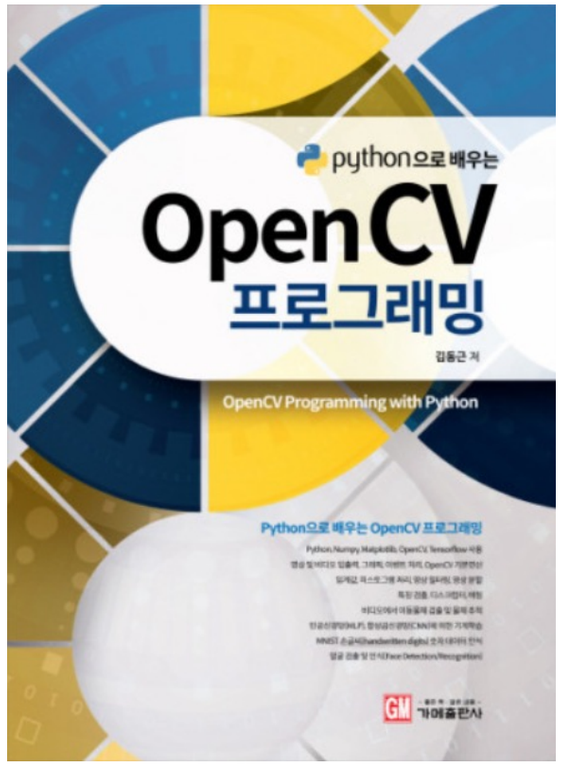
OPENCV를 이용한 영상처리

CONTENTS

- OpenCV라이브러리의 기본 이해
- 기본적인 영상 처리 기법
- 이미지 도형 처리
- 이미지 File I/O
- 기하학적 변환, 영상의 특징 추출
- 영상의 측정 객체 검출 과 필터링
- 특징점 검출과 매칭, 객체 추적과 모션 벡터

컴퓨터 비전

- OpenCV를 처음 배우시는 분이시라면 Python으로 OpenCV를 배우셔도 좋을듯 합니다.
- python으로 배우는 OpenCV프로그래밍, 김동근저
- OpenCV4로 배우는 컴퓨터 비전과 머신 러닝, 황선규저

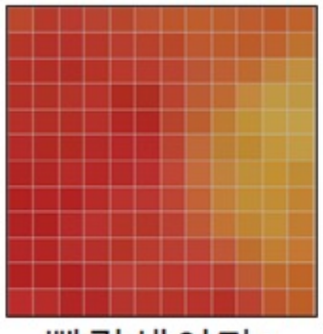


컴퓨터 비전

- 컴퓨터를 이용하여 정지 영상 또는 동영상으로부터 의미 있는 정보를 추출하는 방법을 연구하는 학문
- 사람이 눈으로 사물을 보고 인지하는 작업을 컴퓨터가 수행하게 만드는 기술

사과?

둥글다?



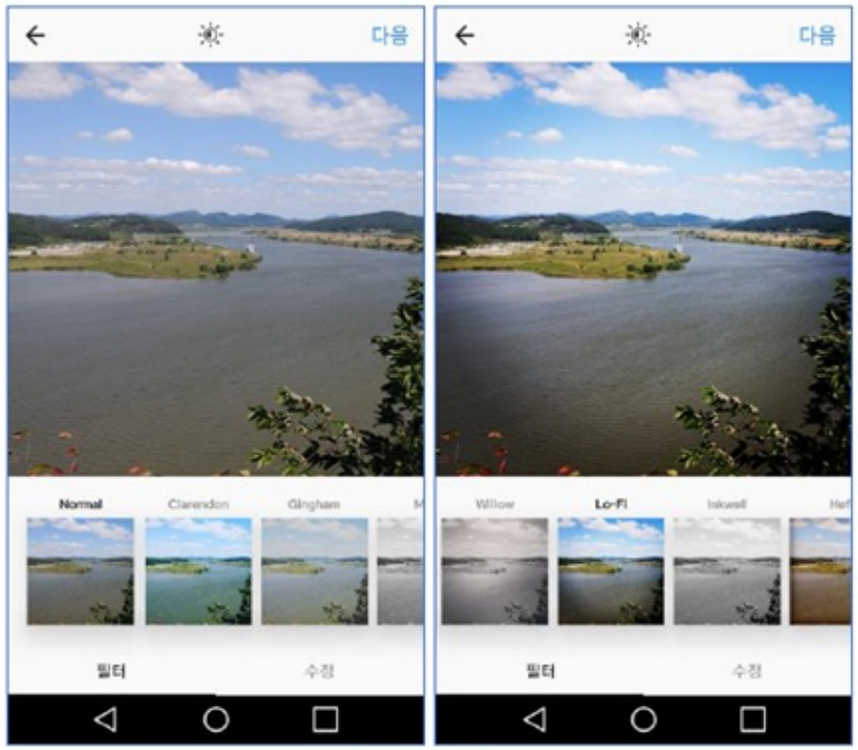
빨간색이다?

꼭지 모양?

사과가 몇 개??

컴퓨터 비전 연구 분야

- 영상의 화질 개선



Filtering App

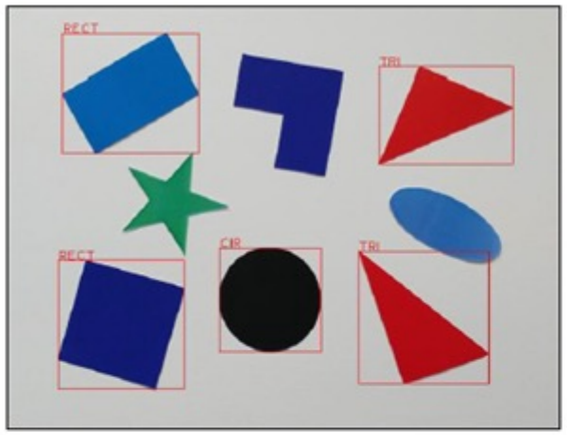
HDR

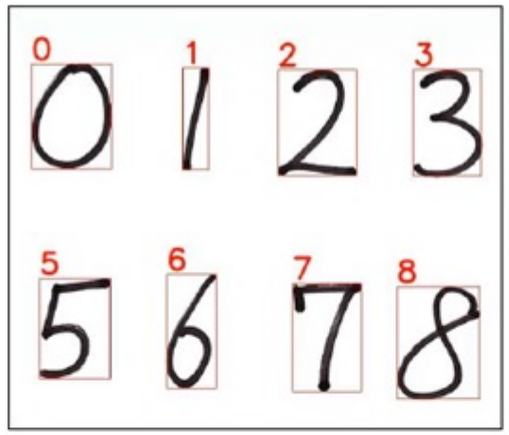
Image Noise Reduction

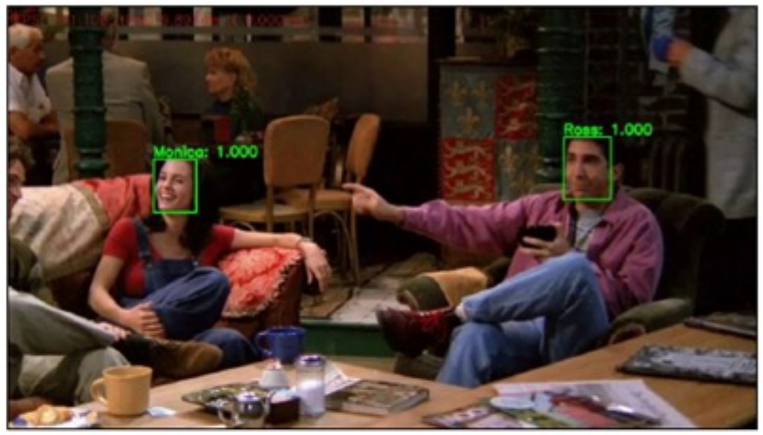
Super Resolution

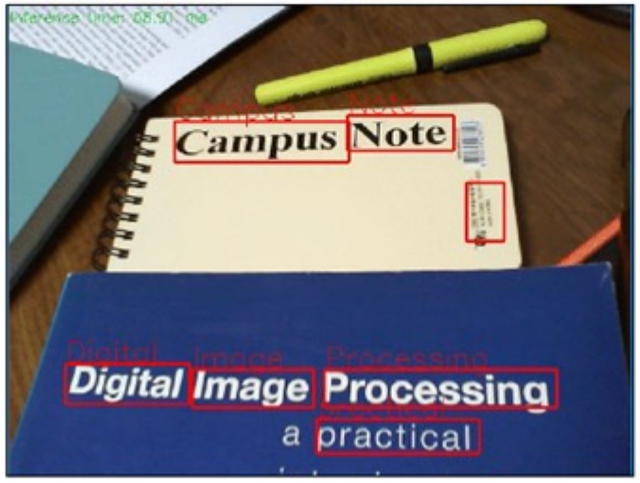
컴퓨터 비전 연구 분야

- 영상 인식









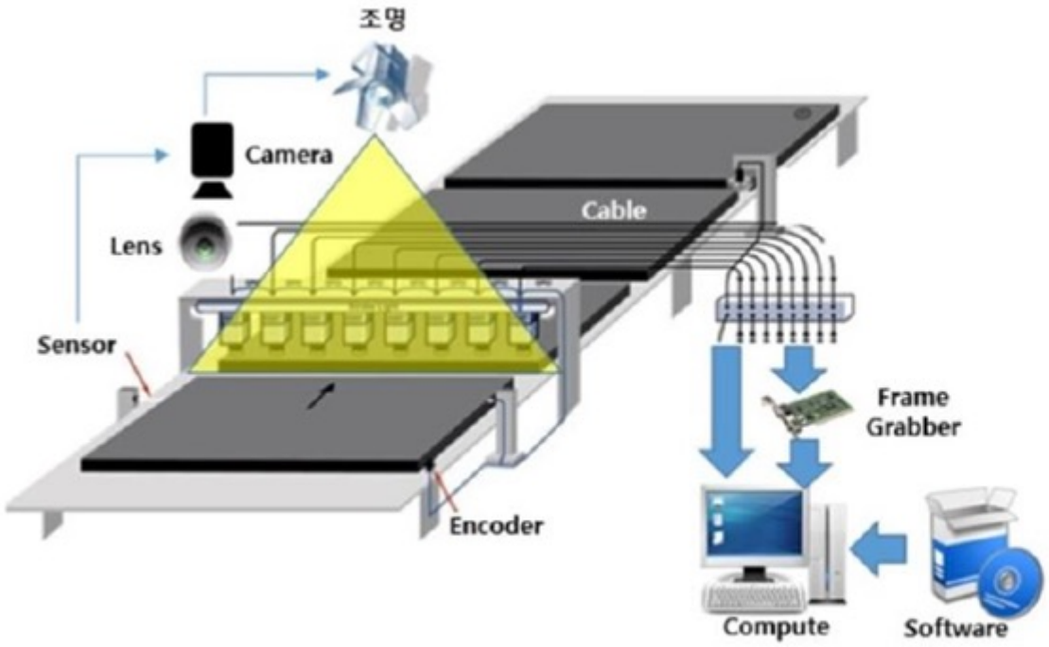
컴퓨터 비전 응용 분야

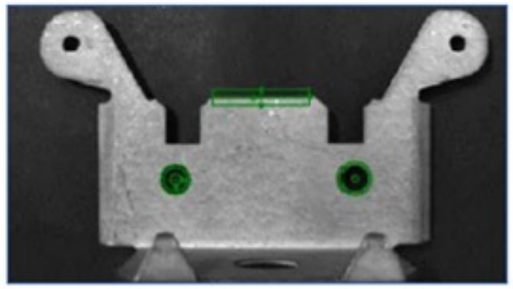
- 머신 비전

  공장 자동화 : 제품의 불량 검사, 위치 확인, 측정 등

  높은 정확도와 빠른 처리 시간 요구

  조명 렌즈, 필터, 실시간 처리





https://laonple.blog.me/, http://www.cognex.com

컴퓨터 비전 응용 분야

- 인공지능 서비스

입력 영상을 객체와 배경으로 분할 -> 객체와 배경 인식 -> 상황인식 -> 로봇과 자동차의 행동 지시

Computer Vision + Sensor Fusion + Deep Learning

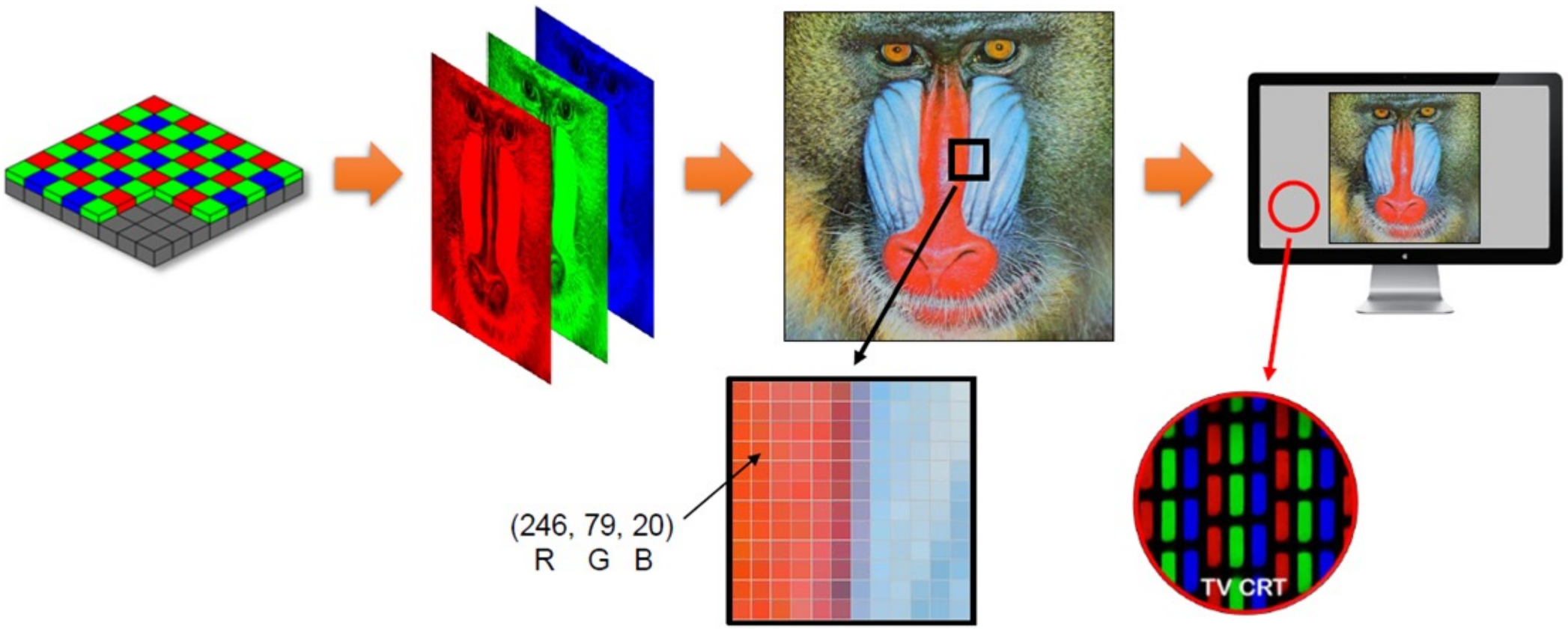
인공지능 로봇, Amazon Go, 구글/테슬라의 자율 주행 자동차

https://youtu.be/NrmMk1Myrxc?t=26

https://youtu.be/wuhbqcMzOaw?t=7

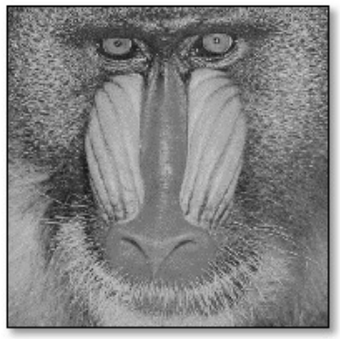
영상의 표현 방법

- 영상이란, 픽셀이 바둑판 모양의 격자에 나열되어 있는 형태의 2차원 행렬입니다.
- 픽셀(화소) : 영상의 기본 단위



영상의 표현 방법

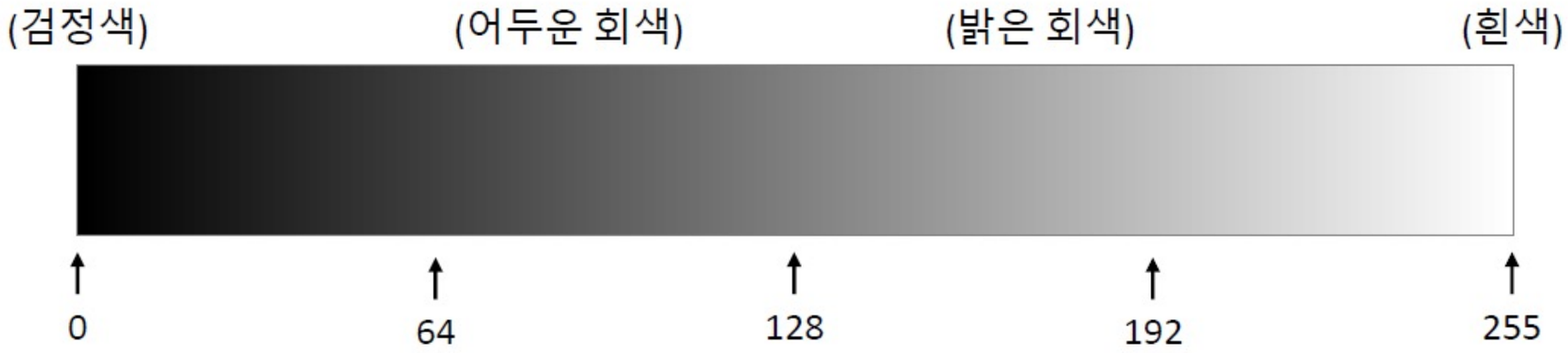
- 그레이스케일(grayscale) 영상
  - 흑백 사진처럼 색상 정보가 없이 오직 밝기 정보만으로 구성된 영상
  - 밝기 정보를 256 단계로 표현




- 트루컬러(truecolor) 영상
  - 컬러 사진처럼 색상 정보를 가지고 있어서 다양한 색상을 표현할 수 있는 영상
  - Red, Green, Blue 색 성분을 256 단계로 표현
    → 256^3 = 16,777,216 색상 표현 가능


영상의 표현 방법

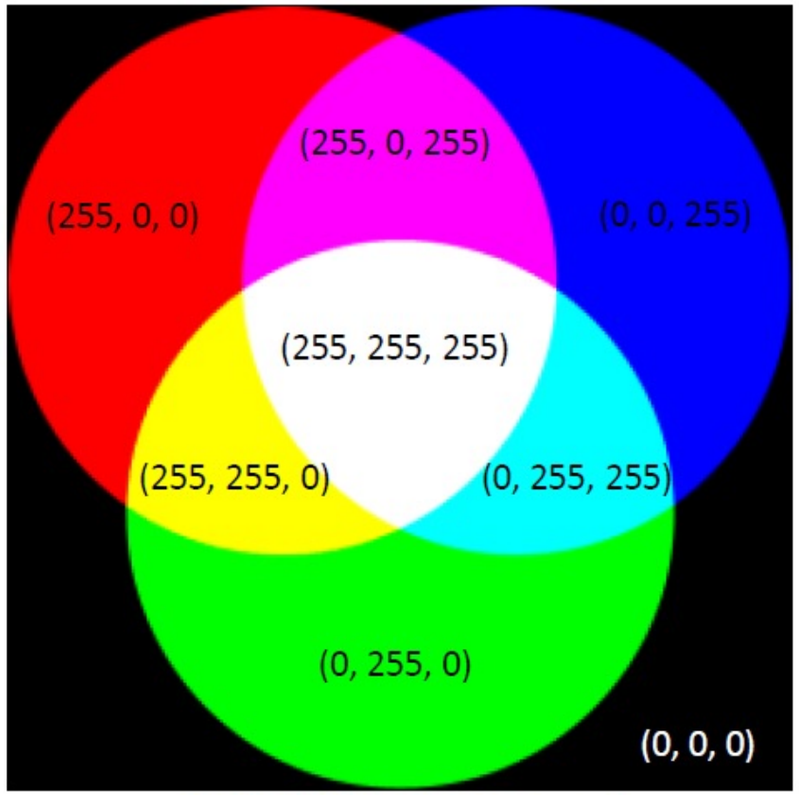
- 그레이스케일 영상의 픽셀 값 표현
  - 밝기 성분을 0 ~ 255 범위의 정수로 표현



- 프로그래밍 언어에서 표현 방법: 1Byte 사용
  - C/C++ ➔ unsigned char
  - Python ➔ numpy.uint8

영상의 표현 방법

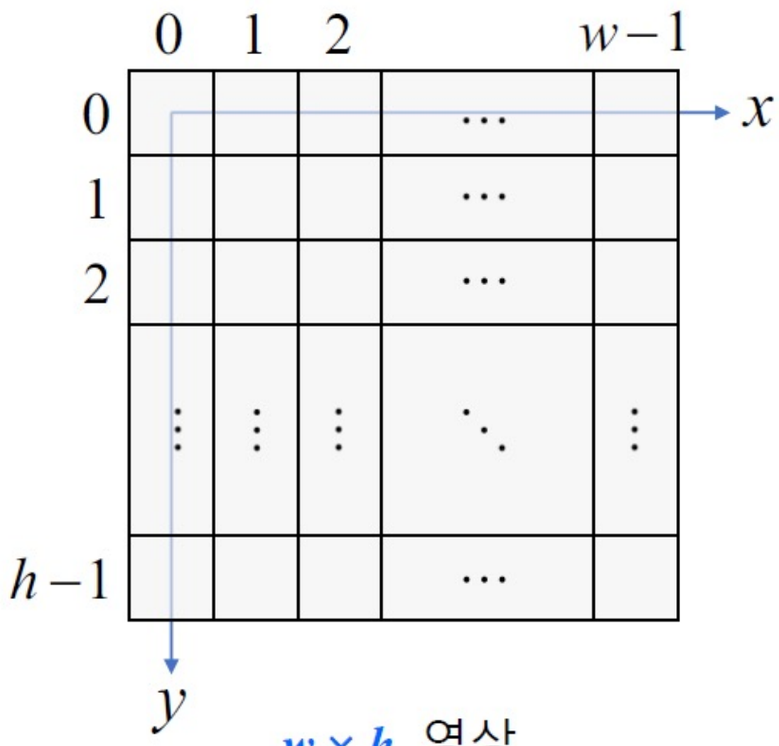
- 컬러 영상의 픽셀 값 표현
  - R, G, B 색 성분의 크기를 각각 0 ~ 255 범위의 정수로 표현
    - 0 : 해당 색 성분이 전혀 없는 상태
    - 255 : 해당 색 성분이 가득 있는 상태
  - 프로그래밍 언어에서 표현 방법: 3Bytes 사용
    - C/C++ ➔ 구조체, 클래스
    - Python ➔ 튜플, numpy.ndarray



빛의 삼원색: (R, G, B)

영상의 표현 방법

- 영상에서 주로 사용되는 좌표계



$w \times h$ 영상
(w-by-h image)

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix}$$

$M \times N$ 행렬
(m-by-n matrix)

영상의 표현 방법

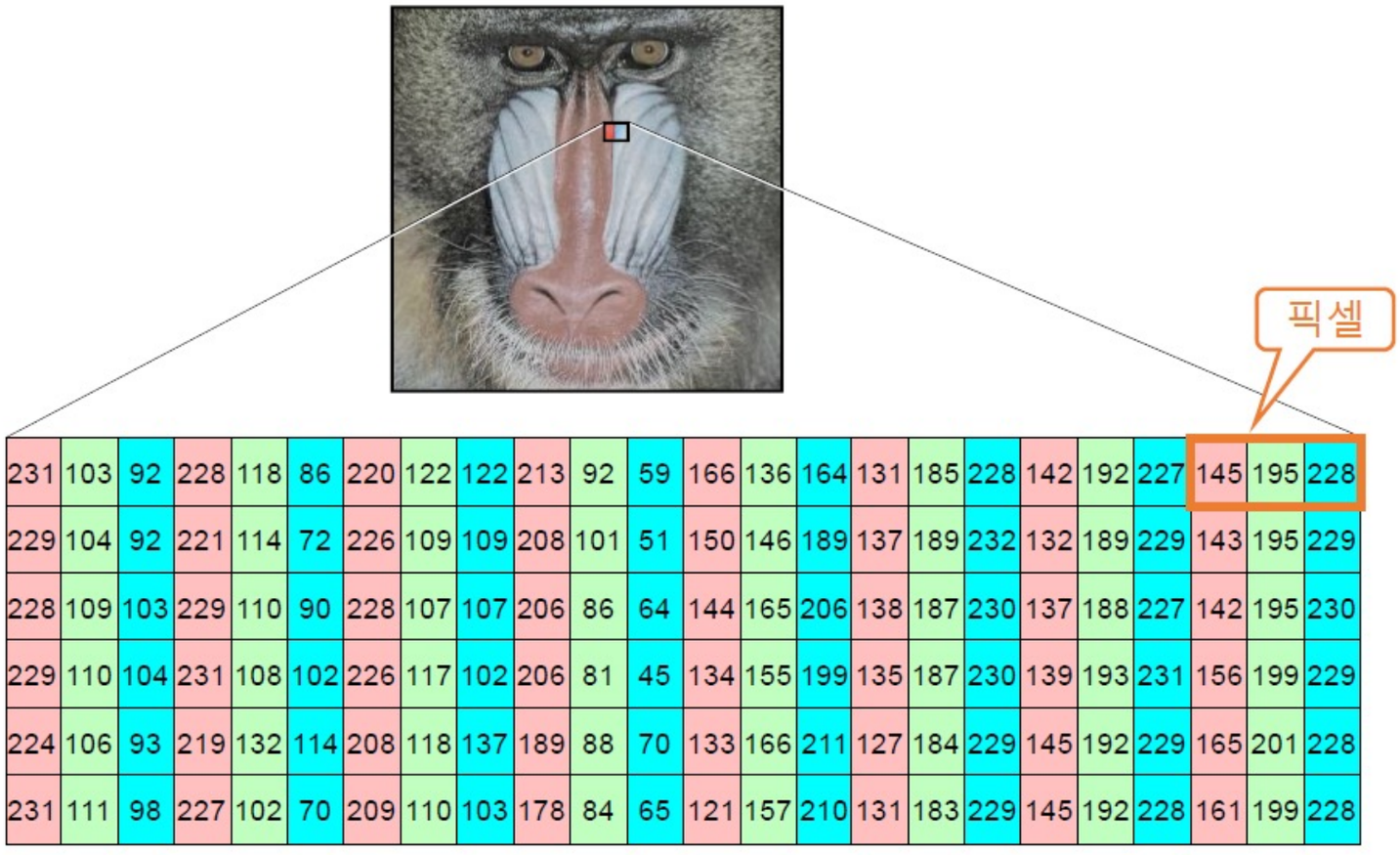
- 그레이스케일 영상에서 픽셀 값 분포의 예

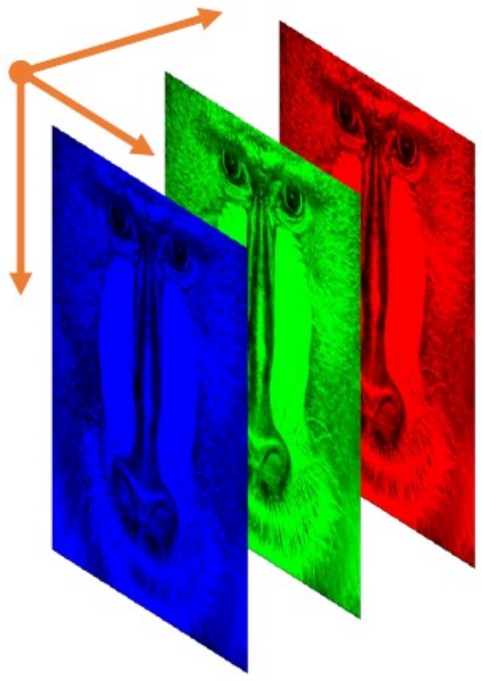
픽셀

187	187	187	194	197	173	77	25	19	19
190	187	190	191	158	37	15	14	20	20
187	182	180	127	32	16	13	16	14	12
184	186	172	100	20	13	15	18	13	18
186	190	187	127	18	14	15	14	12	10
189	192	192	148	16	15	11	10	10	9
192	195	181	37	13	10	10	10	10	10
189	194	54	14	11	10	10	10	9	8
189	194	19	16	11	11	10	10	9	9
192	88	12	11	11	10	10	10	9	9

영상 파일 형식 특징

- 트루컬러 영상에서 픽셀 값 분포의 예





영상 데이터의 크기

- 영상 데이터 크기 분석
  - 그레이스케일 영상: (가로 크기) × (세로 크기) Bytes
  - 트루컬러 영상: (가로 크기) × (세로 크기) × 3 Bytes

$512 \times 512 = 262144$ Bytes

$1920 \times 1080 \times 3 = 6220800$ Bytes

$\approx$ 6 MBytes

영상 파일 형식 특징

BMP	픽셀 데이터를 압축하지 않고 그대로 저장하여 파일 용량이 매우 크다. 파일 구조가 단순하여 별도의 라이브러리 도움 없이 파일 입출력이 가능
JPG	주로 사진과 같은 컬러 영상을 저장 손실 압축 방법 압축률이 좋아서 파일 용량이 크게 감소 디지털 카메라에서 가장 선호하는 사진 포맷
GIF	256색상 이하의 영상을 저장 무손실 압축 움직이는 GIF지원
PNG	Portable Network Graphics 무손실 압축 알파 채널(투명도)을 지원

OpenCV 개요

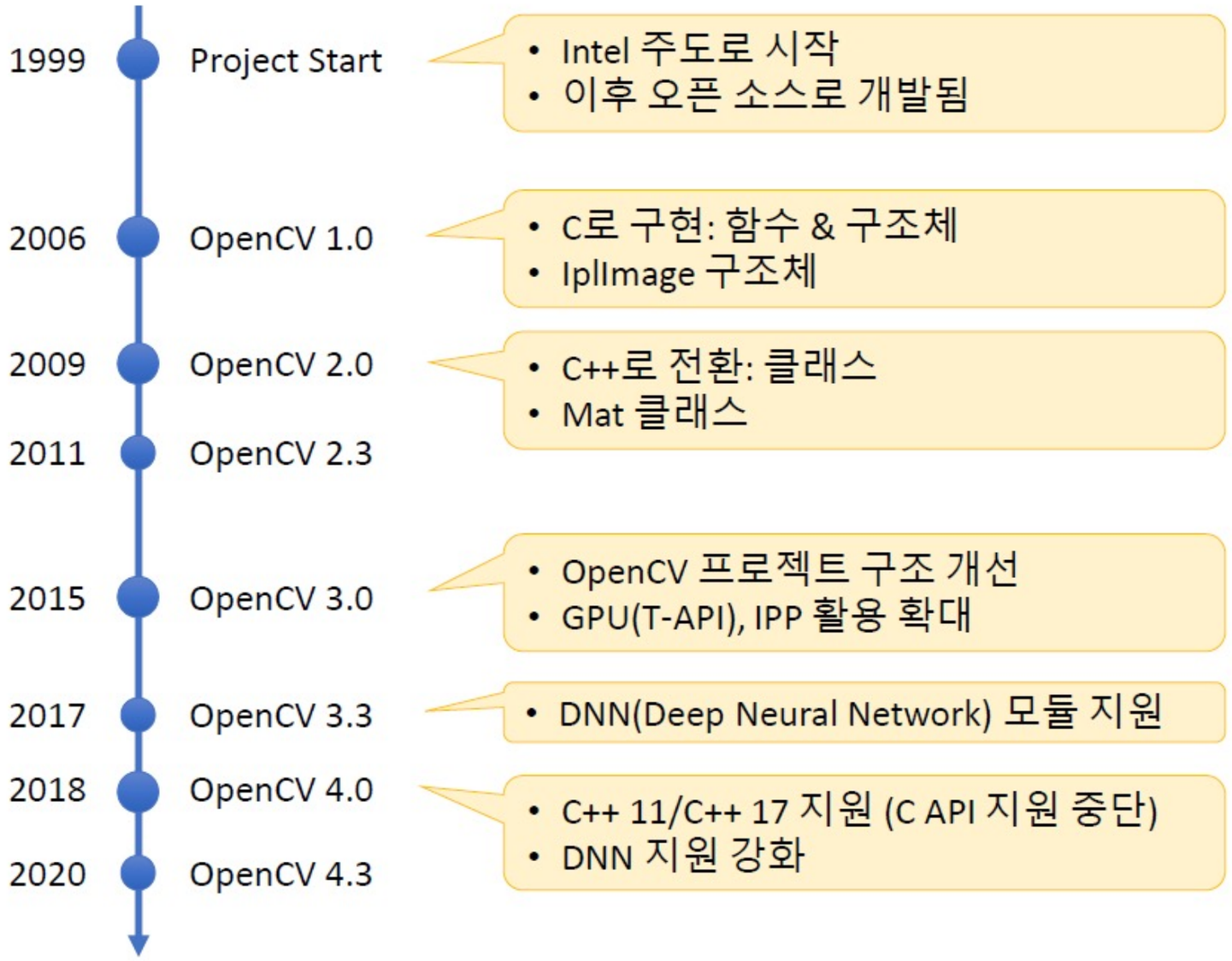
- What is OpenCV?
  - Open source
  - Computer vision & machine learning
  - Software library

- Why OpenCV?
  - BSD license … Free for academic & commercial use
  - Multiple interface … C, C++, Python, Java, JavaScript, MATLAB, etc.
  - Multiple platform … Windows, Linux, Mac OS, iOS, Android
  - Optimized … CPU instructions, Multi-core processing, OpenCL, CUDA
  - Popular … More than 18 million downloads
  - Usage … Stitching streetview images, detecting intrusions, monitoring mine equipment, helping robots navigate and pick up objects, Interactive art, etc.

OpenCV역사

```
import cv
```

```
import cv2
```



영상의 표현 방법

- OpenCV main modules
  - Core, widely used, infrastructures
  - https://github.com/opencv/opencv/

  calib3d, core, dnn, features2d, flann, gapi, highgui, imgcodecs, imgproc, java, js, ml, objdetect, photo, python, stitching, ts, video, videoio, world

- OpenCV extra modules
  - Brand new, unpopular, non-free, HW dependency, etc.
  - https://github.com/opencv/opencv_contrib/

  aruco, bgsegm, bioinspired, ccalib, cnn_3dobj, cudaarithm, cudabgsegm, ... , cudawarping, cudev, cvv, datasets, dnns_easily_fooled, dnn_objdetect, dpm, face, freetype, fuzzy, hdf, hfs, img_hash, line_descriptor, matlab, optflow, ovis, phase_unwrapping, plot, reg, rgbd, saliency, sfm, shape, stereo, structured_light, superres, surface_matching, text, tracking, videostab, viz, xfeatures2d, ximgproc, xobjdetect, xphoto

Anaconda 가상환경 생성

가상환경 생성

```
conda create –n opencv4 python=3.7
```

가상환경 활성화

```
conda activate opencv4
```

opencv 패키지 설치

```
pip install opencv-python==4.2.0.32
```

matplotlib 패키지 설치

```
pip install matplotlib
```

pafy 패키지 설치

```
pip install pafy
```

OpenCV-Python 설치

- pip 명령으로 설치하기

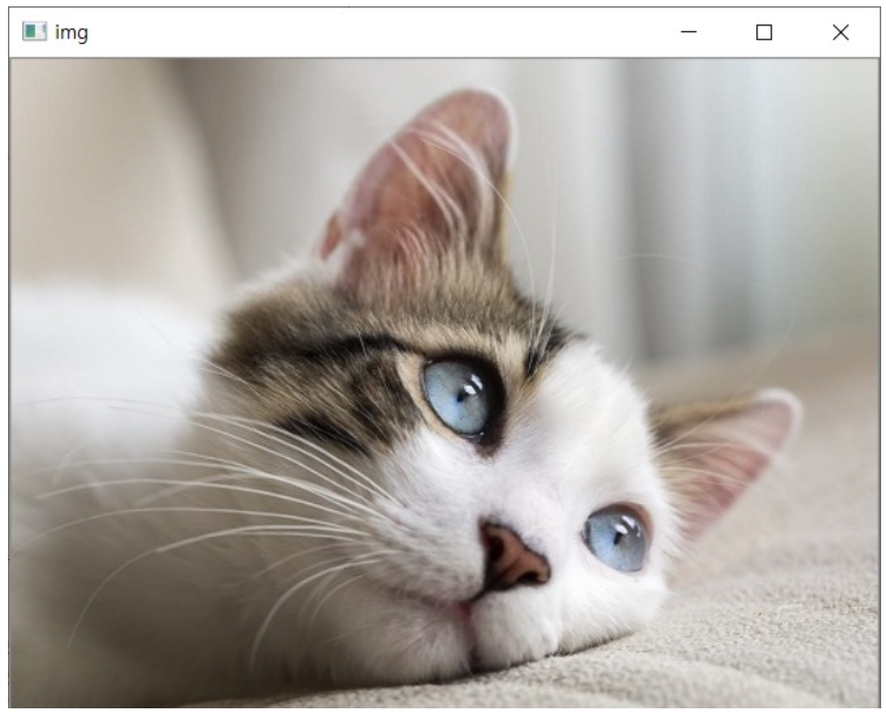
```
> pip install opencv-python
```

  - <PYTHON_PATH>\Lib\site-packages 폴더 아래에 cv, opencv_python-4.2.0.34.dist-info 폴더가 생성되고, 그 아래에 cv2.cp37-win_amd64.pyd 파일이 저장됨.
  - 설치되는 OpenCV 버전은 https://pypi.org/ 에서 확인 가능
  - OpenCV 추가 모듈도 함께 사용하려면 opencv-contrib-python 패키지를 설치
  - OpenCV 4.2.0 버전에서 cv2.imread() 함수를 사용하여 영상을 그레이스케일 형식으로 불러올 때 픽셀 값이 잘못 설정되는 문제가 있음
    → cv2.imread() 함수가 정상 동작하는 OpenCV를 설치하려면 아래 명령 사용

```
> pip install opencv-python==4.1.0.25
```

파일에서 이미지를 읽어서 출력하기

```
import sys
import cv2

print("Hello OpenCV")

img = cv2.imread('cat2.jpg')

if img is None:
    print("Image load fail")
    sys.exit()

cv2.namedWindow('img')
cv2.imshow('img',img)
cv2.waitKey()

cv2.destroyAllWindows()
```



영상 입출력과 디스플레이 함수

함 수	비 고
cv2.imread(filename[, flags]) -> retval	영상입력
cv2.imwrite(filename, img[, flags]) -> retval	영상파일 출력
cv2.namedWindow(winname[, flags])	윈도우 생성
cv2.imshow(winname, mat)	윈도우 표시
cv2.waitkey([, delay]) -> retval	키보드 입력 대기
cv2.destroyWindow(winname)	윈도우 닫기
cv2.destroyAllWindows( )	모든 윈도우 닫기

imread() : 이미지 파일 불러오기

```
cv2.imread(filename, flags=None)   -> retval
```

- Filename : 불러올 영상 파일이름(문자열)
- flags : 영상 파일 불러오기 옵션 플래그

flags	
cv2.IMREAD_COLOR	BGR 컬러 영상으로 읽기(기본값) shape = (rows, cols, 3)
cv2.IMREAD_GRAYSCALE	그레이 영상으로 읽기 shape = (rows, cols, 3)
cv2.IMREAD_UNCHANGED	영상 파일 속성 그대로 읽기 PNG파일 : shape = (rows, cols,4)

- retval : 불러온 영상 데이터 (numpy.ndarray)

imwrite() : 이미지 파일 쓰기

```
cv2.imwrite(filename, img, params=None)   -> retval
```

- Filename : 저장할 영상 파일 이름(문자열)
- img : 저장할 영상 데이터
- params : 파일 저장 옵션 지정 (영상의 화질 지정 등..)
- retval : 정상적으로 저장하면 True, 실패하면 False

namedWindow() : 새 창 생성하기

```
cv2.namedWindow(winname, flags=None)  -> None
```

- winname : 생성할 창의 이름(문자열)
- flags : 창 속성 지정 플래그

flags	
cv2.WINDOW_NORMAL	영상 크기를 창 크기에 맞게 지정
cv2.WINDOW_AUTOSIZE	창 크기를 영상 크기에 맞게 변경(기본값)

destroyWindow() : 열린 창 닫기

```
cv2.destroyWindow(winname)   -> None
cv2.destroyAllWindows(winname)   -> None
```

- winname : 닫고자 하는 창 이름
- destroyWindow()는 하나의 창만 닫고, destroyAllWindows()는 열려 있는 모든 창을 닫는다. destroyAllWindows()함수의 대소문자 주의

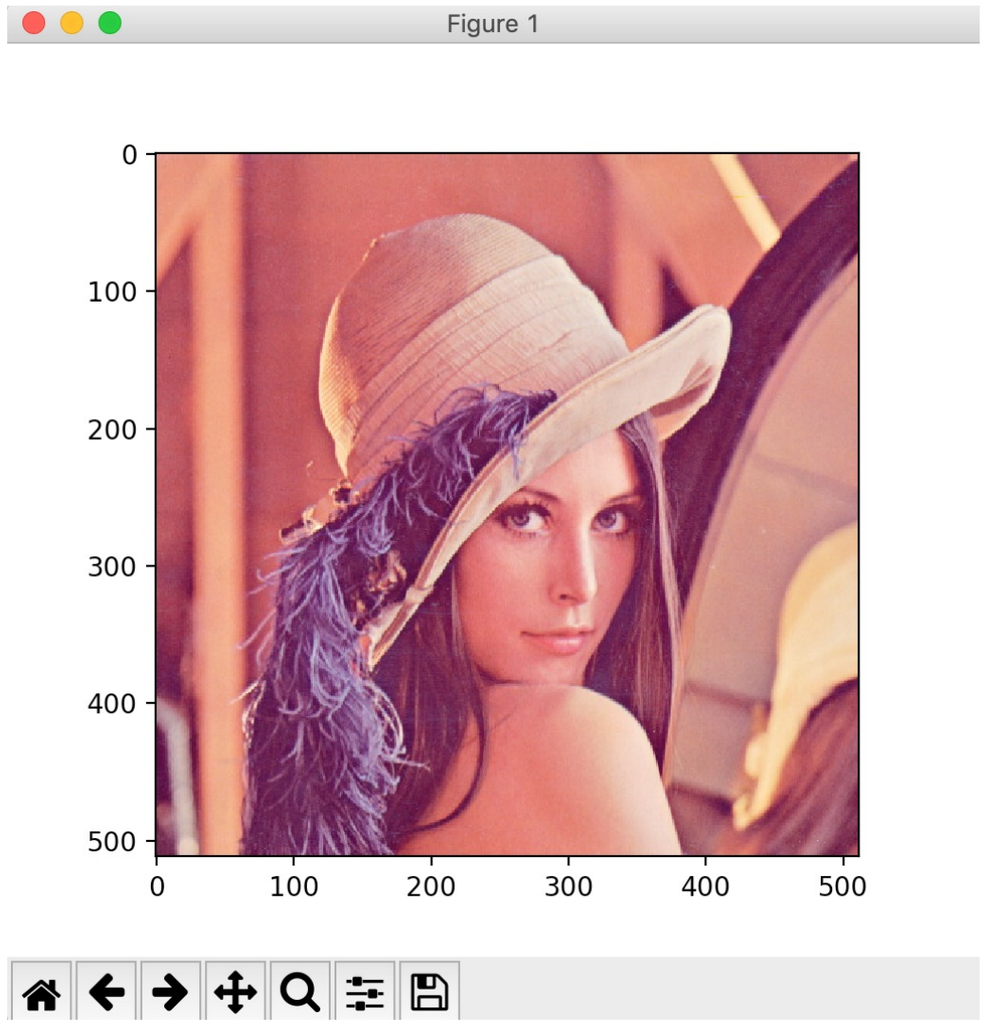
파일에서 이미지를 읽어서 저장하기

```
# 파이썬에서 openCV를 사용하기 위해
import cv2

# 파일이 저장된 경로를 문자열 객체 지정
imageFile = './data/lena.jpg'

# BGR포맷으로 읽어서 img 객체에 저장한다.
img  = cv2.imread(imageFile)    # cv2.IMREAD_COLOR

# Lena.bmp 이름으로 파일을 저장한다.
cv2.imwrite('./data/Lena.bmp', img)

# Lena.png 이름으로 파일을 저장한다.
cv2.imwrite('./data/Lena.png', img)

# Lena2.png 이름으로 파일을 저장한다. 0~9사이의 숫자로 압축률을 정한다. 디폴트는 3
cv2.imwrite('./data/Lena2.png',img, [cv2.IMWRITE_PNG_COMPRESSION, 9])

# Lena.jpg 이름으로 파일을 저장한다. jpg의 품질을 90%로 설정한다. 디폴트는 95%
cv2.imwrite('./data/Lena2.jpg', img, [cv2.IMWRITE_JPEG_QUALITY, 90])
```

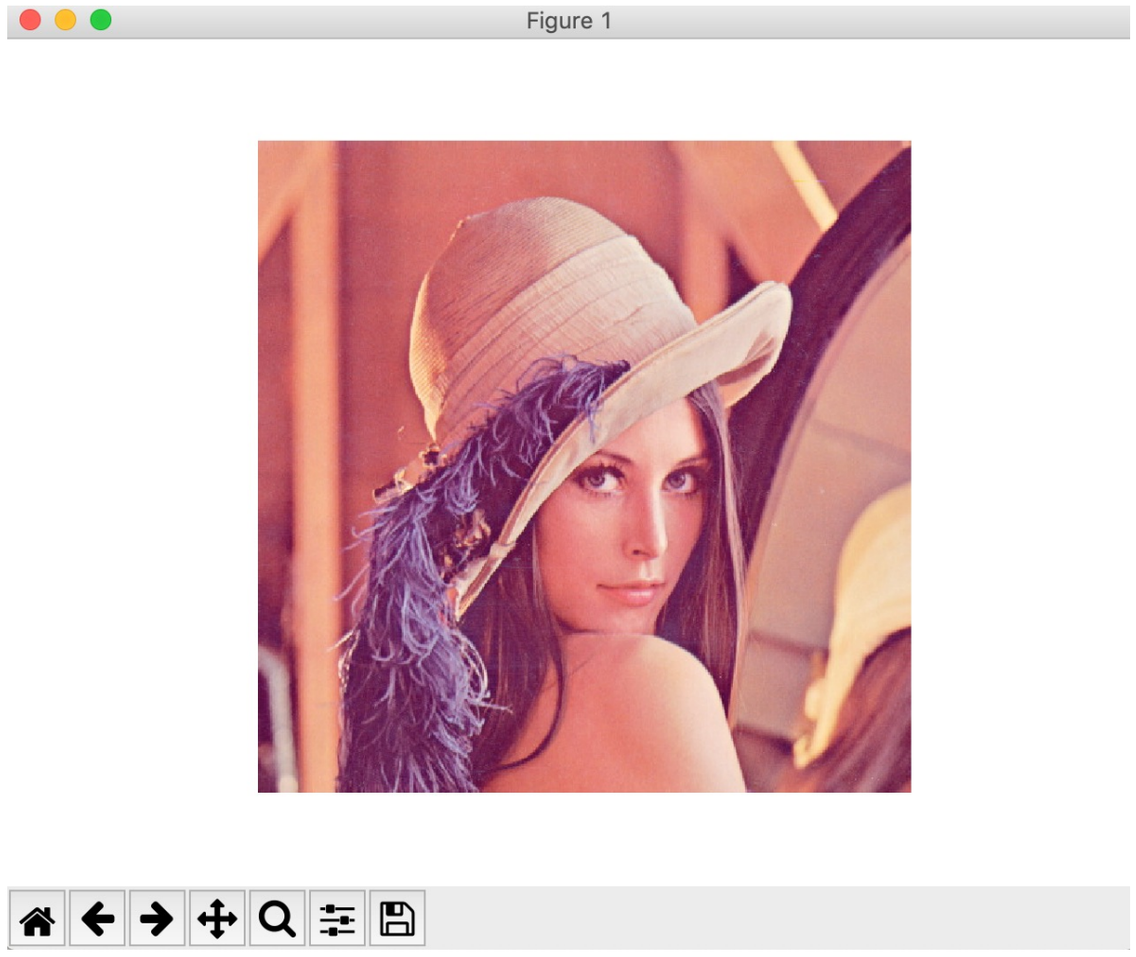
Matplotlib1 : 컬러 영상 표시

```
# 파이썬에서 openCV를 사용하기 위해
import cv2

# matplotlib 패키지에서 pyplot를 plt 라는 이름으로 임포트 한다.
from   matplotlib import pyplot as plt

# 파일이 저장된 경로를 문자열 객체 지정
imageFile = './data/lena.jpg'

# BGR포맷으로 읽어서 img 객체에 저장한다.
img  = cv2.imread(imageFile)    # cv2.IMREAD_COLOR

# X, Y축을 표시하지 않는다.
plt.axis('off')
#plt.imshow(imgBGR)
#plt.show()

# plot에 그리기 위해서는 RGB순서의 포맷이어야 한다.
imgRGB = cv2.cvtColor(imgBGR,cv2.COLOR_BGR2RGB)
plt.imshow(imgRGB)    # plot에 imgRGB를 그리고
plt.show( )           # plot에 그린 내용을 출력한다.
```

Matplotlib1 : 컬러 영상 표시



plt.axis('on')



plt.axis('off')

Matplotlib2 : 그레이 영상 표시

```
# 파이썬에서 openCV를 사용하기 위해
import cv2

# matplotlib 패키지에서 pyplot를 plt 라는 이름으로 임포트 한다.
from   matplotlib import pyplot as plt

# 파일이 저장된 경로를 문자열 객체 지정
imageFile = ‘./data/lena.jpg'

# Grayscale 포맷으로 읽어서 imgGray 객체에 저장한다.
imgGray = cv2.imread(imageFile, cv2.IMREAD_GRAYSCALE)

# X, Y축을 표시하지 않는다.
plt.axis('off')

# plot에 컬러맵을 “gray”로 설정하고 imgGray를 그린다.
plt.imshow(imgGray, cmap = "gray", interpolation=‘bicubic')
plt.show()
```

Matplotlib2 : 그레이 영상 표시

Matplotlib 그레이스케일 영상

Matplotlib3 : 여백 조정 및 영상 저장

```
# 파이썬에서 openCV를 사용하기 위해
 import cv2

 # matplotlib 패키지에서 pyplot를 plt 라는 이름으로 임포트 한다.
 from   matplotlib import pyplot as plt

 # 파일이 저장된 경로를 문자열 객체 지정
 imageFile = ‘./data/lena.jpg'

 # Grayscale 포맷으로 읽어서 imgGray 객체에 저장한다.
 imgGray = cv2.imread(imageFile, cv2.IMREAD_GRAYSCALE)

 # plt.figure 크기를 (6,6) 인치로 설정
 plt.figure(figsize=(6,6))

 # left = 0, right = 1, bottom = 0, top = 1로 설정 (일단 창을 띄워서 값을 변경해 보자)
 plt.subplots_adjust(left=0, right=1, bottom=0, top=1)
 #plt.subplots_adjust(left=0.03, right=0.97, bottom=0.03, top=0.97)
 plt.imshow(imgGray, cmap = 'gray')
 #plt.axis('tight')
 plt.axis(‘off’)

 # plot 배열에 저장된 이미지를 파일로 저장
 plt.savefig('./data/0205.png')
 plt.show()
```

Matplotlib3 : 여백 조정 및 영상 저장

left=0.03, right=0.97
bottom=0.03, top=0.97

left=0, right=1
bottom=0, top=1

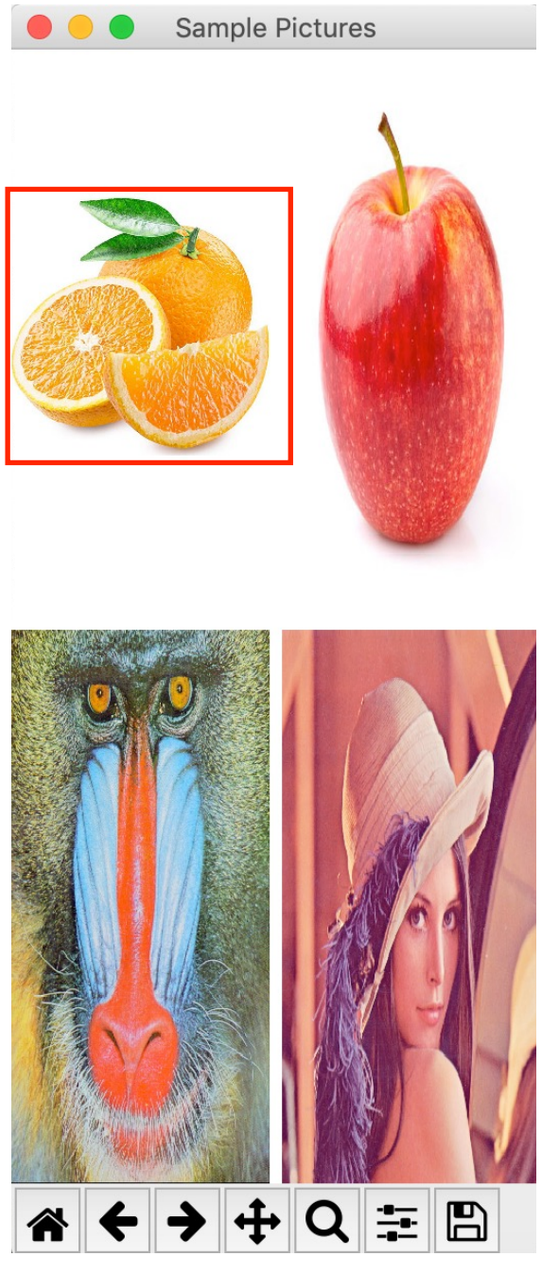
```
# 파이썬에서 openCV를 사용하기 위해
 import cv2

# matplotlib 패키지에서 pyplot를 plt 라는 이름으로 임포트 한다.
from   matplotlib import pyplot as plt

# BGR포맷으로 읽어서 imgBGR1~4 객체에 저장한다.
path = ‘./data/'
imgBGR1 = cv2.imread(path+'orange.jpg')
imgBGR2 = cv2.imread(path+'apple.jpg')
imgBGR3 = cv2.imread(path+'baboon.jpg')
imgBGR4 = cv2.imread(path+'lena.jpg')

# 컬러 순서 변환: BGR -> RGB
imgRGB1 = cv2.cvtColor(imgBGR1, cv2.COLOR_BGR2RGB)
imgRGB2 = cv2.cvtColor(imgBGR2, cv2.COLOR_BGR2RGB)
imgRGB3 = cv2.cvtColor(imgBGR3, cv2.COLOR_BGR2RGB)
imgRGB4 = cv2.cvtColor(imgBGR4, cv2.COLOR_BGR2RGB)

# subplots 2x2 총 4개로 설정하고 Window를 (10,10) 사이즈로 정의하고,
# window의 이름을 ‘Sample Pictures’로 정의 한다.
fig, ax = plt.subplots(2, 2, figsize=(10,10), sharey=True)
fig.canvas.set_window_title('Sample Pictures’)
```

Matplotlib4 : 서브플롯에 영상 표시

```
# 왼쪽 위 Window의 좌표 축을 ‘off’하고,
# imgRGB1(orange이미지) 비율을 ‘auto’로 설정한다.
ax[0][0].axis('off')
ax[0][0].imshow(imgRGB1, aspect = 'auto')

# 오른쪽 위 Window의 좌표 축을 ‘off’하고,
# imgRGB2(apple이미지) 비율을 ‘auto’로 설정한다.
ax[0][1].axis('off')
ax[0][1].imshow(imgRGB2, aspect = 'auto')

# 왼쪽 아래 Window의 좌표 축을 ‘off’하고,
# imgRGB3(baboon이미지) 비율을 ‘auto’로 설정한다.
ax[1][0].axis("off")
ax[1][0].imshow(imgRGB3, aspect = "auto")

# 오른쪽 아래 Window의 좌표 축을 ‘off’하고,
# imgRGB3(lena이미지) 비율을 ‘auto’로 설정한다.
ax[1][1].axis("off")
ax[1][1].imshow(imgRGB4, aspect = 'auto')

# 주어진 사이즈의 Window에서 크기를 최대로 하고,
# sub window간의 공간을 가로(wspace)=0.05, 세로(hspace)=0.05로 설정한다.
plt.subplots_adjust(left=0, bottom=0, right=1, top=1, wspace=0.05, hspace=0.05)
plt.savefig("./data/0206.png", bbox_inches='tight')
plt.show()
```

Matplotlib4 : 서브플롯에 영상 표시

aspect = auto



aspect = default

VideoCapture와 화면 표시1-1

```
# import
import cv2

# 동영상 파일 불러오기
cap = cv2.VideoCapture('./data/vtest.avi')

frame_size = (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
              int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))
print('frame_size =', frame_size)

while True:
    retval, frame = cap.read( )  # 프레임 캡처
    if not retval:
        break

    cv2.imshow('frame',frame)

    key = cv2.waitKey(25)   # 25/1000 sec 대기
    if key == 27:   # Esc
        break

if cap.isOpened( ):
    cap.release( )
cv2.destroyAllWindows( )
```

VideoCapture와 화면 표시1-2

```
# import
import cv2

# 동영상 파일 불러오기
cap = cv2.VideoCapture(0)   # 0번 카메라

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)

frame_size = (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
              int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))
print('frame_size =', frame_size)

while True:
    retval, frame = cap.read( )   # 프레임 캡처
    if not retval:
        break

    cv2.imshow('frame',frame)

    key = cv2.waitKey(25)   # 25/1000 sec 대기
    if key == 27:   # Esc
        break

if cap.isOpened( ):
    cap.release( )
cv2.destroyAllWindows( )
```

비디오 프레임 캡처와 화면 표시 함수

함 수	비 고
cv2.VideoCapture( ) -> <VideoCapture object> cv2.VideoCapture(filename) -> <VideoCapture object> cv2.VideoCapture(device) -> <VideoCapture object>	비디오획득 객체 생성
cv2.VideoCapture.read([image]) -> retval, image	프레임 획득
cv2.VideoCapture.grab( ) -> retval	프레임 잡기
cv2.VideoCapture.retrieve([image[, channel]]) -> retval, image	프레임 획득
cv2.VideoCapture.release( )	비디오 획득 객체 해제
cv2.VideoCapture.get(property_id) -> retval	비디오 특성 얻기
cv2.VideoCapture.set(property_id, value) -> retval	비디오 특성 설정

비디오 프레임 캡처와 화면 표시 함수

cv2.VideoCapture( ) : 비디오획득 객체 생성

비디오 파일 filename 또는 카메라 번호 device로부터 VideoCapture 객체를 생성하여 반환한다. 장치번호는 카메라가 1개 일 경우 0, 두 개 일 경우 0, 1로 지정 한다. 파일명 또는 장치번호 명시 없이 VideoCapture( )를 생성한 경우 Video Capture.open(filename)이나 VideoCapture.open(device)를 함께 사용 한다. VideoCapture.isOpened( )를 사용하면 비디오 객체가 개방 되었는지를 확인 할 수 있다.

cv2.VideoCapture.read(image) : 프레임 획득

장치가 열린 경우(사용 가능한 상태) VideoCapture 객체로 부터 다음 비디오 프레임을 잡아서(grab), 디코딩하여, 프레임을 반환 한다. **VideoCapture.grab( ) + VideoCapture.retrieve( )**로 이해하면 된다. 프레임 캡쳐에 성공하면 retval로 True, 실패 하면 False를 반환 한다.

cv2.VideoCapture.grab( ) : 프레임 잡기

장치가 열린 경우, 다음 프레임을 잡기(grab) 위해 사용한다. 프레임을 캡쳐하기 위해서는 이후에 VideoCapture.retrieve( )를 사용해야 한다. 여러 대의 카메라에서 동기화를 목적으로 사용 한다.

cv2.VideoCapture.retrieve(image, channel) : 프레임 획득

VideoCapture.grab( )된 영상을 디코딩하여 image로 반환 한다. 프레임을 정상적으로 캡처 하면 retval는 True, 실패 하면
False 이다.

cv2.VideoCapture.release( ) : 비디오 획득 객체 해제

장치가 열린 상태의 경우 VideoCapture 객체를 해제하여 닫는다.

cv2.VideoCapture.get(property_id) : 비디오 특성 얻어오기

비디오 파일에서 프레임 레이트, 총 프레임 수 등의 값을 얻어올 때 사용 한다.

cv2.VideoCapture.set(property_id, value) : 비디오 특성 설정하기

비디오 파일의 Property 특성을 설정할 때 사용 한다.

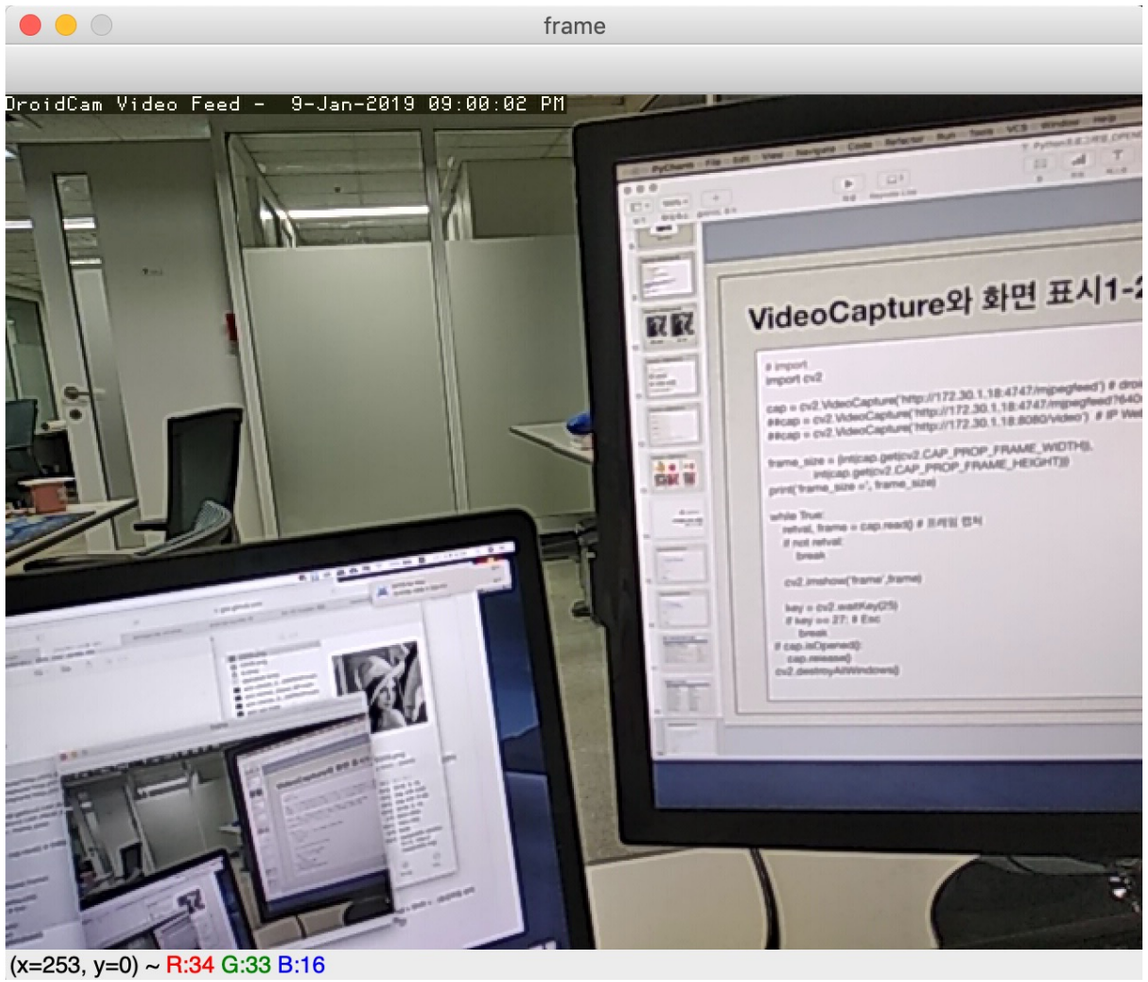
property_id 주요 상수

property_id	설 명
cv2.CAP_PROP_POS_MSEC	밀리초로 현재 위치
cv2.CAP_PROP_POS_FRAMES	캡쳐될 프레임 번호
cv2.CAP_PROP_FRAME_WIDTH	비디오 프레임의 가로 크기
cv2.CAP_PROP_FRAME_HEIGHT	비디오 프레임의 세로 크기
cv2.CAP_PROP_FPS	프레임 속도
cv2.CAP_PROP_FOURCC	코덱의 4문자
cv2.CAP_PROP_FRAME_COUNT	비디오 파일에서 총 프레임 수
cv2.CAP_PROP_CONVERT_RGB	영상이 RGB로 변환해야 하는지 여부
cv2.CAP_PROP_FORMAT	캡쳐된 영상 포맷
CV_CAP_PROP_BRIGHTNESS	카메라에서 영상의 밝기
CV_CAP_PROP_CONTRAST	카메라에서 영상의 대비
CV_CAP_PROP_SATURATION	카메라에서 영상의 포화도
CV_CAP_PROP_HUE	카메라에서 영상의 채도
CV_CAP_PROP_GAIN	카메라에서 영상의 Gain
CV_CAP_PROP_EXPOSURE	카메라에서 영상의 노출

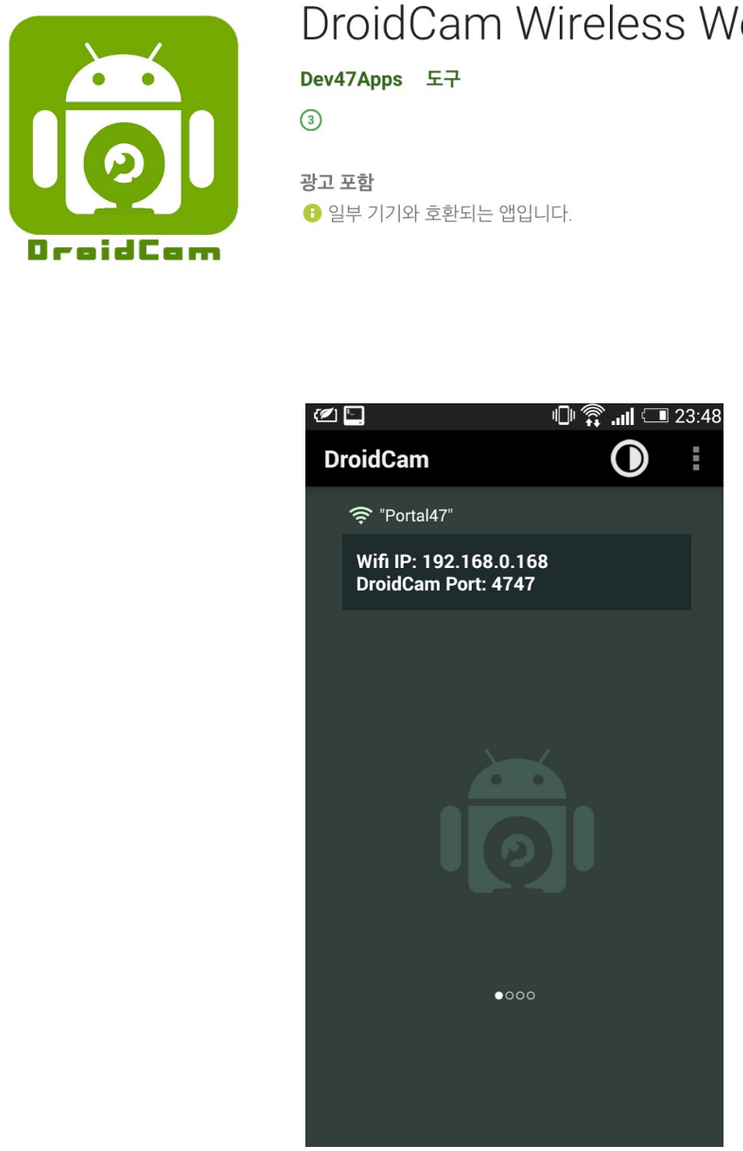
VideoCapture와 화면 표시1-2

```
# import
import cv2

cap = cv2.VideoCapture('http://192.168.1.129:4747/mjpegfeed') # droid cam

frame_size = (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
              int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))
print('frame_size =', frame_size)

while True:
    retval, frame = cap.read() # 프레임 캡처
    if not retval:
        break

    cv2.imshow('frame',frame)

    key = cv2.waitKey(25)
    if key == 27: # Esc
        break

if cap.isOpened():
    cap.release()
cv2.destroyAllWindows()
```

Matplotlib3 : 여백 조정 및 영상 저장



Python OpenCV



Android Phone

Pafy (Retrieve YouTube content and metadata)

pafy 0.5.4

```
pip install pafy
```

Latest version

Last released: Feb 12, 2018

Retrieve YouTube content and metadata

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage
- Download

Statistics

View statistics for this project via Libraries.io, or by using Google BigQuery

Project description

pypi v0.5.4 | downloads 14k/month | coverage unknown | health 92% | build failing | wheel yes

Features

- Retreive metadata such as viewcount, duration, rating, author, thumbnail, keywords
- Download video or audio at requested resolution / bitrate / format / filesize
- Command line tool (ytdl) for downloading directly from the command line
- Retrieve the URL to stream the video in a player such as vlc or mplayer
- Works with age-restricted videos and non-embeddable videos
- Small, standalone, single importable module file (pafy.py)
- Select highest quality stream for download or streaming
- Download video only (no audio) in m4v or webm format
- Download audio only (no video) in ogg or m4a format
- Retreive playlists and playlist metadata
- Works with Python 2.6+ and 3.3+
- Optionally depends on youtube-dl (recommended; more stable)

Pafy (Retrieve YouTube content and metadata)

create a video instance from a YouTube url:

```
>>> url = "https://www.youtube.com/watch?v=bMt47wvK6u0"
>>> video = pafy.new(url)
```

get certain attributes:

```
>>> video.title
'Richard Jones: Introduction to game programming - PyCon 2014'

>>> video.rating
5.0

>>> video.viewcount, video.author, video.length
(1916, 'PyCon 2014', 10394)

>>> video.duration, video.likes, video.dislikes
('02:53:14', 25, 0)

>>> print(video.description)
Speaker: Richard Jones

This tutorial will walk the attendees through development of a simple game using PyGame with time left over for some experimentation and exploration of different types of games.

Slides can be found at: https://speakerdeck.com/pycon2014 and https://github.com/PyCon/2014-slides
```

VideoCapture와 유튜브 동영상

```
# pip install youtube_dl
# pip install pafy

import cv2, pafy
url = 'https://www.youtube.com/watch?v=u_Q7Dkl7AIk'
video = pafy.new(url)
print('title = ', video.title)
print('video.rating = ', video.rating)
print('video.duration = ', video.duration)

best = video.getbest(preftype='webm')    # 'mp4','3gp' 의 최적의 해상도를 찾아준다.
print('best.resolution', best.resolution)

cap=cv2.VideoCapture(best.url)
while(True):
    retval, frame = cap.read()
    if not retval:
        break
    cv2.imshow('frame',frame)

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray,100,200)
    cv2.imshow('edges',edges)

    key = cv2.waitKey(25)
    if key == 27:   # Esc
        break
cv2.destroyAllWindows()
```

비디오 프레임 캡처와 화면 표시 함수

비디오 파일 녹화 함수

함 수	비 고
cv2.VideoWriter([filename, fourcc, fps, frameSize[, isColor]]) -> <VideoWriter object>	비디오 출력 객체 생성
cv2.VideoWriter.write(image)	비디오 파일에 이미지 출력
cv2.VideoWriter.release( )	비디오 출력 객체 해제

주요 fourcc 비디오 코덱 문자

함 수	비 고
cv2.VideoWriter_fourcc(*'PIM1')	MPEG-1
cv2.VideoWriter_fourcc(*'MJPG')	Motion-JPEG
cv2.VideoWriter_fourcc(*'DIVX')	DIVX 4.0 이후 버전
cv2.VideoWriter_fourcc(*'XVID')	XVID, MPEG-4
cv2.VideoWriter_fourcc(*'MPEG')	MPEG
cv2.VideoWriter_fourcc(*'X264')	H.264/AVC

비디오 파일 저장

```
import cv2

cap = cv2.VideoCapture(0) # 0번 카메라
frame_size = (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
              int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))
print('frame_size =', frame_size)

fourcc = cv2.VideoWriter_fourcc(*'XVID')

out1 = cv2.VideoWriter('./data/record0.mp4',fourcc, 20.0, frame_size)
out2 = cv2.VideoWriter('./data/record1.mp4',fourcc, 20.0, frame_size,isColor=False)

while True:
    retval, frame = cap.read()
    if not retval:
        break
    out1.write(frame)

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    out2.write(gray)
    cv2.imshow('frame',frame)
    cv2.imshow('gray',gray)

    key = cv2.waitKey(25)
    if key == 27:
        break
cap.release()
out1.release()
out2.release()
cv2.destroyAllWindows()
```

Matplotlib 비디오 디스플레이

```
import cv2
import matplotlib.pyplot as plt

# 키 입력을 위한 함수
def handle_key_press(event):
    if event.key == 'escape':
        cap.release()
        plt.close()

# 종료 이벤트를 처리하기 위한 함수
def handle_close(evt):
    print('Close figure!')
    cap.release()
```

Matplotlib 비디오 디스플레이

```
# 프로그램 시작
cap = cv2.VideoCapture(0) # 0번 카메라

# interactive on
plt.ion() # 대화모드 설정
fig = plt.figure(figsize=(10, 6)) # fig.set_size_inches(10, 6)
plt.axis('off')
#ax = fig.gca()
#ax.set_axis_off()
fig.canvas.set_window_title('Video Capture')
fig.canvas.mpl_connect('key_press_event', handle_key_press)
fig.canvas.mpl_connect('close_event', handle_close)

retval, frame = cap.read() # 첫 프레임 캡처
im = plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))

#3
while True:
    retval, frame = cap.read() # 프레임 캡처
    if not retval:
        break
#    plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    im.set_array(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    fig.canvas.draw()                    # canvas update
#   fig.canvas.draw_idle()
    fig.canvas.flush_events()  # plt.pause(0.001)
if cap.isOpened():
    cap.release()
```

fig.canvas.mpl_connect(event, func)

https://matplotlib.org/users/event_handling.html

Event connections

```
fig, ax = plt.subplots()
ax.plot(np.random.rand(10))

def onclick(event):
    print('%s click: button=%d, x=%d, y=%d, xdata=%f, ydata=%f' %
          ('double' if event.dblclick else 'single', event.button,
           event.x, event.y, event.xdata, event.ydata))

cid = fig.canvas.mpl_connect('button_press_event', onclick)
```

```
fig.canvas.mpl_disconnect(cid)
```

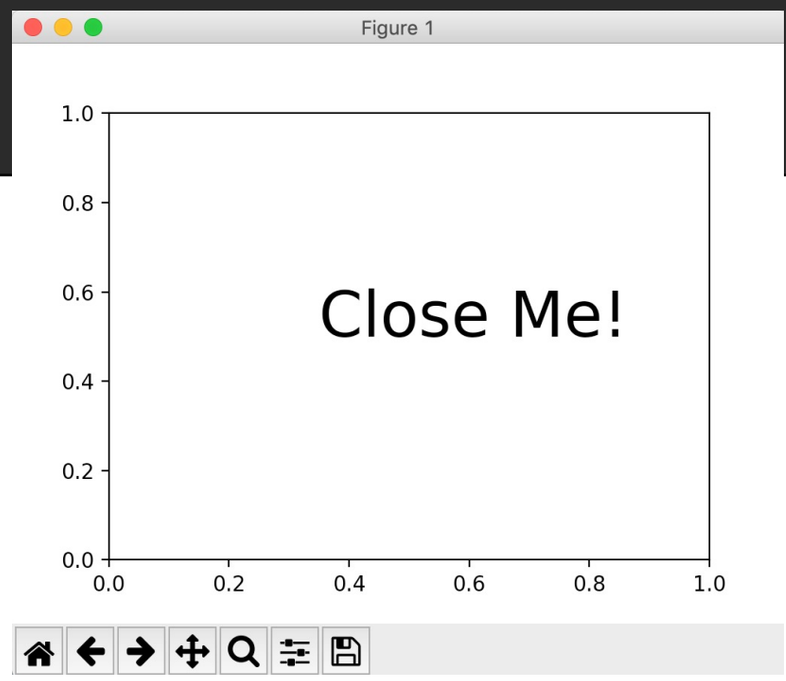
fig.canvas.mpl_connect(event, func)

Event name	Class and description
'button_press_event'	`MouseEvent` - mouse button is pressed
'button_release_event'	`MouseEvent` - mouse button is released
'draw_event'	`DrawEvent` - canvas draw (but before screen update)
'key_press_event'	`KeyEvent` - key is pressed
'key_release_event'	`KeyEvent` - key is released
'motion_notify_event'	`MouseEvent` - mouse motion
'pick_event'	`PickEvent` - an object in the canvas is selected
'resize_event'	`ResizeEvent` - figure canvas is resized
'scroll_event'	`MouseEvent` - mouse scroll wheel is rolled
'figure_enter_event'	`LocationEvent` - mouse enters a new figure
'figure_leave_event'	`LocationEvent` - mouse leaves a figure
'axes_enter_event'	`LocationEvent` - mouse enters a new axes
'axes_leave_event'	`LocationEvent` - mouse leaves an axes

close_event

Example

```
import matplotlib.pyplot as plt

def handle_close(evt):
    print('Closed Figure!')

fig = plt.figure()
fig.canvas.mpl_connect('close_event', handle_close)

plt.text(0.35, 0.5, 'Close Me!', dict(size=30))
plt.show()
```



Animation.FuncAnimation [Basic Form]

```
import cv2
import matplotlib.pyplot as plt
import matplotlib.animation as animation

# 프로그램 시작
cap = cv2.VideoCapture(0)
fig = plt.figure(figsize=(10, 6)) # fig.set_size_inches(10, 6)
fig.canvas.set_window_title('Video Capture ' )
plt.axis('off ' )

def init():
    global im
    retval, frame = cap.read() # 첫 프레임 캡처
    im = plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
##    return im,

def updateFrame(frameCount):
    retval, frame = cap.read()
    print(frameCount)
    if retval:
        im.set_array(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))

# matplotlib.animation.FuncAnimation(fig, func, frames=None, init_func=None, fargs=None, save_count=None,
**kwargs)
# func : The function to call at each frame
#
ani = animation.FuncAnimation(fig, updateFrame, init_func=init, interval=50)
plt.show()
if cap.isOpened():
    cap.release()
```

Animation.FuncAnimation [클래스 생성]

```
import cv2
import matplotlib.pyplot as plt
import matplotlib.animation as animation

class Video:
    def __init__(self, device=0):
        self.cap = cv2.VideoCapture(device)
        self.retval, self.frame = self.cap.read()
        self.im = plt.imshow(cv2.cvtColor(self.frame, cv2.COLOR_BGR2RGB))
        print('start capture ...')

    def updateFrame(self, k):
        self.retval, self.frame = self.cap.read()
        self.im.set_array(cv2.cvtColor(camera.frame, cv2.COLOR_BGR2RGB))
#       return self.im,

    def close(self):
        if self.cap.isOpened():
            self.cap.release()
        print('finish capture.')

# 프로그램 시작
fig = plt.figure()
fig.canvas.set_window_title('Video Capture')
plt.axis("off")

camera = Video()
##camera = Video('./data/vtest.avi')
ani = animation.FuncAnimation(fig, camera.updateFrame, interval=50)
plt.show()
camera.close()
```

Animation.FuncAnimation [클래스 상속]

```
import cv2
import matplotlib.pyplot as plt
import matplotlib.animation as animation

class Video(animation.FuncAnimation):
    # __init__은 객체에 사용할 초기값들을 초기화 한다.
    def __init__(self, device=0, fig=None, frames=None,
                 interval=50, repeat_delay=5, blit=False, **kwargs):

        if fig is None:
            self.fig = plt.figure()
            self.fig.canvas.set_window_title('Video Capture')
            plt.axis("off")

        # 상위 클래스의 생성자를 호출하여 초기화한다.
        super(Video, self).__init__(self.fig, self.updateFrame, init_func=self.init,
                                    frames=frames, interval=interval, blit=blit,
                                    repeat_delay=repeat_delay, **kwargs)
        self.cap = cv2.VideoCapture(device)
        print("start capture ...")

    def init(self):
        retval, self.frame = self.cap.read()
        if retval:
            self.im = plt.imshow(cv2.cvtColor(self.frame, cv2.COLOR_BGR2RGB))
```

Animation.FuncAnimation [클래스 상속]

```
def updateFrame(self, k):
        retval, self.frame = self.cap.read()
        if retval:
            self.im.set_array(cv2.cvtColor(camera.frame, cv2.COLOR_BGR2RGB))
#       return self.im,

    def close(self):
        if self.cap.isOpened():
            self.cap.release()
        print("finish capture.")


# 프로그램 시작
camera = Video()
##camera = Video('./data/vtest.avi' )
plt.show()
camera.close()
```

참조 : Python의 클래스 생성

airtravel.py

```
class Flight:

    def __init__(self):
        print('init')
        super().__init__()

    def __new__(cls):
        print('new')
        return super().__new__(cls)

    def number(self):
        return 'SN060'
```

```
>>> from airtravel import Flight
>>> f = Flight()
>>> f.number()
```

결과값

```
new
init
SN060
```

cv2.split & cv2.merge

```python
import cv2

src = cv2.imread("data/lena.jpg", cv2.IMREAD_COLOR)
b, g, r = cv2.split(src)
inversebgr = cv2.merge((r, g, b))

cv2.imshow("b", b)
cv2.imshow("g", g)
cv2.imshow("r", r)
cv2.imshow("inverse", inversebgr)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

cv2.split & cv2.merge

```python
import cv2
import numpy as np

src = cv2.imread("data/lena.jpg", cv2.IMREAD_COLOR)
b, g, r = cv2.split(src)
imgorg = cv2.merge((b,g,r))
cv2.imshow("ORG", imgorg)

height, width, channel = src.shape
print(height, width)
zero = np.zeros((height, width, 1), dtype = np.uint8)

imgB = cv2.merge((b, zero, zero))
imgG = cv2.merge((zero, b, zero))
imgR = cv2.merge((zero, zero, r))

cv2.imshow("b", imgB)
cv2.imshow("g", imgG)
cv2.imshow("r", imgR)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

파이썬의 또다른 이미지 패키지 Pillow

```
$ pip install Pillow
```

이미지 파일 읽고 쓰기

```
from PIL import Image

# 이미지 열기
im = Image.open('data/lena.jpg' )

# 이미지 크기 및 이미지 출력
print(im.size)
im.show()

# 이미지 JPG로 저장
im.save('data/pillow_Lena.jpg' )
```

Thumbnail 이미지 생성

```
from PIL import Image

im = Image.open('data/lena.jpg' )

# Thumbnail 이미지 생성
size = (64, 64)
im.thumbnail(size)
im.show()

im.save('data/lena-thumb.jpg' )
```

이미지 crop / resize / rotate

```
from PIL import Image
im = Image.open('data/lena.jpg ' )
cropImage = im.crop((100, 100, 350, 350))
cropImage.show()
cropImage.save('data/lena-crop.jpg' )
```

```
from PIL import Image

im = Image.open('data/lena.jpg ' )

# 크기를 600x600 으로
img2 = im.resize((600, 600))
img2.save('lena-1000.jpg ' )

# 90도 회전
img3 = im.rotate(90)
img3.save('lena-rotate.jpg' )
```

이미지 필터링(blur)

```
from PIL import Image, ImageFilter

im = Image.open('data/lena.jpg' )
blurImage = im.filter(ImageFilter.BLUR)
blurImage.show()
blurImage.save( 'data/lena-blur.jpg' )
```

Pillow Alpha blend

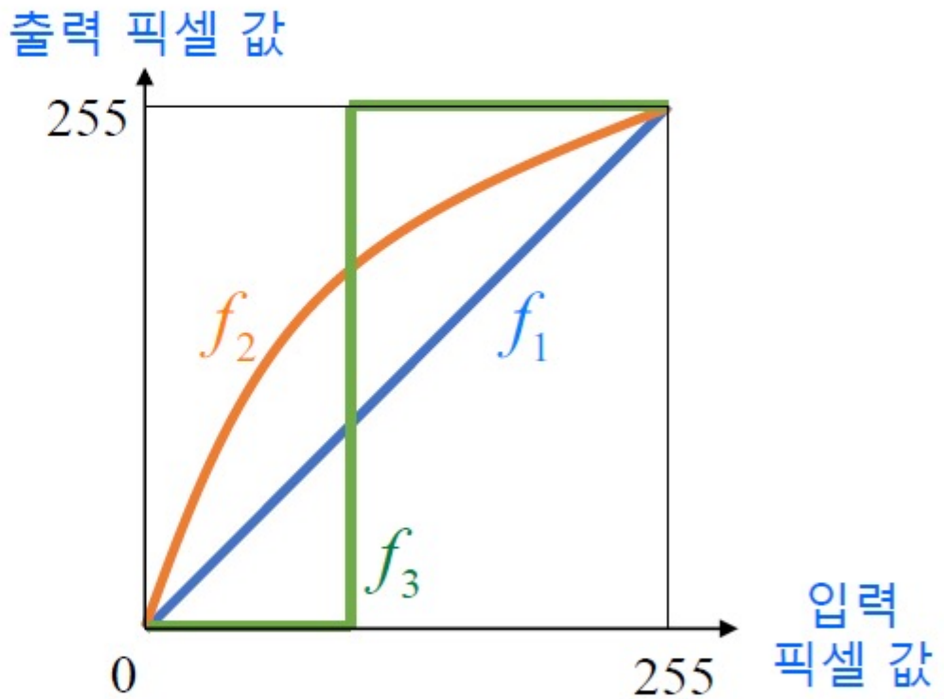
```
from PIL import Image

# Take two images for blending them together
image1 = Image.open("./data/red.jpg")
image2 = Image.open("./data/green.jpg")

# Make sure images got an alpha channel
image5 = image1.convert("RGBA")
image6 = image2.convert("RGBA")

# alpha-blend the images with varying values of alpha
alphaBlended1 = Image.blend(image5, image6, alpha=.2)
alphaBlended2 = Image.blend(image5, image6, alpha=.8)

# Display the alpha-blended images
alphaBlended1.show()
alphaBlended2.show()
```

영상의 화소처리 방법

- 화소 처리(Point processing)
  - 입력 영상의 특정 좌표 픽셀 값을 변경하여 출력 영상의 해당 좌표 픽셀 값으로 설정하는 연산

$$\mathrm{dst}(x, y) = f(\mathrm{src}(x, y))$$

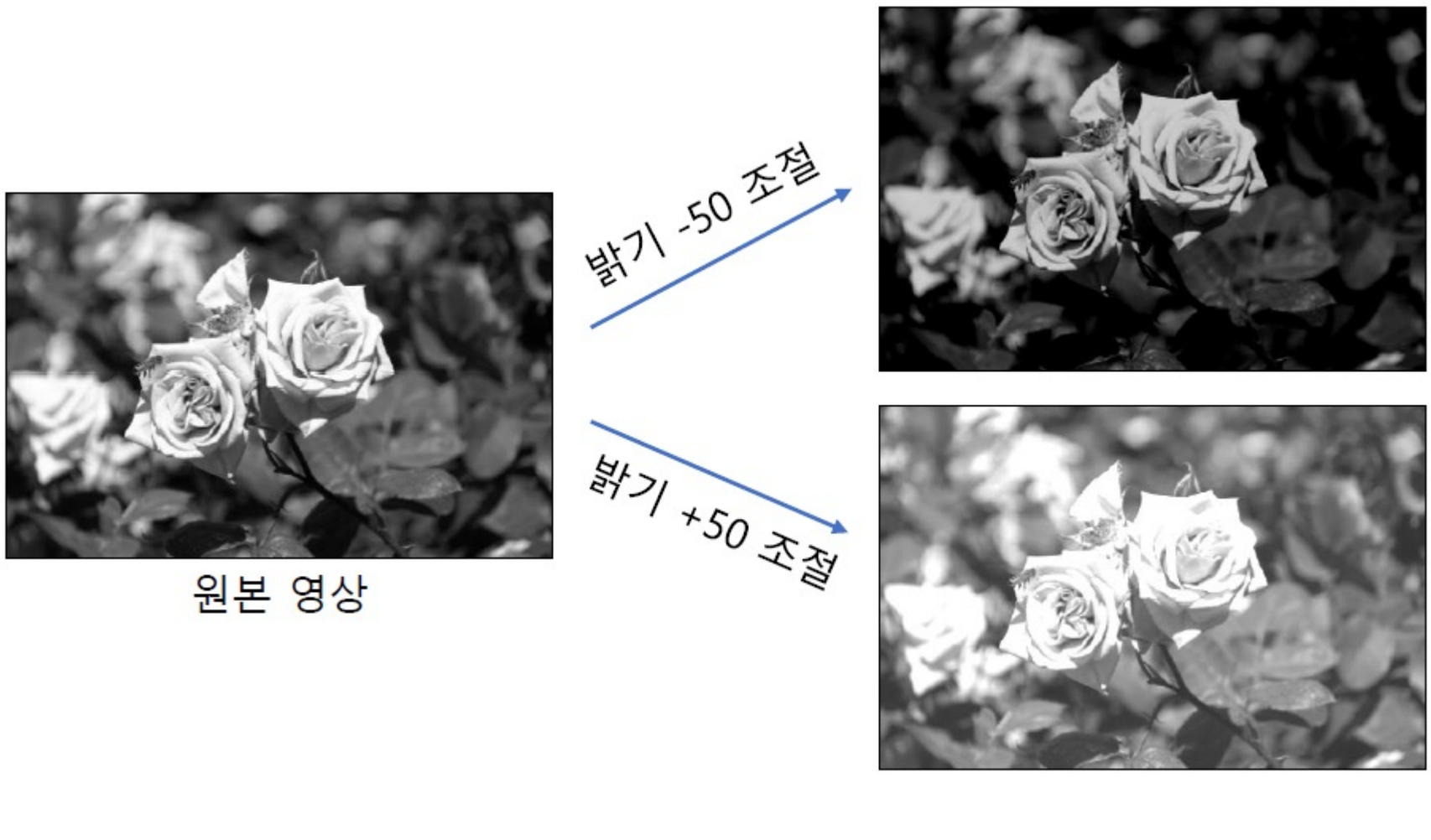
변환 함수
(transfer function)



- 결과 영상의 픽셀 값이 정해진 범위(e.g. 그레이스케일)에 있어야 함
- 반전, 밝기 조절, 명암비 조절 등

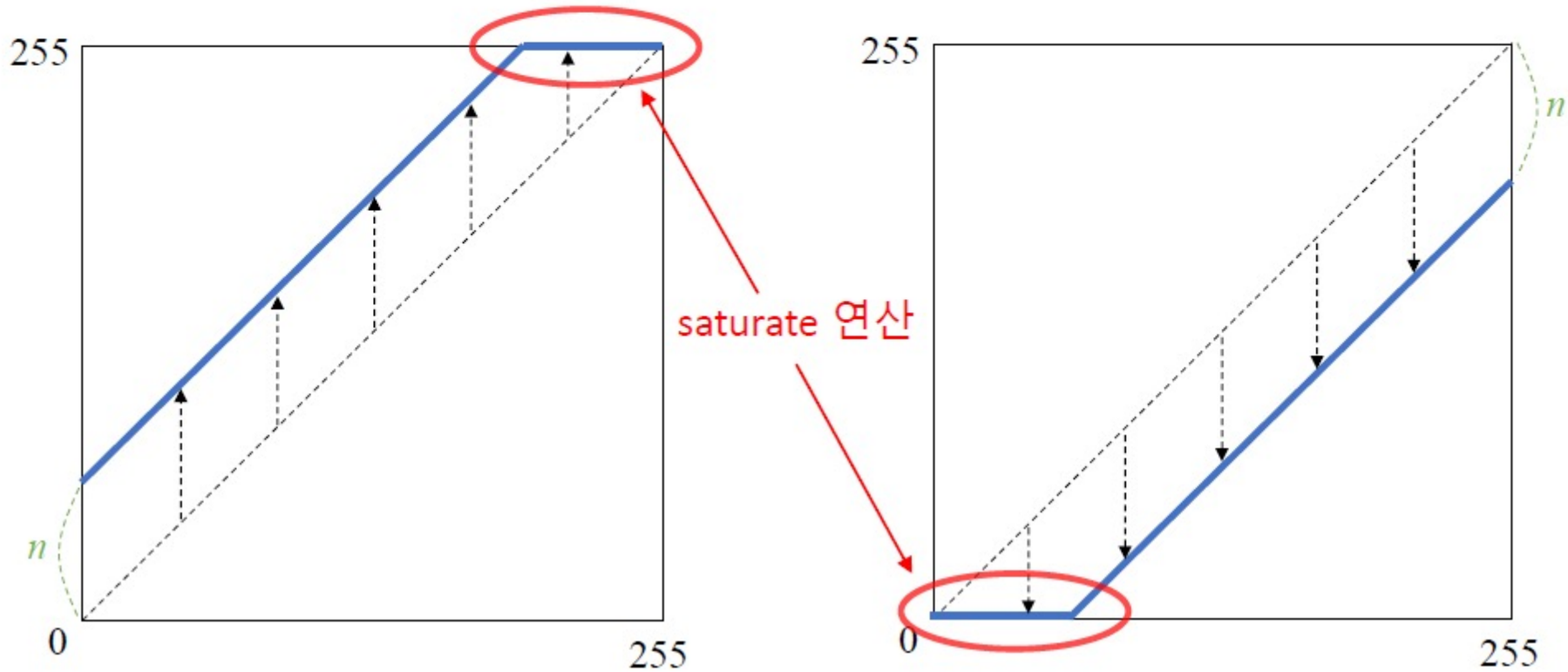
영상의 밝기 조절

- 밝기 조절이란?
- 영상을 전체적으로 더욱 밝거나 어둡게 만드는 연산



영상의 밝기 조절

$$\mathrm{dst}(x, y) = \mathrm{saturate}(\mathrm{src}(x, y) + n)$$



영상의 밝기 조절

- 영상의 밝기 조절을 위한 영상의 덧셈 연산

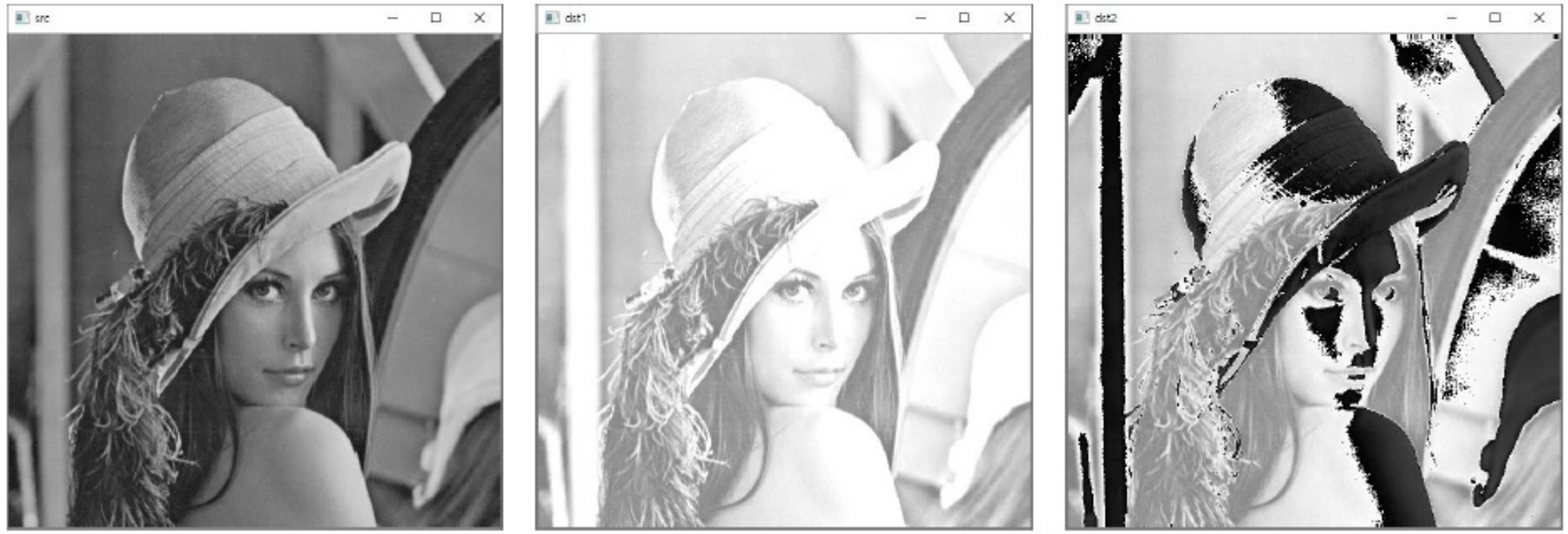
```
cv2.add(src1, src2, dst=None, mask=None, dtype=None) -> dst
```

  - src1: (입력) 첫 번째 영상 또는 스칼라
  - src2: (입력) 두 번째 영상 또는 스칼라
  - dst: (출력) 덧셈 연산의 결과 영상
  - mask: 마스크 영상
  - dtype: 출력 영상(dst)의 타입. (e.g.) cv2.CV_8U, cv2.CV_32F 등
  - 참고사항
    - 스칼라(Scalar)는 실수 값 하나 또는 실수 값 네 개로 구성된 튜플
    - dst를 함수 인자로 전달하려면 dst의 크기가 src1, src2와 같아야 하며, 타입이 적절해야 함

영상의 밝기 조절

- 그레이 스케일 영상의 밝기 100만큼 증가 시키기

```
src = cv2.imread('lenna.bmp', cv2.IMREAD_GRAYSCALE)

dst1 = cv2.add(src, 100)

dst2 = src + 100
#dst2 = np.clip(src + 100., 0, 255).astype(np.uint8)
```

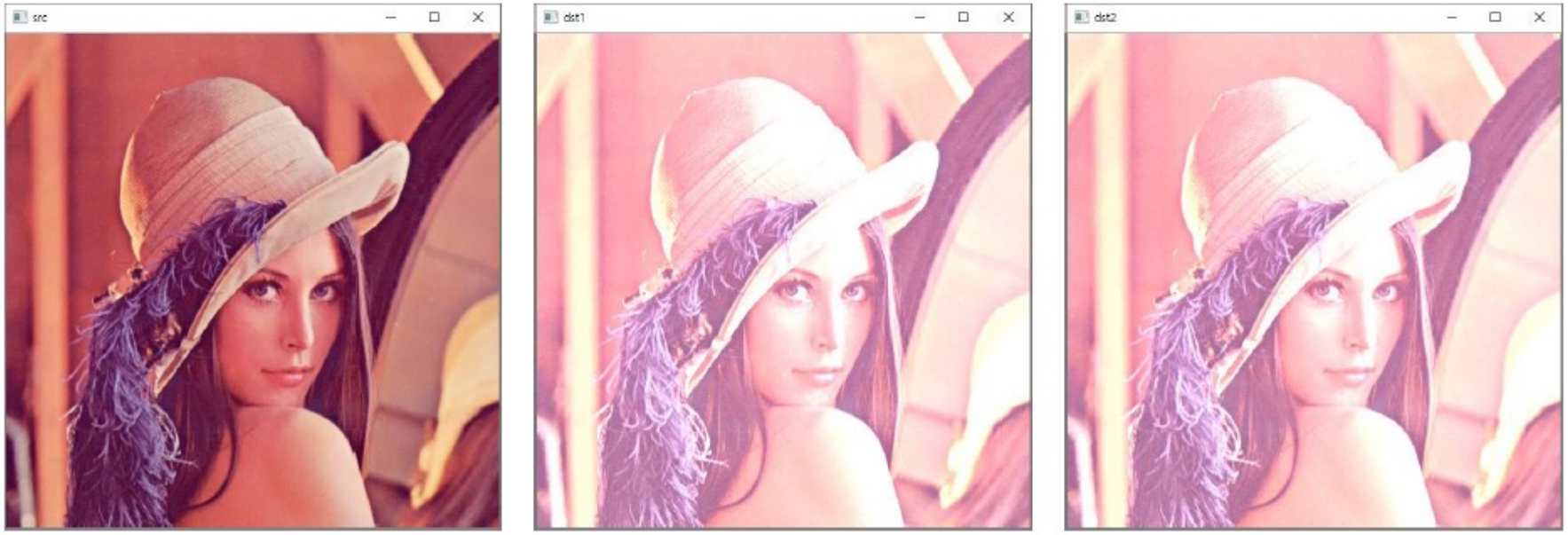


brightness.py

영상의 밝기 조절

- 컬러 영상의 밝기100만큼 증가 시키기

```
src = cv2.imread('lenna.bmp')

dst1 = cv2.add(src, (100, 100, 100, 0))

dst2 = np.clip(src + 100., 0, 255).astype(np.uint8)
```



brightness.py

영상의 밝기 조절

- **덧셈 연산**

$$\mathrm{dst}(x, y) = \mathrm{saturate}(\mathrm{src1}(x, y) + \mathrm{src2}(x, y))$$

  - 두 영상의 같은 위치에 존재하는 픽셀 값을 더하여 결과 영상의 픽셀 값으로 설정
  - 덧셈 결과가 255보다 크면 픽셀 값을 255로 설정 (포화 연산)

영상의 산술 연산

- **덧셈 연산**

```
cv2.add(src1, src2, dst=None, mask=None, dtype=None) -> dst
```

  - src1: (입력) 첫 번째 영상 또는 스칼라
  - src2: (입력) 두 번째 영상 또는 스칼라
  - dst: (출력) 덧셈 연산의 결과 영상
  - mask: 마스크 영상
  - dtype: 출력 영상(dst)의 타입. (e.g.) cv2.CV_8U, cv2.CV_32F 등
  - 참고사항
    - 스칼라(Scalar)는 실수 값 하나 또는 실수 값 네 개로 구성된 튜플
    - dst를 함수 인자로 전달하려면 dst의 크기가 src1, src2와 같아야 하며, 타입이 적절해야 함

영상의 산술 연산

- 가중치 합(weighted sum)

$$\mathrm{dst}(x, y) = \mathrm{saturate}(\alpha \cdot \mathrm{src1}(x, y) + \beta \cdot \mathrm{src2}(x, y))$$

  - 두 영상의 같은 위치에 존재하는 픽셀 값에 대하여 가중합을 계산하여 결과 영상의 픽셀 값으로 설정
  - 보통 $\alpha + \beta = 1$ 이 되도록 설정 ➔ 두 입력 영상의 평균 밝기를 유지

- 평균 연산(average)
  - 가중치를 $\alpha = \beta = 0.5$ 로 설정한 가중치 합

$$\mathrm{dst}(x, y) = \frac{1}{2}\left(\mathrm{src1}(x, y) + \mathrm{src2}(x, y)\right)$$

영상의 산술 연산

- 가중치 합

$$\frac{1}{2}\left(\text{  } + \text{  }\right) = \text{  }$$


$\alpha = 1, \beta = 0$


$\alpha = 0.75, \beta = 0.25$


$\alpha = 0.5, \beta = 0.5$


$\alpha = 0.25, \beta = 0.75$


$\alpha = 0, \beta = 1$

영상의 산술 연산

- 가중치 합(weighted sum)

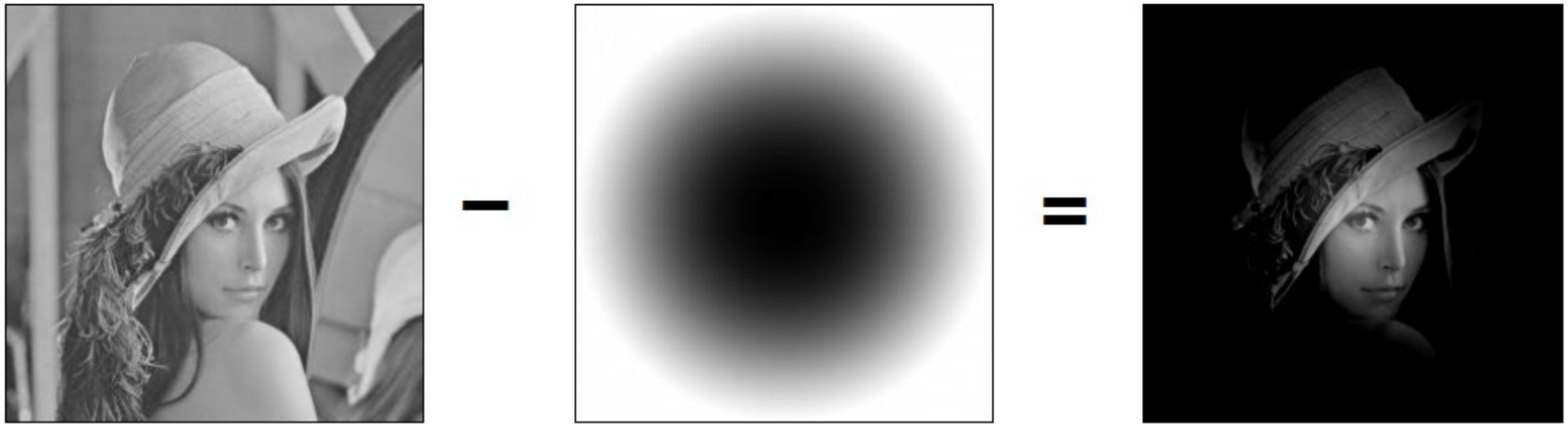
```
cv2.addWeighted(src1, alpha, src2, beta, gamma, dst=None, dtype=None) -> dst
```

- src1: 첫 번째 영상
- alpha: 첫 번째 영상 가중치
- src2: 두 번째 영상. src1과 같은 크기 & 같은 타입
- beta: 두 번째 영상 가중치
- gamma: 결과 영상에 추가적으로 더할 값
- dst: 가중치 합 결과 영상
- dtype: 출력 영상(dst)의 타입

영상의 산술 연산

$$\mathrm{dst}(x, y) = \mathrm{saturate}(\mathrm{src1}(x, y) - \mathrm{src2}(x, y))$$

- 두 영상의 같은 위치에 존재하는 픽셀 값에 대하여 뺄셈 연산을 수행하여 결과 영상의 픽셀 값으로 설정
- 뺄셈 결과가 0보다 작으면 픽셀 값을 0으로 설정 (포화 연산)



영상의 산술 연산

▪ 뺄셈 연산

```
cv2.subtract(src1, src2, dst=None, mask=None, dtype=None) -> dst
```

- src1: 첫 번째 영상 또는 스칼라
- src2: 두 번째 영상 또는 스칼라
- dst: 뺄셈 연산 결과 영상
- mask: 마스크 영상
- dtype: 출력 영상(dst)의 타입

컬러 영상과 색 공간

- **RGB 색상 평면**

B 평면

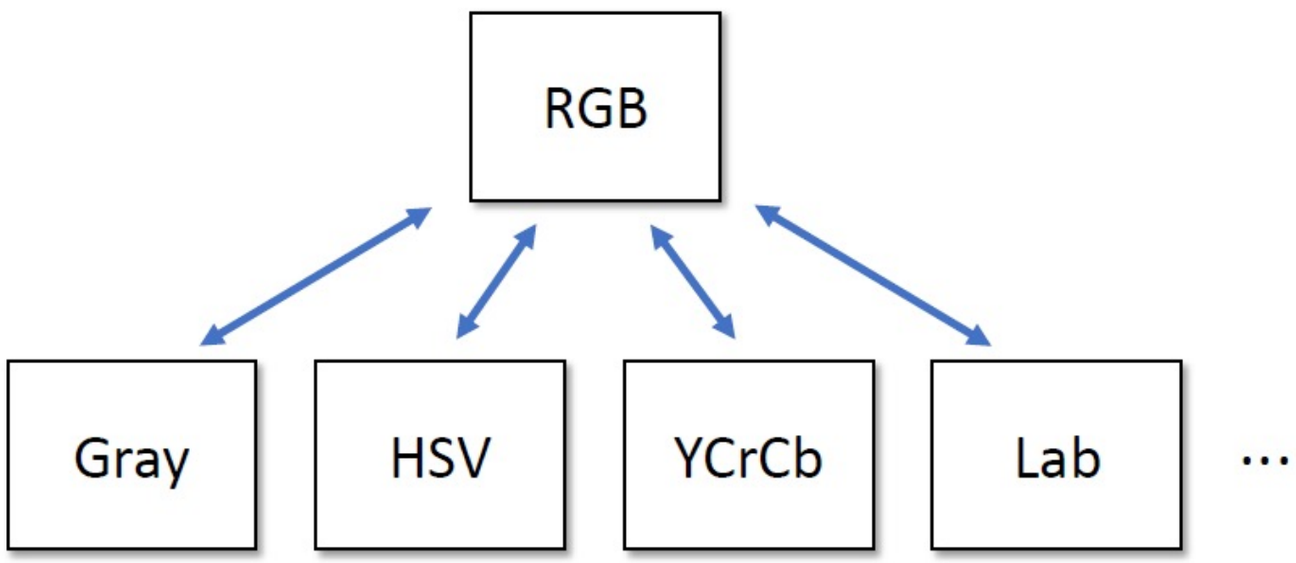
G 평면

R 평면

컬러 영상과 색 공간

- **색공간 변환**

  영상처리를 위한 목적으로 RGB 색 공간을 HSV, YCbCr, Gray등의 색공간으로 변환하는 것



컬러 영상과 색 공간

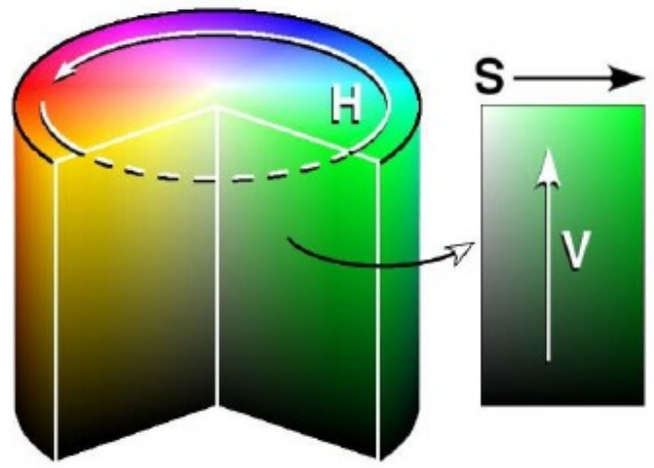
- **RGB 색상을 그레이 스케일로 변환**

$$Y = 0.299R + 0.587G + 0.114B$$

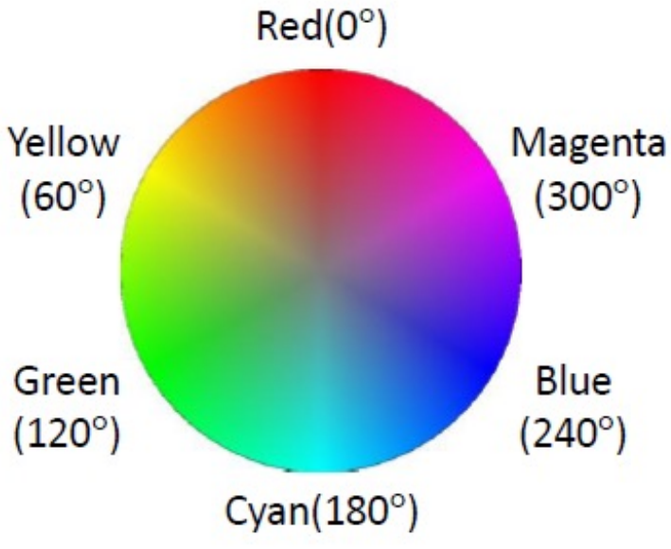
  - 장점 : 데이터 저장 용량 감소, 데이터 처리 속도 향상
  - 단점 : 색 정보 손실

컬러 영상과 색 공간

- HSV 색 공간
  - Hue: 색상, 색의 종류
  - Saturation: 채도, 색의 탁하고 선명한 정도
  - Value: 명도, 빛의 밝기



- HSV 값 범위
  - cv2.CV_8U 영상의 경우
    - $0 \leq H \leq 179$
    - $0 \leq S \leq 255$
    - $0 \leq V \leq 255$



https://ko.wikipedia.org/wiki/HSV_%EC%83%89_%EA%B3%B5%EA%B0%84

히스토그램 분석

- **히스토그램(Histogram)**

  영상의 픽셀 값 분포를 그래프의 형태로 표현한 것
  그레이 스케일 영상의 경우, 각 그레이 스케일 값에 해당하는 픽셀의 개수를 구하고, 이를 막대 그래프의 형태로 표현

$$h(g) = N_g$$

0	0	0	2
1	1	2	0
1	5	6	5
3	6	7	6

히스토그램 분석

▪ 정규화된 히스토그램(Normalized histogram)

- 각 픽셀의 개수를 영상 전체 픽셀 개수로 나누어 준 것
- 해당 그레이 스케일 값을 갖는 픽셀이 나타날 확률

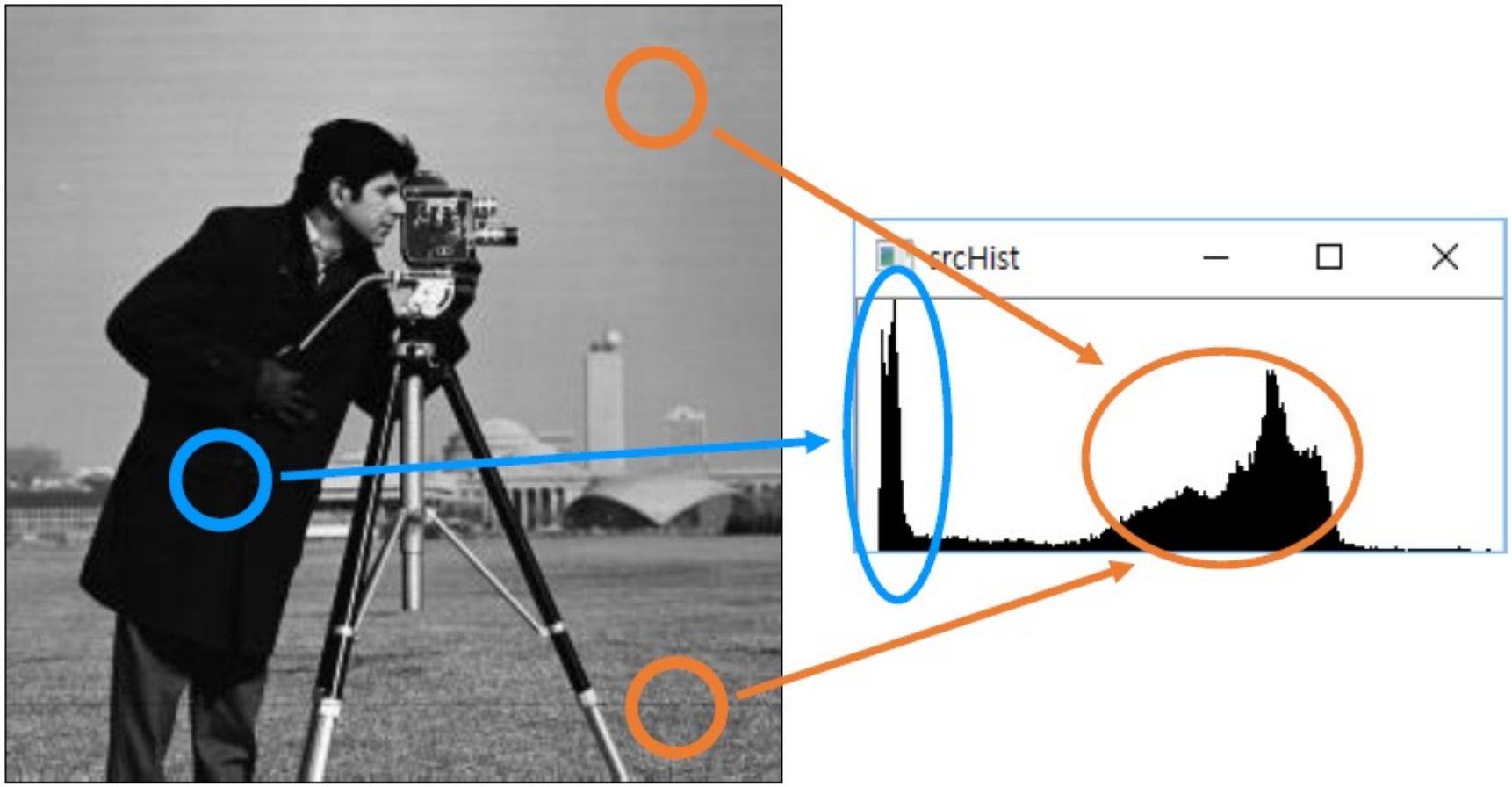
$$p(g) = \frac{N_g}{w \times h} \quad \Rightarrow \quad \sum_{g=0}^{L-1} p(g) = 1$$

0	0	0	2
1	1	2	0
1	5	6	5
3	6	7	6

g	0	1	2	3	4	5	6	7
$h(g)$	4	3	2	1	0	2	3	1
$p(g)$	$\frac{4}{16}$	$\frac{3}{16}$	$\frac{2}{16}$	$\frac{1}{16}$	0	$\frac{2}{16}$	$\frac{3}{16}$	$\frac{1}{16}$

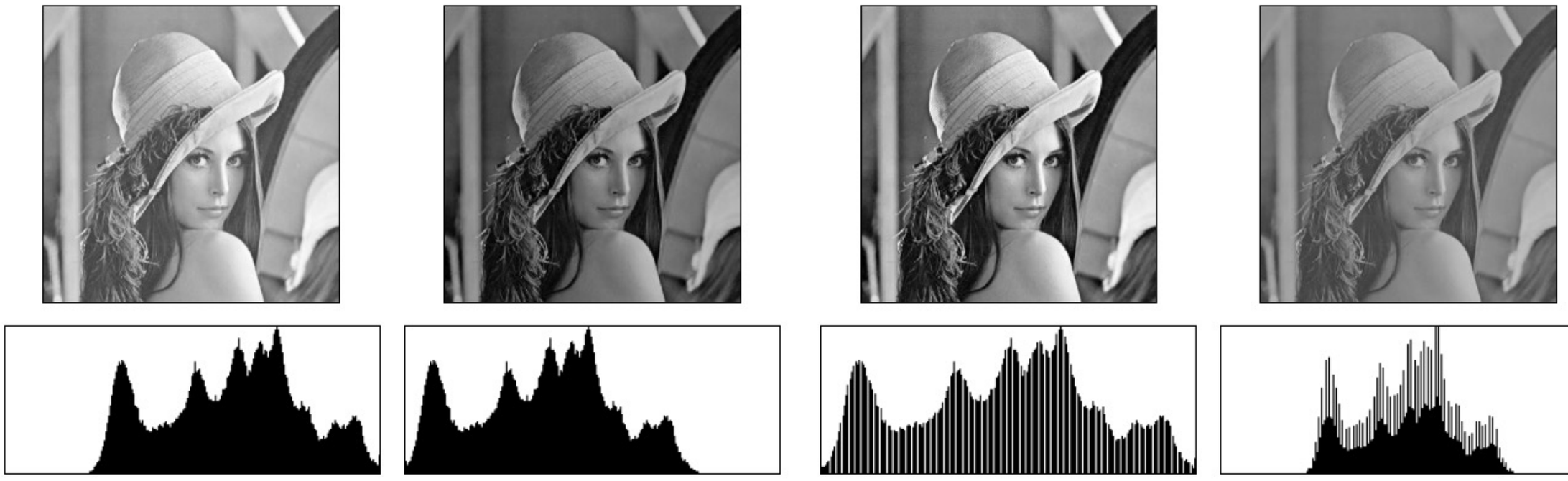
히스토그램 분석

- 영상과 히스토그램의 관계



히스토그램 분석

- 영상과 히스토그램의 관계



히스토그램 분석

▪ 히스토그램 구하기

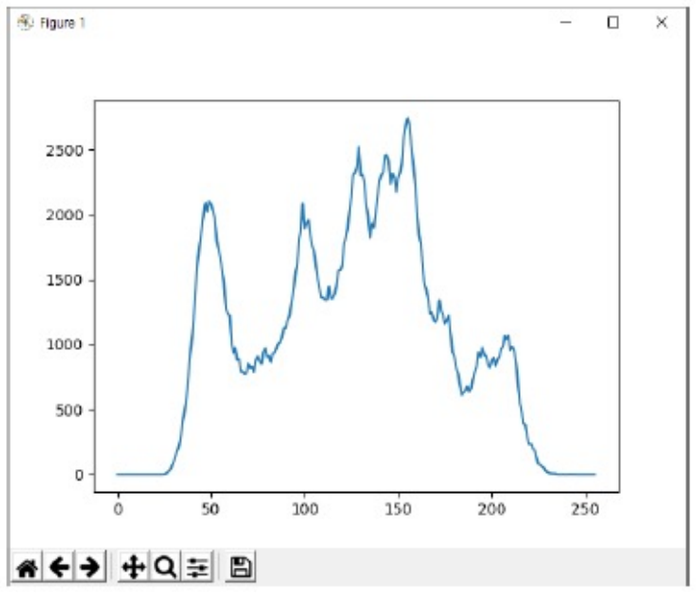
```
cv2.calcHist(images, channels, mask, histSize, ranges, hist=None,
             accumulate=None) -> hist
```

- images: 입력 영상 리스트
- channels: 히스토그램을 구할 채널을 나타내는 리스트
- mask: 마스크 영상. 입력 영상 전체에서 히스토그램을 구하려면 None 지정.
- histSize: 히스토그램 각 차원의 크기(빈(bin)의 개수)를 나타내는 리스트
- ranges: 히스토그램 각 차원의 최솟값과 최댓값으로 구성된 리스트
- hist: 계산된 히스토그램 (numpy.ndarray)
- accumulate: 기존의 hist 히스토그램에 누적하려면 True, 새로 만들려면 False.

히스토그램 분석

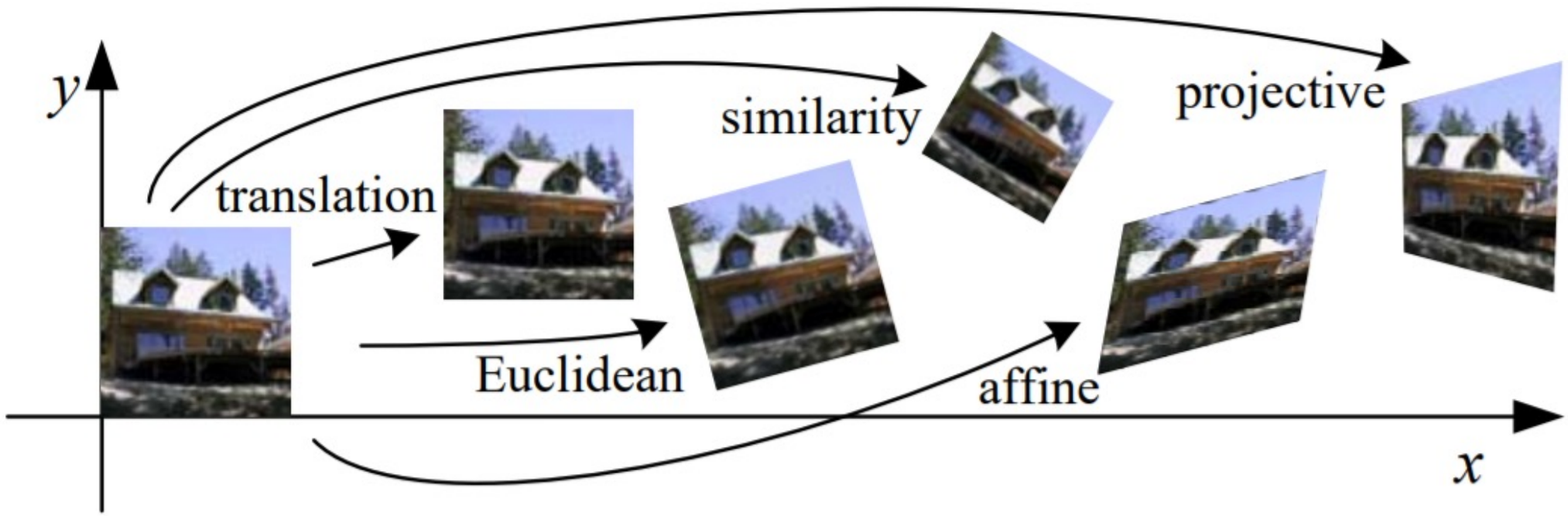
- 그레이스케일 영상의 히스토그램 구하기

```
src = cv2.imread('lenna.bmp', cv2.IMREAD_GRAYSCALE)

hist = cv2.calcHist([src], [0], None, [256], [0, 256])

plt.plot(hist)
plt.show()
```



영상의 기하학적 변환

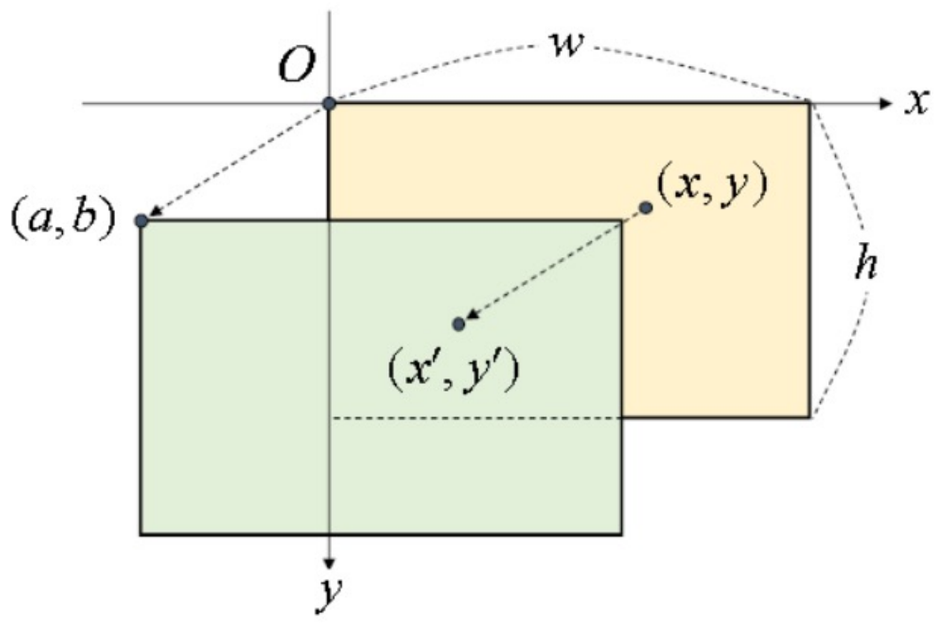
- 영상의 기하학적 변환



http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

영상의 이동 변환(Translation transformation)

- 가로 또는 세로 방향으로 영상을 특정 크기만큼 이동시키는 변환
- X축과 Y축 방향으로의 이동 변위를 지정



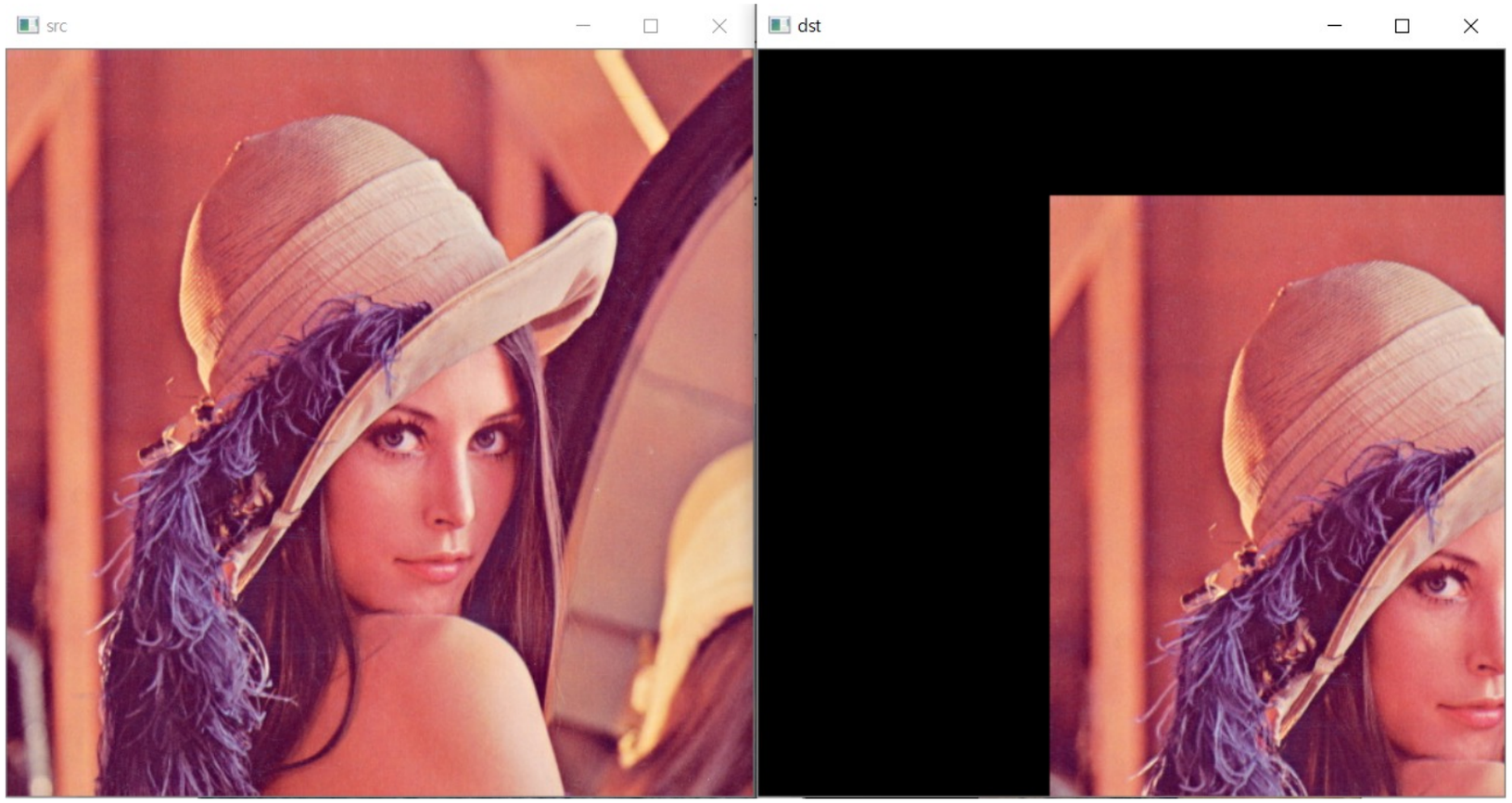
$$\begin{cases} x' = x + a \\ y' = y + b \end{cases} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2x3 어파인 변환 행렬

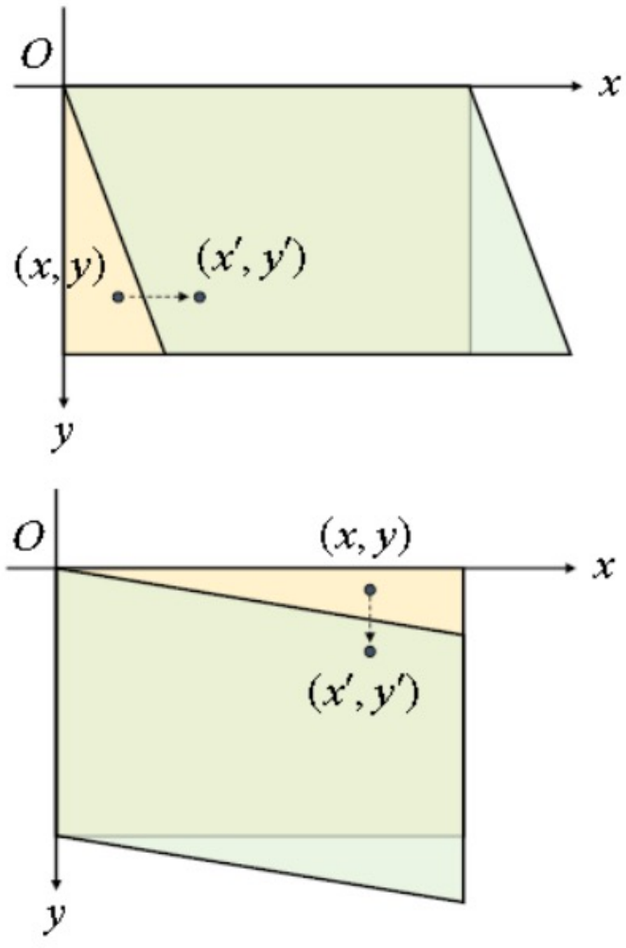
영상의 이동 변환(Translation transformation)

```
aff = np.array([[1, 0, 200],
                [0, 1, 100]], dtype=np.float32)

dst = cv2.warpAffine(src, aff, (0, 0))
```



영상의 전단 변환(Shear transformation)

- 층 밀림 변환, X축과 Y축 방향에 대해 각각 정의

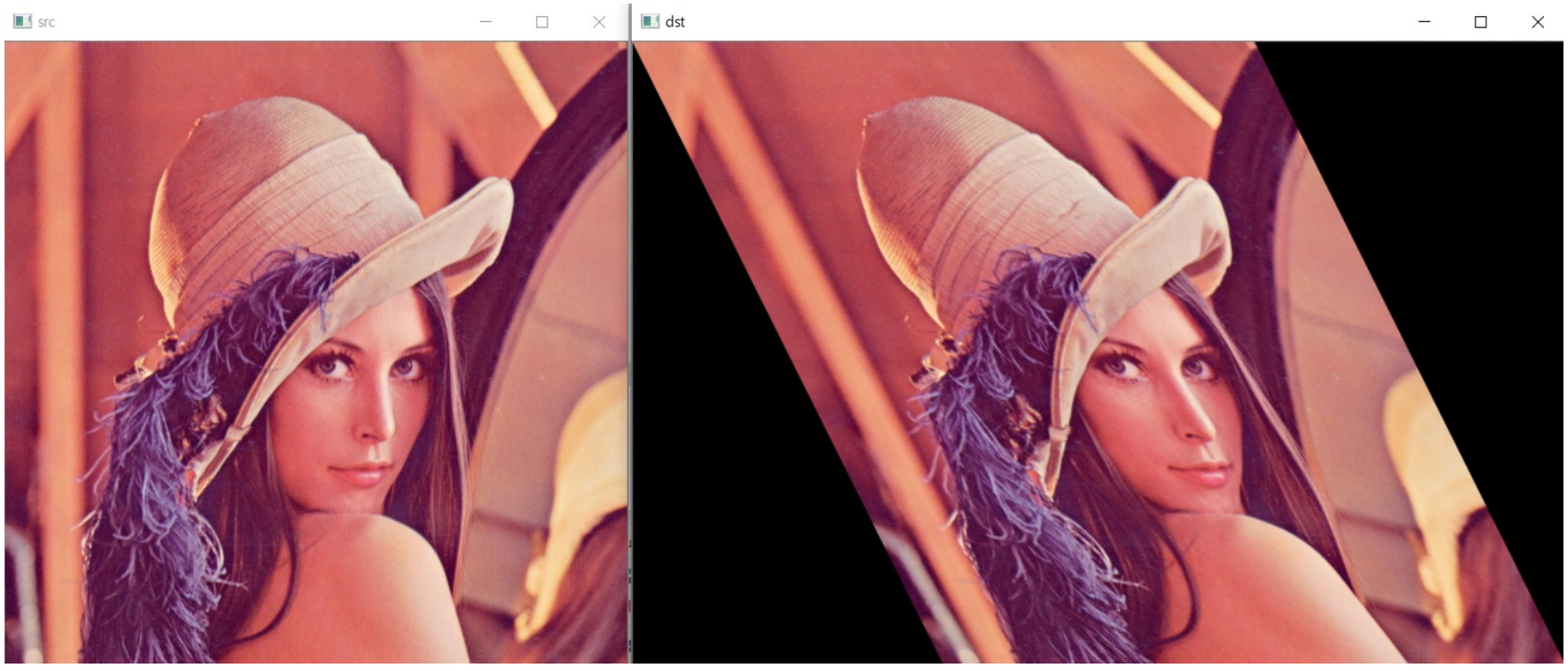


$$\begin{cases} x' = x + my \\ y' = y \end{cases} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & m & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{cases} x' = x \\ y' = mx + y \end{cases} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ m & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

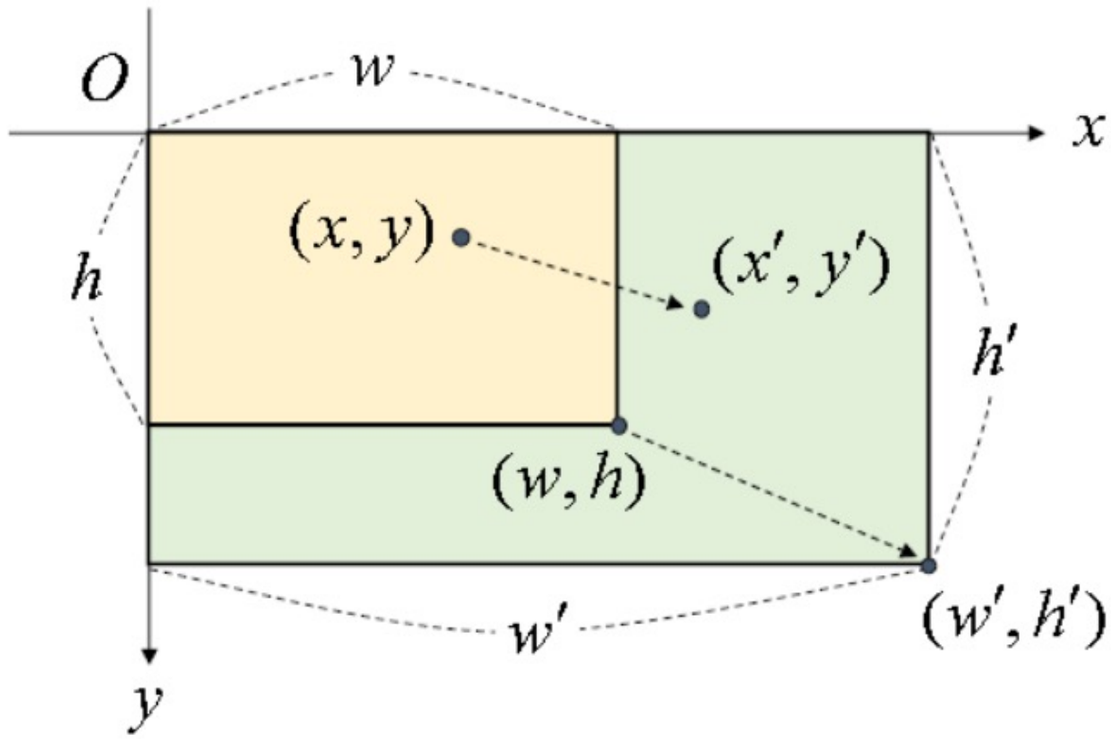
영상의 전단 변환(Shear transformation)

```
aff = np.array([[1, 0.5, 0],
                [0, 1, 0]], dtype=np.float32)

h, w = src.shape[:2]
dst = cv2.warpAffine(src, aff, (w + int(h * 0.5), h))
```



영상의 확대와 축소 (Scale transformation)

- 영상의 크기를 원본 영상보다 크거나 작게 만드는 변환

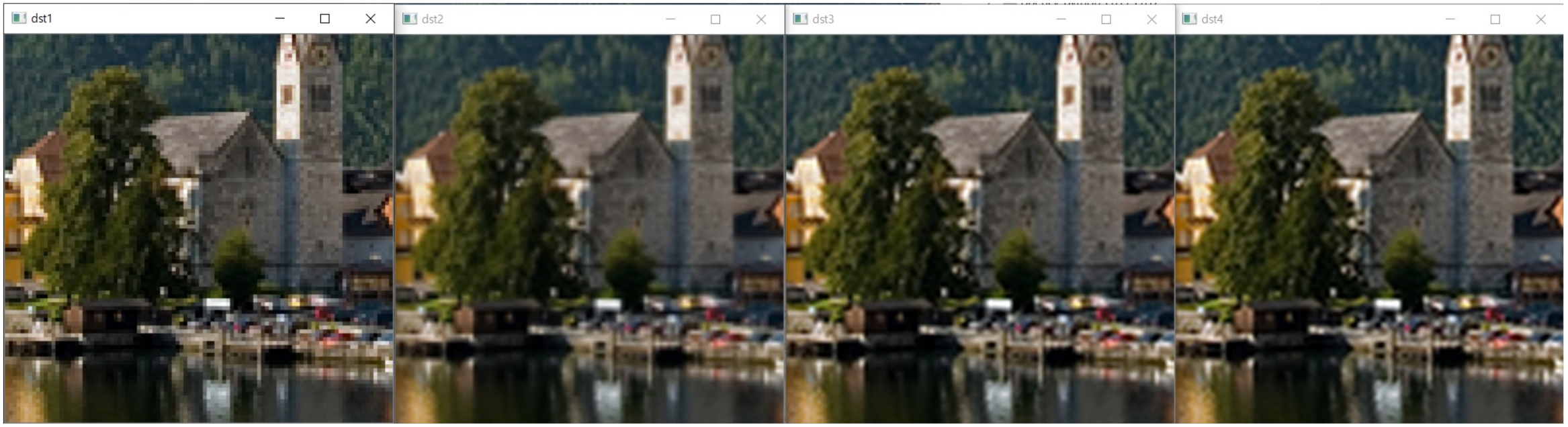


$$\begin{cases} x' = s_x x \\ y' = s_y y \end{cases} \qquad \begin{cases} s_x = w' / w \\ s_y = h' / h \end{cases}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

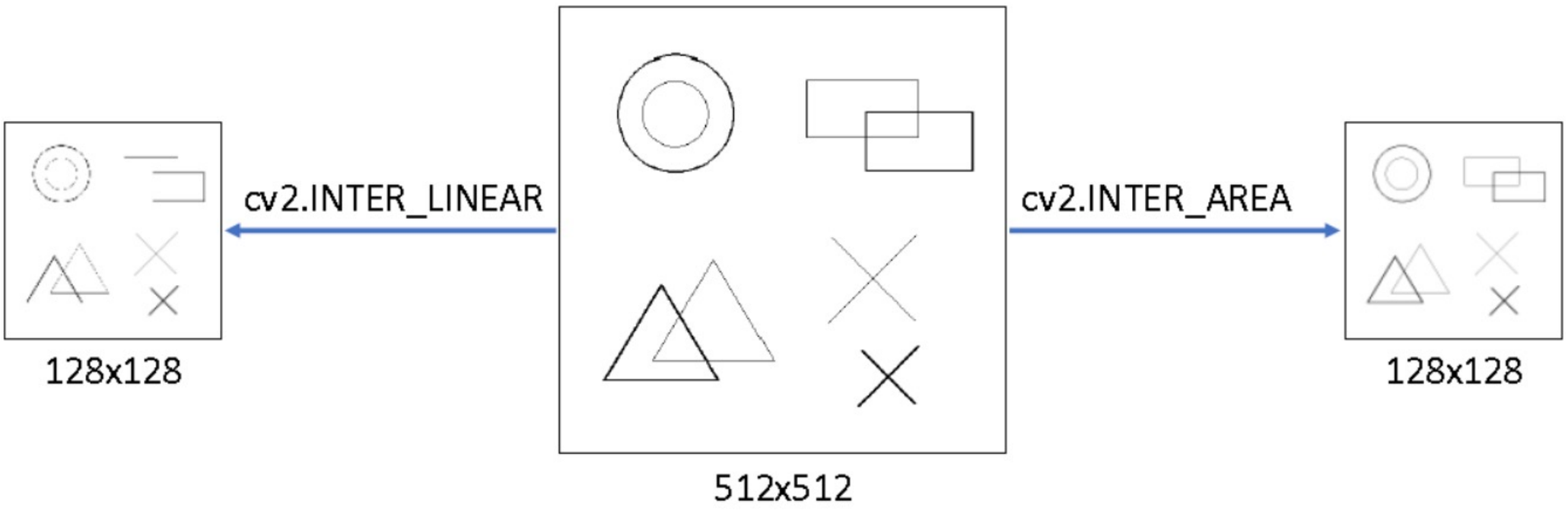
영상의 확대와 축소 (Scale transformation)

```
dst1 = cv2.resize(src, (0, 0), fx=4, fy=4, interpolation=cv2.INTER_NEAREST)
dst2 = cv2.resize(src, (1920, 1280))  # cv2.INTER_LINEAR
dst3 = cv2.resize(src, (1920, 1280), interpolation=cv2.INTER_CUBIC)
dst4 = cv2.resize(src, (1920, 1280), interpolation=cv2.INTER_LANCZOS4)
```



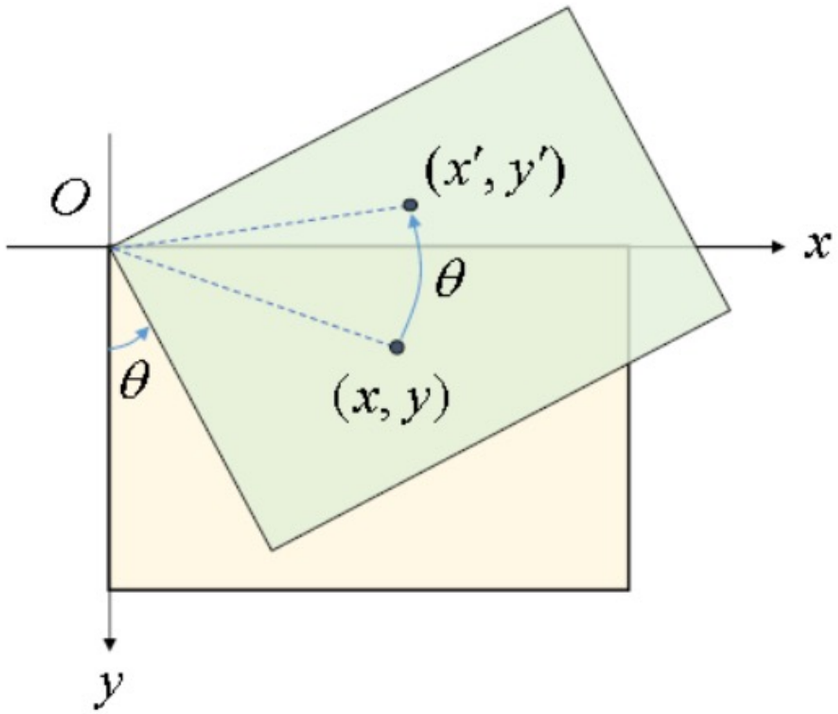
영상의 확대와 축소 (Scale transformation)

- 영상의 축소시 고려할 사항
- 영상 축소시 디테일이 사라지는 경우가 발생(특히, 한 픽셀로 구성된 선)
- 입력 영상을 부드럽게 필터링한 후 축소, 다단계 축소
- OpenCV의 cv2.resize()함수에서는 cv2.INTER_AREA 플래그를 사용



영상의 회전 (rotation transformation)

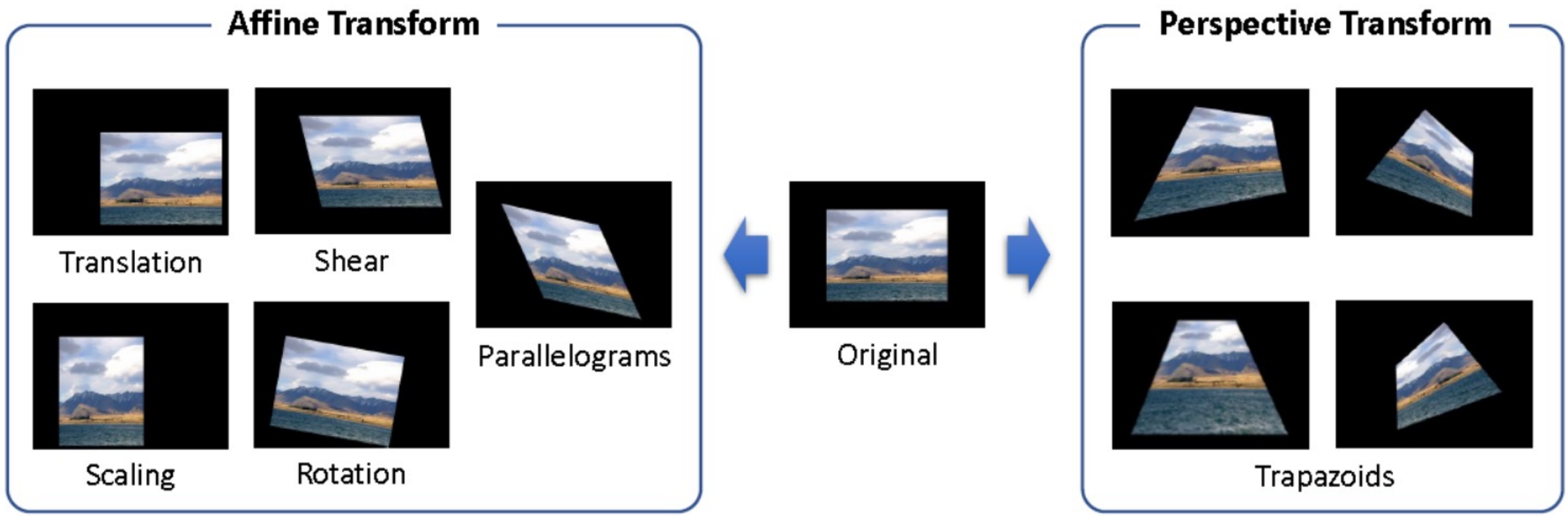
- 영상을 특정 각도만큼 회전시키는 변환(반시계 방향)



$$\begin{cases} x' = \cos\theta \cdot x + \sin\theta \cdot y \\ y' = -\sin\theta \cdot x + \cos\theta \cdot y \end{cases}$$

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

Affine Transform VS Perspective Transform



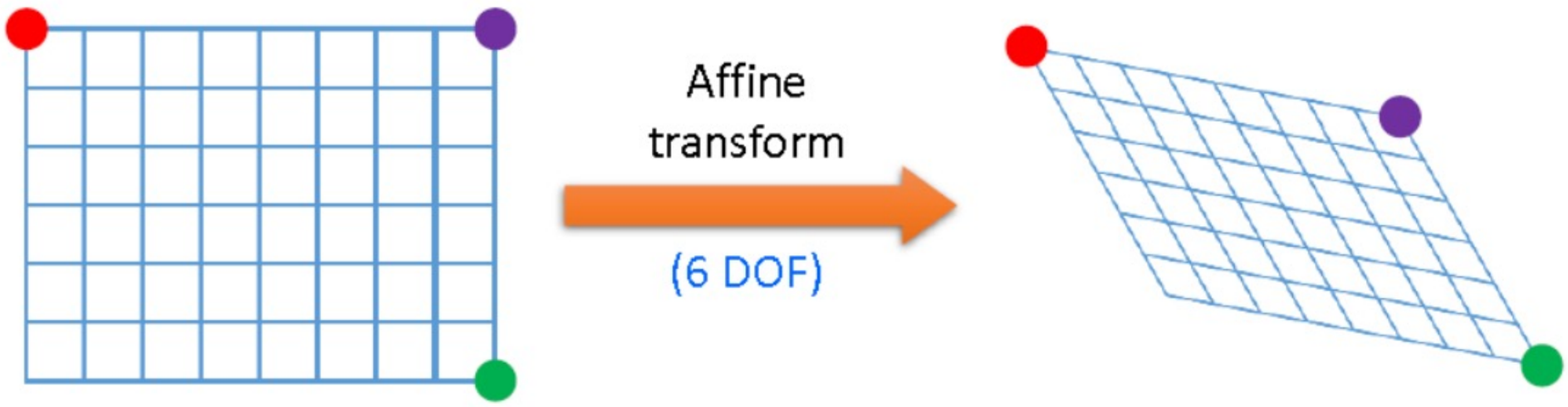
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

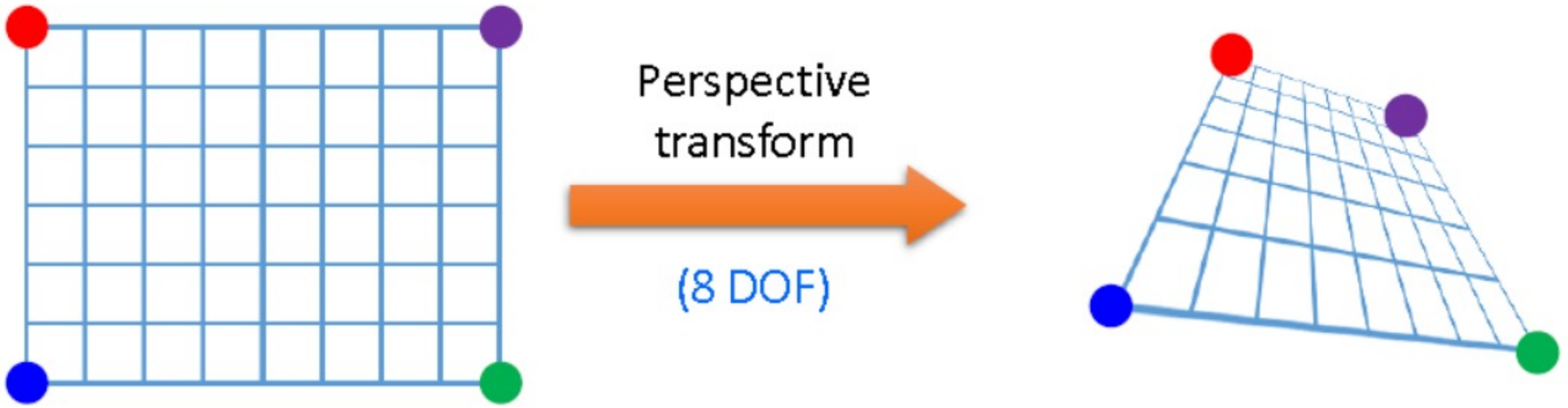
2x3 matrix
(6 DOF)

3x3 matrix
(8 DOF)

$$\begin{pmatrix} wx' \\ wy' \\ w \end{pmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Affine Transform VS Perspective Transform



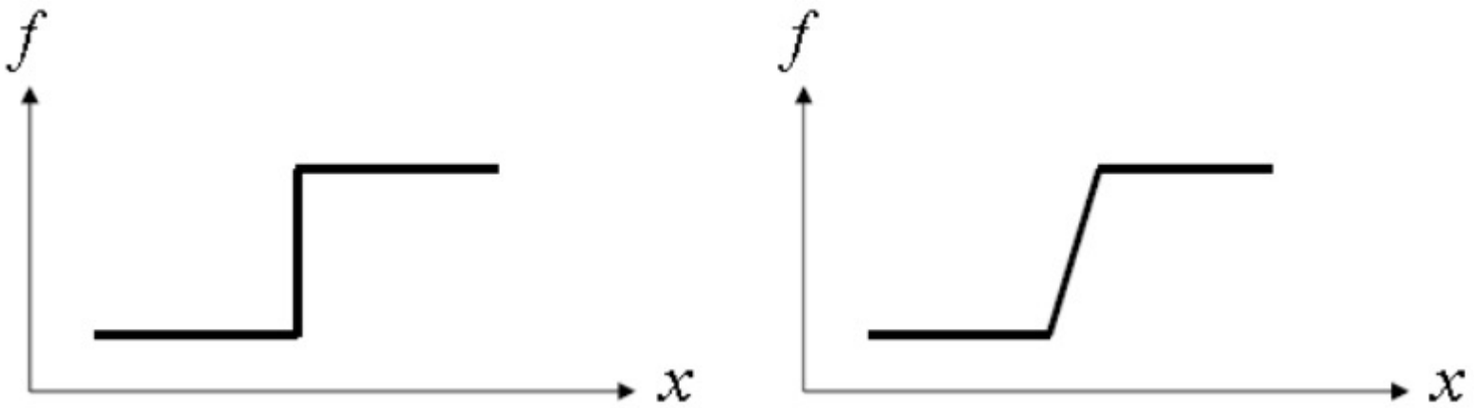


Affine Transform VS Perspective Transform

- getAffineTransform VS getPerspectiveTransform
- warpAffine VS warpPerspective

에지 검출과 미분

- 에지(edge)
- 영상에서 픽셀의 값이 급격하게 변하는 부분



영상의 미분과 소벨 필터

가로 방향:

-1	0	1
-1	0	1
-1	0	1

-1	0	1
-2	0	2
-1	0	1

-3	0	3
-10	0	10
-3	0	3

세로 방향:

-1	-1	-1
0	0	0
1	1	1

Prewitt

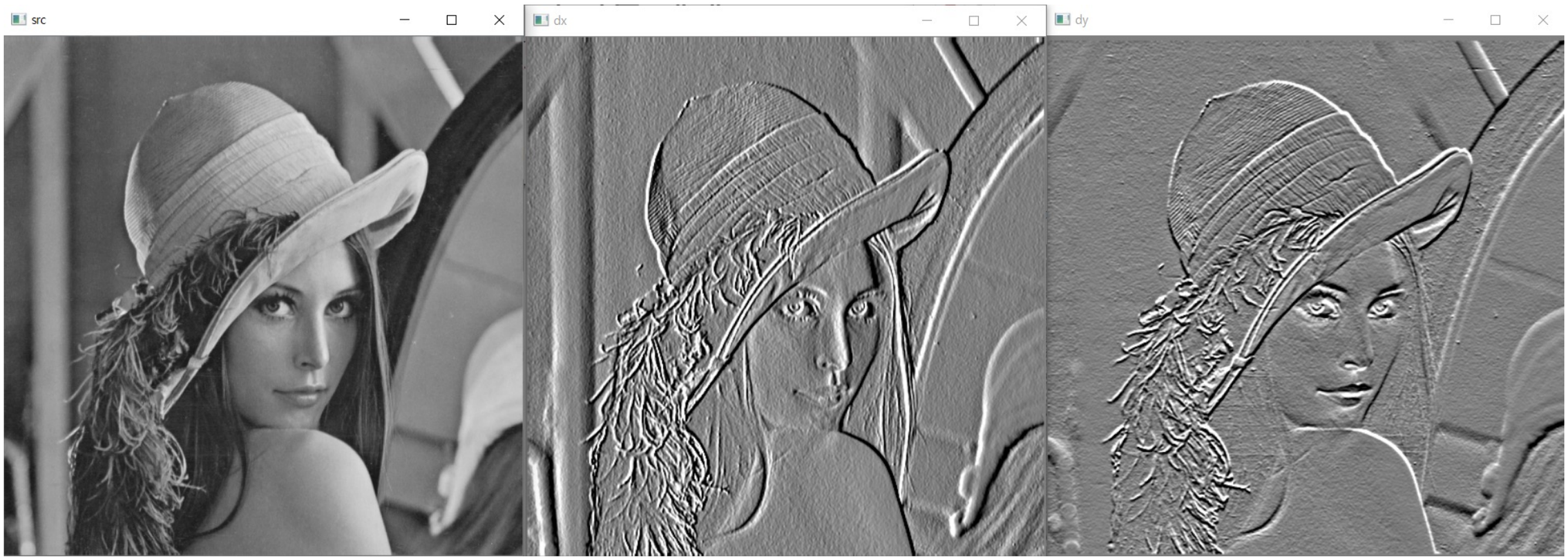
-1	-2	-1
0	0	0
1	2	1

Sobel

-3	-10	-3
0	0	0
3	10	3

Scharr

```
dx = cv2.Sobel(src, -1, 1, 0, delta=128)
dy = cv2.Sobel(src, -1, 0, 1, delta=128)
```

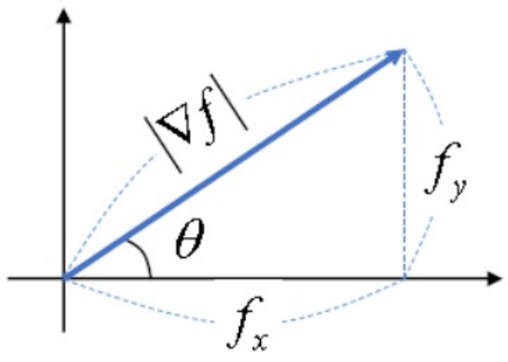


그래디언트

- 영상의 그래디언트
  - 함수 $f(x, y)$를 x축과 y 축으로 각각 편미분(partial derivative)하여 벡터 형태로 표현한 것

$$\nabla f = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = f_x \mathbf{i} + f_y \mathbf{j}$$

- 그래디언트 크기: $|\nabla f| = \sqrt{f_x^2 + f_y^2}$
- 그래디언트 방향: $\theta = \tan^{-1}\left(\frac{f_y}{f_x}\right)$



그래디언트와 에지 검출

```
dx = cv2.Sobel(src, cv2.CV_32F, 1, 0)
dy = cv2.Sobel(src, cv2.CV_32F, 0, 1)

mag = cv2.magnitude(dx, dy)
mag = np.clip(mag, 0, 255).astype(np.uint8)

dst = np.zeros(src.shape[:2], np.uint8)
dst[mag > 120] = 255
```

캐니 에지 검출 단계

- 1.가우시안 필터링 : 잡은 제거 목적
- 2. 그래디언트 계산 : 주로 소벨 마스크 사용

  크기: $\|f\| = \sqrt{f_x^2 + f_y^2}$

  방향: $\theta = \tan^{-1}\left(f_y / f_x\right)$
- 3. 비최대 억제(Non maximum suppression)

  하나의 에지가 여러 개의 픽셀로 표현되는 현상을 없애기 위해 그래디언트 크기가 국지적 최대인 픽셀

  만을 에지 픽셀로 설정
- 4. 히스테리시스 에지 트래킹

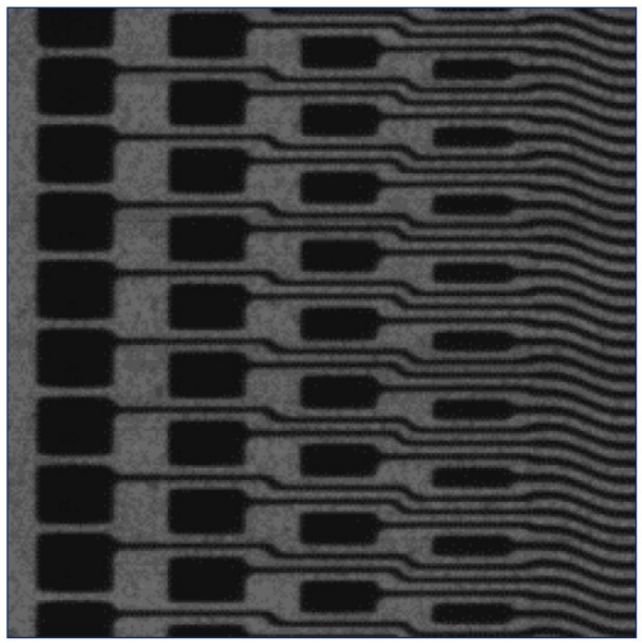
  두 개의 임계값을 사용: T_{Low}, T_{High}

  강한 에지: $\|f\| \geq T_{High}$ → 최종 에지로 선정

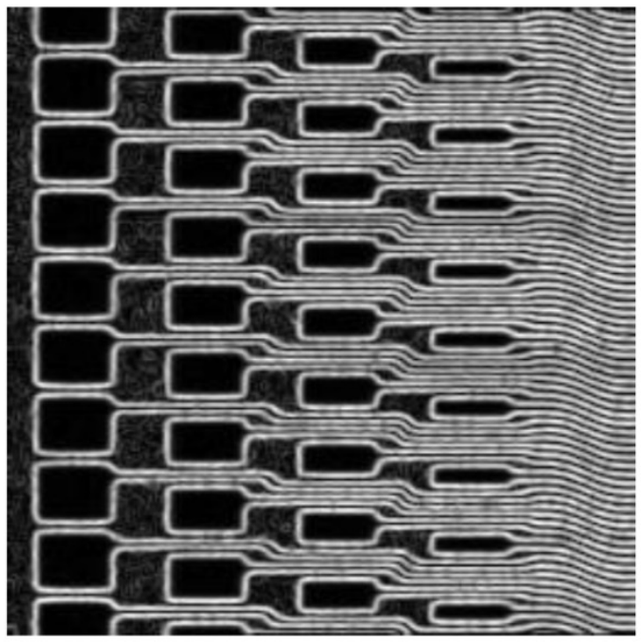
  약한 에지: $T_{Low} \leq \|f\| < T_{High}$

  → 강한 에지와 연결되어 있는 픽셀만 최종 에지로 선정

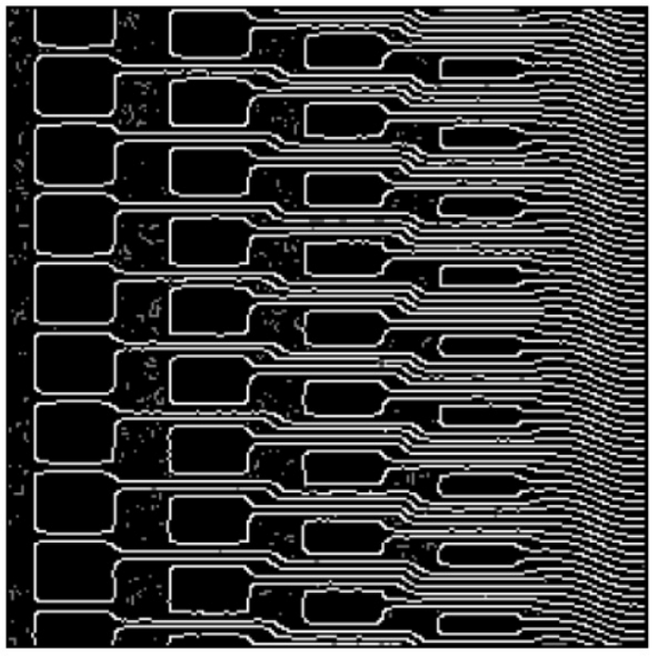
캐니 에지 검출 단계



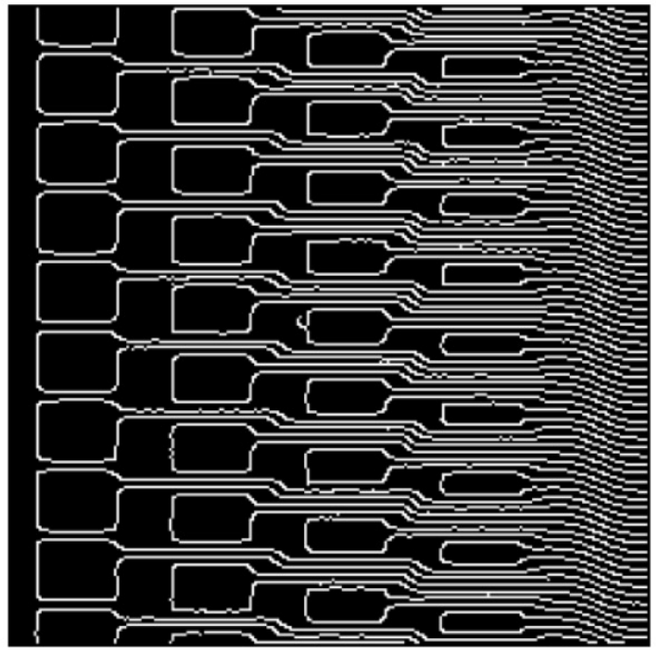
입력 영상



그래디언트 크기



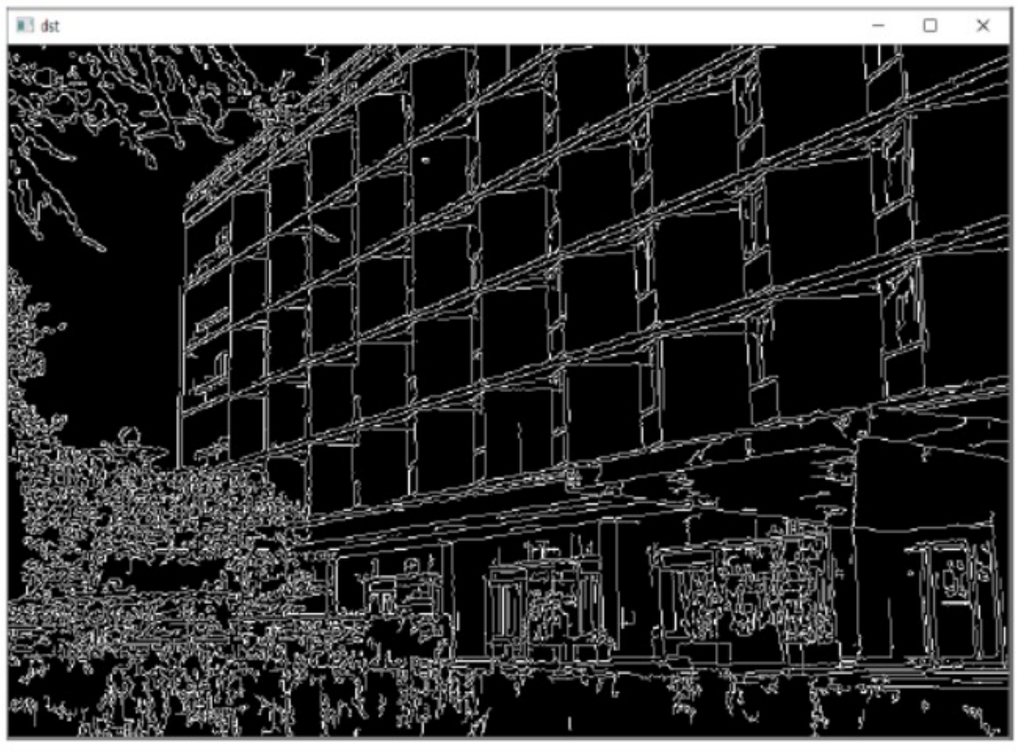
비최대 억제



히스테리시스 에지 트래킹

캐니 에지 검출

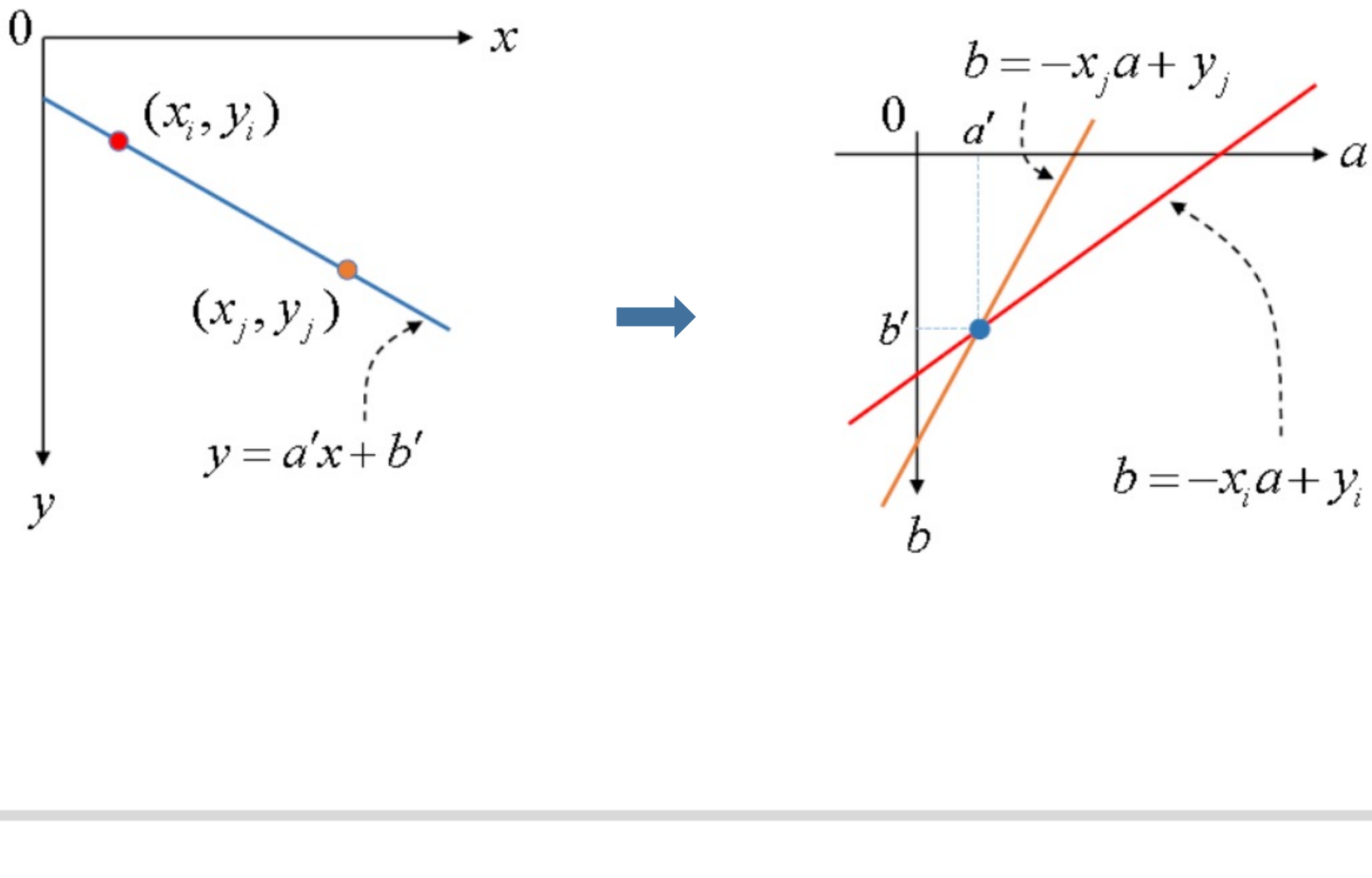
```
dst = cv2.Canny(src, 50, 150)
```



허프변환 : 직선검출

- 2차원 영상 좌표에서 직선의 방정식을 파라미터 공간으로 변환하여 직선을 찾는 알고리즘

$$y = ax + b \quad \longrightarrow \quad b = -xa + y$$



허프변환 : 직선검출

- 직선의 방정식 $y = ax + b$ 를 사용할 때 y축과 평행한 직선을 표현하지 못하여 극좌표계 직선의 방정식을 사용

$$x\cos\theta + y\ \sin\theta = \rho$$

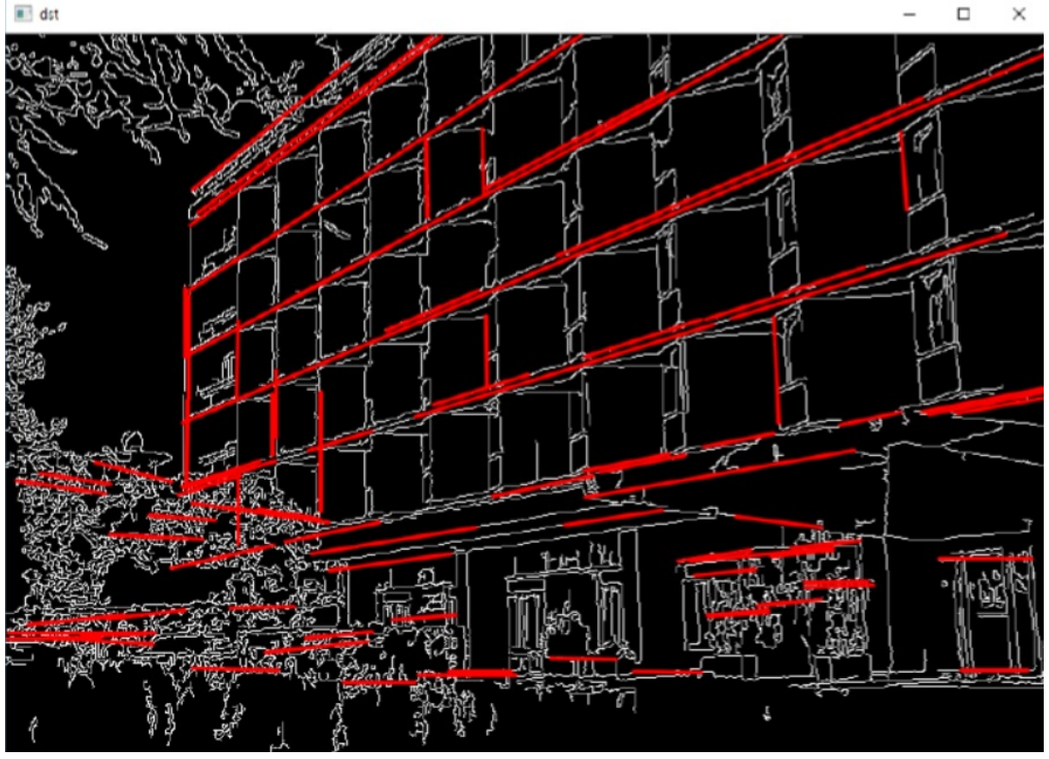
$$\begin{cases} \text{기울기} = -\dfrac{\cos\theta}{\sin\theta} \\ \text{y절편} = \dfrac{\rho}{\sin\theta} \end{cases}$$

$$y = -\frac{\cos\theta}{\sin\theta}x + \frac{\rho}{\sin\theta}$$

허프변환 : 직선검출

- 허프 변환 직선 검출 예제

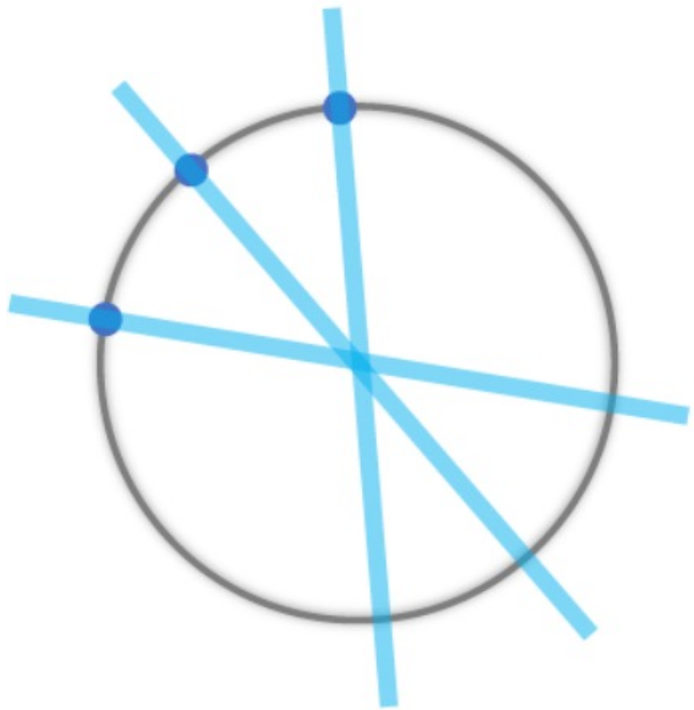
```
edges = cv2.Canny(src, 50, 150)

lines = cv2.HoughLinesP(edges, 1, np.pi / 180., 160,
                        minLineLength=50, maxLineGap=5)
```



허프변환 : 원검출

- 허프 변환을 응용하여 원을 검출

원의 방정식 : $(x-\mathrm{a})^2+(x-\mathrm{b})^2=c^2$ ⟶ 3차원 축적 평면



허프변환 : 원검출

- 허프 변환 직선 검출 예제

```
blr = cv2.GaussianBlur(gray, (0, 0), 1.0)
circles = cv2.HoughCircles(blr, cv2.HOUGH_GRADIENT, 1, 50,
                           param1=120, param2=th, minRadius=rmin, maxRadius=rmax)
```