ Match Shape

- Example <2/2>

```
matchs = [] # List to save contour and matching points
for contr in cntrs_shapes:
    match = cv2.matchShapes(cntrs_target[0], contr,
cv2.CONTOURS_MATCH_I2, 0.0)
    matchs.append( (match, contr) )
    cv2.putText(shapes, '%.2f'%match, tuple(contr[0][0]),
cv2.FONT_HERSHEY_PLAIN, 1,(0,0,255),1 )
matchs.sort(key=lambda x : x[0])

cv2.drawContours(shapes, [matchs[0][1]], -1, (0,255,0), 3)
cv2.imshow('target', target)
cv2.imshow('Match Shape', shapes)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Example <Result>



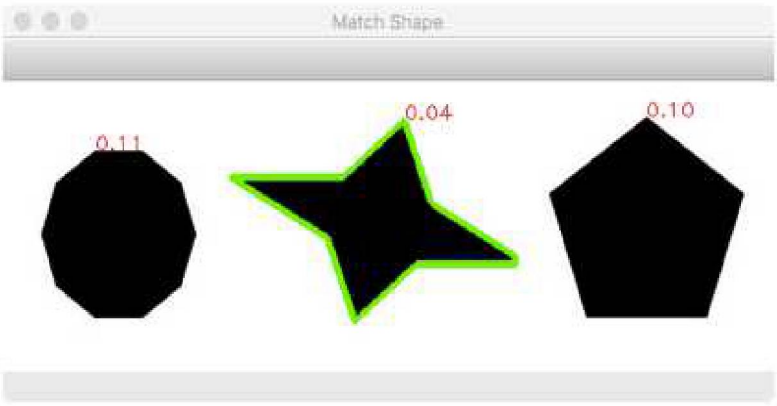
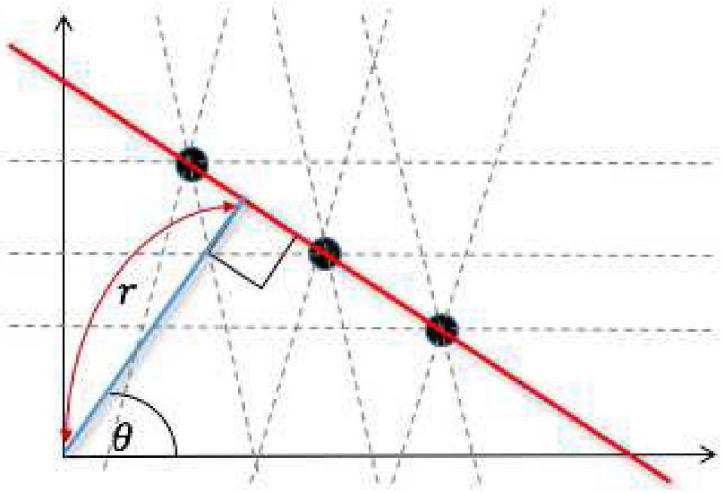


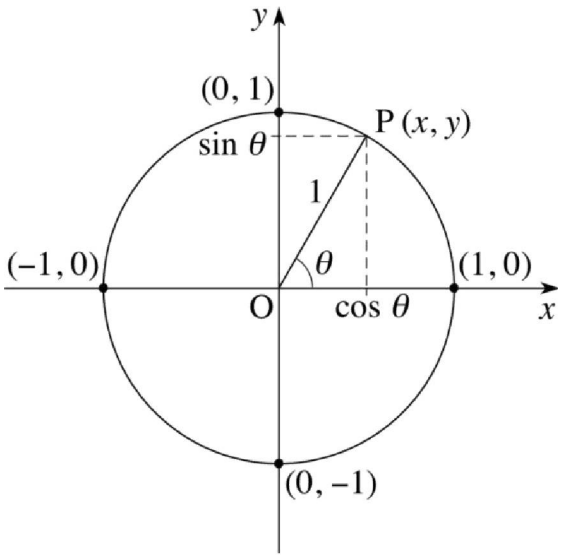
Image Segmentation

1. Contour
2. **Hough Transform**
3. Connected Component
4. Workshop

❖ Hough Transform

- Distinguish line, circle simple shame
- Expanded to various figures, starting from line finding method
- Paul Hough First Discovered (bubble chamber), 1962
- Richard Duda and Peter Hart Generalized, 1972
- Popularized by Dana H. Ballard's Computer Vision, 1981
- Hough Line Transform
- Hough Circle Transform
- Detect straight line from numerous dots in images
- r = cos θx + sin θy
- For one dot (r, θ)
- Straight line connecting all same (r, θ) of all dots





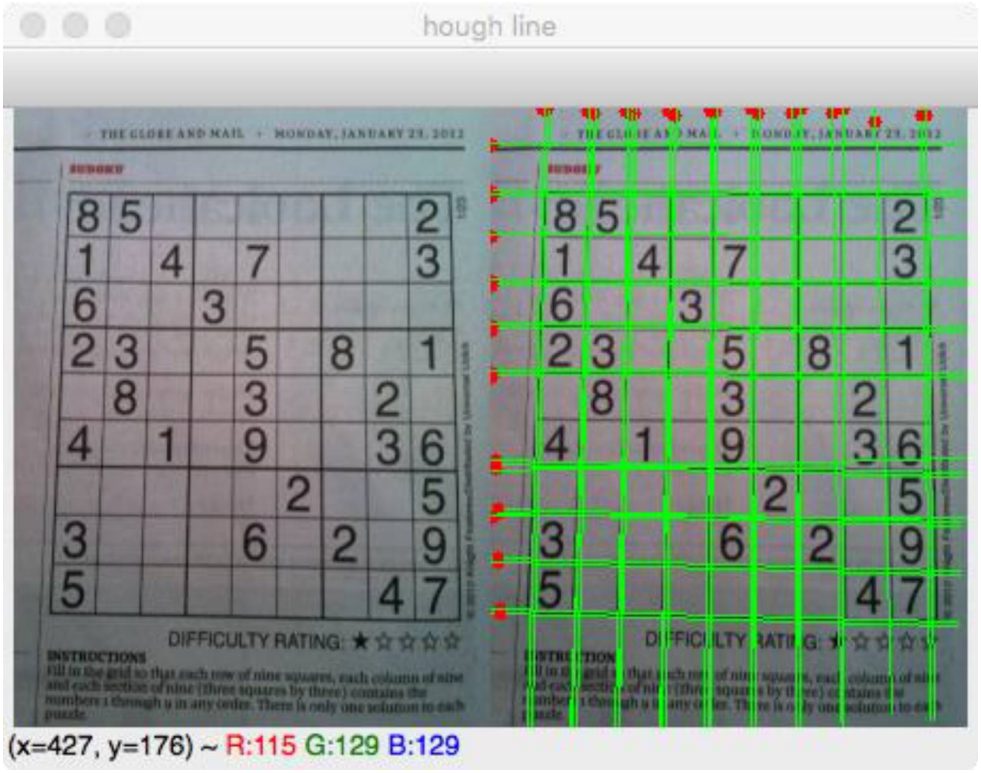
❖ Hough Line Transform

- lines = cv2.HoughLines(img, rho, theta, threshold[, lines, srn=0, stn=0, min_theta, max_theta])
  - img: Input image, 1 channel binary scale
  - rho: Resolution of distance, 0~1
  - theta: Resolution of angle, Radian unit (np.pi/0 ~180)
  - threshold: Minimum number of matching to judge straight line
    - Small value: Accuracy drops but # of detected value increases
    - Big Value: Accuracy increases but # of detected value decreases
  - lines : Detecting Result, N x 1 x 2 array (r,θ)
  - srn, stn : Use for multi scale huff transform, not used in detection
  - min_theta, max_theta : Max and min angle to use for detection
- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/sudoku.jpg')
img2 = img.copy()
h, w = img.shape[:2]
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(imgray, 100, 200 )
lines = cv2.HoughLines(edges, 1, np.pi/180, 130)
for line in lines: # Tour all detected lines
    r,theta = line[0] # Distance and angle
    tx, ty = np.cos(theta), np.sin(theta) # Trigonometric ration to x, y
    x0, y0 = gx*r, gy*r  #Coordination based on x, y
    cv2.circle(img2, (abs(x0), abs(y0)), 3, (0,0,255), -1)
    x1, y1 = int(x0 + w*(-ty)), int(y0 + h * tx)
    x2, y2 = int(x0 - w*(-ty)), int(y0 - h * tx)
    cv2.line(img2, (x1, y1), (x2, y2), (0,255,0), 1)
merged = np.hstack((img, img2))
cv2.imshow('hough line', merged)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Hough Line Transform

- Example <Result>



❖ Probabilistic Hough Line Transform

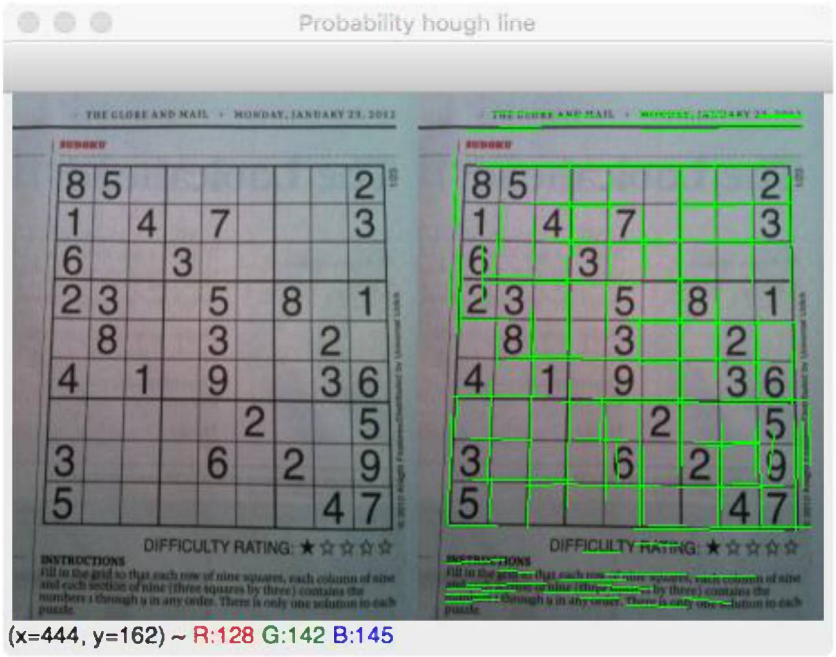
- Avoid inefficiency of calculating all pixels
- Randomly select pixel
- lines = cv2.HoughLineP(src, rho, theta, threshold, minLineLen, maxLineGap)
  - src : Input image
  - rho : Distance accuracy (0~1)
  - theta : Angle accuracy, radian (np.pi/0~180)
  - threshold : Min, matching value found to be straight line
  - minLineLen : If the length is shorter than this, drop
  - maxLineGap : If the distance is bigger than this, it is different line
  - lines : Detected line coordination
    - nx1x 4 array for two lines in straight line (x1, y1, x2, y2)

❖ Probabilistic Hough Line Transform

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/sudoku.jpg')
img2 = img.copy()
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(imgray, 100, 200 )

lines = cv2.HoughLinesP(edges, 1, np.pi/180, 50, 5, 10)
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(img, (x1,y1), (x2, y2), (0,255,0), 1)

cv2.imshow('Probability hough line', img)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Example <Result>



❖ Hough Circle Transform

- Transform rectangular coordination to polarity coordination
- Able to detect circle with straight line transform algorithm
- Too much calculation, slow

- Hough Gradient Method
  - OpenCV operation
  - Detect edge with Canny
  - Calculate gradient orientation with Sobel
  - Calculate center and radius of circle, 3D accumulator
  - Inaccurate b/c it gets gradient orientation using Sobel
  - Difficult to detect overlapping concentric circle
- circles = cv2.HoughCircle(src, method, dp, minDst, circles, param1, param2, minRds, maxRds)
  - src : Input image
  - method : cv2.HOUGH_GRADIENT, only available
    - cv.HOUGH_STANDARD
    - cv.HOUGH_PROBABILISTIC
    - cv.HOUGH_MULTI_SCALE
    - cv.HOUGH_GRADIENT
  - dp : Inverse proportion rate of resolution between accumulator and input image
    - 1: same, 2: ½, small value : Increase accuracy,
    - Big value: Increase # detected
  - minDst : Min value between center points of circle, 0=err, unable to detect concentric circle
  - circles : Circle Result, N x 1 x 3 (x, y, r)
  - param1 : maxValue used for Canny Edge
  - param2 : HUGH_GRADENT threshold ,
    - Small value: increase # detected, decrease accuracy
  - minRadius, maxRadius: Min radius of circule, max radius (if 0, size of image)

❖ Hough Circle Transform

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/coins_spread1.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (3,3), 0)
circles = cv2.HoughCircles(blur, cv2.HOUGH_GRADIENT, 1.5, 30, None,
200)
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0,:]:
        cv2.circle(img,(i[0], i[1]), i[2], (0, 255, 0), 2)
        cv2.circle(img, (i[0], i[1]), 2, (0,0,255), 5)

cv2.imshow('hough circle', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>

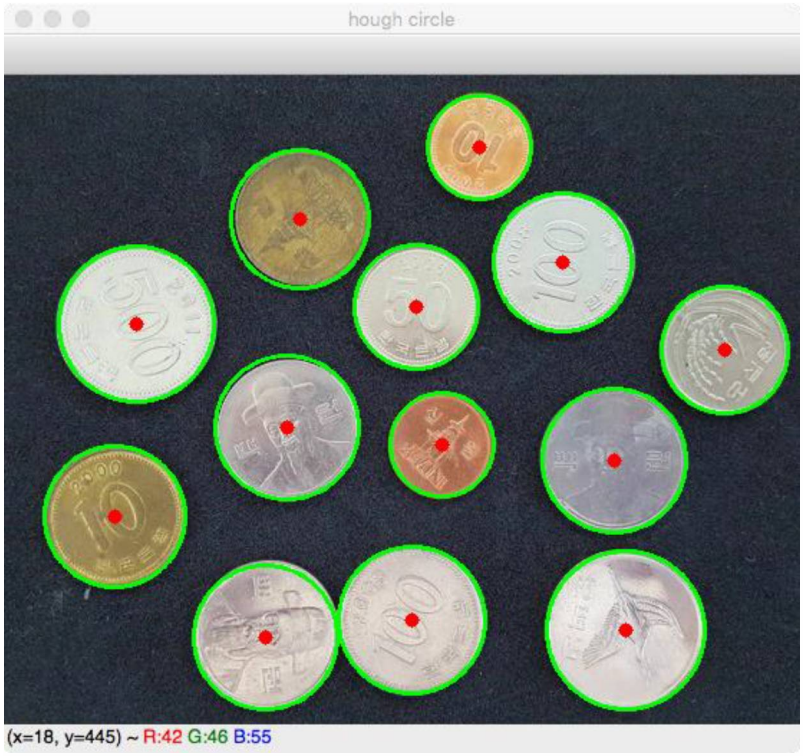
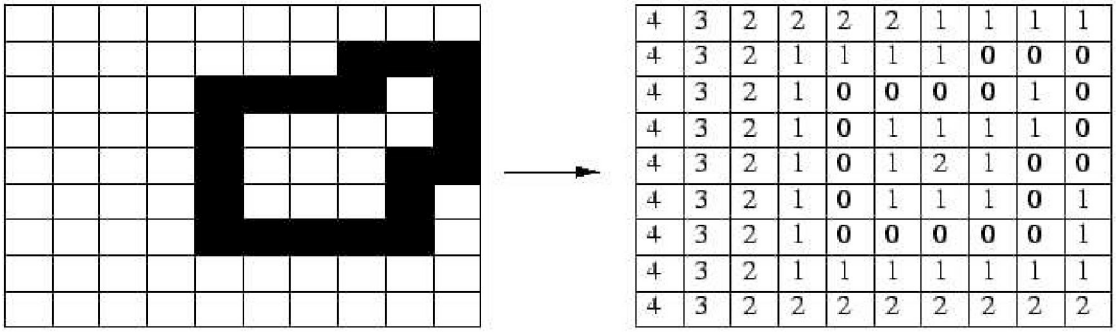


Image Segmentation

1. Contour
2. Hough Transform
3. **Connected Component**
4. Workshop

❖ Distance Transform

- Finding center point of object
- Farthest from the boundary
- For binary scale image
- Starting from 0 where pixel value is 0, it increases by 1 as it distances from 0
- cv2.distanceTransform(src, distanceType, maskSize)
  - src : Input image, binary scale
  - distanceType : Selecting distance calculation type
    - cv2.DIST_L2, cv2.DIST_L1, cv2.DIST_L12, cv2.DIST_FAIR, cv2.DIST_WELSCH, cv2.DIST_HUBER
  - maskSize : Distance transform kernel size



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

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/full_body.jpg', cv2.IMREAD_GRAYSCALE)
_, biimg = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY_INV)

dst = cv2.distanceTransform(biimg, cv2.DIST_L2, 5)
dst = (dst/(dst.max()-dst.min()) * 255).astype(np.uint8)
skeleton = cv2.adaptiveThreshold(dst, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 7, -3)
cv2.imshow('origin', img)
cv2.imshow('dist', dst)
cv2.imshow('skel', skeleton)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Example <Result>

❖ Labeling

- 연결된 요소들 끼리 분리
- 픽셀 값이 0으로 끊어 지지 않는 여역 끼리 같은 레이블 값 부여
- retval, labels = cv.connectedComponents(src[,labels, connectivity=8, ltype]) : 연결 요소 레이블링과 갯수 반환
  - src : input image, 바이너리 스케일 이미지
  - labels : 레이블링 된 input image과 같은 크기의 배열
  - connectivity : 연결성 검사할 방향 갯수, 4, 8 중 선택
  - ltype : Result 레이블 배열 dtype
  - retval : 레이블 갯수
- retval, labels, stats, centroids = cv.connectedComponentsWithStats(src[, labels, stats, centroids, connectivity, ltype]) : 레이블링과 각종 상태 정보 반환
  - stats : N x 5 행렬(N:레이블 갯수)
    - [x좌표, y좌표, 폭, 높이, 넓이]
  - centroids : 각 레이블의 중심점 좌표, N x 2 행렬(N:레이블 갯수)

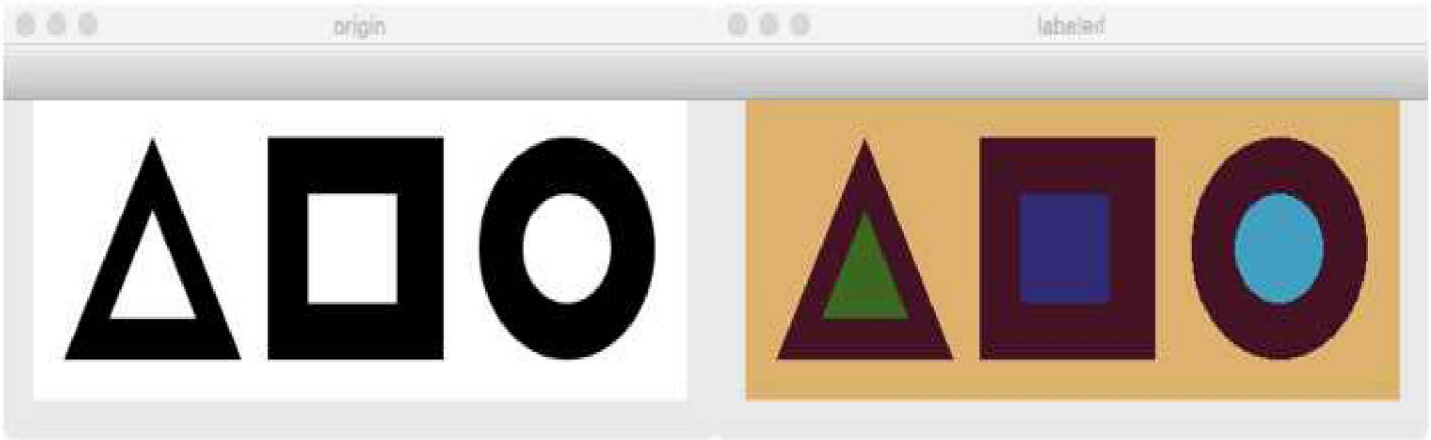
	1	1	1						
	1	1	1		2	2	2	2	
	1	1	1						
			1			3			
						3			
				3	3	3	3	3	

❖ Labeling

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/shapes_donut.png')
img2 = np.zeros_like(img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, th = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

cnt, labels = cv2.connectedComponents(th)
#retval, labels, stats, cent = cv2.connectedComponentsWithStats(th)
#---②

for i in range(cnt):
    img2[labels==i] = [int(j) for j in np.random.randint(0,255, 3)]

cv2.imshow('origin', img)
cv2.imshow('labeled', img2)
```

- Example <Result>



❖ Flood Fill

- Filling same color to consecutive area
- retval, img, mask, rect = cv2.floodFill(img, mask, seed, newVal[, loDiff, upDiff, flags])
  - img : input image, 1 or 3 channels
  - mask : Larger by 2x2 pixel than input image, If it meets area that is not 0, stop filling
  - seed : Coordination to start filling
  - newVal : Color value used to fill
  - loDiff, upDiff : Min and max value used to decide to fill
  - flags : Flag for filling type,
    - 4 or 8 distance filling
    - cv2.FLODDFILL_MASK_ONLY : Fill only mask not img
      - Need to include 8-16 bits for the value to be used in fillings
      - cv2.FLOODFILL_FIXED_RANGE: Compare it with seed pixel not neighboring pixel
  - retval: # of pixels filled
  - rect : Rectangle surrounding areas done filling
- Example

- $src(x', y') - \text{loDiff} \leq src(x, y) \leq src(x', y') + \text{upDiff}$
  - $src(x, y)$: 현재 픽셀
  - $src(x', y')$: 이웃 픽셀

```
import cv2
import numpy as np
img = cv2.imread('../img/taekwonv1.jpg')
rows, cols = img.shape[:2]
mask = np.zeros((rows+2, cols+2), np.uint8)
newVal = (255,255,255)
loDiff, upDiff = (10,10,10), (10,10,10)
def onMouse(event, x, y, flags, param):
    global mask, img
    if event == cv2.EVENT_LBUTTONDOWN:
        seed = (x,y)
        retval = cv2.floodFill(img,mask,seed,newVal,loDiff, upDiff)
        cv2.imshow('img', img)
cv2.imshow('img', img)
cv2.setMouseCallback('img', onMouse)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Flood Fill

- Example <Result>

❖ Watershed

- Divide subject by considering edge balanced area as valley
- Designate maker to divide by user (or algorithm)
- Fill water to each different marker area, stop when they meet.
- Marker
  - Apply ID (1~) for each different object, Mark subject to distinguish
  - 0 : Unknown
  - i.e.) Front Image=1, Background=2
  - I.e.)Orange=1, lime=2, Background=3
- markers = cv2.watershed(img, markers)
  - img : input image
  - markers
    - Marker boarder set as -1, same size as input

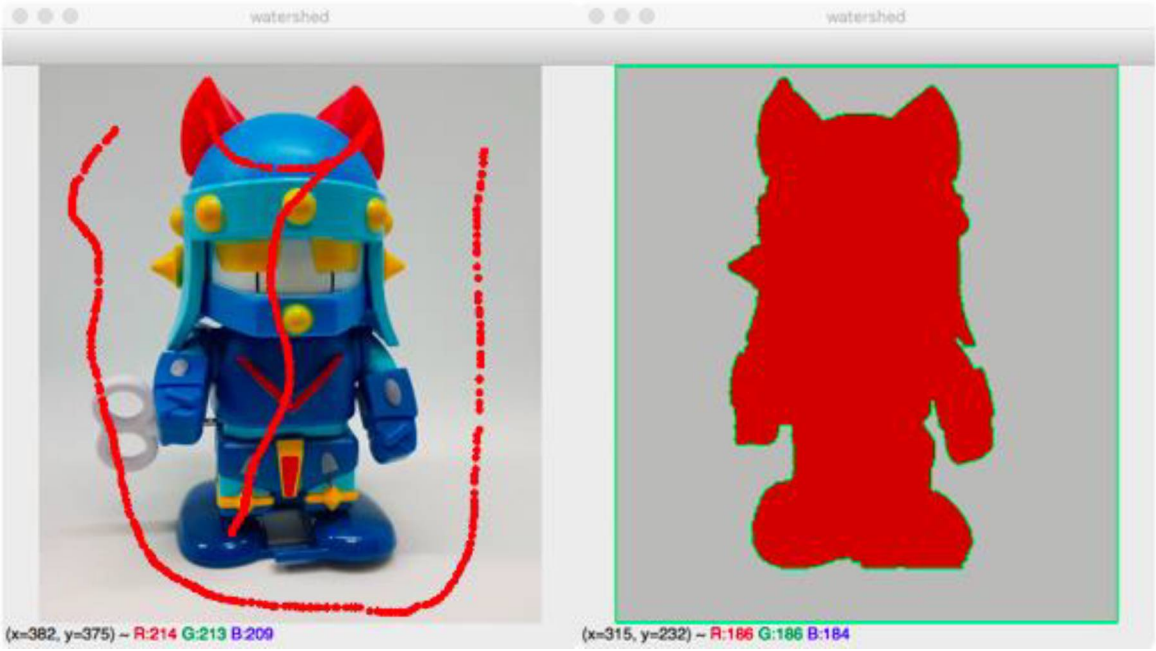
Watershed

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/taekwonv1.jpg')
rows, cols = img.shape[:2]
img_draw = img.copy()
marker = np.zeros((rows, cols), np.int32)
markerId = 1
colors = []
isDragging = False
def onMouse(event, x, y, flags, param):
    global img_draw, marker, markerId, isDragging
    if event == cv2.EVENT_LBUTTONDOWN:  # Drag down left mouse
button, start dragging
        isDragging = True
        colors.append((markerId, img[y,x]))
elif event == cv2.EVENT_MOUSEMOVE:  # Mouse moving
        if isDragging:                  # Dragging
            marker[y,x] = markerId
            cv2.circle(img_draw, (x,y), 3, (0,0,255), -1)
            cv2.imshow('watershed', img_draw)
    elif event == cv2.EVENT_LBUTTONUP:  # Up left mouse button
        if isDragging:
            isDragging = False          # Stop dragging
            markerId +=1
    elif event == cv2.EVENT_RBUTTONDOWN: # Click right mouse button
            cv2.watershed(img, marker)
            img_draw[marker == -1] = (0,255,0)
            for mid, color in colors: # Repeat as the number of marker ID
                img_draw[marker==mid] = color
            cv2.imshow('watershed', img_draw) # Print marked result
cv2.imshow('watershed', img)
cv2.setMouseCallback('watershed', onMouse)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Watershed

- Example <Result>



❖ Grabcut

- Expand based on Graph Cut algorithm
- Assume color distribution for subject that will separate front image
- At the connected area with same label, separate front image and background
- mask, bgdModel, fgdModel = cv2.grabCut(img, mask, rect, bgdModel, fgdModel, iterCount[, mode])
- img : input image
- mask : 1 channel array, same size as input image, save value that separtes background and front page
  - cv2.GC_BGD : Absolutely Background (0)
  - cv2.GC_FGD : Absolutely Front Image(1)
  - cv2.GC_PR_BGD : Maybe background(2)
  - cv2.GC_PR_FGD : Maybe front image (3)
- rect : Rectangular coordination for area that front image is assumed, Tuple (x1, y1, x2, y2)

- bgdModel, fgdModel: Temporary array buffer to be used within function, do not edit when re used
- iterCount : # of repetition
- mode :
  - cv2.GC_INIT_WITH_RECT : Grabcut based on set coordination at rect
  - cv2.GC_INIT_WITH_MASK : Grabcut based on set coordination at mask
  - cv2.GC_EVAL : Re-try

❖ Grabcut

- Example <1/2>

```
import cv2
import numpy as np
img = cv2.imread('../img/taekwonv1.jpg')
img_draw = img.copy()
mask = np.zeros(img.shape[:2], dtype=np.uint8)
rect = [0,0,0,0]
mode = cv2.GC_EVAL
bgdmodel = np.zeros((1,65),np.float64)
fgdmodel = np.zeros((1,65),np.float64)
def onMouse(event, x, y, flags, param):
    global mouse_mode, rect, mask, mode
    if event == cv2.EVENT_LBUTTONDOWN :
        if flags <= 1: # If no key input
            mode = cv2.GC_INIT_WITH_RECT
            rect[:2] = x, y
elif event == cv2.EVENT_MOUSEMOVE and flags &
cv2.EVENT_FLAG_LBUTTON :
        if mode == cv2.GC_INIT_WITH_RECT:
            img_temp = img.copy()
cv2.rectangle(img_temp, (rect[0], rect[1]), (x, y), (0,255,0), 2)
            cv2.imshow('img', img_temp)
        elif flags > 1:
                 mode = cv2.GC_INIT_WITH_MASK
     if flags & cv2.EVENT_FLAG_CTRLKEY :
                cv2.circle(img_draw,(x,y),3, (255,255,255),-1)
                cv2.circle(mask,(x,y),3, cv2.GC_FGD,-1)
```

❖ Grabcut

- Example <2/2>

```
            if flags & cv2.EVENT_FLAG_SHIFTKEY :
                 cv2.circle(img_draw,(x,y),3, (0,0,0),-1)
                 cv2.circle(mask,(x,y),3, cv2.GC_BGD,-1)
             cv2.imshow('img', img_draw)
elif event == cv2.EVENT_LBUTTONUP:
        if mode == cv2.GC_INIT_WITH_RECT :
             rect[2:] =x, y
 cv2.rectangle(img_draw, (rect[0], rect[1]), (x, y), (255,0,0), 2)
             cv2.imshow('img', img_draw)
cv2.grabCut(img, mask, tuple(rect), bgdmodel, fgdmodel, 1, mode)
        img2 = img.copy()
        img2[(mask==cv2.GC_BGD) | (mask==cv2.GC_PR_BGD)] = 0
        cv2.imshow('grabcut', img2)
        mode = cv2.GC_EVAL
cv2.imshow('img', img)
cv2.setMouseCallback('img', onMouse)
while True:
    if cv2.waitKey(0) & 0xFF == 27 : # esc
        break
cv2.destroyAllWindows()
```

- Example <Result>

❖ MeanShift Filter

- It is kernel window with certain boarder size kernel, move center to average pixel value
- Find connecting area by binding starting point and stop point as one
- Change connecting area color value as most frequent color value
- dst = cv.pyrMeanShiftFiltering(src, sp, sr[, dst, maxLevel, termcrit])
  - src : input image
  - sp : Space window radius
  - sr : Color window radius
  - maxLevel : Max level of image pyramid
  - termcrit : Standard for mean move cut
- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/taekwonv1.jpg')
def onChange(x):
    sp = cv2.getTrackbarPos('sp', 'img')
    sr = cv2.getTrackbarPos('sr', 'img')
    lv = cv2.getTrackbarPos('lv', 'img')
    mean = cv2.pyrMeanShiftFiltering(img, sp, sr, None, lv)
    cv2.imshow('img', np.hstack((img, mean)))

cv2.imshow('img', np.hstack((img, img)))
cv2.createTrackbar('sp', 'img', 0,100, onChange)
cv2.createTrackbar('sr', 'img', 0,100, onChange)
cv2.createTrackbar('lv', 'img', 0,5, onChange)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ MeanShift Filter

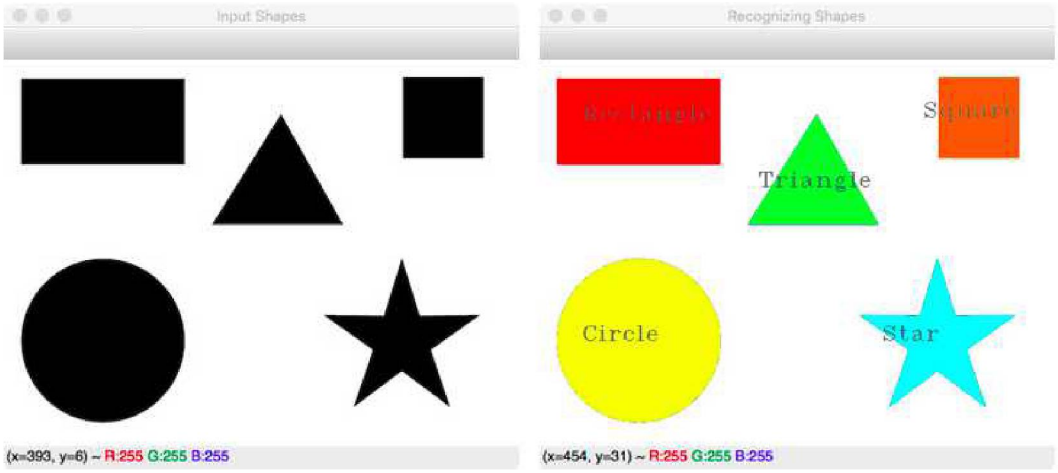
- Example <Result>

Image Segmentation

1. Contour
2. Hough Transform
3. Connected Component
4. **Workshop**

❖ Recognizing Shapes

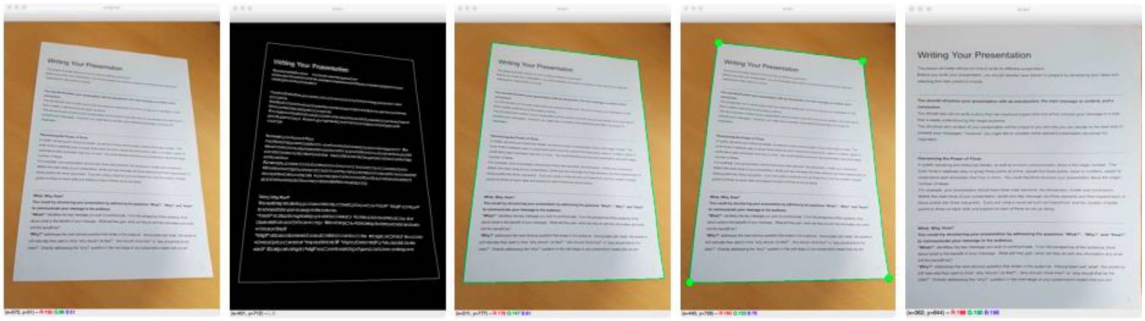
- Make a program that gets the name of figures in a image with five figures.
- Result example



- Hint
  - Find contour and count the number of vertex by simplifying them as approximate value

❖ Document Scanner

- Draft a program that gives scanning effect without mouse input
- Result Example



- Hint
  - Get 4 vertex
    - Detect boarder with canny edge
      - Find biggest contour with findCotour() and simplify to approxPolyDP()

❖ Coin Counter

- Make a program that separates coins and counts.
- Result Example

- Hint
  - pyMeanShiftFiltering() Smudge coin surface
  - Otsu Threshold
  - distanceTransform() Center point of coin
  - Find local max value
  - floodfill() local max value to seed
  - Secure absolute front image of coin with distance transform
  - Secure absolute background of coin with distance transform
  - Absolute background – Absolute front image = Unknown area
  - connectComponents() labeling
  - watershed()
  - Separate coin area with findCountour(), boundingRect()

Matching & Tracking

1. **Matching with a similar image**
2. Feature and Key Point
3. Descriptor Extractor
4. Feature Matching
5. Tracking
6. Workshop

❖ Average Hash Matching

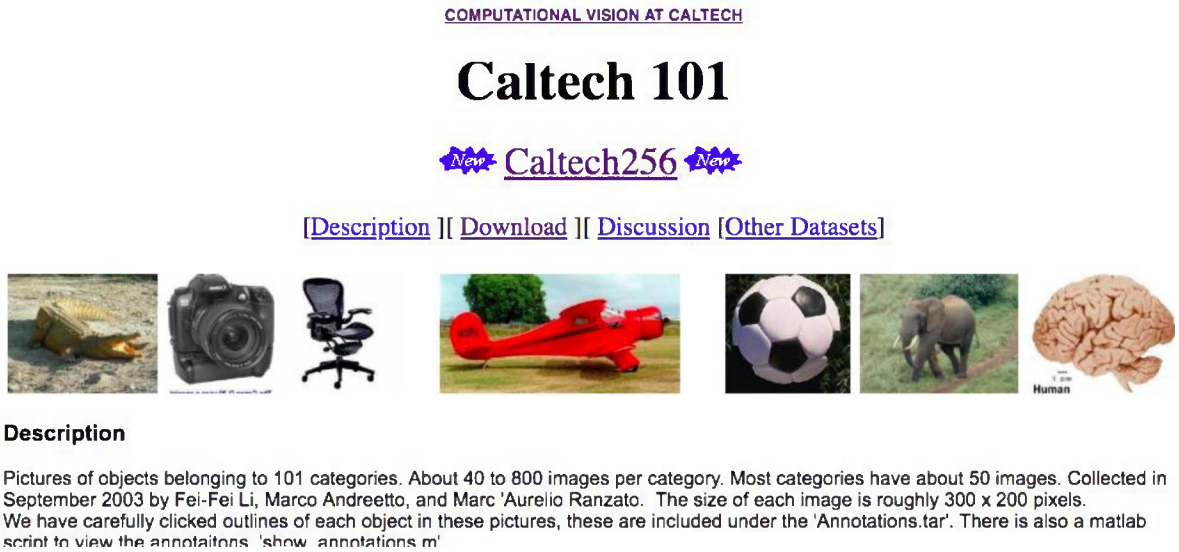
- Fit for understanding similar pictures
- Average Hash
  - Transform image into one number by calculating average value
    - Reduce image into a certain size regardless of the ration of width and height
    - Calculate all pixel average and compare
    - px > Average? 1 : 0
    - Change it to 1 line, and transform that into one binary number
    - Able to transform into hexadecimal
- Analogous Distance
  - Measure distance between the two value, if the number is small, similarity is high
  - Euclidian distance
    - Find the differences between the number as distance
    - 예) 5:3 = 2, 5:8=3
  - Hamming distance
    - Showing the number of differences in digits in between two numbers
      - 12345:12354=2, 12345:92345=1
- Example

```
import cv2

img = cv2.imread('../img/pistol.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray = cv2.resize(gray, (8,8))
avg = gray.mean()
bin = 1 * (gray > avg)
print(bin)
dhash = []
for row in bin.tolist():
    s = ''.join([str(i) for i in row])
    dhash.append('%02x'%(int(s,2)))
dhash = ''.join(dhash)
print(dhash)
cv2.imshow('pistol', img)
cv2.waitKey(0)
```

❖ Average Hash Matching


```
[[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1]
 [1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1]
 [1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1]
 [1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1]
 [1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1]
 [1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
 [1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
 [1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
 [1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
 [1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1]
 [1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1]]
ffff8000800080008000813fc1ffc1ffc07fc3ffc7ffc7ff87ff87ff87ffc7ff
```

- 101 Image Dataset
  - Caltech USA
  - Image set with 101 objects
  - http://www.vision.caltech.edu/Image_Datasets/Caltech101/#Download



❖ Average Hash Matching

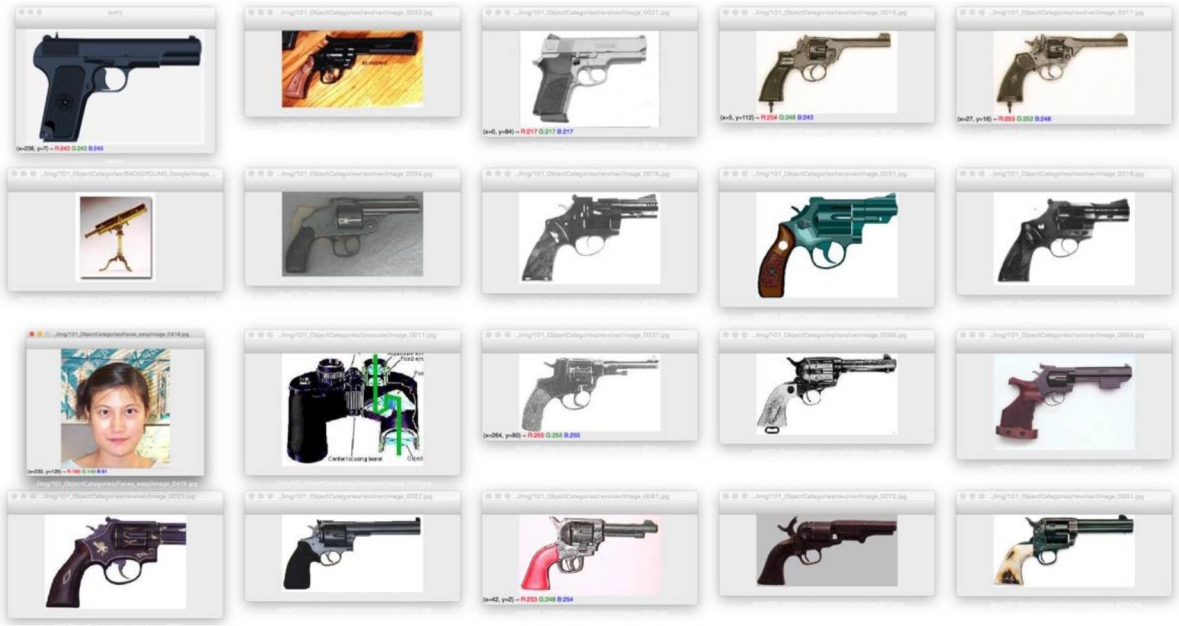
- Finding Gun Image

```
import cv2
import numpy as np
import glob

img = cv2.imread('../img/pistol.jpg')
cv2.imshow('query', img)
search_dir = '../img/101_ObjectCategories'

def img2hash(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray = cv2.resize(gray, (16, 16))
    avg = gray.mean()
    bi = 1 * (gray > avg)
    return bi
def hamming_distance(a, b):
    a = a.reshape(1,-1)
    b = b.reshape(1,-1)
    distance = (a !=b).sum()
    return distance

query_hash = img2hash(img)

img_path = glob.glob(search_dir+'/**/*.jpg')
for path in img_path:
    img = cv2.imread(path)
    cv2.imshow('searching...', img)
    cv2.waitKey(5)
    a_hash = img2hash(img)
    dst = hamming_distance(query_hash, a_hash)
    if dst/256 < 0.25:
        print(path, dst/256)
        cv2.imshow(path, img)
cv2.destroyWindow('searching...')
cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ Average Hash Matching

- Finding Gun Image <Result 1>

```
../img/101_ObjectCategories/revolver/image_0033.jpg 0.2421875
../img/101_ObjectCategories/revolver/image_0019.jpg 0.23828125
../img/101_ObjectCategories/revolver/image_0031.jpg 0.21875
../img/101_ObjectCategories/revolver/image_0018.jpg 0.1953125
../img/101_ObjectCategories/revolver/image_0034.jpg 0.23046875
../img/101_ObjectCategories/revolver/image_0021.jpg 0.171875
../img/101_ObjectCategories/revolver/image_0037.jpg 0.2421875
../img/101_ObjectCategories/revolver/image_0023.jpg 0.21875
../img/101_ObjectCategories/revolver/image_0022.jpg 0.21484375
../img/101_ObjectCategories/revolver/image_0081.jpg 0.23046875
../img/101_ObjectCategories/revolver/image_0068.jpg 0.24609375
../img/101_ObjectCategories/revolver/image_0064.jpg 0.18359375
../img/101_ObjectCategories/revolver/image_0072.jpg 0.203125
../img/101_ObjectCategories/revolver/image_0001.jpg 0.2421875
../img/101_ObjectCategories/revolver/image_0015.jpg 0.24609375
../img/101_ObjectCategories/revolver/image_0017.jpg 0.23828125
../img/101_ObjectCategories/binocular/image_0011.jpg 0.23828125
../img/101_ObjectCategories/BACKGROUND_Google/image_0398.jpg 0.234375
../img/101_ObjectCategories/Faces_easy/image_0419.jpg 0.2421875
```



❖ Template Matching

- The most basic methods of Object Detecting
- Search image to find from all images
- cv2. matchTemplate(image, templ, method) : result
  - image : Input Image
  - templ : Object of Detection
  - method : Matching Algorithm
    - cv2.TM_CCOEFF, cv2.TM_CCOEFF_NORMED
    - cv2.TM_CCORR, cv2.TM_CCORR_NORMED
    - cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED
  - Turnaround : Array result value of matching ,
    - TM_SQDIFF, For TM_SQDIFF_NORMED, min_loc
    - 니머지는 max_loc
- cv2.minMaxLoc(matched)
  - Turnaround : min_val, max_val, min_loc, max_loc

❖ Matching Method (I : Image, T : Template, R : Result)

- method=CV_TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

- method=CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- method=CV_TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

- method=CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- method=CV_TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

- method=CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

Template Matching

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/figures.jpg')
template = cv2.imread('../img/taekwonv1.jpg')
th, tw = template.shape[:2]
cv2.imshow('template', template)

methods = ['cv2.TM_CCOEFF_NORMED',
'cv2.TM_CCORR_NORMED',  'cv2.TM_SQDIFF_NORMED']
for i, method_name in enumerate(methods):
    img_draw = img.copy()
    method = eval(method_name)
    res = cv2.matchTemplate(img, template, method)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
    print(method_name, min_val, max_val, min_loc, max_loc)
if method in [cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED]:
        top_left = min_loc
        match_val = min_val
    else:
        top_left = max_loc
        match_val = max_val
    bottom_right = (top_left[0] + tw, top_left[1] + th)
    cv2.rectangle(img_draw, top_left, bottom_right, (0,0,255),2)
    cv2.putText(img_draw, str(match_val), top_left,
cv2.FONT_HERSHEY_PLAIN, 2,(0,255,0), 1, cv2.LINE_AA)
    cv2.imshow(method_name, img_draw)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
cv2.TM_CCOEFF_NORMED -0.17781560122966766 0.5126562714576721 (42, 0) (208, 43)
cv2.TM_CCORR_NORMED 0.8271383047103882 0.9236084818840027 (85, 6) (208, 43)
cv2.TM_SQDIFF_NORMED 0.1706070452928543 0.36892083287239075 (208, 43) (86, 7)
```

❖ Template Matching

- Example <Result>

❖ Multi Object Matching

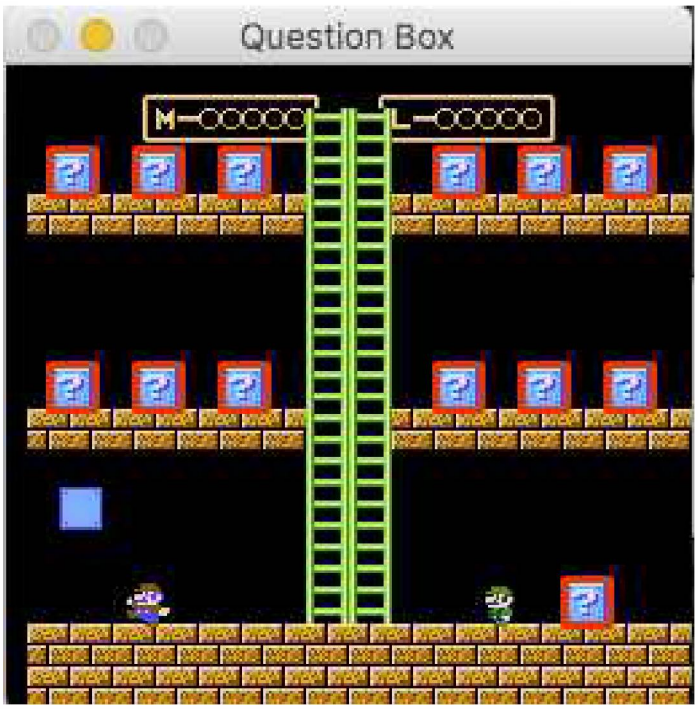
- When you want to find same template in one image
- cv2.matchTemplate() turns around only one result
- cv2.minMaxLoc(): Problem of turning around max or minimum one
- Find smallest differences in max or min value .
  - threshold = max * (1-0.01)
  - np.where(res >= threshold)

❖ Multi Object Matching

- Example

```
img = cv2.imread('../img/mario.png')
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

template = cv2.imread('../img/mario_template.png',0)
w, h = template.shape[::-1]

res = cv2.matchTemplate(imgray,template,cv2.TM_CCOEFF_NORMED)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
threshold = max_val * (1-0.01)
print("max:%f, threshold : %f" %(max_val, threshold))

loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)

cv2.imshow('Question Box',img)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Example <Result>



Matching & Tracking

1. Matching with a similar image
2. **Feature and Key Point**
3. Descriptor Extractor
4. Feature Matching
5. Tracking
6. Workshop

❖ Corner Feature Detector

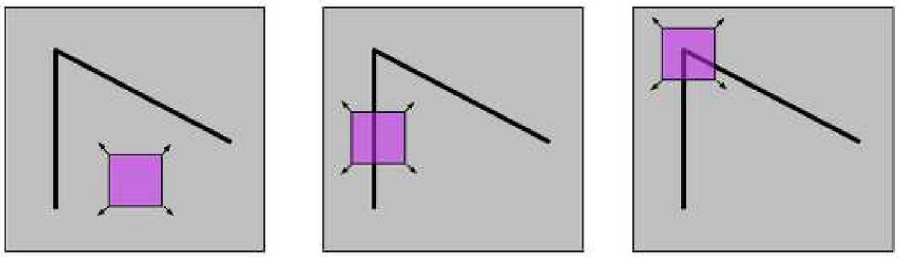
- Limitation of Template Matching
  - Rotate, Scale, Brightness, Contrast, Hue, Unable Affine transform
- Image's Corner :
  - Easy to find characters from image
  - Unable to detect even human other than Corner
  - Detect Image Feature

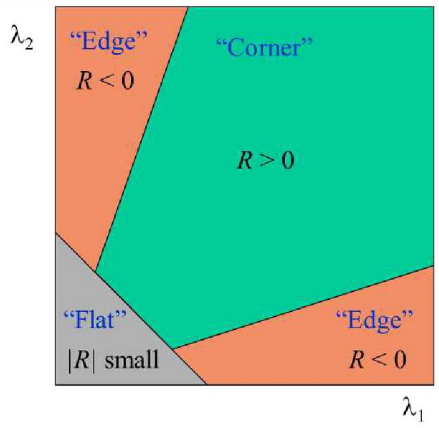
❖ Harris Corner Detection

- Chris harris, Mike Stephens's paper, 1988
- Representative methods of finding Corner Keypoint
- When move kernel, all direction changes
- cv2.cornerHarris(img, blockSize, ksize, k)
  - img : Input image, float32
  - blocksize : Size of neighboring pixel
  - ksize : Kernel size to use in Sobel
  - k : K Parameter (Empirical Integer, 0.04~0.06)
  - Turnaround : Same size as input image
    - Changed value

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$



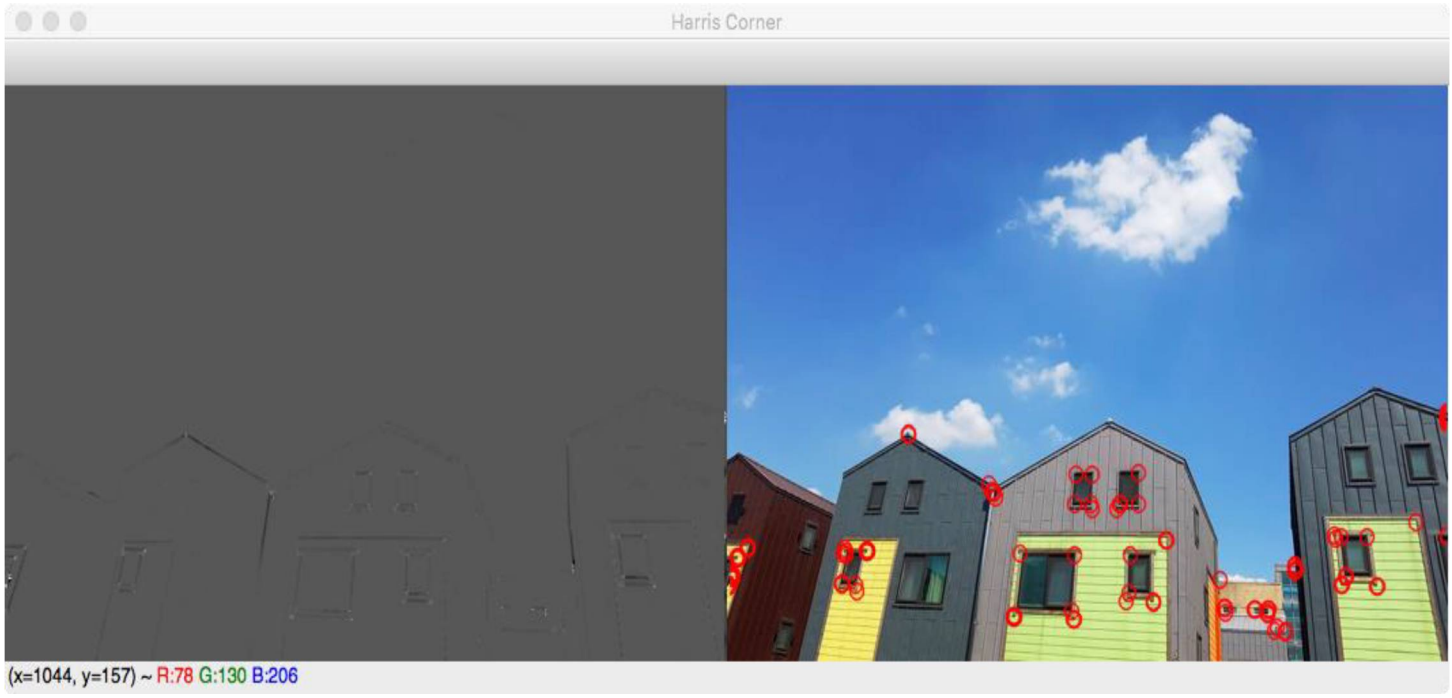


❖ Harris Corner Detection

```
import cv2
import numpy as np
img = cv2.imread('../img/house.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
corner = cv2.cornerHarris(gray, 2, 3, 0.04)
coord = np.where(corner > 0.1* corner.max())
coord = np.stack((coord[1], coord[0]), axis=-1)
for x, y in coord:
    cv2.circle(img, (x,y), 5, (0,0,255), 1, cv2.LINE_AA)
corner_norm = cv2.normalize(corner, None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)
corner_norm = cv2.cvtColor(corner_norm, cv2.COLOR_GRAY2BGR)
merged = np.hstack((corner_norm, img))
cv2.imshow('Harris Corner', merged)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Harris Corner Detection

- Example <Result>



❖ Shi-Tomasi

- J.Shi. C. Tomasi's Paper, 1994
- Good for finding features to detect using Optical Flow
- Consider only min value λ1, λ2 in Harris method
- Apply threshold value in Result value, If it is bigger than that, recognize it as corner
- **R = min(λ1, λ2) > T**
- cv2.goodFeaturesToTrack(img, maxCorner, qualityLev, minDistance)
  - img : input image
  - maxCorner : # of coners wish to detect, From the strongest one
  - qualityLev: Threshold to consider it as corners
  - mask : Mask to exclude in detection
  - blockSize = 3 : Size of area surrounding corners
    - useHarrisDetectior=False: Select method to detect corner

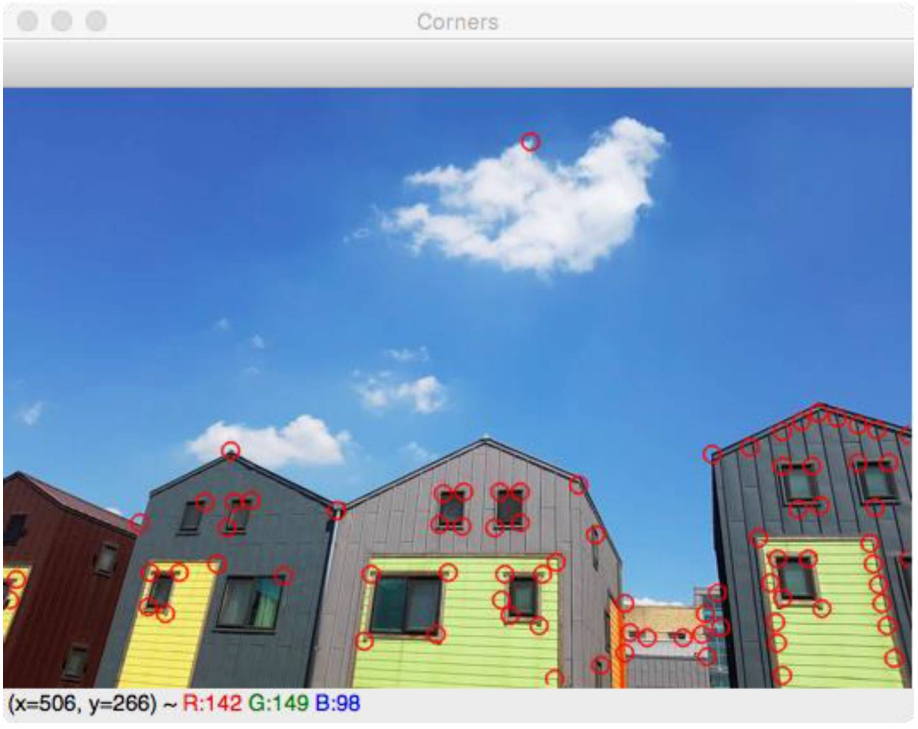
- True = Harris corner detection. False = Siwa and Tomasian detection
- K : k Coefficient to use in Harris Corner detection
- corners : Corner detection coordination result, Array size of N x 1 x 2, it is in real number value, so need to change them into integer

❖ Shi-Tomasi

- Example

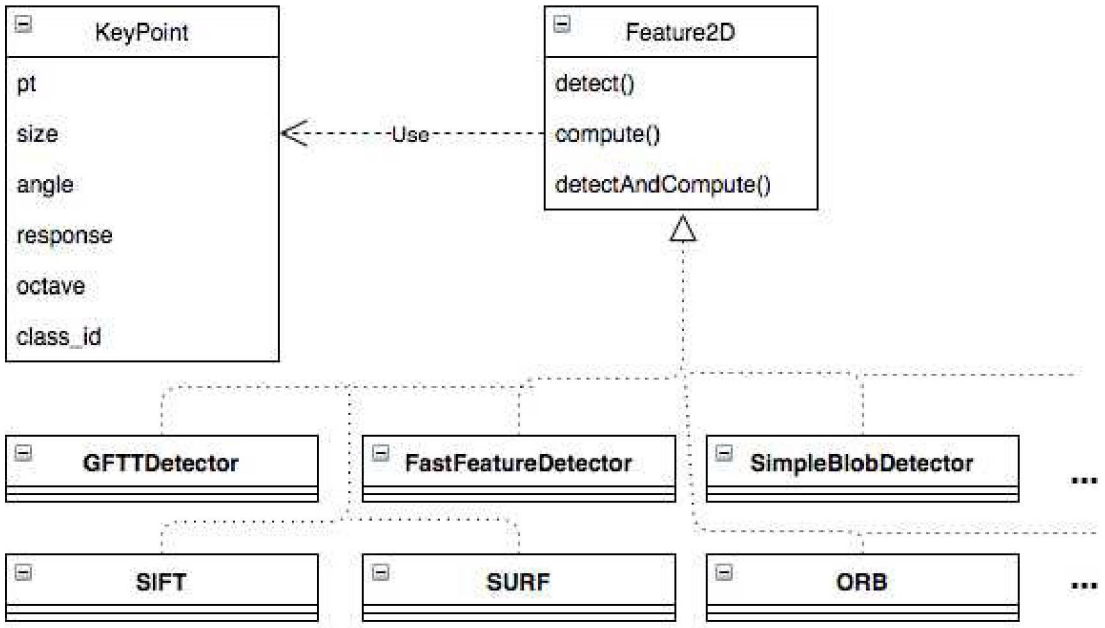
```
import cv2
import numpy as np
img = cv2.imread('../img/house.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
corners = cv2.goodFeaturesToTrack(gray, 80, 0.01, 10)
corners = np.int32(corners)
for corner in corners:
    x, y = corner[0]
    cv2.circle(img, (x, y), 5, (0,0,255), 1, cv2.LINE_AA)
cv2.imshow('Corners', img)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Example <Result>



❖ Key Point and Feature Detector

- Offer common interface regardless of feature detecting algorithm
- Feature detector: Operate after inheriting cv2.Feature2D, 12 types (as of 3.4.1 ver.)
- Feature : cv2.KeyPoint



❖ cv2.Feature2D

- keypoints = detector.detect(img [, mask]) : Function to detect key point
  - img : Input image, binary scale
  - mask : Mask to exclude in detection
  - keypoints : Feature detection Result, List of KeyPoint
- KeyPoint : Object with feature info
  - pt : Coordination of key point (x, y), Need to change into integer, iti sin float type
  - size : Radius of meaningful key point neighbors
  - angle : Orientation of feature point (Clockwise, -1=Meaningless)
  - response : Strength of feature point response (Depends on the detector)
  - octave : Layers of pyramid in detected images
  - class_id : Object ID with key point

❖ Show Key Point

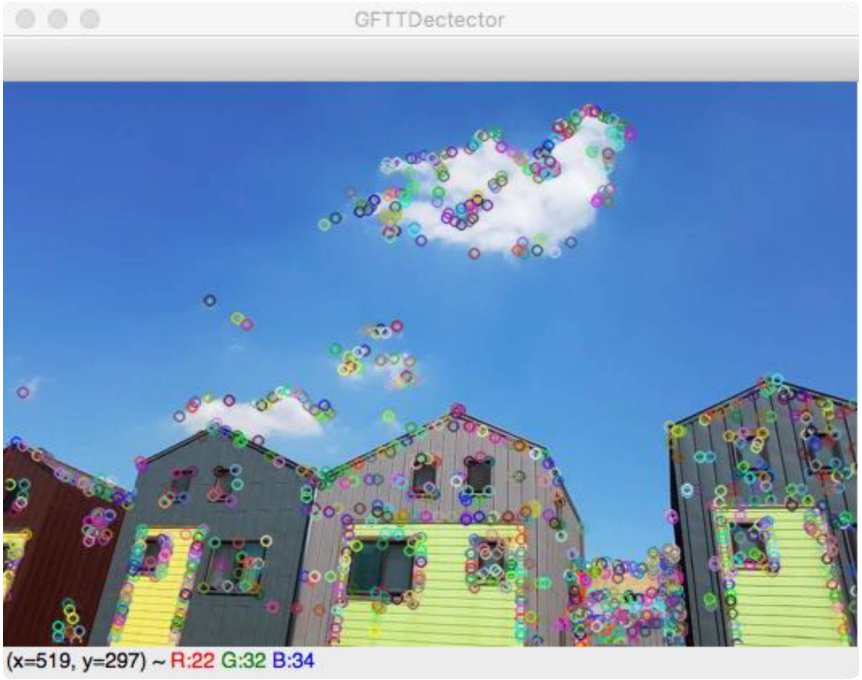
- outImg = cv2.drawKeypoints(img, keypoints, outImg[, color[, flags]])img : Input Image
- keypoints : Key Point List to show
- outImg : Result image with key point drawn
- color : Color to show, (Basic Value : Random)
- flags : Flag to show marking methods
  - cv2.DRAW_MATCHES_FLAGS_DEFAULT : Draw only circle in the center of coordination
  - cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS : Draw circle considering size and angles

❖ GFTTDetector

- cv2.goodFeaturesToTrack() Detector show in function
  - detector = cv2.GFTTDetector_create([, maxCorners[, qualityLevel, minDistance, blockSize, useHarrisDetector, k])
  - All details in parameters are same as cv2.goodFeaturesToTrack()
  - There is no value other than PT feature coordination in KeyPoint Object

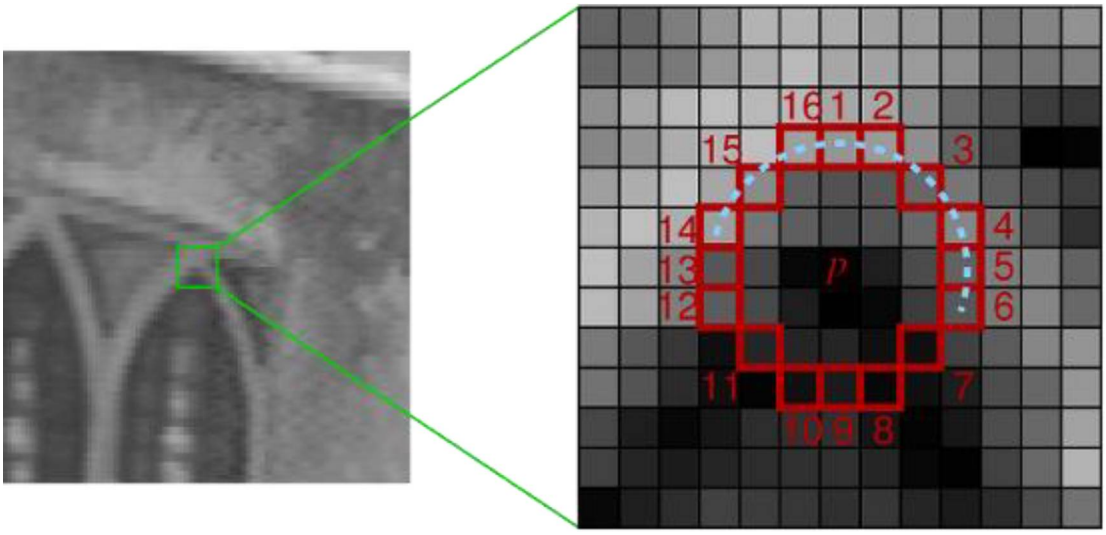
```
import cv2
import numpy as np

img = cv2.imread("../img/house.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

gftt = cv2.GFTTDetector_create()

keypoints = gftt.detect(gray, None)

img_draw = cv2.drawKeypoints(img, keypoints, None)


cv2.imshow('GFTTDectector', img_draw)
cv2.waitKey(0)
cv2.destrolyAllWindows()
```

❖ GFTTDetector



❖ FAST(Feature from Accelerated Segment Test)

- Edward Rosten, Tom Drummon, Cambrige Univ, 2006
- Improve speed in real time
  - Check 16 pixels that passes a circle with radius 3 as point P as center
  - Recognize them as corner if it is brighter than certain value (t) than P
  - n : FAST-9, 10, 11, 12, 13, 14, 15, 16
- Detects only key point without descriptor

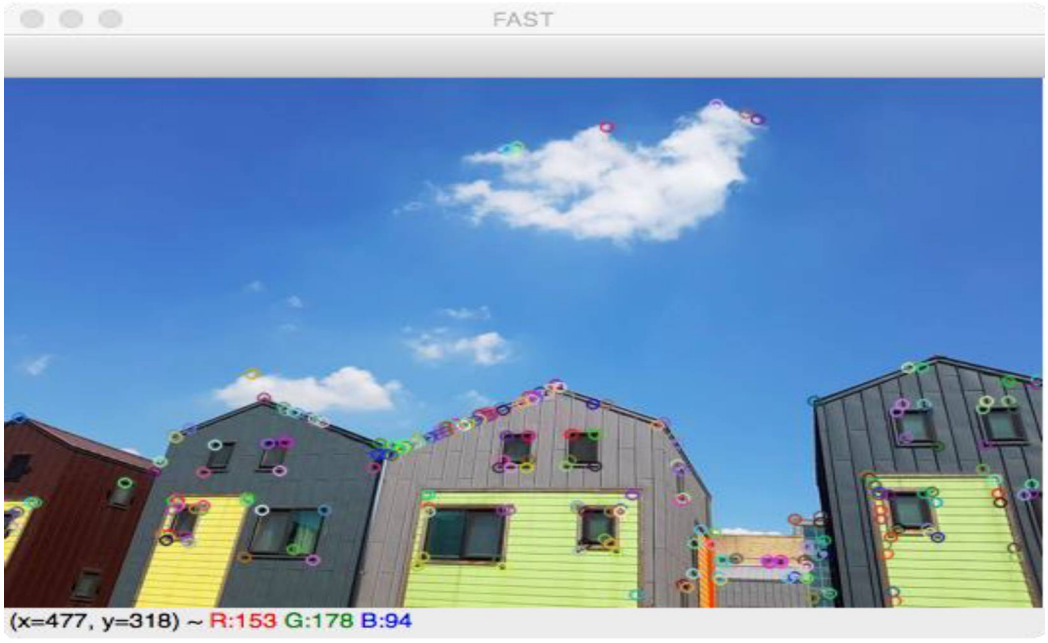


❖ FAST(Feature from Accelerated Segment Test)

- detector = cv.FastFeatureDetector_create([threshold[, nonmaxSuppression, type])
  - threshold=10 : Corner Detecting Threshold
  - nonmaxSuppression = True : Control corner not max value
  - type : Edge detecting pattern
    - cv2.FastFeatureDetector_TYPE_9_16: 9 of 16 consecutive values (basic value)
    - cv2.FastFeatureDetector_TYPE_7_12: 7 of 12 consecutive
    - cv2.FastFeatureDetector_TYPE_5_8: 5 of 8 consecutive
- Example

```
import cv2
import numpy as np
img = cv2.imread('../img/house.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
fast = cv2.FastFeatureDetector_create(50)
keypoints = fast.detect(gray, None)
img = cv2.drawKeypoints(img, keypoints, None)
cv2.imshow('FAST', img)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Example <result>

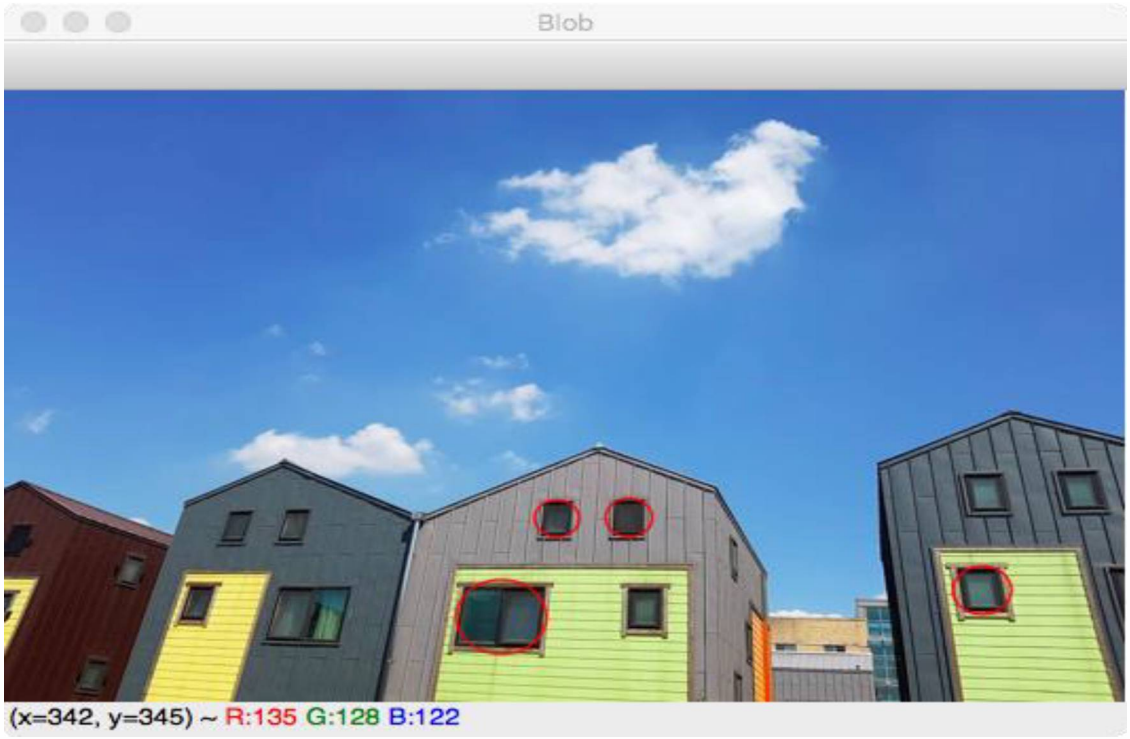


❖ SimpleBlobDetctor

- BLOB(Binary Large Object)
  - Connected pixel group of binary scale image
  - Recognize small objects as noise
  - Only interested in objects bigger than
- detector = cv2.SimpleBlobDetector_create([parameters]) : Blob detector generator
  - paramters : Object for Blob detecting filter parameter
- Example

```
mport cv2
import numpy as np
img = cv2.imread("../img/house.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
detector = cv2.SimpleBlobDetector_create()
keypoints = detector.detect(gray)
img = cv2.drawKeypoints(img, keypoints, None,
(0,0,255),flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow("Blob", img)
cv2.waitKey(0)
```

- Example <result>



❖ SimpleBlobDetctor

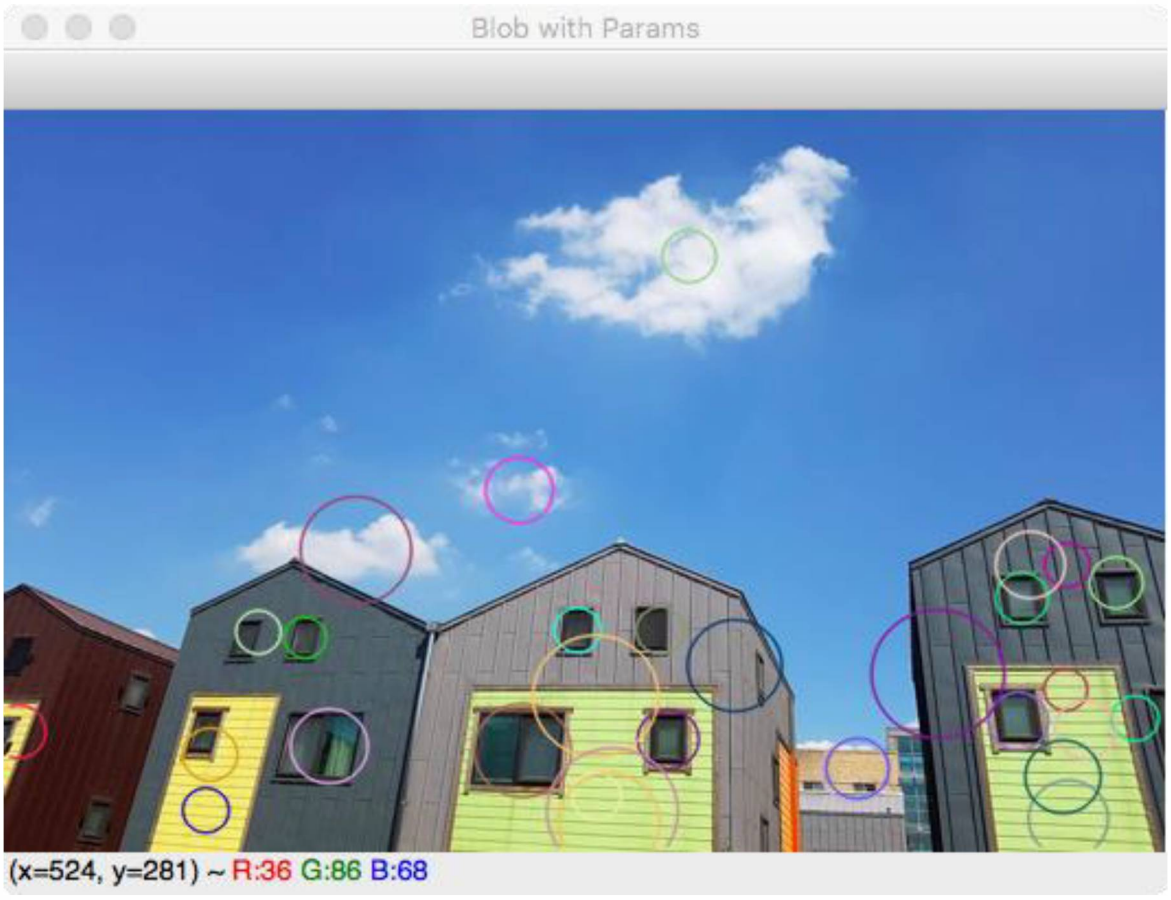
- cv2.SimpleBlobDetector_Params()
  - minThreshold, maxThreshold, thresholdStep : Boundary value to generate Blob
    - Increase by thresholdStep until it goes over max threshold from minThreshold
  - minRepeatability: # of consecutive boundary value to participate in Blob
  - minDistBetweenBlobs: Distance recognizing two blobs as one blobs
  - filterByArea: Area filter option
  - minArea, maxArea: Detect it as blob for min~max area
  - filterByCircularity: Circle ration filter option
  - minCircularity, maxCircularity : Detect only circle ration within min ~ max as blob
  - filterByColor: Filter option using brightness
  - blobColor : 0=detect black blob, 255= Detect white blob
  - filterByConvexity: Block ration filter option
  - minConvexity, maxConvexity : Detect it as blob for min~max area
  - filterByInertia : Consistency ration filter option
  - minInertiaRatio, maxInertiaRatio: Detect it as blob for min~max area consistency ration
- Example <1/2>

```
import cv2
import numpy as np

img = cv2.imread("../img/house.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

params = cv2.SimpleBlobDetector_Params()

params.minThreshold = 10
params.maxThreshold = 240

params.thresholdStep = 5

params.filterByArea = True
params.minArea = 200
```

- Example <2/2>

```
params.filterByColor = False
params.filterByConvexity = False
params.filterByInertia = False
params.filterByCircularity = False

detector = cv2.SimpleBlobDetector_create(params)
keypoints = detector.detect(gray)
img_draw = cv2.drawKeypoints(img, keypoints, None, None,
cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

cv2.imshow("Blob with Params", img_draw)
cv2.waitKey(0)
```

- Example <result>

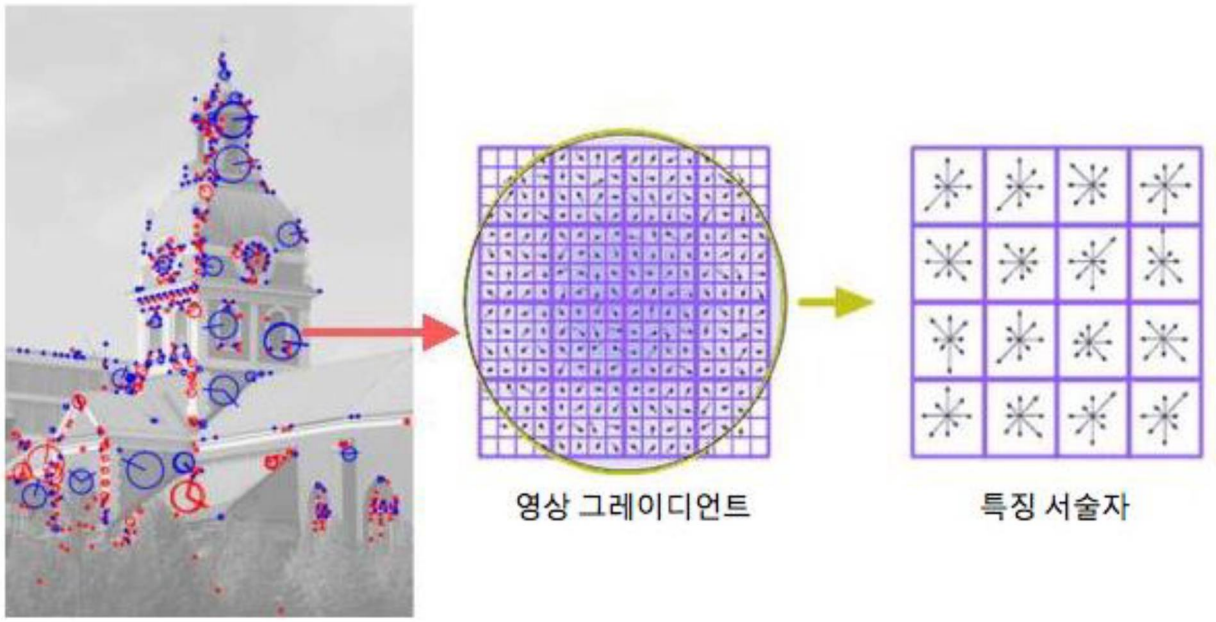


Matching & Tracking

1. Matching with a similar image
2. Feature and Key Point
3. **Descriptor Extractor**
4. Feature Matching
5. Tracking
6. Workshop

❖ Feature Descriptor

- Demand of feature narrator regardless of rotation, size and orientation
- Divide pixels surrounding Key Point into certain size blocks
- Calculated gradient histogram of pixel in block
- Generally, it shows 8 direction slope
  - 4 x 4 x 8 for each key point = Composed of 128 values

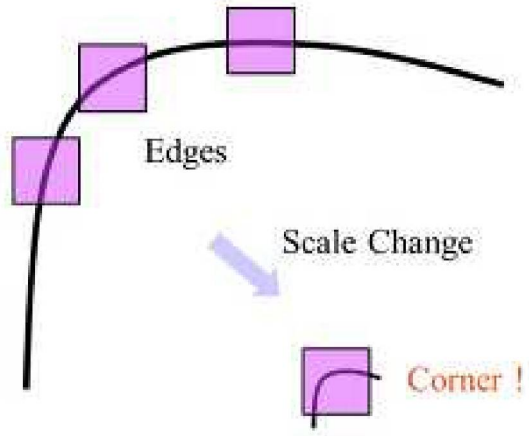


❖ Descriptor Extractor Interface

- cv2.Feature2D Abstract Class
- keypoints, descriptors = detector.compute(image, keypoints[, descriptors]) : Calculate certain narrator while delivery key point to turnaround
- keypoints, descriptors = detector.detectAndCompute(image, mask[, descriptors, useProvidedKeypoints]): Run both 0 key point detection and feature narration calculation at once
  - image : Input Image
  - keypoints : Key point necessary for narrator calculation
  - descriptors : Calculated narrator
  - mask : Mask to use in key point detection
  - useProvidedKeypoints : When True, do not run key point detection (do not use)

❖ SIFT (Scale-Invariant Feature Transform)

- D.Lowe, Univ. of British Columbia, 2004
- Corner Matching Issue
  - Intensity: Brightness, move, rotation, affine strength
  - Scaling: Detect more corner if the size is reduced
- Solve scale issue using Image Pyramid
  - Find feature point at maximized Laplacian value
- It has patent right; only allowed to use at school and research
- From OpenCV3.0 package, it is not included in basic package
  - Included in Extra(contrib) package



- Generate objects
- detector = cv.xfeatures2d.SIFT_create([, nfeatures[, nOctaveLayers[, contrastThreshold[, edgeThreshold[, sigma]]]]])
  - nfeatures : Max feature number to detect
  - nOctaveLayers : # of layers to use image pyramid
  - contrastThreshold : Weak feature threshold value to filter
  - edgeThreshold : Weak edge threshold value to filter
  - sigma : Sigma value used in Gaussian filter to be used in image pyramid 0 layers

❖ SIFT (Scale-Invariant Feature Transform)

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/house.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


sift = cv2.xfeatures2d.SIFT_create()
keypoints, descriptor = sift.detectAndCompute(gray, None)
print('keypoint:',len(keypoints), 'descriptor:', descriptor.shape)
print(descriptor)

img_draw = cv2.drawKeypoints(img, keypoints, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

cv2.imshow('SIFT', img_draw)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Example <Result>

```
keypoint: 413 descriptor: (413, 128)
[[  1.   1.   1. ...   0.   0.   1.]
 [  8.  24.   0. ...   1.   0.   4.]
 [  0.   0.   0. ...   0.   0.   2.]
 ...
 [  1.   8.  71. ...  73. 127.   3.]
 [ 35.   2.   7. ...   0.   0.   9.]
 [ 36.  34.   3. ...   0.   0.   1.]]
```

❖ SURF (Speeded-Up Robust Feature)

- Improve SIFT performance
  - Changing size of filter instead of Image Pyramid
- It has patent right; only allowed to use at school and research
- From OpenCV3.0 package, it is not included in basic package, but included in Extra(contrib) package
- detector = cv.xfeatures2d.SURF_create([hessianThreshold, nOctaves, nOctaveLayers, extended, upright])
  - hessianThreshold : Boundary value to detect features (100)
  - nOctaves : # of image pyramid layer (3)
  - extended : Narrator generating flag(False), True : 128개, False : 64개
  - upright : Orientation calculating flag (False), True : Ignore orientation, False: Apply orientation
- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/house.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

surf = cv2.xfeatures2d.SURF_create(1000, 3, True, True)

keypoints, desc = surf.detectAndCompute(gray, None)
print(desc.shape, desc)

img_draw = cv2.drawKeypoints(img, keypoints, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('SURF', img_draw)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ SURF (Speeded-Up Robust Feature)

- Example <Result>

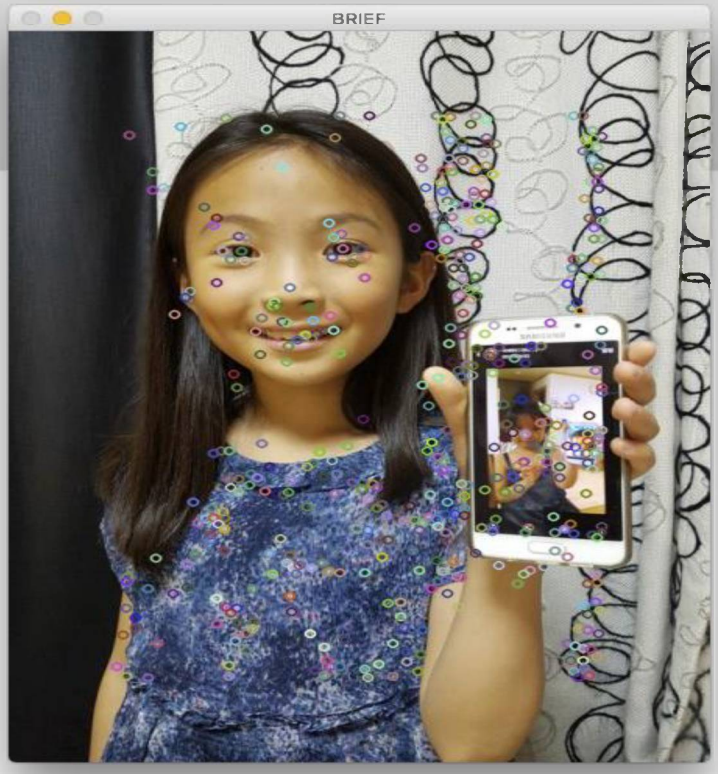
❖ BRIEF (Binary Robust Independent Elementary Features)

- Transform floating point data to binary string
- Quick processing, memory use decreases
- Only able Descriptor calculation
  - KeyPoint detection is depended on SIFT/SURF/Star
  - KeyPoint: Coordination of image for feature
  - Descriptor :
    - Feature information of detected local image
    - Gradient distribution histogram information
    - Mostly used for deciding matching level of two images

❖ BRIEF (Binary Robust Independent Elementary Features)

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/model.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

star = cv2.xfeatures2d.StarDetector_create()
brief = cv2.xfeatures2d.BriefDescriptorExtractor_create()

kp1 = star.detect(img, None)
kp2, desc = brief.compute(img, kp1)

print(desc.shape)
print(desc)
img = cv2.drawKeypoints(img, kp2, img)

cv2.imshow('BRIEF', img)
cv2.waitKey()
cv2.destroyAllWindows()
```



❖ ORB(Oriented FAST and Rotated BRIEF)

- Rublee, Rabaud, Konolige, Bradski, 2011
- Detect features with FAST
- Consider rotation and orientation at BRIEF
  - SIFT/SURF/BRIEF alternative
- dectector = cv.ORB_create([nfeatures, scaleFactor, nlevels, edgeThreshold, firstLevel, WTA_K, scoreType, patchSize, fastThreshold])
  - nfeatures=500 : Max # of features to detect
  - scaleFactor = 1.2 : Image pyramid ratio
  - nlevels = 8 : # of layers in image pyramid
  - edgeThreshold = 31 : Size of boarder excluding search, match with patchSize
  - firstLevel = 0 : Initial image pyramid layer level
  - WTA_K = 2 : # of generated temporary coordination
  - scoreType : Type to be used in key point detection
    - cv2.ORB_HARRIS_SCORE : Harris corner detection (Basic Value)
    - cv2.ORB_FAST_SCORE : FAST Corner detection
  - patchSize = 31 : Patch size of narrator
  - fastThreshold = 20 : Threshold value to be used in FAST
- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/house.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

orb = cv2.ORB_create()
keypoints, descriptor = orb.detectAndCompute(img, None)
img_draw = cv2.drawKeypoints(img, keypoints,
None,  flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

cv2.imshow('ORB', img_draw)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ ORB(Oriented FAST and Rotated BRIEF)

- Example <Result>

Matching & Tracking

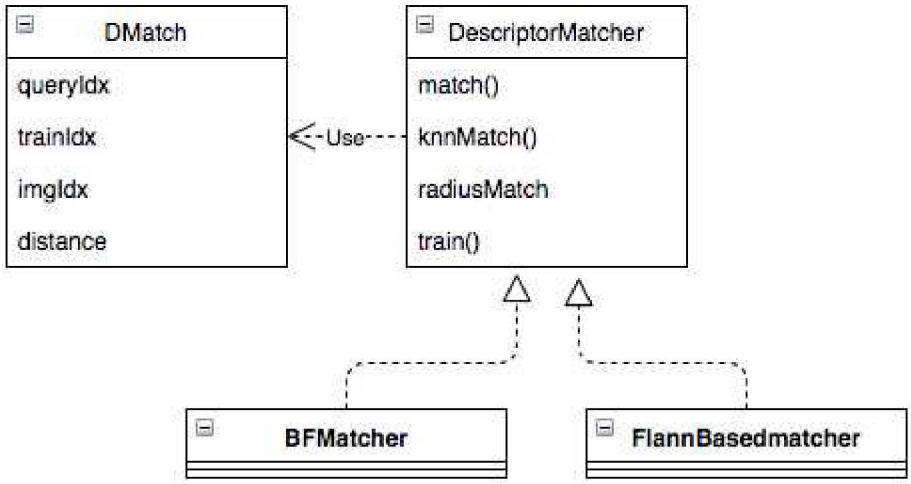
1. Matching with a similar image
2. Feature and Key Point
3. Descriptor Extractor
4. **Feature Matching**
5. Tracking
6. Workshop

❖ Matching

- Matching similarities by comparing key points and descriptor of different images
- Measure and detect similarity of images depends on # of matching point
- Image search, generate panorama image, use in object recognition

❖ Common Interface

- cv2.DescriptorMatcher : Inherit matching abstract classes
- cv2.DMatch : Object to save matching result



❖ Common Interface

- matcher = cv2.DescriptorMatcher_create(matcherType) : Matching Generator
  - matcherType: Algorithm of operation classes to generate, string
    - "BruteForce": BFMatcher using NORM_L2r
    - "BruteForce-L1": BFMatcher using NORM_L1
    - "BruteForce-Hamming": BFMatcher using NORM_HAMMING
    - "BruteForce-Hamming(2)": BFMatcher using NORM_HAMMING2
    - "FlannBased" : FlannBasedMatcher using NORM_L2
- matches = matcher.match(queryDescriptors, trainDescriptors[, mask]): 1 optimal matching
- queryDescriptors: Arrau feature narrator, narrator to be standard of matching
- trainDescriptors: Arrau feature narrator, narrator to be object of matching

- mask: Mask for matching
- matches: Matching Result, DMatch List of objects

- matches = matcher.knnMatch(queryDescriptors, trainDescriptors, k[, mask[, compactResult]] : K # of most closest matching
  - k : # of neighbors to match
  - compactResult=False : True: If no match, exclude in the result
- matches = matcher.radiusMatch(queryDescriptors, trainDescriptors, maxDistance[, mask, compactResult]): Distance matching within maxDistance
  - maxDistance : Distance to match
- DMatch
  - queryIdx : queryDescriptor's index
  - trainIdx : trainDescriptor's index
  - imgIdx : trainDescriptor's image index
  - distance : Similarity distance
- cv2.drawMatches(img1, kp1, img2, kp2, matches, flags): Mark matching point in img1, kp1 : queryDescirptor's image and key point
- img1, kp1 : trainDescirptor's image and key point
- matches : Matching Result
- flags : Matching point drawing option
  - cv2.DRAW_MATCHES_FLAGS_DEFAULT : Newly generate result image (Base value)
  - cv2.DRAW_MATCHES_FLAGS_DRAW_OVER_OUTIMG : Do not newly generate result image
  - cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS: Draw size and orientation of key point
  - cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS : Exclude one-side matching result drawing

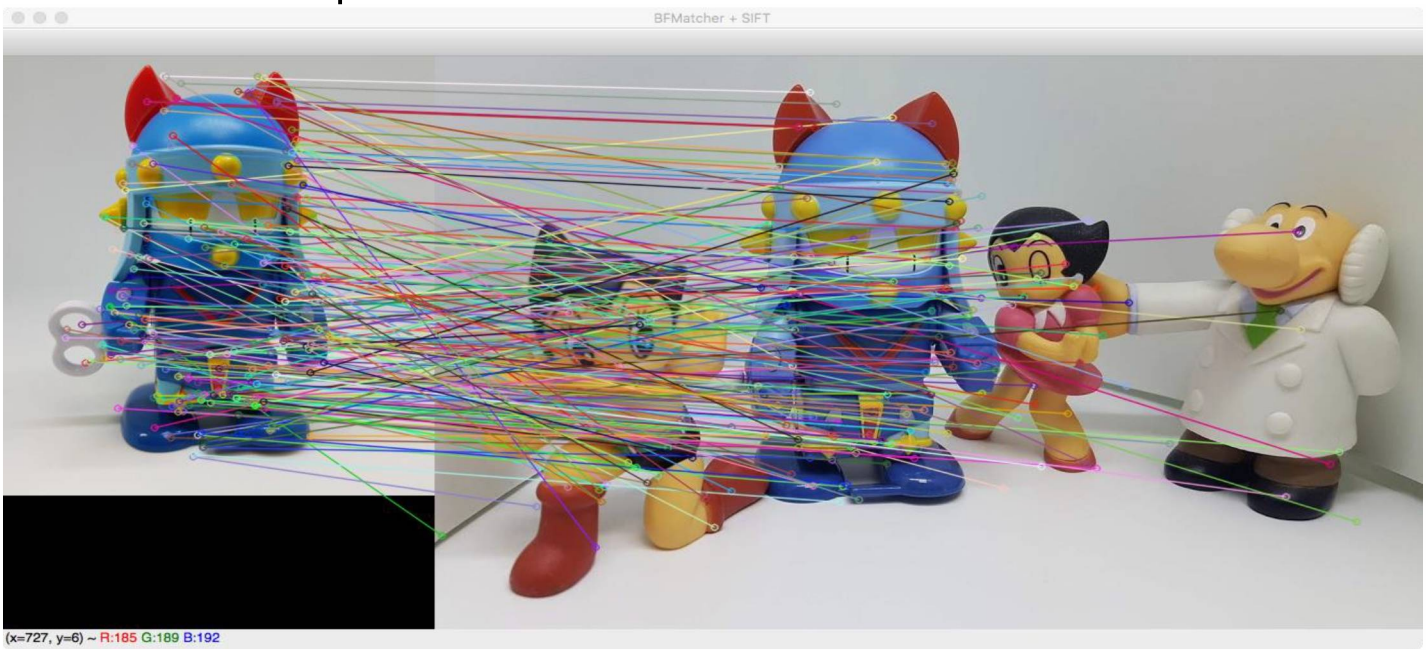
❖ BF(Brute-Force) Matcher

- Investigate all feature descriptor of two images
- matcher = cv.BFMatcher_create([normType[, crossCheck])
  - normType : Distance metering algorithm
    - cv2.NORM_L1
    - cv2.NORM_L2 : Base value
    - cv2.NORM_L2SQR
    - cv2.NORM_HAMMING
    - cv2.NORM_HAMMING2
  - crossCheck=False : Apply only ones with mutual matching
- Selecting normType by feature
  - SIFT SURF : NORM_L1, NORM_L2
  - ORB : NORM_HAMMING
  - ORB , WTA_K : NORM_HAMMING2
- SIFT-BF Example

```
import cv2, numpy as np

img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

detector = cv2.xfeatures2d.SIFT_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)

matcher = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
matches = matcher.match(desc1, desc2)
res = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, \
                      flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_
POINTS)

cv2.imshow('BFMatcher + SIFT', res)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ BF(Brute-Force) Matching

- SIFT-BF Example <Result>

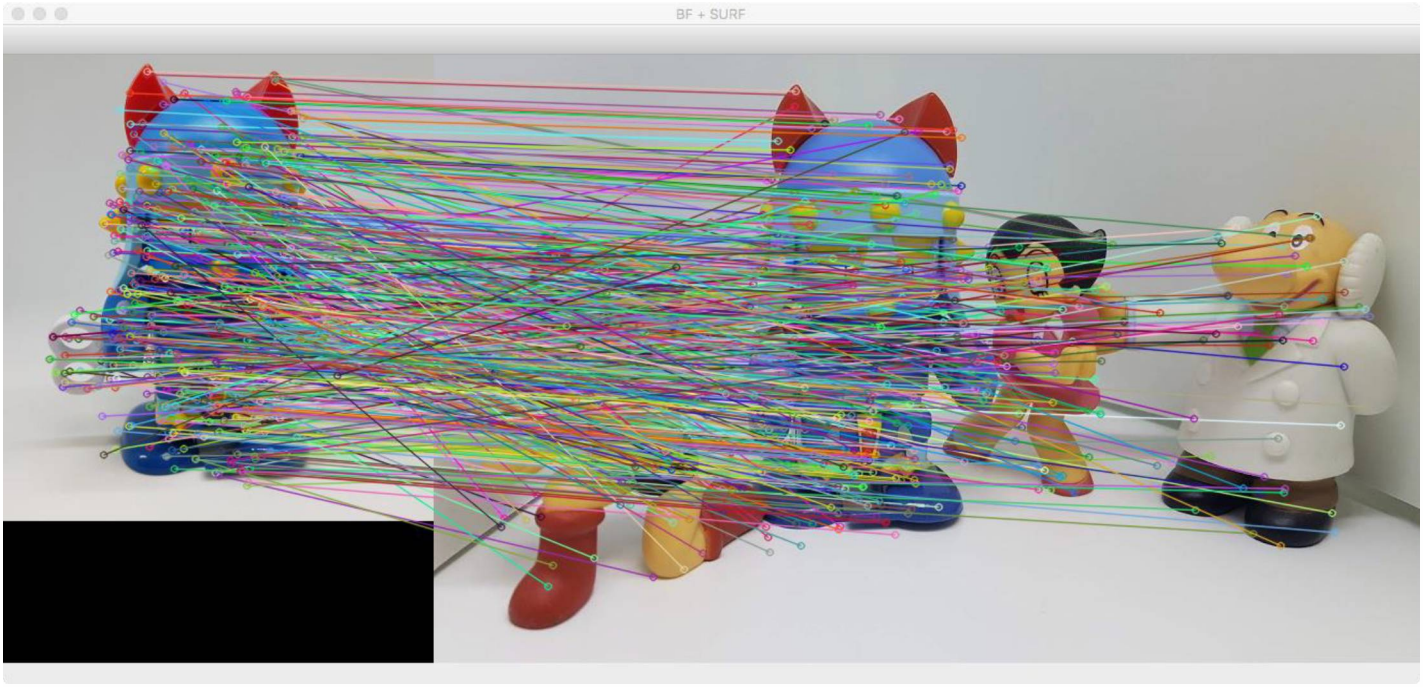


- SURF– BF Example

```
import cv2
import numpy as np
img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
detector = cv2.xfeatures2d.SURF_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)
matcher = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
matches = matcher.match(desc1, desc2)
res = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, \
                      flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_P
OINTS)
cv2.imshow('BF + SURF', res)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ BF(Brute-Force) Matching

- SURF– BF Example <Result>

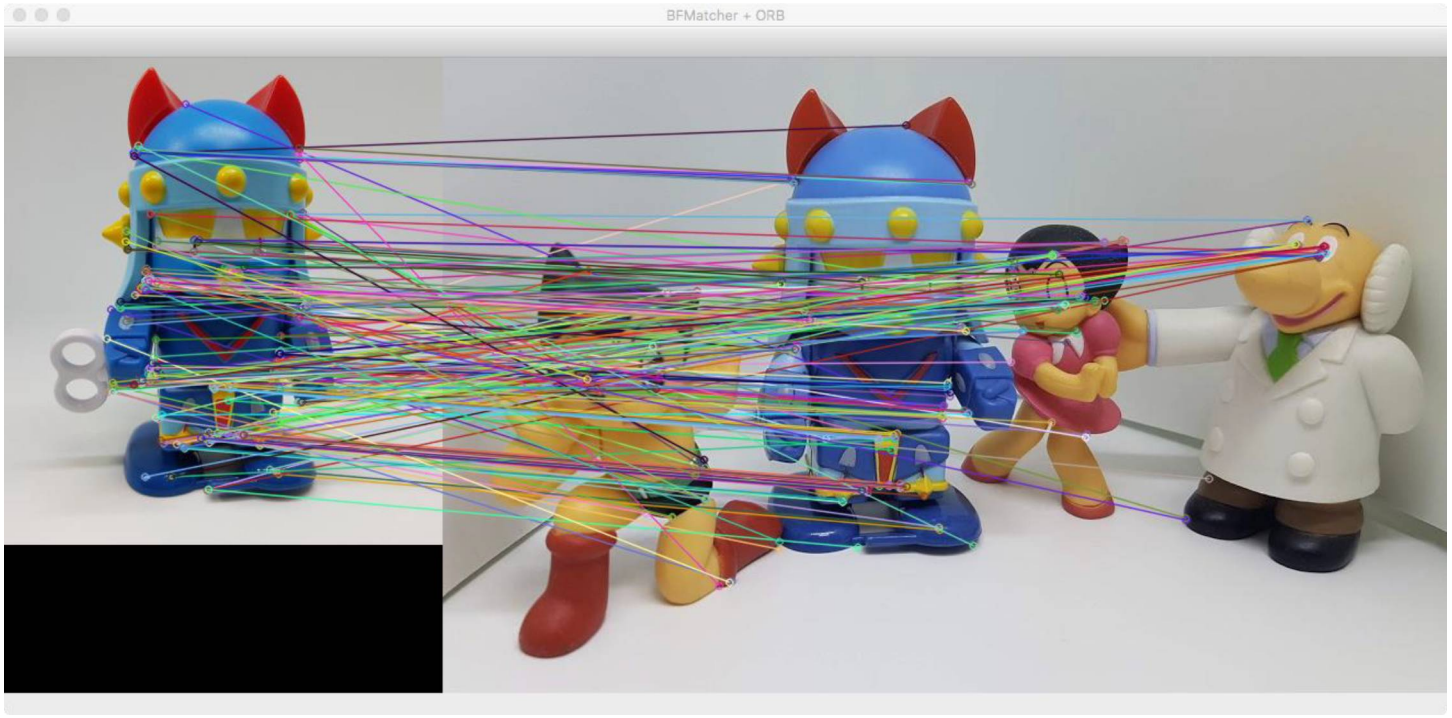


- ORB– BF Example

```
import cv2, numpy as np

img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
detector = cv2.ORB_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)
matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = matcher.match(desc1, desc2)
res = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, \
                      flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_P
OINTS)
cv2.imshow('BFMatcher + ORB', res)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ BF(Brute-Force) Matching

- ORB– BF Example <Result>



❖ FLANN Matching

- Fast library for Approximate Nearest Neighbors Matching
- Disadvantage of BF matching
  - Inspect All: If image size is large, performance decreases
- Do not compare all descriptor, only match most close neighboring value
  - Need extra parameter for algorithm selection
- Select search parameter and index parameter
  - C++ class operation is omitted in python binding
  - Draft key – value couple in dictionary objects
- matcher = cv2.FlannBasedMatcher([indexParams[, searchParams]])
  - indexParams : Index parameter, dictionary
    - algorithm : Algorithm select key, Decide subsidiary key according to selected key

        - FLANN_INDEX_LINEAR = 0: Linear indexing, same as BFMatcher
        - FLANN_INDEX_KDTREE = 1 : KD-Tree Indexing
        - FLANN_INDEX_KMEANS = 2 : K-Average Tree Indexing
        - FLANN_INDEX_COMPOSITE = 3 : KD Tree, K Average mixed indexing
        - FLANN_INDEX_LSH = 6 : LSH Indexing
        - FLANN_INDEX_AUTOTUNED = 255 : Auto index
    - searchParams : Search Parameter, dictionary objects
        - checks=32 : Number of candidnate to search
        - eps =0.0 : Do not use
        - sorted=True : Organize and Turnaround

❖ FLANN Matching

- Difficult and picky to draft index parameter
- Suggested index parameter value
    - SIFT, SURF

```
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
```

    - ORB

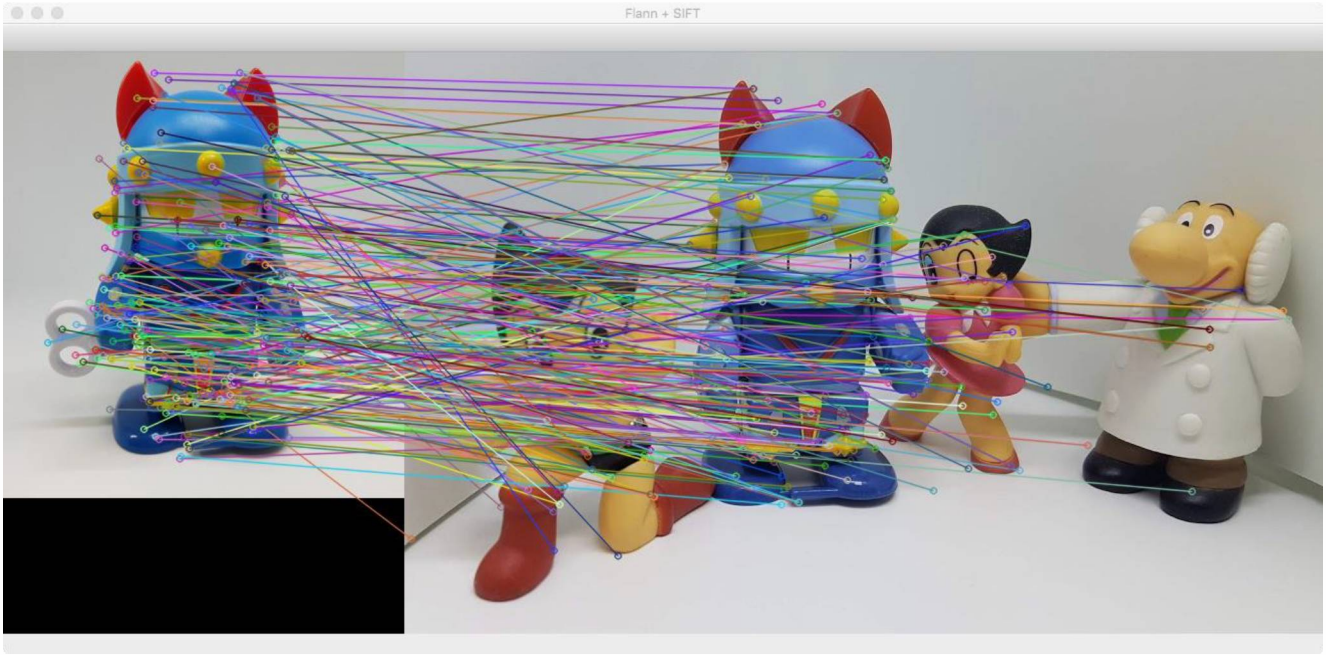
```
FLANN_INDEX_LSH = 6
index_params= dict(algorithm = FLANN_INDEX_LSH,
                   table_number = 6,
                   key_size = 12,
                   multi_probe_level = 1)
```

❖ FLANN Matching

- SIFT-FLANN Example

```
import cv2, numpy as np
img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
detector = cv2.xfeatures2d.SIFT_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)
matcher = cv2.FlannBasedMatcher(index_params, search_params)
matches = matcher.match(desc1, desc2)
res = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, \
                flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Flann + SIFT', res)
cv2.waitKey()
cv2.destroyAllWindows()
```

- SIFT-FLANN Example

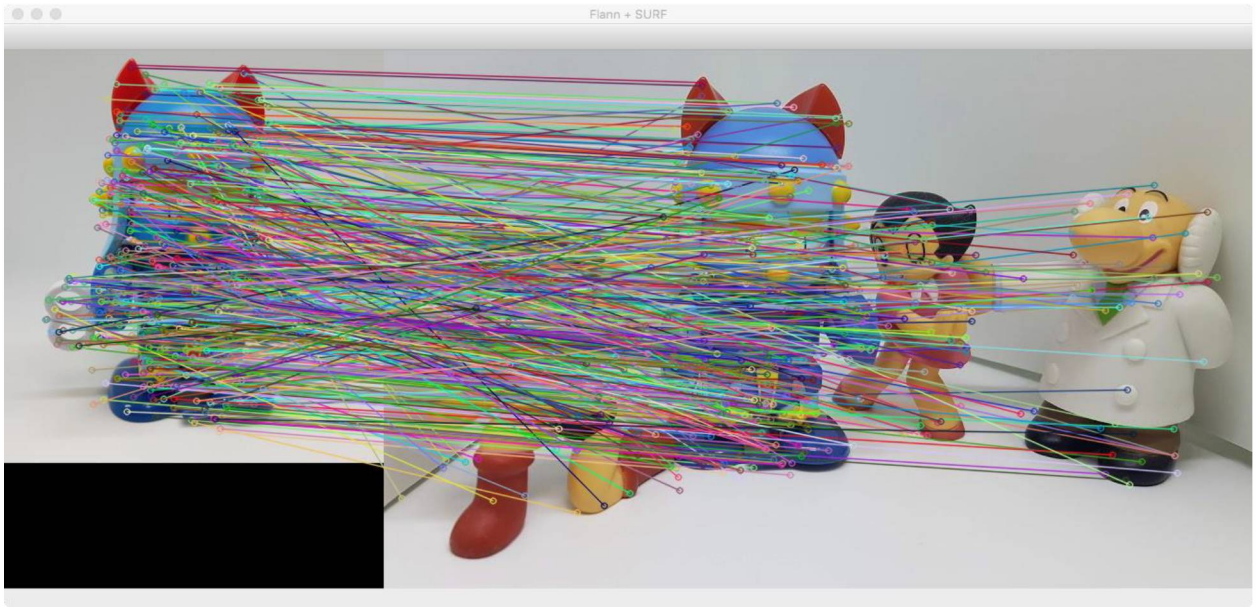


❖ FLANN Matching

- SURF-FLANN Example

```
import cv2, numpy as np
img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
detector = cv2.xfeatures2d.SURF_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)
matcher = cv2.FlannBasedMatcher(index_params, search_params)
matches = matcher.match(desc1, desc2)
res = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, \
                flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS
)
cv2.imshow('Flann + SURF', res)
cv2.waitKey()
cv2.destrovAllWindows()
```

- SURF-FLANN Example <result>

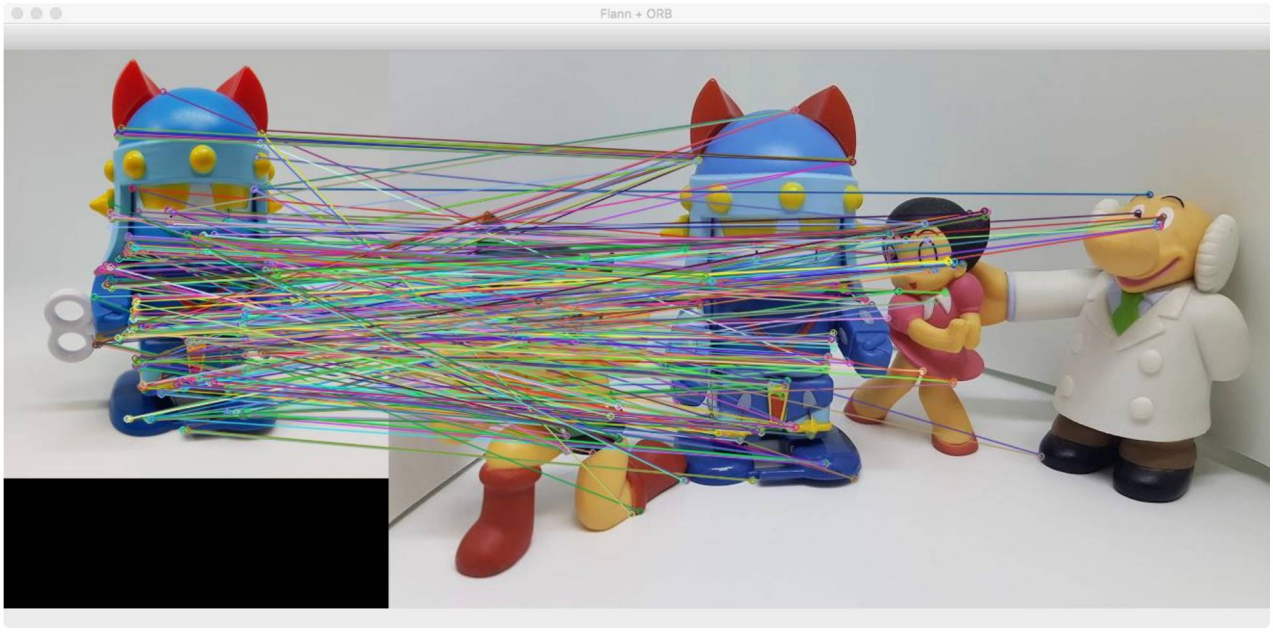


❖ FLANN Matching

- ORB-FLANN Example

```
import cv2, numpy as np
img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
detector = cv2.ORB_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)
FLANN_INDEX_LSH = 6
index_params= dict(algorithm = FLANN_INDEX_LSH,
                   table_number = 6,
                   key_size = 12,
                   multi_probe_level = 1)
search_params=dict(checks=32)
matcher = cv2.FlannBasedMatcher(index_params, search_params)
matches = matcher.match(desc1, desc2)
res = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, \
            flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Flann + ORB', res)
cv2.waitKey()
cv2.destroyAllWindows()
```

- SURF-FLANN Example <result>



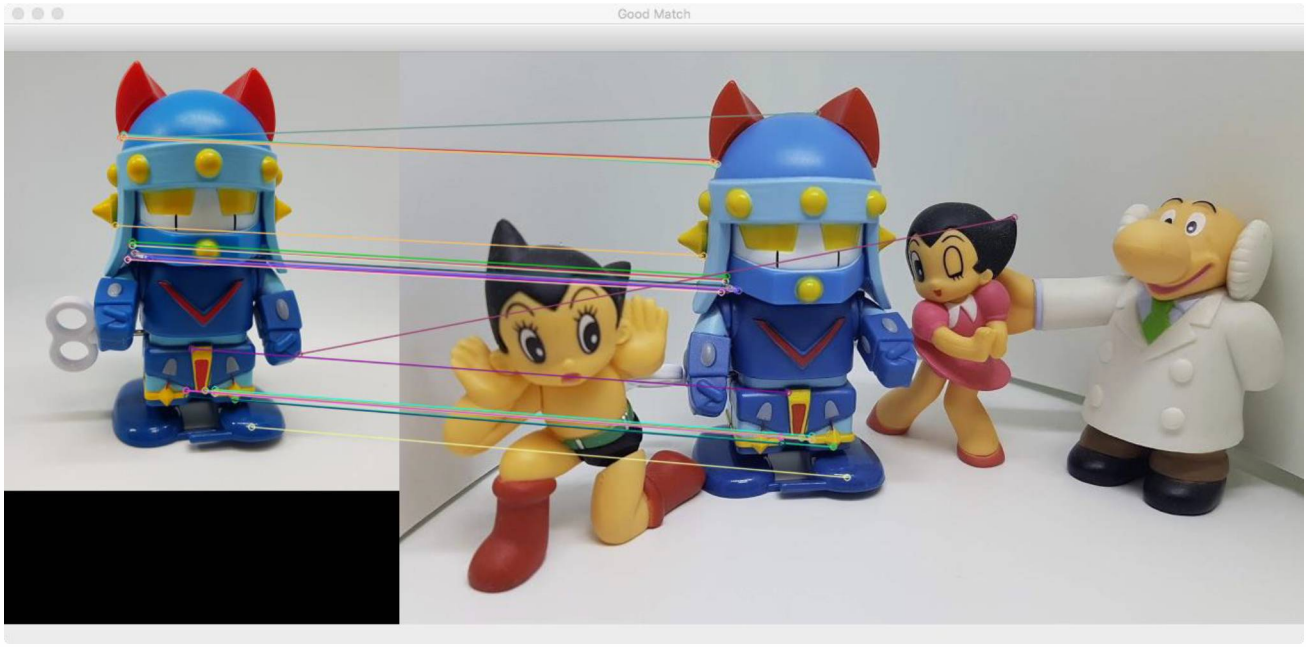
❖ Finding Good Match Points

- match()
  - One matching point turnaround for one narrator
  - Use small distance high rank percentage
- knnMatch()
  - K # of closest neighbor matching point for narrator
  - One that is close among neighbor matching point
- radiusMatch()
  - Within certain distance
  - Meaningless to find good matching point
- match() example

```
import cv2, numpy as np

img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

detector = cv2.ORB_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)
matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = matcher.match(desc1, desc2)
matches = sorted(matches, key=lambda x:x.distance)
min_dist, max_dist = matches[0].distance, matches[-1].distance
ratio = 0.2
good_thresh = (max_dist - min_dist) * ratio + min_dist
good_matches = [m for m in matches if m.distance < good_thresh]
print('matches:%d/%d, min:%.2f, max:%.2f, thresh:%.2f' \
        %(len(good_matches),len(matches), min_dist, max_dist,
good_thresh))
res = cv2.drawMatches(img1, kp1, img2, kp2, good_matches, None, \
                flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Good Match', res)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ FLANN Matching

- match() example <result>

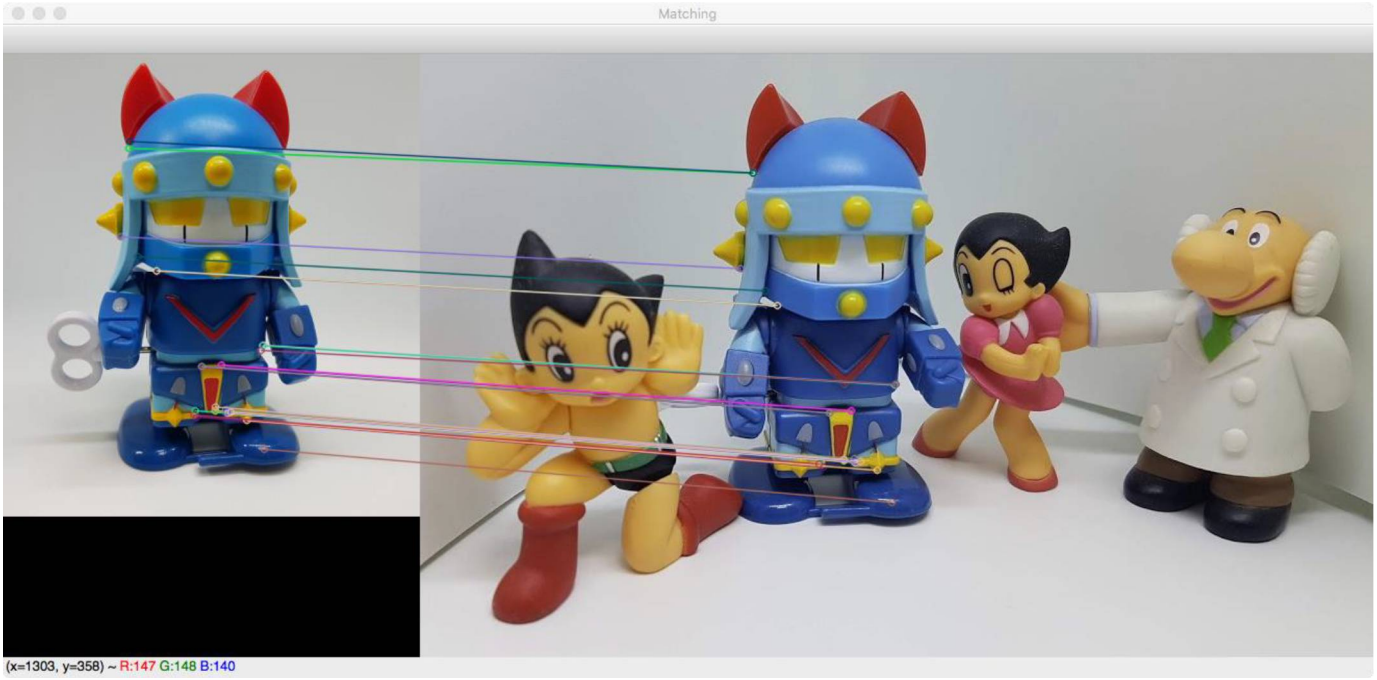


- knnMatch() Example

```
import cv2, numpy as np
img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
detector = cv2.ORB_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)
matcher = cv2.BFMatcher(cv2.NORM_HAMMING2)
matches = matcher.knnMatch(desc1, desc2, 2)
ratio = 0.75
good_matches = [first for first,second in matches \
                    if first.distance < second.distance * ratio]
print('matches:%d/%d' %(len(good_matches),len(matches)))
res = cv2.drawMatches(img1, kp1, img2, kp2, good_matches, None, \
                  flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Matching', res)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Finding Good Matching

- kNNmatch() Example



❖ Matching Homography to find Object

- Calculate distance transform array of two images with good matching points
- Find object location
- Eliminate bad matching point above transform array
- cv2.findHomograpy()
  - Calculate distance transform array approx. with multiple matching couple
- cv2.perspectiveTransform()
  - Distance transform original coordination fit to transform array
- mtrx, mask = cv.findHomography(srcPoints, dstPoints[, method[, ransacReprojThreshold[, mask[, maxIters[, confidence]]]]])
  - srcPoints: Original coordination array
  - dstPoints: Result coordination array
  - method=0 : Select approx. calculation algorithm

        - 0 : Calculate min square errors to all points
        - cv2.RANSAC
        - cv2.LMEDS
        - cv2.RHO
    - ransacReprojThreshold=3: Normal distance threshold value, when RANSAC,RHO
    - maxIters=2000 : # of approximate calculation repetition
    - confidence=0.995 : Reliability, 0 ~ 1
    - mtrx : Result transform array
    - mask : Normal level judging Result, N x 1 array (0:Normal, 1:abnormal)
- dst = cv.perspectiveTransform(src, m[, dst])
- src : Input coordination array
- m : Transform array
- dst : Output coordination array
- Approximate Calculation Algorithm
    - RANSAC(Random Sample Consensus)
        - Select random points, calculate approximate points for most satisfactory points
        - Normal Value (Inlier), Abnormal Value (outlier)
    - LMEDS(Least Median of Squares)
        - Least median value of square
        - No need for extra parameter
        - Run only when normal value is more than 50%
    - RHO
        - Use PROSAC (Progressive Simple Consensus); improved RANSAC
        - Faster when there is more outlier

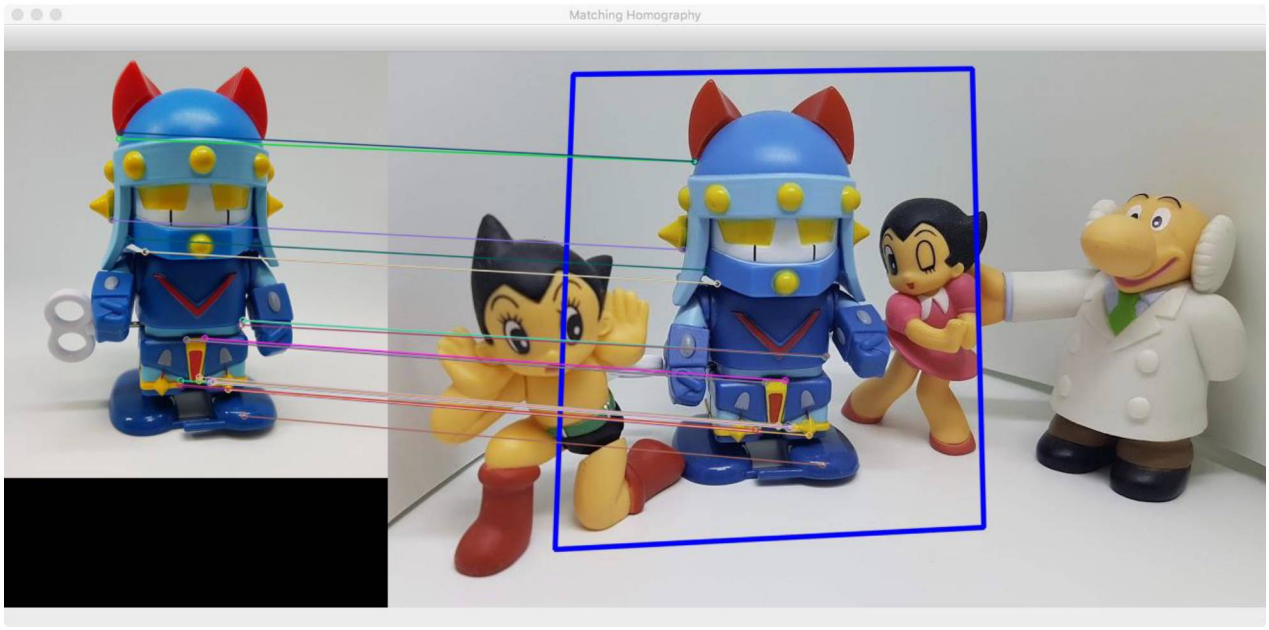
❖ Matching Homography to find Object

- Example

```
import cv2, numpy as np

img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

detector = cv2.ORB_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)
matcher = cv2.BFMatcher(cv2.NORM_HAMMING2)
matches = matcher.knnMatch(desc1, desc2, 2)

ratio = 0.75
good_matches = [first for first,second in matches \
                    if first.distance < second.distance * ratio]
print('good matches:%d/%d' %(len(good_matches),len(matches)))

src_pts = np.float32([ kp1[m.queryIdx].pt for m in good_matches ])
dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good_matches ])
mtrx, mask = cv2.findHomography(src_pts, dst_pts)
h,w, = img1.shape[:2]
pts = np.float32([ [[0,0]],[[0,h-1]],[[w-1,h-1]],[[w-1,0]] ])
dst = cv2.perspectiveTransform(pts,mtrx)
img2 = cv2.polylines(img2,[np.int32(dst)],True,255,3, cv2.LINE_AA)

res = cv2.drawMatches(img1, kp1, img2, kp2, good_matches, None, \
                    flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_PO
INTS)
cv2.imshow('Matching Homography', res)
cv2.waitKey()
cv2.destroyAllWindows()
```

❖ Matching Homography to find Object

- Example <result>



❖ Remove Bad Mathcing with RANSAC

- Example <1/2>

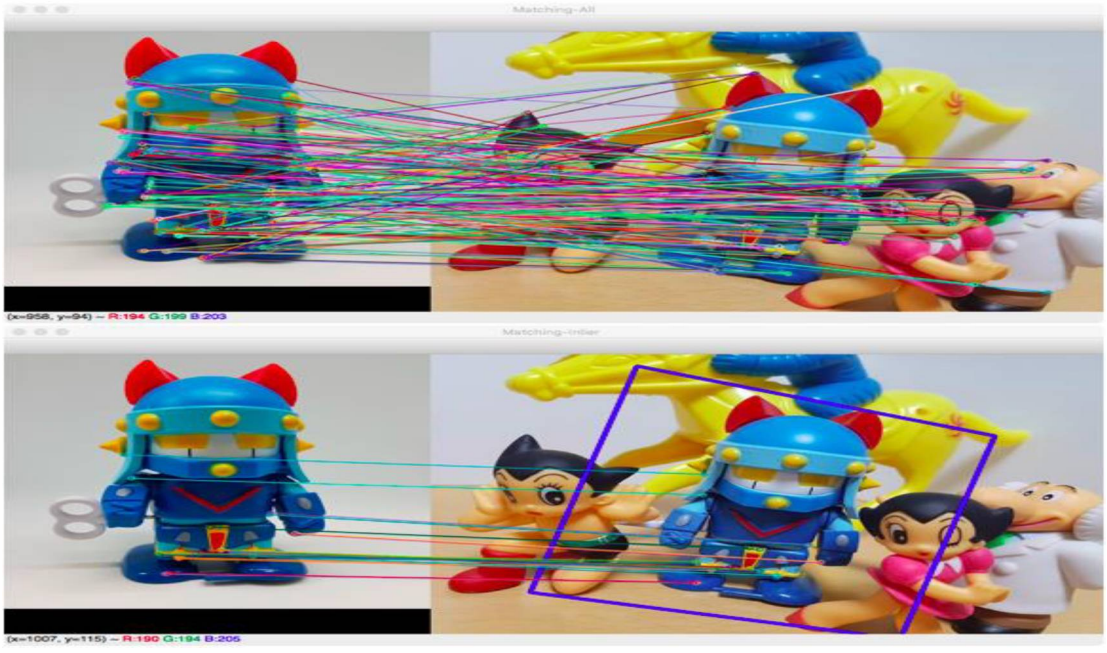
```
import cv2, numpy as np

img1 = cv2.imread('../img/taekwonv1.jpg')
img2 = cv2.imread('../img/figures2.jpg')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

detector = cv2.ORB_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)
matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = matcher.match(desc1, desc2)

matches = sorted(matches, key=lambda x:x.distance)
res1 = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, \
                       flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_PO
INTS)
```

❖ Remove Bad Mathcing with RANSAC

- Example <2/2>

```
src_pts = np.float32([ kp1[m.queryIdx].pt for m in matches ])
dst_pts = np.float32([ kp2[m.trainIdx].pt for m in matches ])
mtrx, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
h,w = img1.shape[:2]
pts = np.float32([ [[0,0]],[[0,h-1]],[[w-1,h-1]],[[w-1,0]] ])
dst = cv2.perspectiveTransform(pts,mtrx)
img2 = cv2.polylines(img2,[np.int32(dst)],True,255,3, cv2.LINE_AA)
matchesMask = mask.ravel().tolist()
res2 = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, \
                    matchesMask = matchesMask,
                    flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_PO
INTS)
accuracy=float(mask.sum()) / mask.size
print("accuracy: %d/%d(%.2f%%)"% (mask.sum(), mask.size, accuracy))
cv2.imshow('Matching-All', res1)
cv2.imshow('Matching-Inlier ', res2)
cv2.waitKey()
cv2.destroyAllWindows()
```

- Example <result>



❖ Finding Object with Camera

- Example <1/2>

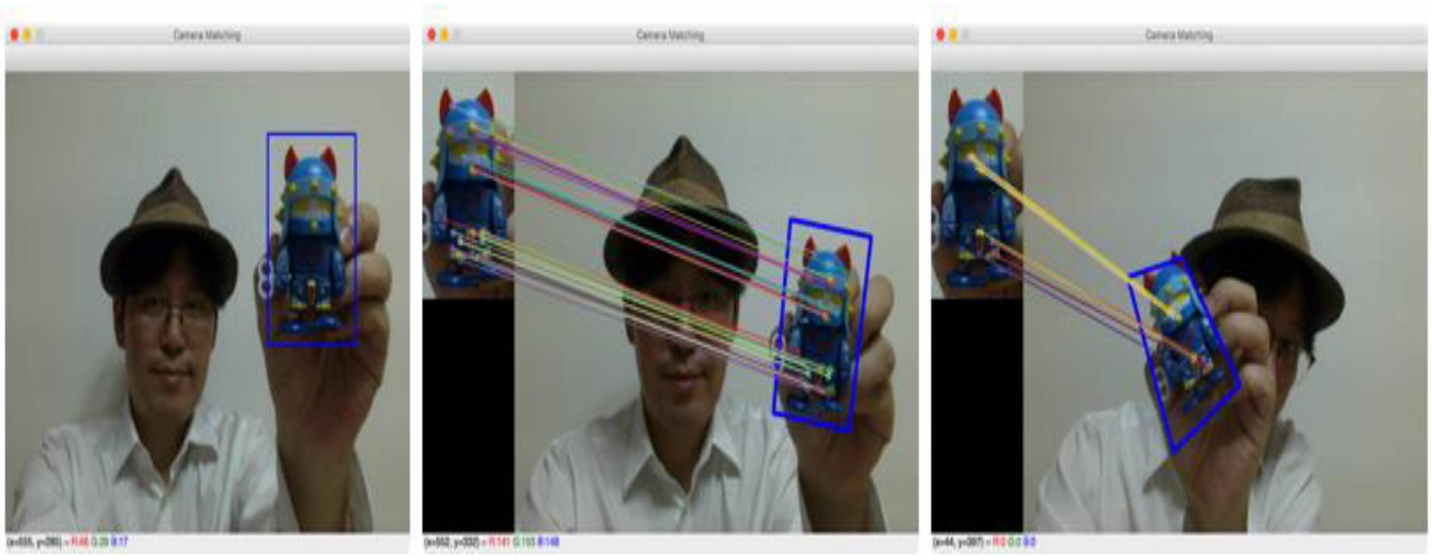
```
import cv2, numpy as np
img1 = None
win_name = 'Camera Matching'
MIN_MATCH = 10
detector = cv2.ORB_create(1000)
FLANN_INDEX_LSH = 6
index_params= dict(algorithm = FLANN_INDEX_LSH,
                   table_number = 6,
                   key_size = 12,
                   multi_probe_level = 1)
search_params=dict(checks=32)
matcher = cv2.FlannBasedMatcher(index_params, search_params)
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
while cap.isOpened():
    ret, frame = cap.read()
    if img1 is None:  # No registered image, camera by-pass
        res = frame
    else:             # Registered image exist, matching starts
        img2 = frame
        gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
        gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
        kp1, desc1 = detector.detectAndCompute(gray1, None)
        kp2, desc2 = detector.detectAndCompute(gray2, None)
        matches = matcher.knnMatch(desc1, desc2, 2)
        ratio = 0.75
        good_matches = [m[0] for m in matches \
                            if len(m) == 2 and m[0].distance <
m[1].distance * ratio]
        matchesMask = np.zeros(len(good_matches)).tolist()
        if len(good_matches) > MIN_MATCH:
            src_pts = np.float32([ kp1[m.queryIdx].pt for m in
good_matches ])
            dst_pts = np.float32([ kp2[m.trainIdx].pt for m in
good_matches ])
            mtrx, mask = cv2.findHomography(src_pts, dst_pts,
cv2.RANSAC, 5.0)
```

❖ Finding Object with Camera

- Example <2/2>

```
                if mask.sum() > MIN_MATCH: # Min # of normal matching point is
                                                above certain number
                matchesMask = mask.ravel().tolist()
                h,w, = img1.shape[:2]
                pts = np.float32([ [[0,0]],[[0,h-1]],[[w-1,h-1]],[[w-1,0]] ])
                dst = cv2.perspectiveTransform(pts,mtrx)
                img2 = cv2.polylines(img2,[np.int32(dst)],True,255,3,
cv2.LINE_AA)
        res = cv2.drawMatches(img1, kp1, img2, kp2, good_matches, None,\
                              matchesMask=matchesMask,
                              flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINT
S)
    cv2.imshow(win_name, res)
    key = cv2.waitKey(1)
    if key == 27:    # Esc, End
            break
    elif key == ord(' '): # Set ROI with spacebar, set img1
        x,y,w,h = cv2.selectROI(win_name, frame, False)
        if w and h:
            img1 = frame[y:y+h, x:x+w]
else:
    print("can't open camera.")
cap.release()
cv2.destroyAllWindows()
```

- Example <Result>



Matching & Tracking

1. Matching with a similar image
2. Feature and Key Point
3. Descriptor Extractor
4. Feature Matching
5. **Tracking**
6. Workshop

❖ Background Subtraction

- Need to track visitors or traffics
- Easy calculation with prepared background image
- But it often does not have prepared background images
- cv2.BackgroundSubtractor Abstraction Class
  - fgmask = bgsubtractor.apply(image[, fgmask[, learningRate]])
    - image : Input Image
    - fgmask : Front Image Mask
    - learningRate=-1 : Background, Training Speed, 0 ~ 1, -1:Auto
  - backgroundImage = bgsubtractor.getBackgroundImage([,backgroundImage])
    - backgroundImage : Training background image
- Algorithms to eliminate background
  - BackgroundSubtractorMOG
  - BackgroundSubtractorMOG2
  - BackgroundSubtractorGMG

❖ BackgroundSubtractorMOG

- Gausisian Mixture-based Backgorund/Fofeground Segmentation Algorithm
- K Gaussian Distribution (From K=3 to 5)
- Use blend to each background pixel
- Weight is the ratio of how much each pixel stays in one places
  - Should background stays long or fixed
- cv2.bgsegm.createBackgroundSubtractorMOG()
  - Must set history length, number of Gaussian mixture, threshold valu
  - Call apply (frame) only, basic value is set
  - contrib module

BackgroundSubtractorMOG

- Example

```
import numpy as np, cv2

cap = cv2.VideoCapture('../img/walking.avi')
fps = cap.get(cv2.CAP_PROP_FPS) # Get # of frames
delay = int(1000/fps)

fgbg = cv2.bgsegm.createBackgroundSubtractorMOG()
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    fgmask = fgbg.apply(frame)
    cv2.imshow('frame',fgmask)
    cv2.imshow('bg', fgbg.getBackgroundImage())
    if cv2.waitKey(1) & 0xff == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

- Example <Result>

BackgroundSubtractorMOG2

- Based on 2 papers.
- Z.Zivkovic, "Improved adaptive Gausian mixture model for background subtraction" in 2004
- "Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction" in 2006.
- Select appropriate Gaussian distribution for each pixel
- Good for various scenes as for transformation by light
- retval = cv.createBackgroundSubtractorMOG2([, history[, varThreshold[, detectShadows]]])
  - history=500 : # of history
  - varThreshold=16 : Distribution Threshold value
  - detectShadows=True : Mark shadow
- fgbg.apply(frame)
  - detecShadows=Set True(Basic Value)
    - Mark shadow in gray, it gets slow.
- Example

```
import numpy as np, cv2

cap = cv2.VideoCapture('../img/walking.avi')
fps = cap.get(cv2.CAP_PROP_FPS) # Count frame number
delay = int(1000/fps)
fgbg = cv2.createBackgroundSubtractorMOG2()
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    fgmask = fgbg.apply(frame)
    cv2.imshow('frame',frame)
    cv2.imshow('bgsub',fgmask)
    if cv2.waitKey(delay) & 0xff == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

❖ BackgroundSubtractorMOG2

- Example <Result>

❖ BackgroundSubtractorGMG

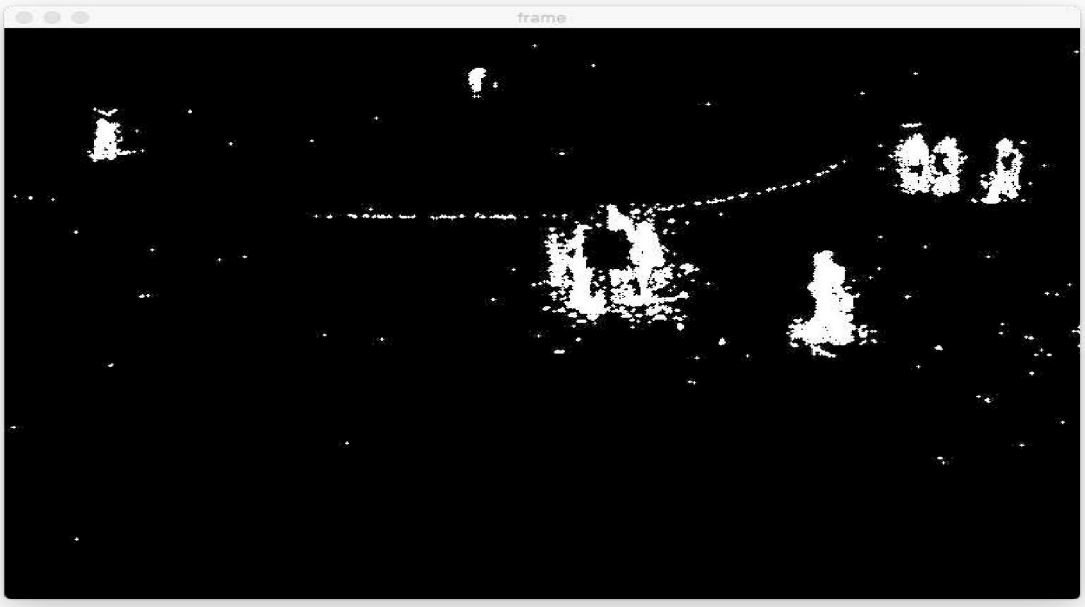
- Andrew B. Godbehere, Akihiro Matsukawa, Ken Goldberg , "Visual Tracking of Human Visitors under Variable-Lighting Conditions for a Responsive Audio Art Installation" , 2012.
- An algorithm combining statistical background image estimation and Bayesian division by pixel.
- Used in background modeling of initial frames (120, Base value)
  - First few frame will be all black
- Use Bayesian assumption to recognize front objects
- Place more weight to most recent view differences to accept lighting changes
- Use morphology filter to eliminate noise (closing, opening)
- fgbg = cv2.bgsegm.createBackgroundSubtractorGMG() # create object
- kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
- fgmask = fgbg.apply(frame)
- fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel

❖ BackgroundSubtractorGMG

- Example

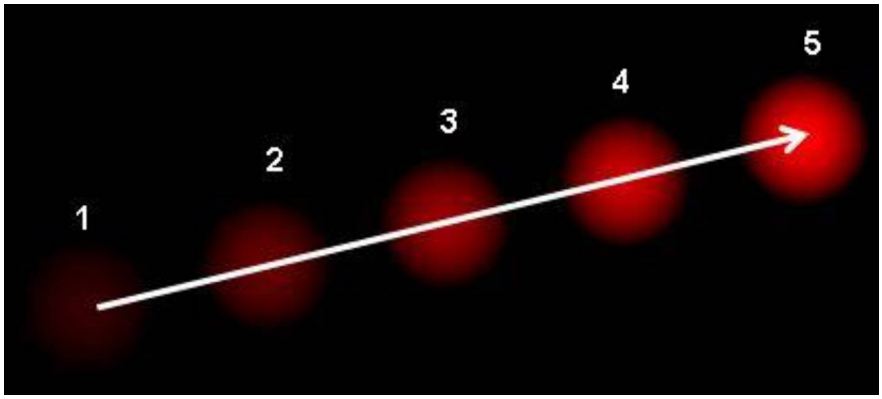
```
import numpy as np
import cv2
cap = cv2.VideoCapture('../img/walking.avi')
#fgbg = cv2.createBackgroundSubtractorGMG() # create object
fgbg = cv2.bgsegm.createBackgroundSubtractorGMG() # create
object
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3
while(1):
    ret, frame = cap.read()
    fgmask = fgbg.apply(frame)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN,
kernel) # filtering for reduce noise
    cv2.imshow('frame',fgmask)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

- Example <Result>



❖ Optical Flow

- Distribution of move orientation and distance in between previous frame and current frame
- Method of chasing movement without previous information in frame
- Sparse Optical Flow (Rare)
  - Pre-set part of points to chase
  - Lucas-Kanade: Square methods
- Dense Optical Flow(Density)
  - Calculating velocity in all pixels
  - Large quantity of calculation



❖ Luca—Kaneda

- 3 assumption of LK algorithm
  - Brightness Constancy
    - Pixel brightness of chasing object do not change
  - Temporal Persistence
    - Time moves faster than object movement in images
    - Differences between frame is not big
  - Spatial Consistency
    - High possibility for neighboring dots (neighboring pixels) to be included in the same objects and
  - Have same movement
- It is used in small window, it cannot calculate big movements
- Use image pyramid to improve it

- OpenCV Function
  - cv2.calcOpticalFlowLK : Do not use image pyramid
  - cv2.calcOpticalFlowPyrLK : Use image pyramid
  - Use dots selected by goodToTrack as input
- calcOpticalFlowPyrLK(prevImg, nextImg, prevPts, nextPts)
  - prevImg : Previous Frame, n*1*2
  - nextImg : Next Frame, n*1*2
  - prevPts : Point of move chanse , n*1*2
  - nextPts : Position of dots chasing object passed by, None, use return value
  - winSize : Window size of each pyramid levels,
  - maxLevel : # of layers of image pyramid, Do not use if it is 0
  - criteria : Search halt criteria
- return:
  - nextPts : Position of dots chasing object passed by
  - status : Same # as nextPts
    - 1 or 0 in each index, 0: Did not find corresponding pint (n*1)
  - err : Distance between previous frame feature point and curent frame moving distance point, use to eliminate
- Flow Example <1/3>

```
import numpy as np, cv2

cap = cv2.VideoCapture('../img/walking.avi')
fps = cap.get(cv2.CAP_PROP_FPS) # Count frame #
delay = int(1000/fps)

color = np.random.randint(0,255,(200,3))
lines = None
prevImg = None
# calcOpticalFlowPyrLK Set halt condition
termcriteria =  (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT,
10, 0.03)
```

Luca—Kaneda

- Flow Example <2/2>

```
while cap.isOpened():
    ret,frame = cap.read()
    if not ret:
        break
    img_draw = frame.copy()
    if prevImg is None:
        prevImg = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        lines = np.zeros_like(frame)
        prevPt = cv2.goodFeaturesToTrack(prevImg, 200, 0.01, 10)

     else:
        nextImg = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        nextPt, status, err = cv2.calcOpticalFlowPyrLK(prevImg,
nextImg, prevPt, None, criteria=termcriteria)
        prevMv = prevPt[status==1]
        nextMv = nextPt[status==1]

 for i,(p, n) in enumerate(zip(prevMv, nextMv)):
            px,py = p.ravel()
            nx,ny = n.ravel()
            cv2.line(lines, (px, py), (nx,ny), color[i].tolist(),2)
            cv2.circle(img_draw, (nx,ny), 2, color[i].tolist(), -1)
        img_draw = cv2.add(img_draw, lines)
        prevImg = nextImg
        prevPt = nextMv.reshape(-1,1,2)
    cv2.imshow('OpticalFlow-LK', img_draw)
    key = cv2.waitKey(delay)
    if key == 27 : # Esc: end
        break
    elif key == 8: # Backspace: Delete chasing log
        prevImg = None
cv2.destroyAllWindows()
cap.release()
```

❖ Luca—Kaneda

- Flow Example <Result>

❖ Dense Optical Flow

- cv2.calcOpticalFlowFarneback(prev, next, flow, pyr_scale, levels, winsize, iterations, poly_n, poly_sigma, flags)
  - prev : Previous image
  - next : Next Frame
  - flow : Calculation having same size as previous frame for 32F type Resultflow, None
  - pyr_scale : Pyramid scale, 0.5
  - levels : # of pyramid levels
  - winsize : Average window size, Larger value makes stronger algorithm.
  - iteration : # of repeats in each pyramid

- poly_n : # of neighboring pixel to use polynomial equation in each pixel, 5 or 7
- poly_sigma: Sigma value to use in Gaussian variation poly_n=5:1.1,poly_n=7:1.5
- flags : Calculation Mode
  - cv2.OPTFLOW_USE_INITIAL_FLOW: Use flow value as initial value
  - cv2.OPTFLOW_FARNEBACK_GAUSSIAN: Use box filter instead of Gaussian filter
- Turnaround: Same size as input image, Distance value of each pixel move

- Example <1/2>

```
import cv2, numpy as np

def drawFlow(img,flow,step=16):
  h,w = img.shape[:2]
  idx_y,idx_x =
np.mgrid[step/2:h:step,step/2:w:step].astype(np.int)
  indices =  np.stack( (idx_x,idx_y), axis =-1).reshape(-1,2)

  for x,y in indices:   # Index Tour
    cv2.circle(img, (x,y), 1, (0,255,0), -1)
    dx,dy = flow[y, x].astype(np.int)
    cv2.line(img, (x,y), (x+dx, y+dy), (0,255, 0),2, cv2.LINE_AA )


prev = None # Stored variable of previous frame

cap = cv2.VideoCapture('../img/walking.avi')
fps = cap.get(cv2.CAP_PROP_FPS) # Counting frame number
delay = int(1000/fps)
while cap.isOpened():
  ret,frame = cap.read()
  if not ret: break
  gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
  if prev is None:
    prev = gray
  else:
    flow =
```

❖ Dense Optical Flow

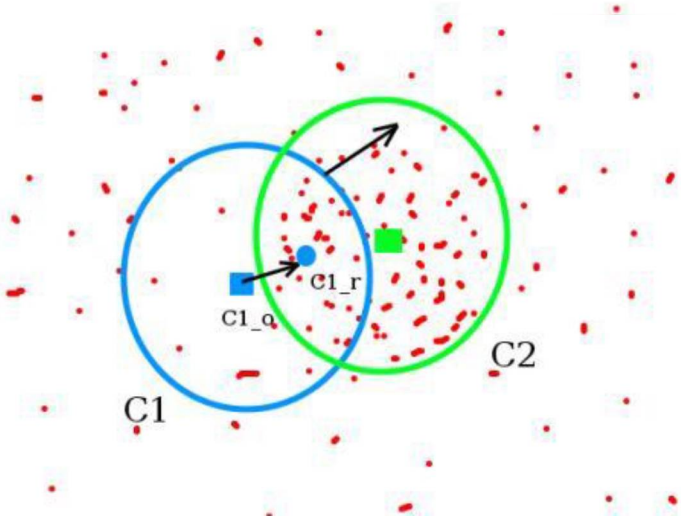
- Example <2/2>

```
cv2.calcOpticalFlowFarneback(prev,gray,None,0.5,3,15,3,5,1.1,cv2.OP
TFLOW_FARNEBACK_GAUSSIAN)
    drawFlow(frame,flow)
    prev = gray
  cv2.imshow('OpticalFlow-Farneback', frame)
  if cv2.waitKey(delay) == 27:
      break
cap.release()
cv2.destroyAllWindows()
```

- Example <result>

❖ Meanshift

- Chase ROI objects based on density distribution (feature or color)
  - Decide initial window
  - Calculate center of mass of data within window
  - Move center of window to center of mass
  - Repeat until window movement stops
- Applying Steps
  - Select chasing object, calculate H histogram of HSV color
  - Back projects to histogram of all imagesn
  - Chase object that moved from back projection by MeanShift



- cv2.meanShift(img, window, criteria)
  - img : Input Image
  - window : Initial window coordination
  - criteria : Criteria to stop search repeat
    - type :cv2.TERM_CRITERIA_*
      - EPS : When accuracy is smaller than epsilon
      - MAX_ITER : Repeat max_iter #
      - COUNT : Same as MAX_ITER
    - max_iter : Max # of repeat
    - epsilon : Minimum accuracy

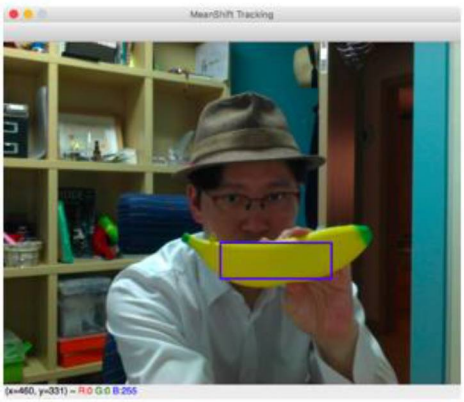
Meanshift

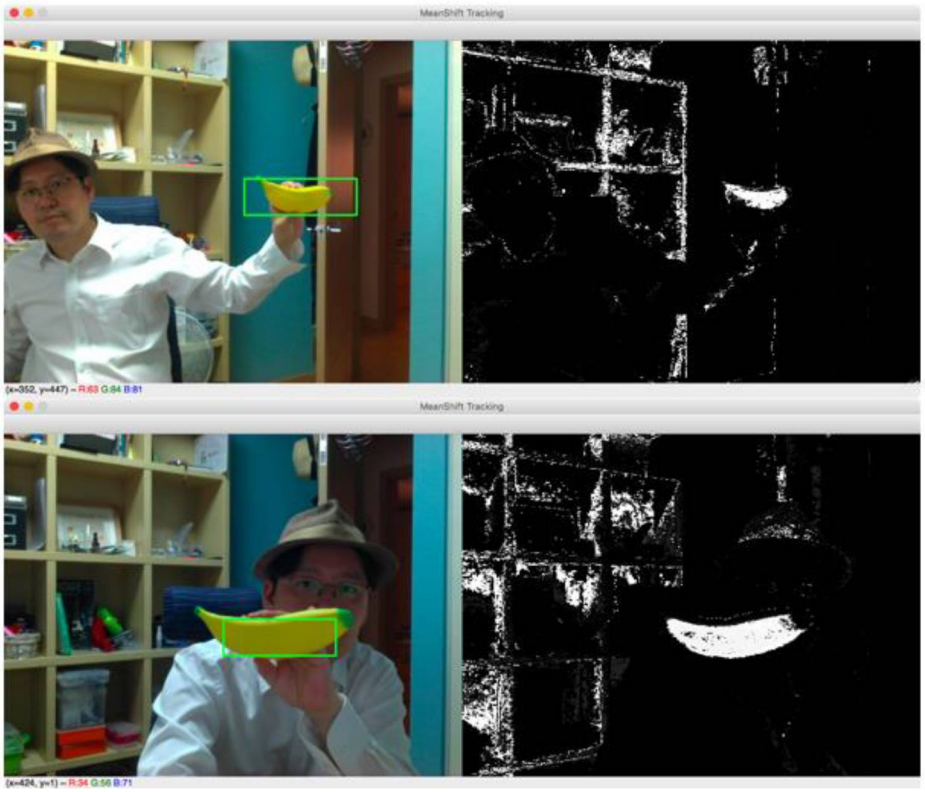
- Example

```
import numpy as np, cv2

roi_hist = None     # Histogram storage vraiable for chasing
objects
win_name = 'MeanShift Tracking'
termination =  (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT,
10, 1)

cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    img_draw = frame.copy()

    if roi_hist is not None:  # Histogram for chasing object
registered
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([hsv], [0], roi_hist, [0,180], 1)
        ret, (x,y,w,h) = cv2.meanShift(dst, (x,y,w,h), termination)
        cv2.rectangle(img_draw, (x,y), (x+w, y+h), (0,255,0), 2)
        result = np.hstack((img_draw, cv2.cvtColor(dst,
cv2.COLOR_GRAY2BGR)))
    else:  # Histogram for chasing object not registered
        cv2.putText(img_draw, "Hit the Space to set target to
track", (10,30),cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 1,
cv2.LINE_AA)
        result = img_draw
```

❖ Meanshift

- Example <Result>





❖ Camshift

- Continuously Adaptive Meanshift
- Improve window fix issues of Meanshift
- Calculate center point using meanshift
- Reset window size and orientation

❖ cv2.CamShift(img, window, criteria)

- img : Input image
- window : Initial search window
- criteria : Criteria to halt search

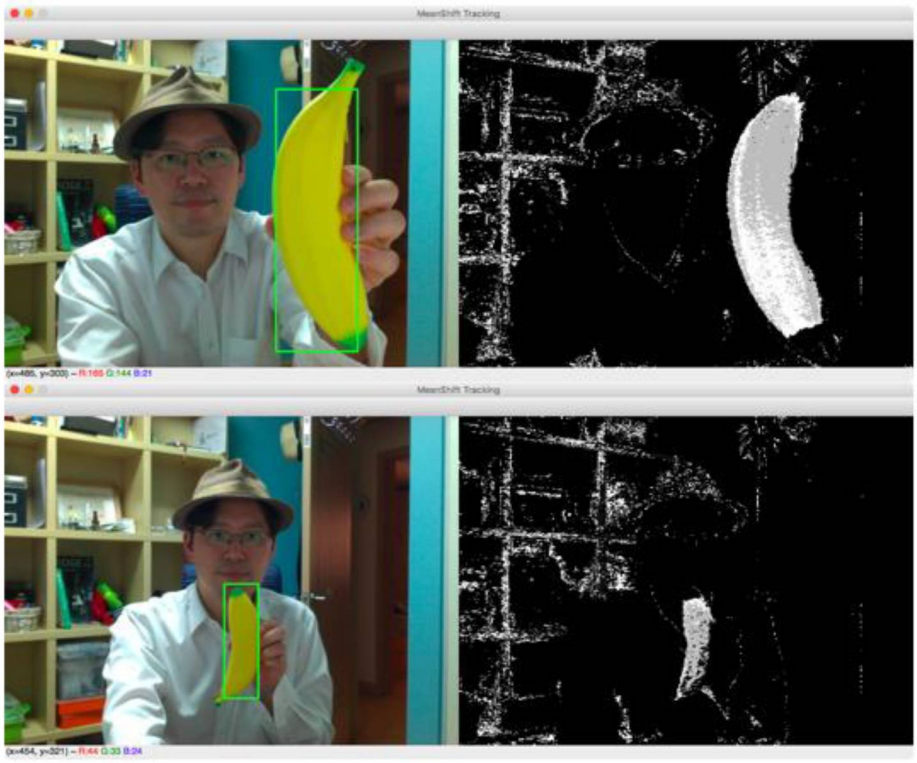
Camshift

- Example <1/2>

```
import numpy as np, cv2

roi_hist = None     # Saved variables for detecting object histogram
win_name = 'MeanShift Tracking'
termination =  (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
while cap.isOpened():
    ret, frame = cap.read()
    img_draw = frame.copy()

    if roi_hist is not None:  # detecting object histogram registered
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([hsv], [0], roi_hist, [0,180], 1)
        ret, (x,y,w,h) = cv2.CamShift(dst, (x,y,w,h), termination)
        cv2.rectangle(img_draw, (x,y), (x+w, y+h), (0,255,0), 2)
        result = np.hstack((img_draw, cv2.cvtColor(dst, cv2.COLOR_GRAY2BGR)))
    else:  # detecting object histogram not registered
        cv2.putText(img_draw, "Hit the Space to set target to track",
(10,30),cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 1, cv2.LINE_AA)
        result = img_draw
cv2.imshow(win_name, result)
    key = cv2.waitKey(1) & 0xff
    if  key == 27: # Esc
        break
    elif key == ord(' '): # Space bar, Set ROI
        x,y,w,h = cv2.selectROI(win_name, frame, False)
        if w and h :    # ROI is set properly
            roi = frame[y:y+h, x:x+w]
            roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
            mask = None
            roi_hist = cv2.calcHist([roi], [0], mask, [180], [0,180])
            cv2.normalize(roi_hist, roi_hist, 0, 255, cv2.NORM_MINMAX)
        else:                       # ROI is not set properly
            roi_hist = None
else:
    print('no camera!')
cap.release()
cv2.destroyAllWindows()
```

❖ Camshift

- Example <result>



❖ Tracking API

- Add to OpenCV3 contrib
- Tracking API operated by machine learning algorithm
- Even users do not know algorithm, them can select various algorithms in interface to select tracker
- retval = cv.Tracker.init(image, boundingBox) : Initialize Trackers
  - image : Input Image
  - boundingBox : Coordination with object to detect (x,y,w,h)
- retval, boundingBox = cv.Tracker.update(image) : Find location of objects to detect in new frame
  - image : New frame image
  - retval : Detection pass/fail
  - boundingBox : New position of detected objects in new frame (x,y,w,h)

❖ Tracking API

- Tracker Generator
  - tracker = cv2.TrackerBoosting_create()
    Based on AdaBoot algorithm
  - tracker = cv2.TrackerMIL_create()
    Based on MIL(Multiple Instance Learning) algorithm
  - tracker = cv2.TrackerKCF_create()
    Based on KCF(Kernelized Correlation Filters) algorithm
  - tracker = cv2.TrackerTLD_create()
    Based on TLD(Tracking, learning and detection) algorithm
  - tracker = cv2.TrackerMedianFlow_create()
    Measure mismatch of objects by detecting front/back orientation
  - tracker = cv2.TrackerGOTURN_create()
    Based on CNN(Convolutional Neural Networks), Does not work for OpenCV 3.4 version; it recognizes it as bug
  - tracker = cv.TrackerCSRT_create()
    CSRT(Channel and Spatial Reliability)
  - tracker = cv.TrackerMOSSE_create()
    Use grey scale internally
- Example <1/3>

```
import cv2
trackers = [cv2.TrackerBoosting_create,
            cv2.TrackerMIL_create,
            cv2.TrackerKCF_create,
            cv2.TrackerTLD_create,
            cv2.TrackerMedianFlow_create,
            cv2.TrackerGOTURN_create, #Error
            cv2.TrackerCSRT_create,
            cv2.TrackerMOSSE_create]
trackerIdx = 0  # Selection index for tracker generator function
tracker = None
isFirst = True
video_src = "../img/highway.mp4"
cap = cv2.VideoCapture(video_src)
fps = cap.get(cv2.CAP_PROP_FPS) # Get frame number
delay = int(1000/fps)
win_name = 'Tracking APIs'
```

Tracking API

- Example <2/3>

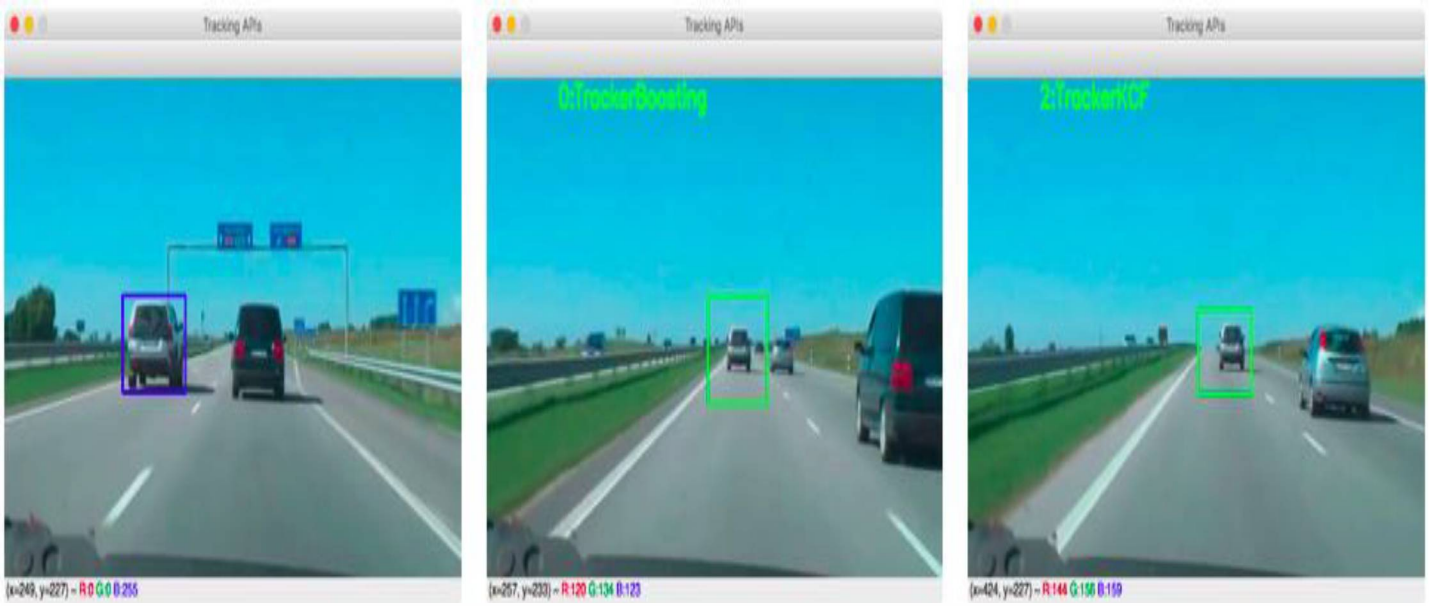
```
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print('Cannot read video file')
        break
    img_draw = frame.copy()
    if tracker is None: # When tracker is not generated
        cv2.putText(img_draw, "Press the Space to set ROI!!",
(100,80), cv2.FONT_HERSHEY_SIMPLEX, 0.75,(0,0,255),2,cv2.LINE_AA)
    else:
        ok, bbox = tracker.update(frame)
        (x,y,w,h) = bbox
        if ok:
            cv2.rectangle(img_draw, (int(x), int(y)), (int(x + w),
int(y + h)), (0,255,0), 2, 1)
        else :
            cv2.putText(img_draw, "Tracking fail.", (100,80),
              cv2.FONT_HERSHEY_SIMPLEX,
              0.75,(0,0,255),2,cv2.LINE_AA)
                  trackerName = tracker.__class__.__name__
                  cv2.putText(img_draw, str(trackerIdx) +
              ":"+trackerName , (100,20), cv2.FONT_HERSHEY_SIMPLEX,
              0.75, (0,255,0),2,cv2.LINE_AA)
    cv2.imshow(win_name, img_draw)
    key = cv2.waitKey(delay) & 0xff
    if key == ord(' ') or (video_src != 0 and isFirst):
        isFirst = False
        roi = cv2.selectROI(win_name, frame, False)
        if roi[2] and roi[3]:
            tracker = trackers[trackerIdx]()
            isInit = tracker.init(frame, roi)
```

❖ Tracking API

- Example <3/3>

```
   elif key in range(48, 56):
        trackerIdx = key-48
        if bbox is not None:
            tracker = trackers[trackerIdx]()
            isInit = tracker.init(frame, bbox)
    elif key == 27 :
        break
else:
    print( "Could not open video")
cap.release()
cv2.destroyAllWindows()
```

- Example <Result>

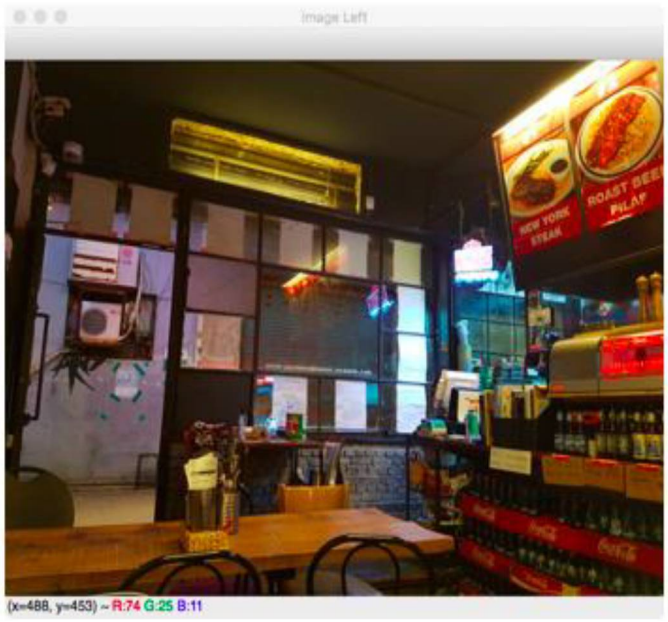


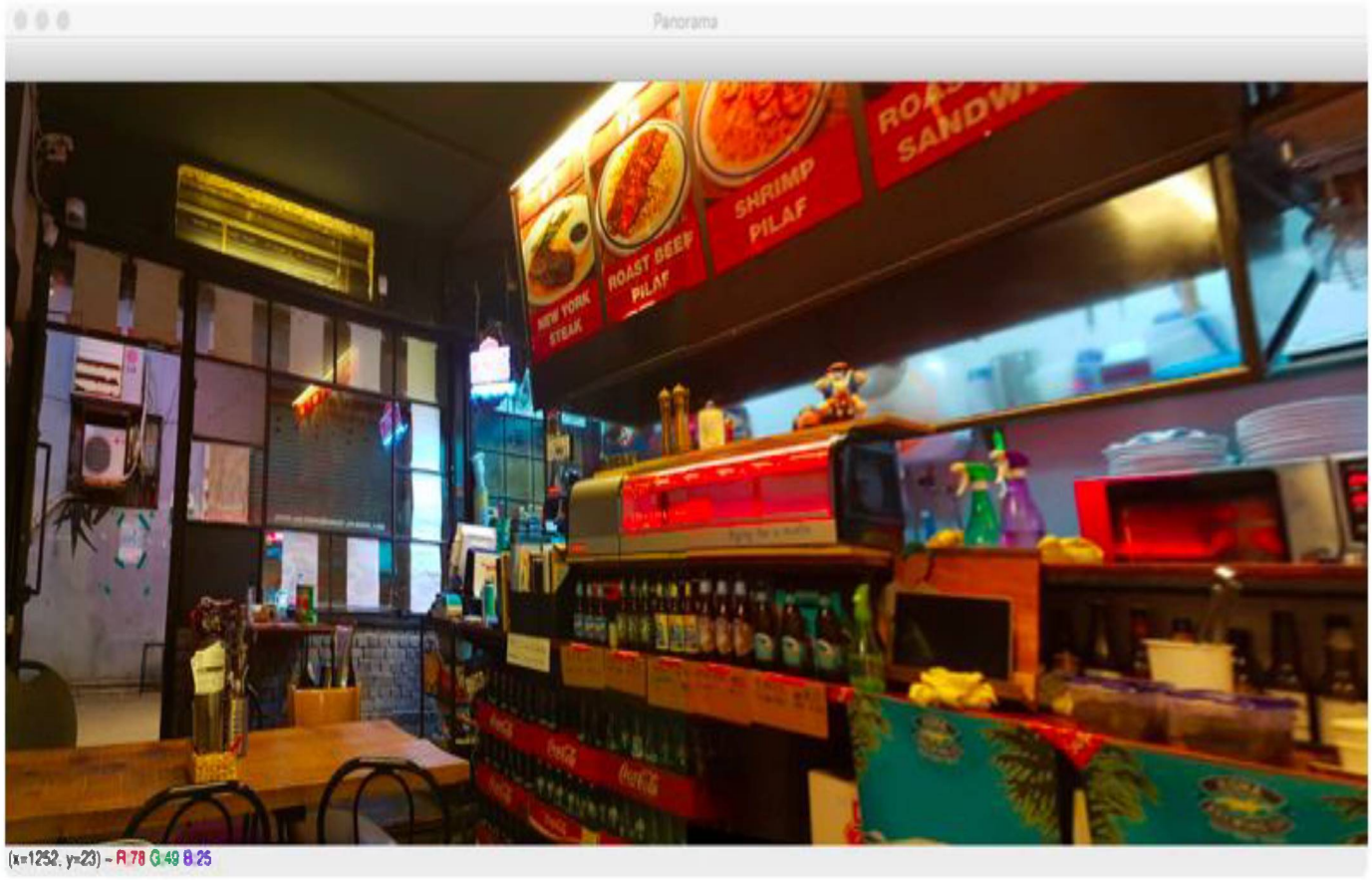
Matching & Tracking

1. Matching with a similar image
2. Feature and Key Point
3. Descriptor Extractor
4. Feature Matching
5. Tracking
6. **Workshop**

❖ Panorama Picture

- Draft a program that makes panorama images with two consecutive photoes.
- Result Example





❖ Panorama Picture

- Hint
  - Decide left and right connecting order, detect features and narrator
  - Calculate distance transform array after matching left picture to right picture
  - cv2.warpPerspective(): distance transform
  - Compound images in each position after generating sizes as big as two images

❖ Book cover Searcher

- Place book cover that will be searched by camera, and print original image by clicking space bar
- Result Example

- Result Example

❖ Book cover Searcher

- Hint
  - Mark square area of camera
  - Detect feature narrator after setting ROI in square area by clicking space bar on top of book cover
  - Read objects directory file one-by-one and detect feature and narrator of each objects
  - Detect search image and matching and good matching
  - Save normal ration by executing distance ration operation
  - After all matching is done, print the highest result after arranging the result from the highest normalization ratio

Machine Learning

1. **Meachine Learning**
2. K-Means Clustering
3. k-NN
4. SVM and HOG
5. Cascade Classifier
6. Workshop

AI

- In 1956 Dartmouth University in USA, Prof. John McCarthy first used the word AI at a conference
- General AI: AI thinking like human
- Narrow AI: AI only performing certain function i.e.) Sorting images, recognizing faces etc.

Machine Learning

- Analyze data with algorithms, learn the analyzed data and judge or predict with the learned details
- Various Algorithm

Deep Learning

- One algorithm of machine learning
- Artificial neural network: it is a network inspired by how human brain operates; it has massive inspiration to calculate hence it requires massive time and high performance devices.
- In 2012, Prof. Andrew NG of Google and Stanford realized deep neural network.

Machine Learning

- Through preconditioning process of data, transfer the data into features and build model through produced feature vector.
- Work analyzing characteristics of various algorithms, weights, threshold and other parameter via models to achieve target goals.
- Of the data set, 90% is for training set and 10% for test set

Supervised Learning

- Learning with data with label (answer)
- classification: name to label
- Regression: numerical label
- Linear regression: Score to study time (0~100)
  - H(x) = Wx + b, H: hypothesis, W: weight, b:bias
- Binary classification: Pass/Fail to study time
- Multi-labeled classification: GPA to study time (A,B,C,D,F)

❖ Unsupervised Learning

- Learning without label
- Clustering: Concern only in which data go to which group

❖ Realizing OpenCV

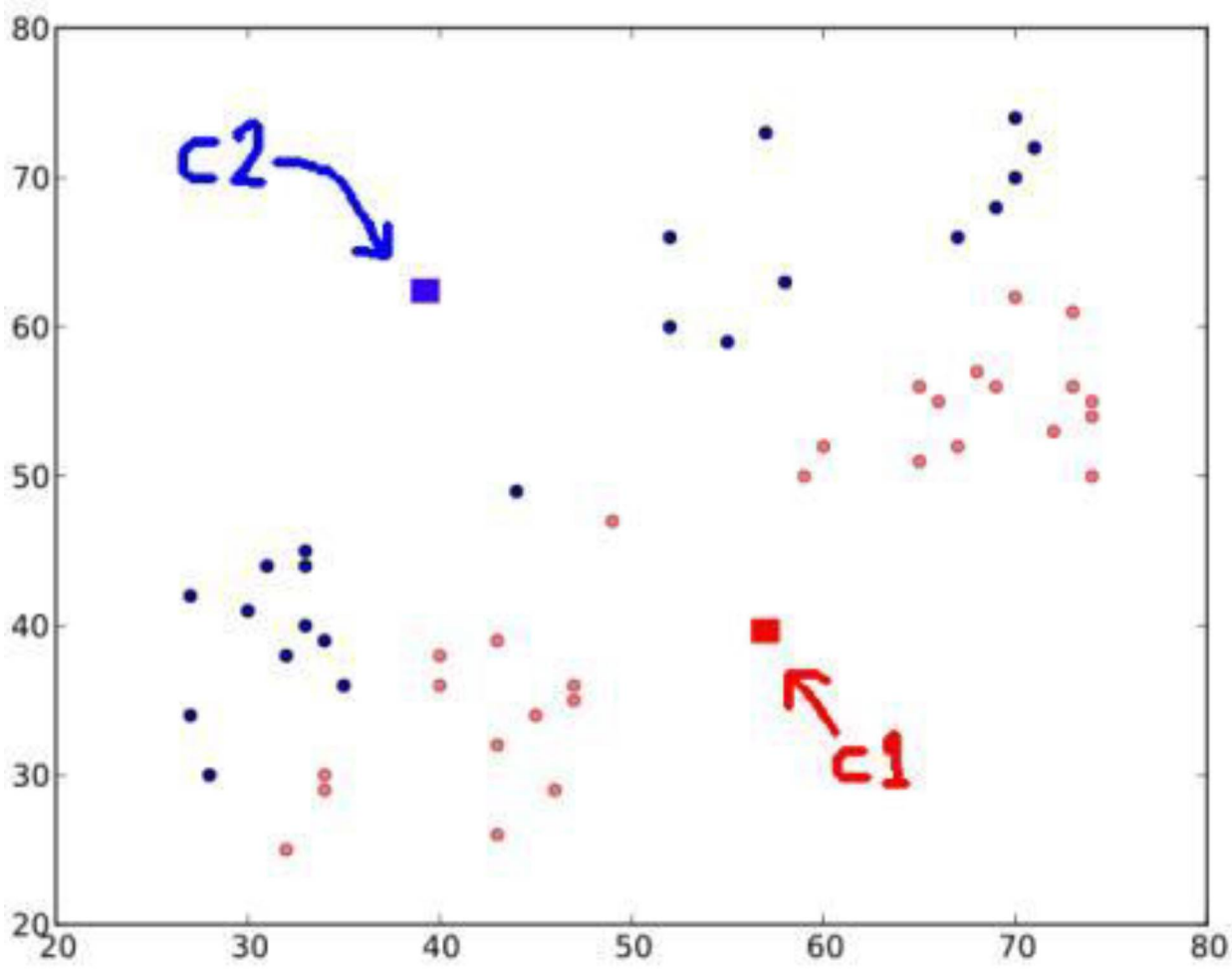
- Legacy Routine: Realize even before the standardized interface design
  - core Module
    - k-Means, Mahalanobis
- Standardized Interface
  - ml Module, Run StatModel Abstract Class Inheritance
    - Normal Bayes(Normal/Pure size Bayes divider)
    - Decision trees
    - Boosting
    - Random tree
    - Expected Value - maximize
    - k-Nearest Neighbor
    - Mumltilayer Perceptron (MLP, Multilayer perceptron)
    - Suport Vector Machine(SVM)
- Objective Detector
  - Upper level class using learning method
    - CascadeClassifier, Latent SVM(DPMDetector), Bag of Word

Machine Learning

1. Meachine Learning
2. **K-Means Clustering**
3. k-NN
4. SVM and HOG
5. Cascade Classifier
6. Workshop

❖ K-Means Clustering

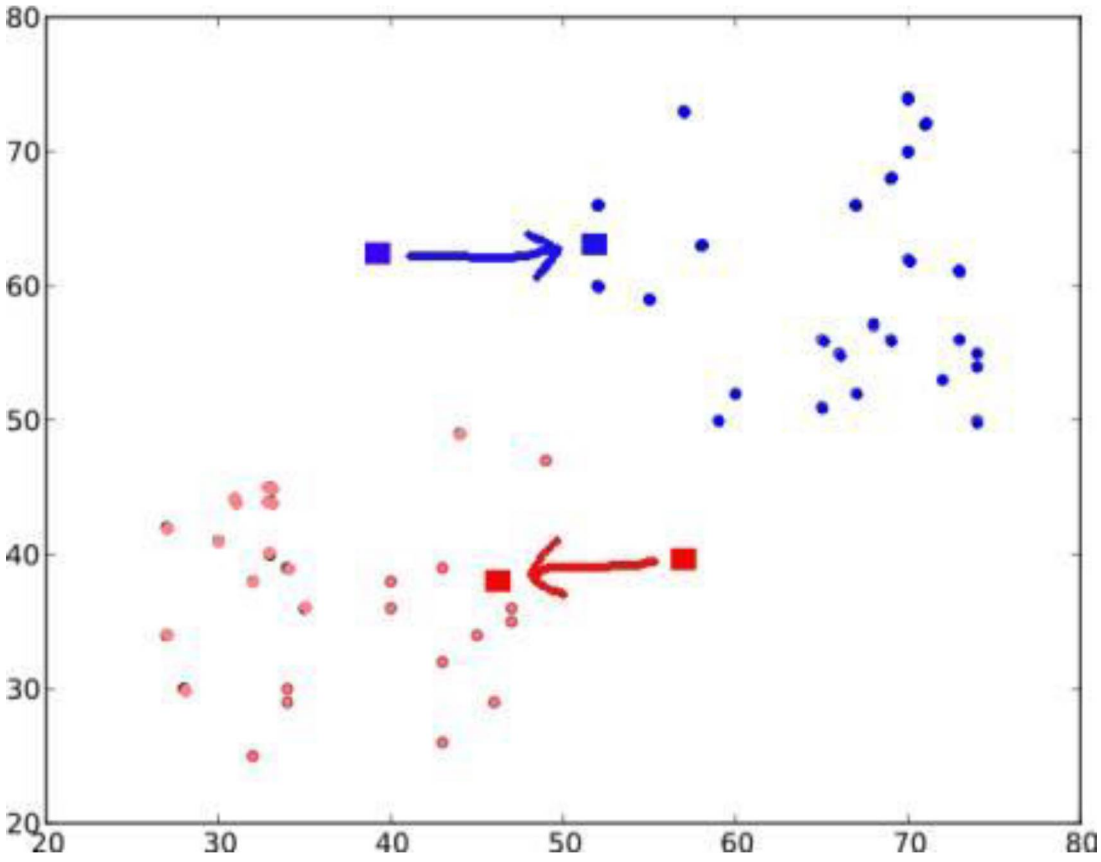
- Clustered algorithm, non-guide learning
- Cluster mixed data as wanted
- Algorithm
  - Step1.
    - Set two center points(C1, C2) randomly.
  - Step2.
    - Calculate distance at the two center points.
    - If test data is close to C1, 0, close to C2, 1.
    - If center point is C3, C4, mark as 2,3.
  - Step3
    - Picture mark 0 data close to C1 as red, and 1 data as blue.



❖ K-Means Clustering

- Algorithm
  - Step4.
    - Average red and blue, set is as new center point and as in step 2, calculate distance and mark as 0, 1
    - Repeat step 2 to 4, until each center point is fixed in as center point.
    - Or, repeat it until it satisfies certain criteria; set given # of repetition or accuracy.
    - Center point is dots with smallest value of distance of data added.

- There are various algorithm of getting first center points, repeating the calculation and etc.



$$minimize\left[J = \sum_{All\ Red\_Points} distance(C1, Red\_Point) + \sum_{All\ Blue\_Points} distance(C2, Blue\_Point)\right]$$

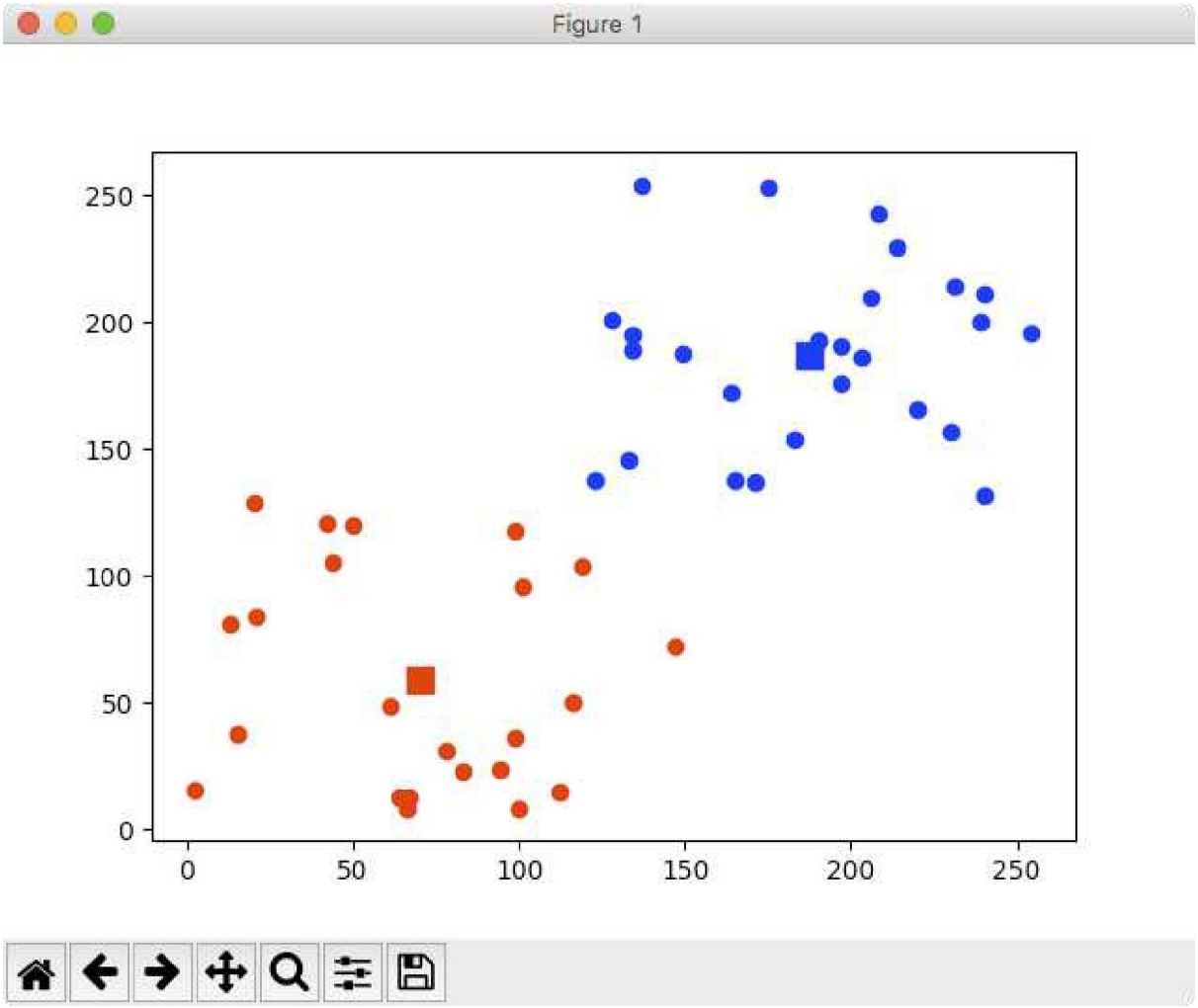
K-Means Clustering

- cv2.kmeans(data, K, bestLabels, criteria, attempts, flags) : retval, bestLabels, centers
  - data : np.float32, should be saved at 1column
  - k : # of cluster wanted
  - bestLabels : Result data, None
  - criteria : termination criteria, tuple(type, max_iter, epsilon)
  - attempts : # of run to be done through different initial label
  - flags : Selecting initial center points,
    - cv2.KMEASNS_PP_CENTERS
    - cv2.KMEASNS_RANDOM_CENTERS
    - cv2.KMEANS_USE_INITIAL_LABELS
- Turnaround:
  - retval : Square of center points and each data distance addition,
  - bestLabels : Labels, 0, 1, ...
  - centers : Array of cluster's center point
- Cluster of random number example

```
import numpy as np, cv2
import matplotlib.pyplot as plt

x = np.random.randint(0,150,(25,2))
y = np.random.randint(128, 255,(25,2))
z = np.vstack((x,y)).astype(np.float32)

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10,
1.0)
ret,label,center=cv2.kmeans(z,2,None,criteria,10,cv2.KMEANS_RANDOM_
CENTERS)

a = z[label.ravel()==0]
b = z[label.ravel()==1]
plt.scatter(a[:,0],a[:,1], c='b')
plt.scatter(b[:,0],b[:,1], c='r')
plt.scatter(center[:,0],center[:,1],s = 80,c = 'y', marker = 's')
plt.show()
```

❖ K-Means Clustering

- Cluster of random number example <result>



❖ K-Means Clustering

- 사진속 영상 색상 군집화 Example

```
import numpy as np
import cv2
img = cv2.imread('../img/taekwonv1.jpg')

z = img.reshape((-1,3))
z = np.float32(z)

criteria=(cv2.TERM_CRITERIA_EPS +cv2.TERM_CRITERIA_MAX_ITER,10,1.0)
K = 8
ret,label,center=cv2.kmeans(z,K,None,criteria,10,cv2.KMEANS_RANDOM_
CENTERS)
center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((img.shape))
merged = np.hstack((img, res2))
cv2.imshow('KMeans Color',merged)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

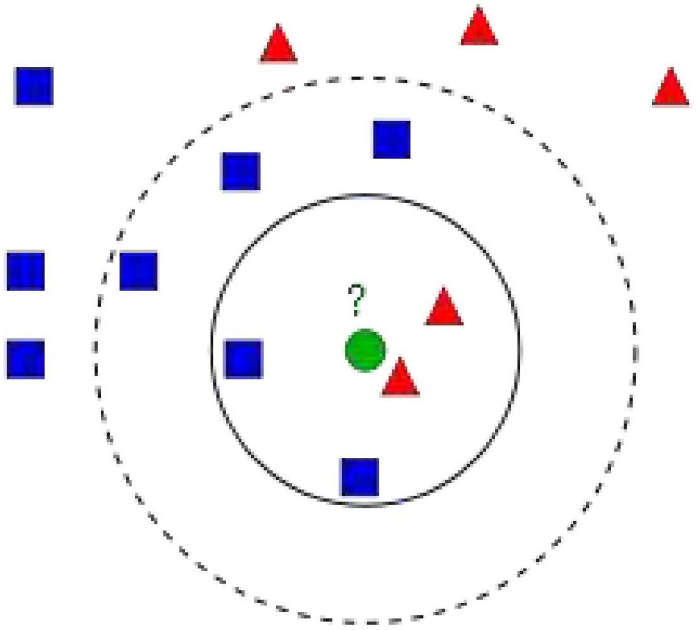
- 16 color clustering

Machine Learning

1. Meachine Learning
2. K-Means Clustering
3. **k-NN**
4. SVM and HOG
5. Cascade Classifier
6. Workshop

❖ k-NN(k-Nearest Neighbour)

- If new point is entered into exiting points that in divided into two groups, how the new points will be categorized?
- Categorizing to the closest member
- K is setting the scope of closest member
- You can give different weight to close member and farthest member.
- knn = cv2.ml.KNearest_create()
- knn.train(trainData, cv2.ml.ROW_SAMPLE, responses)
- ret, results, neighbours ,dist = knn.findNearest(newcomer, 3)



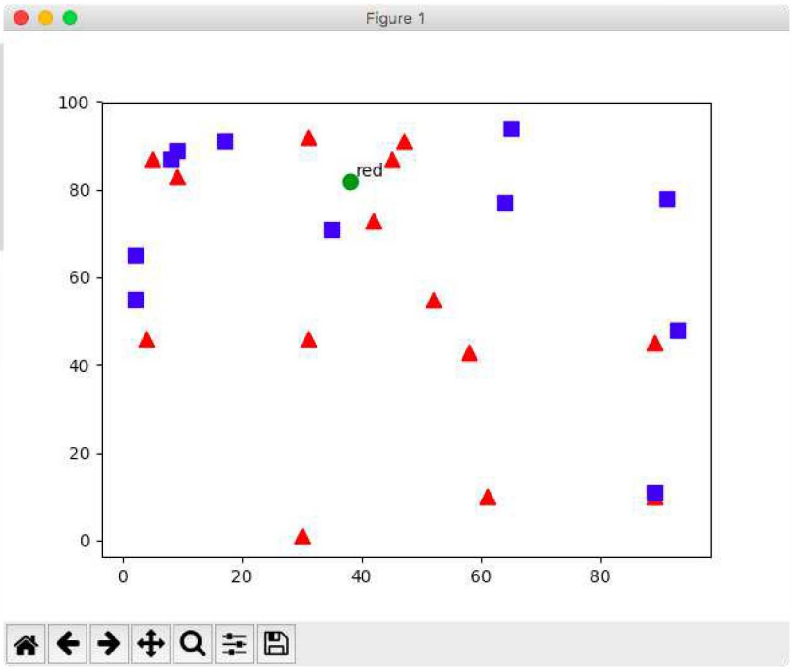
❖ k-NN(k-Nearest Neighbour)

- Simple Example

```
import cv2, numpy as np, matplotlib.pyplot as plt

trainData = np.random.randint(0,100,(25,2)).astype(np.float32)
responses = np.random.randint(0,2,(25,1))#.astype(np.float32)
red = trainData[responses.ravel()==0]
plt.scatter(red[:,0],red[:,1],80,'r','^')
blue = trainData[responses.ravel()==1]
plt.scatter(blue[:,0],blue[:,1],80,'b','s')
newcomer = np.random.randint(0,100,(1,2)).astype(np.float32)
plt.scatter(newcomer[:,0],newcomer[:,1],80,'g','o')
knn = cv2.ml.KNearest_create()
knn.train(trainData, cv2.ml.ROW_SAMPLE, responses)
#ret, res = knn.predict(newcomer)
ret, results, neighbours ,dist = knn.findNearest(newcomer, 3)#K=3
plt.annotate('red' if ret==0.0 else 'blue', xy=newcomer[0],
xytext=(newcomer[0]+1))
print( "result, 0=red, 1=blue:  {}\n".format(results) )
print( "neighbours:  {}\n".format(neighbours) )
print( "distance:  {}\n".format(dist) )
plt.show()
```

- Simple Example <result>

```
result, 0=red, 1=blue:  [[0.]]
neighbours:  [[0. 0. 1.]]
distance:  [[ 74.  97. 130.]]
```



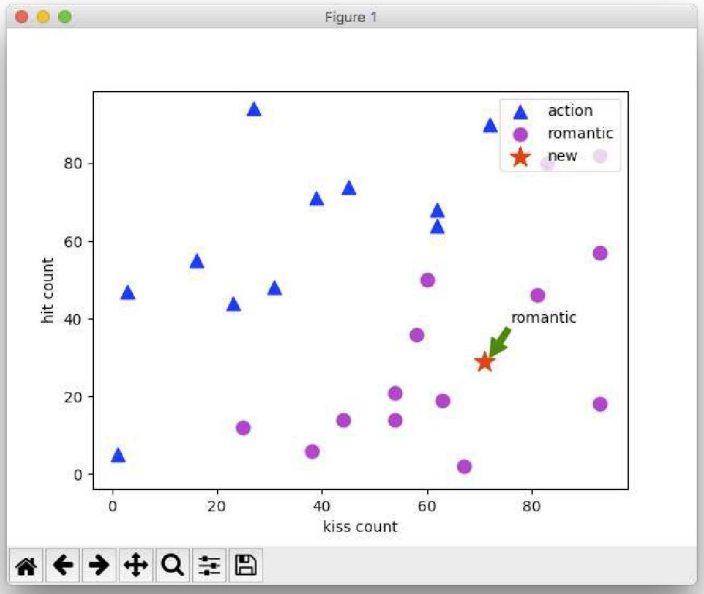
❖ k-NN(k-Nearest Neighbour)

- Move Categorizing Example

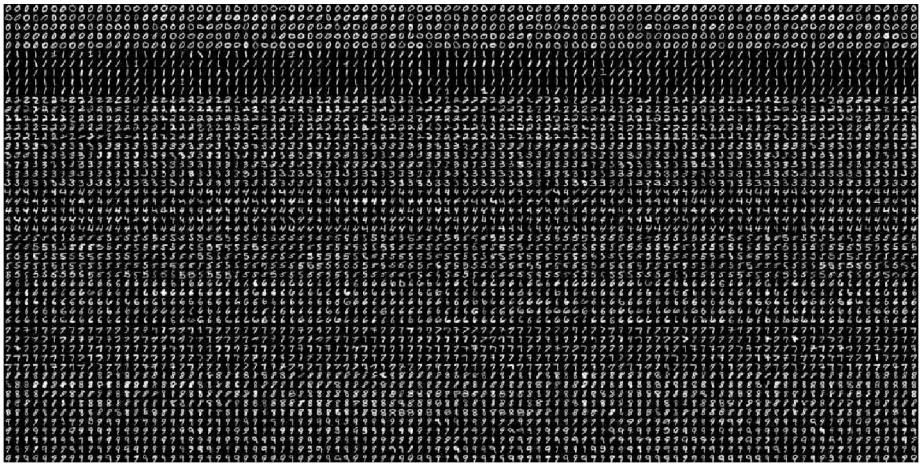
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

trainData = np.random.randint(0,100,(25,2)).astype(np.float32)
responses = (trainData[:, 0] >trainData[:,1]).astype(np.float32)

romantic = trainData[responses==1]
action = trainData[responses==0]

plt.scatter(action[:,0],action[:,1],s=80,marker='^', c="blue",
label='action')
plt.scatter(romantic[:,0],romantic[:,1], s=80, c='m', marker='o',
label="romantic")
newcomer = np.random.randint(0,100,(1,2)).astype(np.float32)
plt.scatter(newcomer[:,0],newcomer[:,1],200,'r','*', label="new")
knn = cv2.ml.KNearest_create()
knn.train(trainData, cv2.ml.ROW_SAMPLE, responses)
ret, results, neighbours ,dist = knn.findNearest(newcomer, 3)#K=3

print( "result(0=action, 1:romantic):  {}\n".format(results) )
print( "neighbours:  {}\n".format(neighbours) )
print( "distance:  {}\n".format(dist) )
label = results == 1 and "romantic" or "action"

anno_x, anno_y = newcomer.ravel()
plt.annotate(label, xy=(anno_x + 1, anno_y+1), xytext=(anno_x+5,
anno_y+10), arrowprops={'color':'green'})
plt.xlabel('kiss count')
plt.ylabel('hit count')
plt.legend(loc="upper right")
plt.show()
```

❖ k-NN(k-Nearest Neighbour)

- Move Categorizing Example <Result>



- Recognize Handwriting
  - 5000 letters of handwritten images
  - 500 letters per number (0 ~ 9)
  - 20 x 20=400 pixels by each letter
  - 5000 X 400 Vector
    - 4500 : train set
    - 500 : test set



❖ k-NN(k-Nearest Neighbour)

- Recognize Handwriting Training

```
import numpy as np, cv2

def load():
    image = cv2.imread('../img/digits.png')
    gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

    cells = [np.hsplit(row,100) for row in np.vsplit(gray,50)]
    x = np.array(cells)

    train = x[:,:90]
    test = x[:,90:100]
    train = train.reshape(-1,400).astype(np.float32) # Size =
(4500,400)
    test = test.reshape(-1,400).astype(np.float32) # Size =
(500,400)

    k = [0,1,2,3,4,5,6,7,8,9]
    train_labels = np.repeat(k,450).reshape(-1,1)
    test_labels = np.repeat(k,50).reshape(-1,1)
    return (train, train_labels, test, test_labels)

if __name__ == '__main__':
    train, train_labels, test, test_labels = load()
    knn = cv2.ml.KNearest_create()
    knn.train(train, cv2.ml.ROW_SAMPLE, train_labels)

    #ret, result = knn.predict(test, k=3)
    ret, result, neighbors, distance = knn.findNearest(test, k=3)

    correct = np.sum(result == test_labels)
    accuracy = correct * (100.0 / result.size)
    print("correct : %d/%d , Accuracy :%.2f%%" % (correct,
result.size, accuracy) )
    #knn.save('digits.yaml') # but knn.load(..) is not implemented
```

❖ k-NN(k-Nearest Neighbour)

- Recognize Handwriting Training

```
correct : 475/500 , Accuracy :95.00%
```

- Recognize Handwriting <1/3>

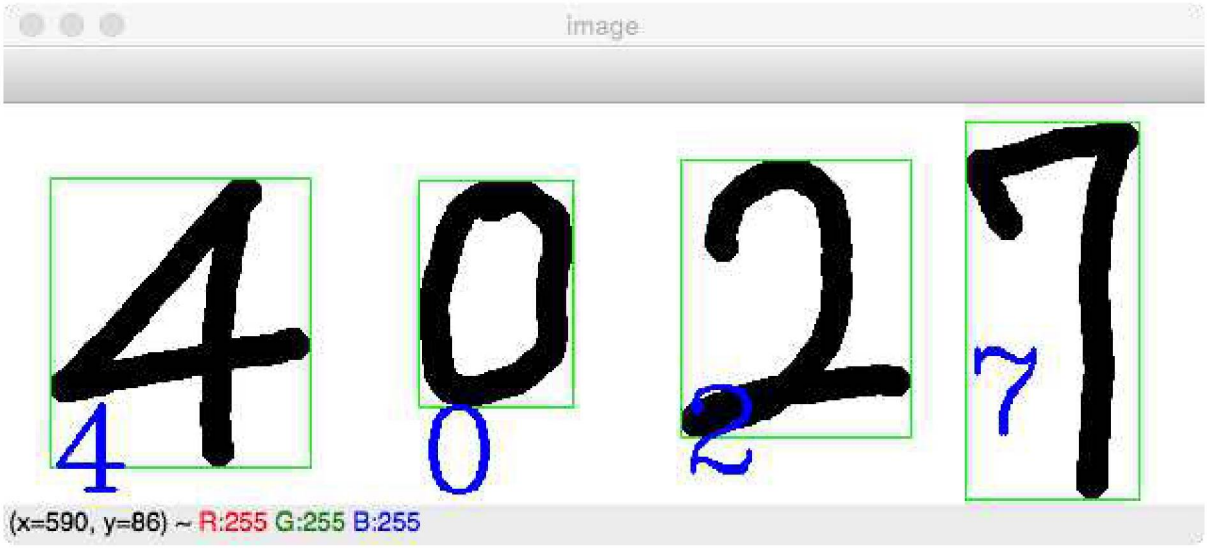
```
import numpy as np, cv2
import knn_mnist_train

def flatten_img(src):
    h, w = src.shape[:2]
    square = src
    if h > w:
        pad = (h - w)//2
        square = np.zeros((h, h), dtype=np.uint8)
        square[:, pad:pad+w] = src
    elif w > h :
        pad = (w - h)//2
        square = np.zeros((w, w), dtype=np.uint8)
        square[pad:pad+h, :] = src
    px20 = np.zeros((20,20), np.uint8)
    px20[2:18, 2:18] = cv2.resize(square, (16,16),
interpolation=cv2.INTER_AREA)
    flatten = px20.reshape((1,400)).astype(np.float32)
    return flatten
knn = cv2.ml.KNearest_create()
train, train_labels = knn_mnist_train.load()[:2]
knn.train(train, cv2.ml.ROW_SAMPLE, train_labels)
image = cv2.imread('../img/4027.png')
cv2.imshow("image", image)
cv2.waitKey(0)
```

❖ k-NN(k-Nearest Neighbour)

- Recognize Handwriting <2/3>

```
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (5, 5), 0)
_, gray = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV)
img, contours, _ = cv2.findContours(gray, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
for c in contours:
    (x, y, w, h) = cv2.boundingRect(c)
    if w >= 5 and h >= 25:
        roi = gray[y:y + h, x:x + w]
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 1)
        flatten = flatten_img(roi)
        ret, result, neighbours, dist = knn.findNearest(flatten, k=1)
        cv2.putText(image, "%d"%ret, (x , y + 155),
cv2.FONT_HERSHEY_COMPLEX, 2, (255, 0, 0), 2)
        cv2.imshow("image", image)
        cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Recognize Handwriting <Result>

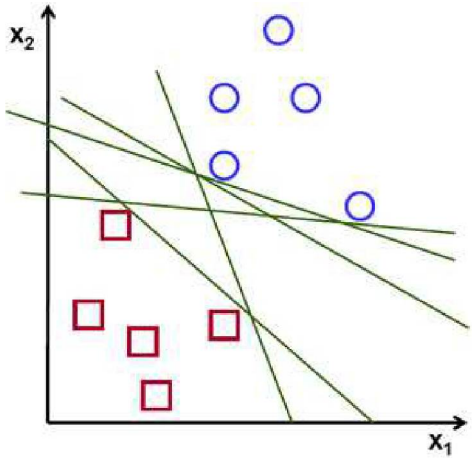


Machine Learning

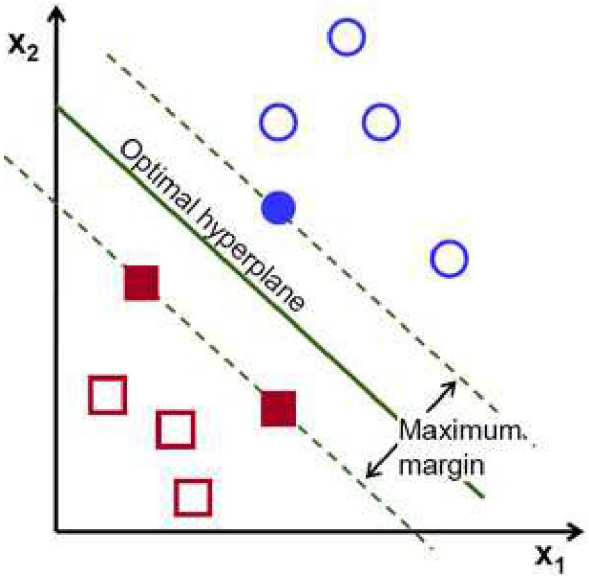
1. Meachine Learning
2. K-Means Clustering
3. k-NN
4. **SVM and HOG**
5. Cascade Classifier
6. Workshop

❖ Support Vector Machine

- One of the classifications
- k-NN needs to calculate all data distance, it takes a lot of time and memory
- Distinguish parameters, find lines and recognizing patterns based on that
  - SVM,NuSVM, LinearSVM (Specialized in linear kernel)
- Lines that can be categorized into two is called Decision Boundary, and the data categorized is called Linear Separable



- Find line first in all SVM parameters because it is strong in noise
- No need training data; if it is close to the counter category, it is enough
- Support Vectors: Blue circle filled and read quadrangle filled
- Support Planes: Line that passes
- Blue Circle: Wx + b > 1
- Red Data: Wx + b < -1
  - W : weight, vector decides orientation
  - b: bias, Decide location
- Decision Boundary :
  - Crossing center point of two hyperplane
  - Wx +b = 0



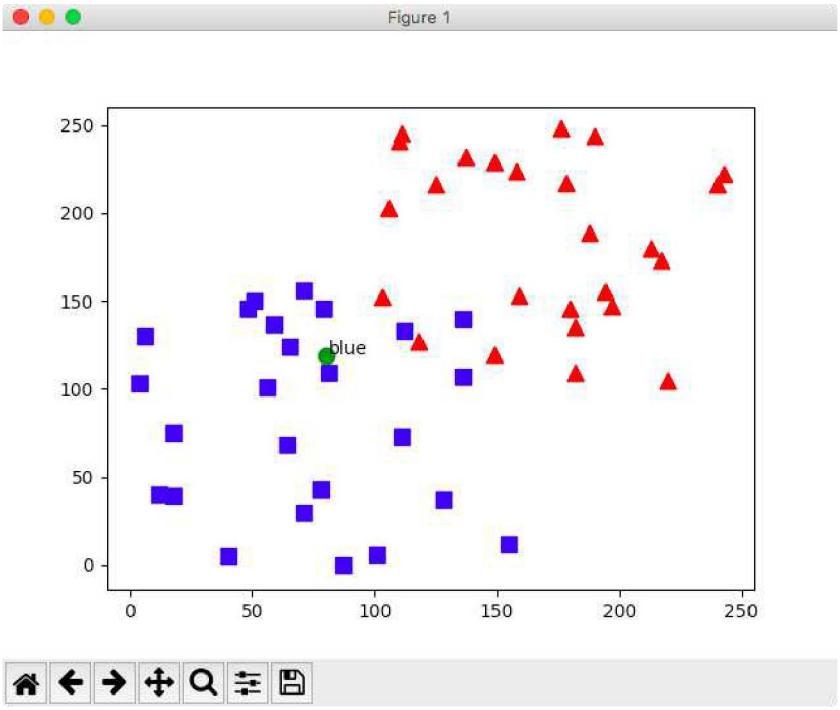
SVM

- Example <1/2>

```
import cv2
import numpy as np
import matplotlib.pylab as plt

x = np.random.randint(0,158,(25,2))
y = np.random.randint(98, 255,(25,2))
trainData = np.vstack((x,y))
trainData = np.float32(trainData)

responses = np.zeros((50,1), np.int32)
responses[0:25] = 1

red = trainData[responses.ravel()==0]
plt.scatter(red[:,0],red[:,1],80,'r','^')
blue = trainData[responses.ravel()==1]
plt.scatter(blue[:,0],blue[:,1],80,'b','s')

newcomer = np.random.randint(0,255,(1,2)).astype(np.float32)
plt.scatter(newcomer[:,0],newcomer[:,1],80,'g','o')

C=1
model = cv2.ml.SVM_create()
model.trainAuto(trainData, cv2.ml.ROW_SAMPLE, responses)

ret, results = model.predict(newcomer)
plt.annotate('red' if results[0]==0.0 else 'blue', xy=newcomer[0],
xytext=(newcomer[0]+1))
print("result, 0=red, 1=blue:  {}\n".format(results))

plt.show()
```

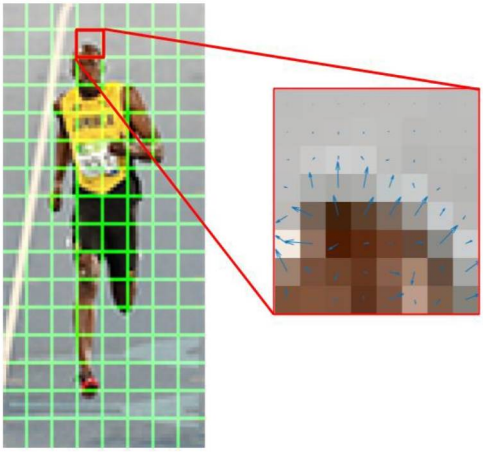
❖ SVM

- Example <Result>



❖ HOG

- Histogram of Oriented Gradient Feature Descriptor
- Histogram for edge slope orientation
- Suggested to recognize pedestrian
- Normalized Histogram

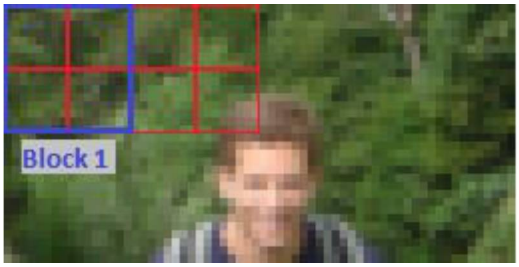


2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction



- hogDesc = cv2.HOGDescriptor(winSize,blockSize,blockStride,cellSize,nbins)
  - windsize : Size of HOG extraction area
  - blockSize : Size of normalization area
  - blockStride : Size of normalization overlapping block
  - cellSize : Size of HOG calculation
  - nbins : # of histogram level
  - hodDesc : HOG specialized narrator
    - Vector Size :

$$\text{nbins} \times (\frac{\text{blockSize}}{\text{cellSize}})^2 \times (\frac{(\text{winSize} - \text{blockSize})}{\text{blockStride}} + 1)^2$$

❖ HOG SVM

- MNIST Handwriting Study <1/3>

```
import cv2 as cv
import numpy as np

winSize = (20,20)
blockSize = (10,10)#blockSize = (8,8)
blockStride = (5,5)#blockStride = (4,4)#half of blocSize
cellSize = (10,10)#cellSize = (8,8)
nbins = 9
hogDesc =
cv.HOGDescriptor(winSize,blockSize,blockStride,cellSize,nbins)

affine_flags = cv.WARP_INVERSE_MAP|cv.INTER_LINEAR
def deskew(img):
    m = cv.moments(img)
    if abs(m['mu02']) < 1e-2:
        return img.copy()
    skew = m['mu11']/m['mu02']
    M = np.float32([[1, skew, -0.5*20*skew], [0, 1, 0]])
    img = cv.warpAffine(img,M,(20, 20),flags=affine_flags)
    return img
```

HOG SVM

- MNIST Handwriting Study <2/3>

```
if __name__ == '__main__':
    img = cv.imread('../img/digits.png',0)
    cells = np.array([np.hsplit(row,100) for row in
np.vsplit(img,50)])
    train_cells = cells[:, :90]
    test_cells = cells[:, 90:100]

    deskewed = [list(map(deskew,row)) for row in train_cells]
    hogdata = [list(map(hogDesc.compute,row)) for row in deskewed]
    trainData = np.float32(hogdata)
    print(trainData.shape)
    trainData = trainData.reshape(-1,trainData.shape[2])

    responses = np.repeat(range(10),450).reshape(-1, 1)
    svm = cv.ml.SVM_create()

     '''
    svm.setKernel(cv.ml.SVM_RBF)
    svm.setType(cv.ml.SVM_C_SVC)
    svm.setC(12.5)
    svm.setGamma(0.50625)
    svm.train(trainData, cv.ml.ROW_SAMPLE, responses)
    '''
    svm.trainAuto(trainData, cv.ml.ROW_SAMPLE, responses)
    svm.save('svm_data.yml')

    deskewed = [list(map(deskew,row)) for row in test_cells]
    hogdata = [list(map(hog2,row)) for row in deskewed]
    testData = np.float32(hogdata)
    testData = testData.reshape(-1,testData.shape[2])
    test_labels = np.repeat(range(10),50).reshape(-1,1)
    ret, result = svm.predict(testData)
    correct = (result==test_labels).sum()
    print(correct*100.0/result.size)
```

HOG SVM

- MNIST Handwriting Study <Result>

```
accuracy : 97.4%
```

- Recognize Handwriting <1/3>

```
import cv2
import numpy as np
import svm_mnist_train

#svm=ocr.svm
#svm = cv2.ml.SVM_create()
#svm = svm.load('./svm_data.yml')
svm = cv2.ml.SVM_load('./svm_data.yml')

def hog_img(src):
    h, w = src.shape[:2]
    square = src
    if h > w:
        pad = (h - w)//2
        square = np.zeros((h, h), dtype=np.uint8)
        square[:, pad:pad+w] = src
    elif w > h :
        pad = (w - h)//2
        square = np.zeros((w, w), dtype=np.uint8)
        square[pad:pad+h, :] = src
    px20 = np.zeros((20,20), np.uint8)
    px20[2:18, 2:18] = cv2.resize(square, (16,16),
interpolation=cv2.INTER_AREA)
```

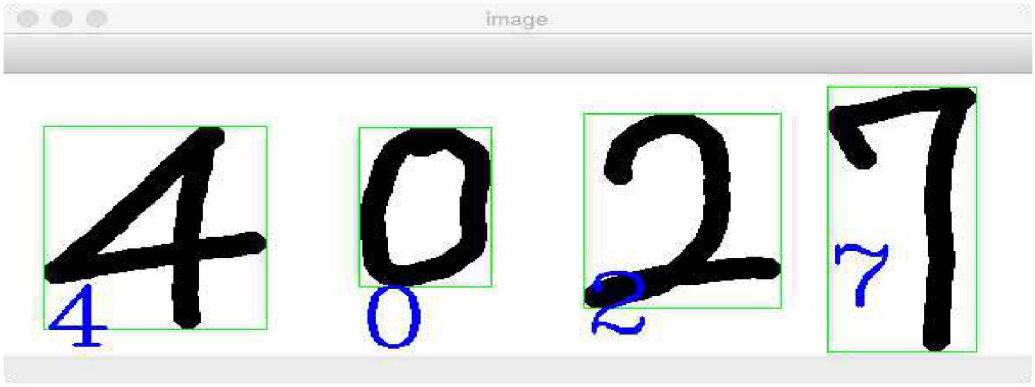
❖ HOG SVM

- Recognize Handwriting <2/3>

```
    deskewed = svm_mnist_train.deskew(px20)
    hogdata = svm_mnist_train.hogDesc.compute(deskewed)
    testData = np.float32(hogdata).reshape(-1, hogdata.shape[0])
    return testData

image = cv2.imread('../img/4027.png')
cv2.imshow("image", image)
cv2.waitKey(0)

gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (5, 5), 0)
_, gray = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV)
img, contours, _ = cv2.findContours(gray, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
for c in contours:
    (x, y, w, h) = cv2.boundingRect(c)
    if w >= 5 and h >= 25:
        roi = gray[y:y + h, x:x + w]
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0),
1)
        testData = hog_img(roi)
        ret, result = svm.predict(testData)
        cv2.putText(image, "%d"%result[0], (x , y + 155),
cv2.FONT_HERSHEY_COMPLEX, 2, (255, 0, 0), 2)
        cv2.imshow("image", image)
        cv2.waitKey(0)
cv2.destroyAllWindows()
```

❖ HOG SVM

- Recognize Handwriting <Result>



❖ Pedstrian Detecting

- HOG training object to detect pedestrian
  - cv2.HOGDescriptor_getDefaultPeopleDetector()
  - cv2.HOGDescriptor_getDaimlerPeopleDetector()
  - cv2.HOGDescriptor.setSVMDetector()
  - cv2.HOGDescriptor.detectMultiScale
- Detect Pedestrian

```
import cv2
hogdef = cv2.HOGDescriptor()
hogdef.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
cap = cv2.VideoCapture('../img/walking.avi')
while cap.isOpened():
    ret, img = cap.read()
    if ret is None:
        print('no frame')
        break
    found, _ = hogdef.detectMultiScale(img)
    for (x,y,w,h) in found:
        cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,255))
    cv2.imshow('frame', img)
    if cv2.waitKey(1) == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

❖ Pedstrian Detecting

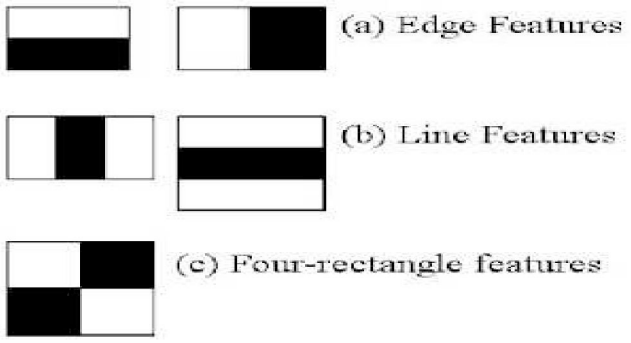
- Detect Pedestrian

Machine Learning

1. Meachine Learning
2. K-Means Clustering
3. k-NN
4. SVM and HOG
5. **Cascade Classifier**
6. Workshop

❖ Cascade

- Tree base object divider
- Effective in recognizing faces
- Use Haar Feature while adding or subtracting certain quadrangle
- Offer trainer and detector
- Detector: Recognize pre-trained object file
  - haarcascades.xml
- Trainer: New image training



❖ Cascade

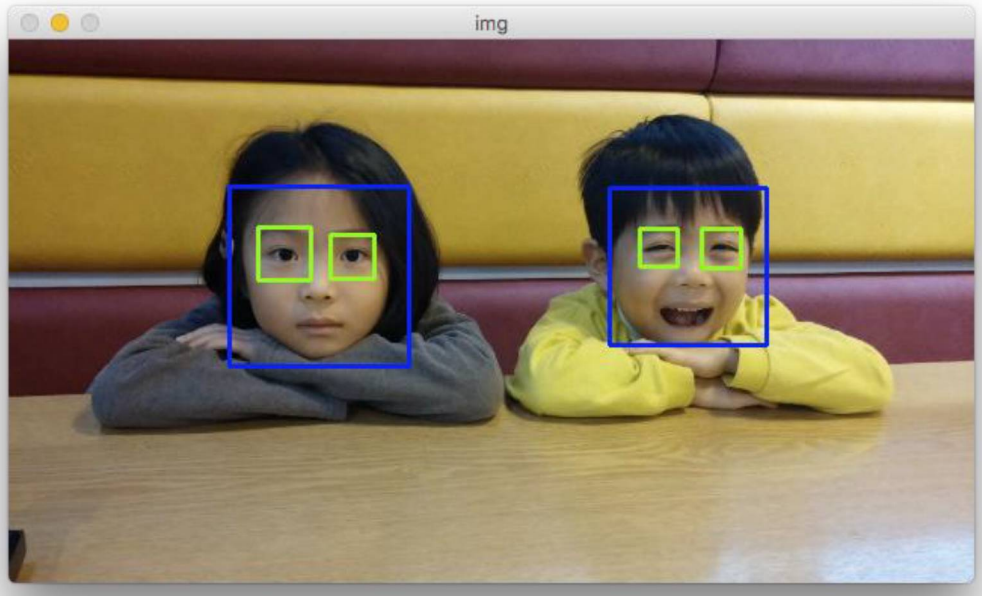
- classifier = cv2.CascadeClassifier('file.xml')
- classifier.detectMultiScale(img, scaleFactor, minNeighbors[, flags, minSize, maxSize)
  - img : Input image
  - scaleFactor: Limit image enlarging size 1.3~1.5
    - Use for scale pyramid; if value is big, chances for recognizing increases but it gets slow
  - minNeighbors: # of pixels to maintain
    - If the number is big, quality gets better, but # of detection decreases
  - flags : For old API; not used now
  - minSize, maxSize: Ignore detection if it goes over given size area

❖ Cascade

- Face Detection Example

```
import cv2
img = cv2.imread('../img/children.jpg')
face_cascade =
cv2.CascadeClassifier('../data/haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('../data/haarcascade_eye.xml')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray)
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),1)
cv2.imshow('children',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Face Detection Example <Result>

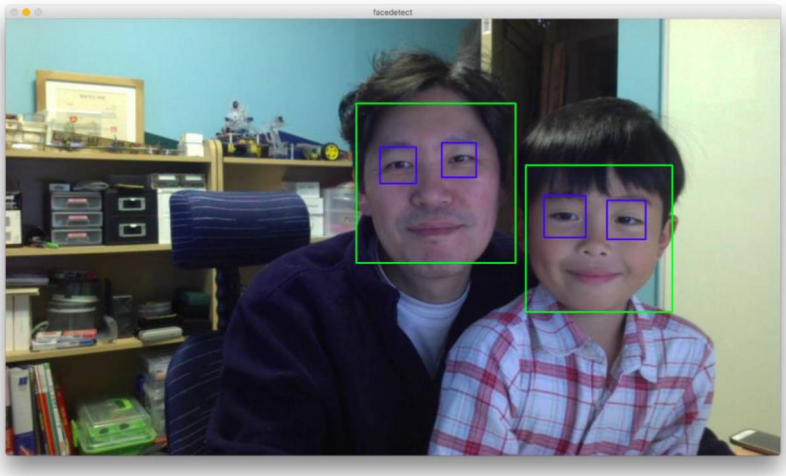


❖ Cascade

- Face Detection Example

```
cascade_xml = '../data/haarcascade_frontalface_default.xml'
cascade = cv2.CascadeClassifier(cascade_xml)
eye_cascade = cv2.CascadeClassifier('../data/haarcascade_eye.xml')
cam = cv2.VideoCapture(0)
while True:
    ret, img = cam.read()
    if not ret:
        print('no frame')
        break
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = cascade.detectMultiScale(gray, 1.3, 5)
    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w, y+h), (0, 255,0),2)
        roi = gray[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi)
        for(ex, ey, ew, eh) in eyes:
            cv2.rectangle(img[y:y+h, x:x+w], (ex,ey), (ex+ew,
ey+eh), (255,0,0),2  )
    cv2.imshow('facedetect', img)
    if 0xFF & cv2.waitKey(5) == 27:
        break
cv2.destroyAllWindows()
```

- Face Detection Example <Result>

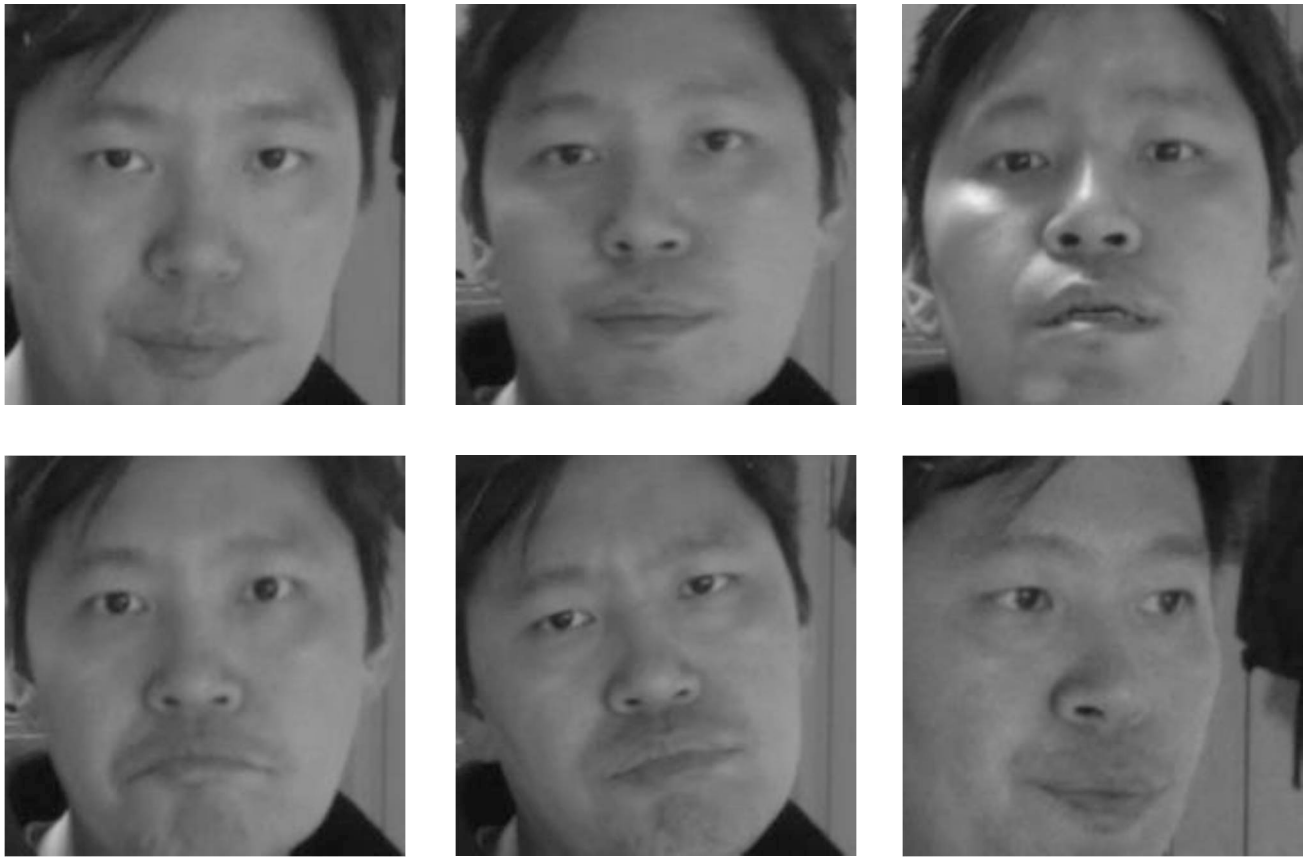


❖ Face Recognition

- Recognizing your own face
- Take your own photo in multiple angles
- Train with the pictures take
  - mode = cv2.face.LBPHFaceRecognizer_create()
  - model.train(train_data, label_data)
- Save the training result in xml
  - mode.write('file.xml')
- Read the saves xml to read faces
  - mode = cv2.face.LBPHFaceRecognizer_create()
  - Mode.read('file.xml')
- result : model.predict(face)
  - result : (label, confidence)
    - label : Label of most similar photo to the registered one
    - confidence : distance (0 : 100%, ∞ : 0%)

❖ Recognize Faces

- Take Photo
  - Input name and ID (in number)
  - Take 100 photos
  - Including front angle, take the photos in variety of angles
  - Various faces

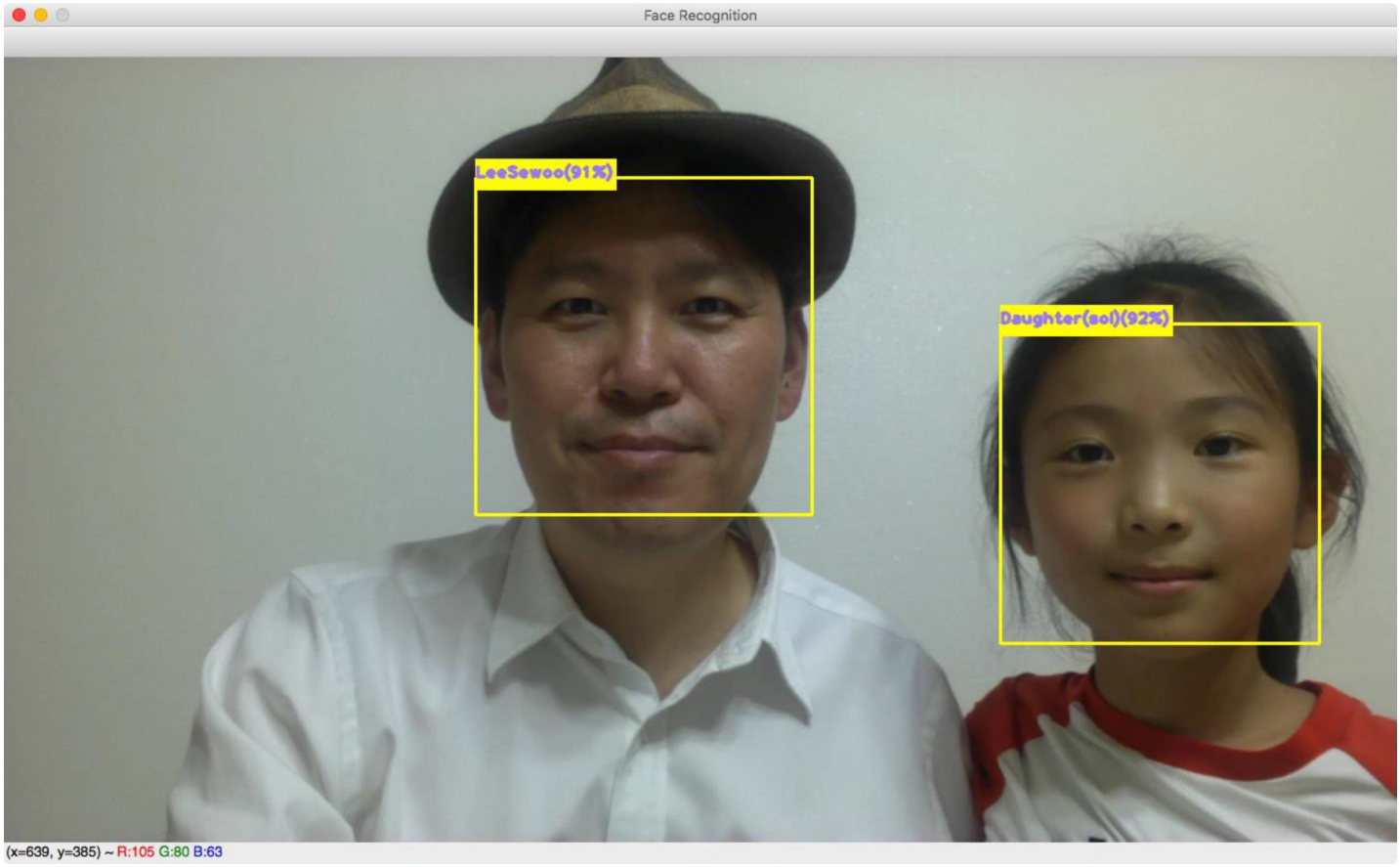


❖ Recognize Faces

- Training

❖ Recognize Faces

- Result



Machine Learning

1. Meachine Learning
2. K-Means Clustering
3. k-NN
4. SVM and HOG
5. Cascade Classifier
6. **Workshop**

❖ Face Mosaic

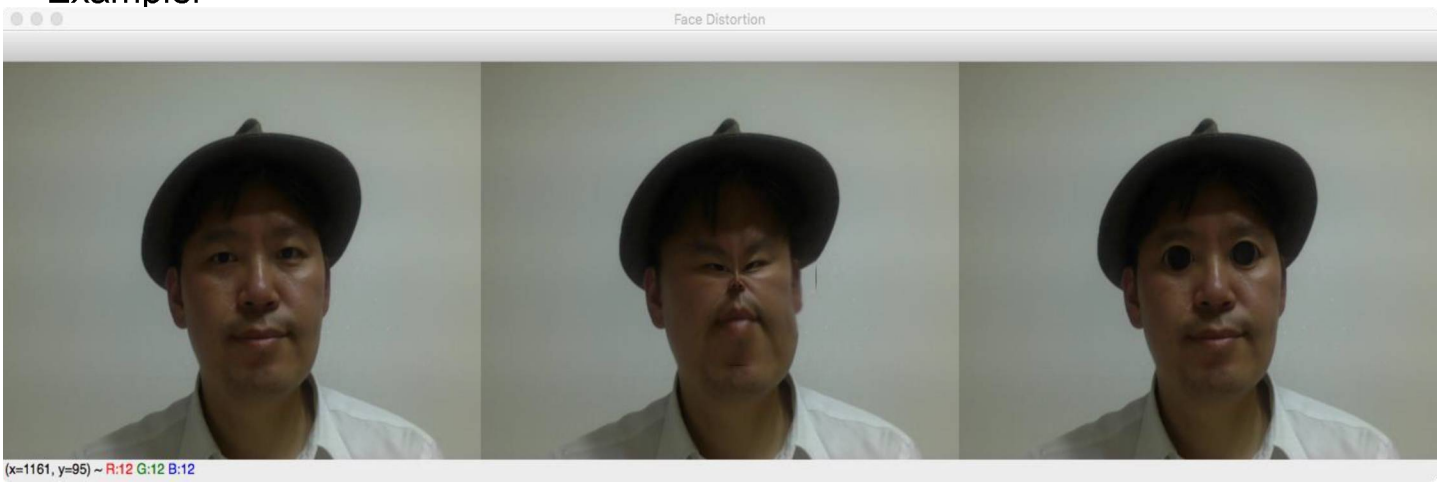
- Give auto mosaic effects to photos with human faces.
- Result Example:

❖ Hannibal Mask

- Place Hannibal bask from the Silence of The Lambs on a face taken from the camera
- Result Example

❖ Face Distortion

- Make a camera that distorts human face like Snapchat or Snow
- Example:



OpenCV3 with Python