

파이썬 OpenCV 영상처리 프로그래밍



<https://github.com/yisy0703/ai>



A4 Screen

1장. 영상처리 개요

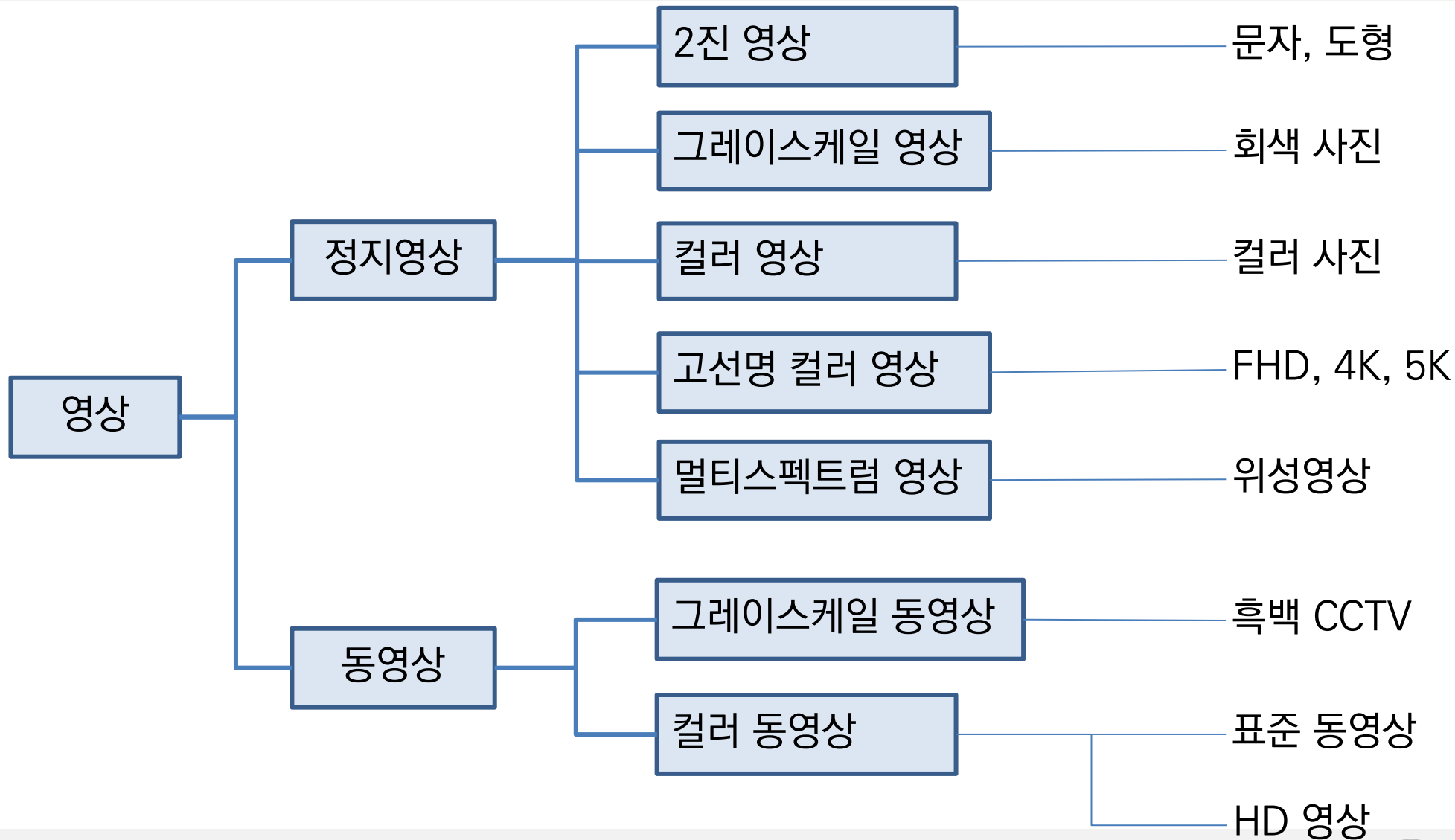
영상처리 개요에 대해 설명합니다.



- 1절. 이미지 표현
- 2절. OpenCV와 이미지 읽기
- 3절. 디지털 화상과 컬러
- 4절. 동영상 처리하기
- 5절. 이벤트 처리하기
- 6절. 도형 그리기

영상처리 개요

1장. 영상처리 개요 / 1절. 이미지 표현



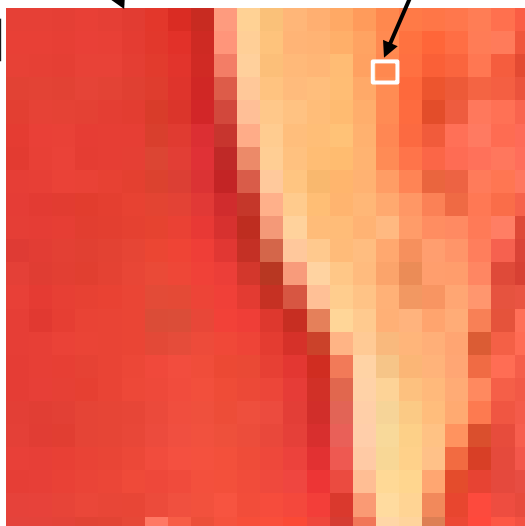
디지털 이미지와 좌표 표현 방법

1장. 영상처리 개요 / 1절. 이미지 표현

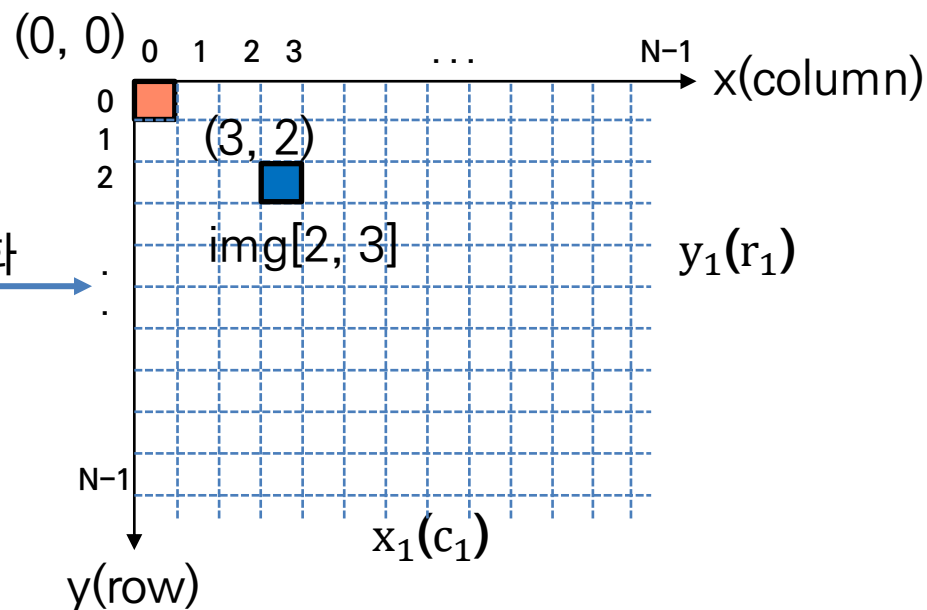
사진, 도형 등 아날로그 이미지



확대



디지털화



좌표의 표현 : (x, y)
배열에서의 표현 : $[y, x]$

표본화와 양자화

1장. 영상처리 개요 / 1절. 이미지 표현

표본화 : 공간적으로 연속되는 화상을 이산적인 화소의 집합으로 분할하는 것

같은 이미지라도 가로*세로의 크기가 큰 이미지가 더 선명

양자화 : 이미지의 농담을 이산적인 정수값으로 변환하는 조작

흑(1) 또는 백(0)으로 하는 2진(1비트)로 양자화 하는 것은 2진 화상

화소를 저장하는 비트 수를 증가시킬수록 표현이 풍부해 짐

-> 6비트 이상일 경우 사람의 눈으로 거의 구별할 수 없음

-> 컴퓨터에서 다룰 수 있도록 고려하여 256단계(8비트, 1바이트/화소)를 사용

-> 화소의 농담이 0~255의 수치로 표현 됨

레나 이미지

1장. 영상처리 개요 / 1절. 이미지 표현

레나(Lenna 또는 Lena) 이미지는 플레이보이 잡지 1972년 11월자 센터폴드(잡지 중간에 접어서 들어가 있는 페이지)에 실린 스웨덴의 모델인 레나 포르센(1951년 3월 31일 -)의 사진의 일부분



OpenCV

1장. 영상처리 개요 / 2절. OpenCV와 이미지 읽기

Open Source Computer Vision Library

- C++, Python, Java, MATLAB 인터페이스 제공

<https://opencv.org/>

- 파이썬용 OpenCV 설치
`pip install opencv-python`
- OpenCV 주요 모듈과 함께 추가 모듈도 설치
`pip install opencv-contrib-python`
- OpenCV를 사용하려면 넘파이 패키지가 설치되어 있어야 함
`pip install numpy`



이미지 불러오기

1장. 영상처리 개요 / 2절. OpenCV와 이미지 읽기

cv2.imread(fileName, flags=1) -> img

- ▶ fileName (str) – 이미지파일의 경로
flag (int) – 이미지 파일을 읽을 때의 Option.
- ▶ Return : numpy.ndarray



cv2.imread()의 flags

- ▶ cv2.IMREAD_COLOR : 이미지 파일을 컬러로 읽어들이니다. 투명한 부분은 무시되며, 기본값입니다.
- ▶ cv2.IMREAD_GRAYSCALE : 이미지를 그레이스케일로 읽어 들입니다. 실제 이미지 처리시 중간단계로 많이 사용합니다.
- ▶ cv2.IMREAD_UNCHANGED : 이미지파일을 알파채널까지 포함하여 읽어 들입니다. 알파채널은 불투명도를 표시하며 값이 0이면 완전하게 투명합니다.
- ▶ 3개의 상수 대신 1, 0, -1을 사용해도 됩니다.

imread flags.

cv2.imread(filename[, flags])

1장. 영상처리 개요 / 2절. OpenCV와 이미지 읽기

IMREAD_UNCHANGED	-1	설정된 경우 로드된 이미지를 그대로 (알파 채널이 있는 경우 자르기) 반환합니다. EXIF 방향을 무시합니다.
IMREAD_GRAYSCALE	0	설정하면 항상 이미지를 단일 채널 회색조 이미지로 변환합니다.(코덱 내부 변환).
IMREAD_COLOR	1	설정된 경우 항상 이미지를 3 채널 BGR 컬러 이미지로 변환합니다.
IMREAD_ANYDEPTH	2	설정된 경우 입력에 해당 깊이가 있으면 16 비트 / 32 비트 이미지를 반환하고, 그렇지 않으면 8 비트로 변환합니다.
IMREAD_ANYCOLOR	4	설정된 경우 이미지를 가능한 모든 색상 형식으로 읽습니다.
IMREAD_LOAD_GDAL	8	설정되어 있으면 gdal 드라이버를 사용하여 이미지를 로드합니다.
IMREAD_REDUCED_GRAYSCALE_2	16	설정하면 항상 이미지를 단일 채널 회색조 이미지로 변환하고 이미지 크기를 1/2로 줄입니다.
IMREAD_REDUCED_COLOR_2	17	설정하면 항상 이미지를 3 채널 BGR 컬러 이미지로 변환하고 이미지 크기를 1/2로 줄입니다.
IMREAD_REDUCED_GRAYSCALE_4	32	설정하면 항상 이미지를 단일 채널 회색조 이미지로 변환하고 이미지 크기를 1/4로 줄입니다.
IMREAD_REDUCED_COLOR_4	33	설정하면 항상 이미지를 3 채널 BGR 컬러 이미지로 변환하고 이미지 크기를 1/4로 줄입니다.
IMREAD_REDUCED_GRAYSCALE_8	64	설정하면 항상 이미지를 단일 채널 회색조 이미지로 변환하고 이미지 크기가 1/8로 줄어 듭니다.
IMREAD_REDUCED_COLOR_8	65	설정하면 항상 이미지를 3 채널 BGR 컬러 이미지로 변환하고 이미지 크기를 1/8로 줄입니다.
IMREAD_IGNORE_ORIENTATION	128	설정된 경우 EXIF의 방향 플래그에 따라 이미지를 회전시키지 않습니다.

<https://en.wikipedia.org/wiki/Exif>

이미지 출력하기

1장. 영상처리 개요 / 2절. OpenCV와 이미지 읽기

cv2.imshow(title, image)

- ▶ title (str) – 윈도우 창의 Title
image (numpy.ndarray) – cv2.imread() 의 return값

```
1 cv2.imshow("Lena", lena_img)
2 cv2.waitKey()
3 cv2.destroyAllWindows()
```



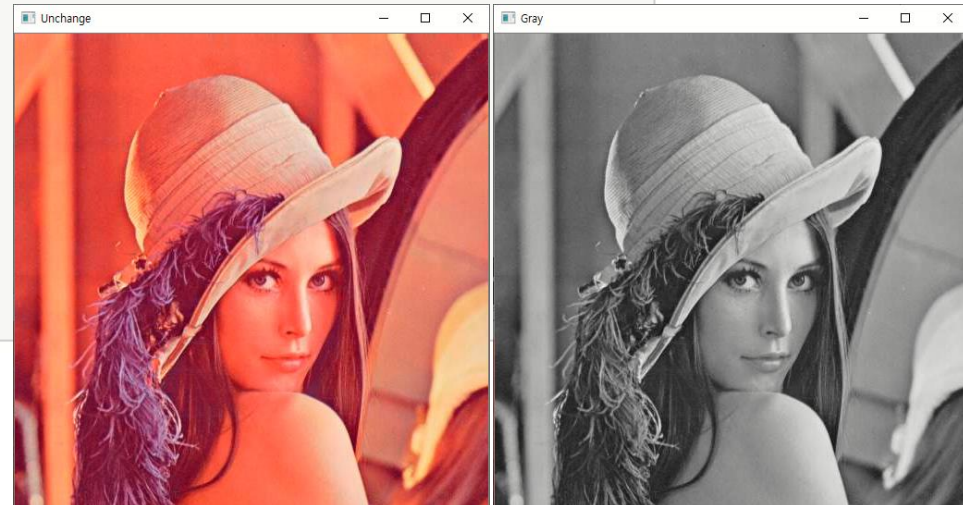
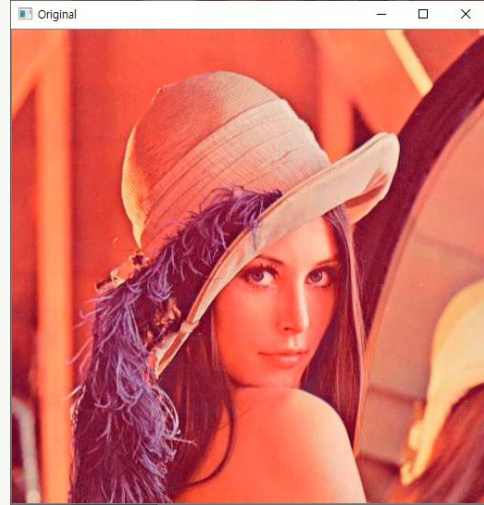
함께 사용하는 코드

- ▶ cv2.waitKey() : keyboard입력을 대기하는 함수로 0이면 key입력까지 무한대기이며 특정 시간동안 대기하려면 milisecond값을 넣어주면 됩니다.
- ▶ cv2.destroyAllWindows() : 화면에 나타난 윈도우를 종료합니다.
- ▶ 일반적으로 위 3개는 같이 사용됩니다.

이미지 출력하기

1장. 영상처리 개요 / 2절. OpenCV와 이미지 읽기

```
1 import cv2
2
3 fname = 'lena.jpg'
4
5 original = cv2.imread(fname, cv2.IMREAD_COLOR)
6 gray = cv2.imread(fname, cv2.IMREAD_GRAYSCALE)
7 unchange = cv2.imread(fname, cv2.IMREAD_UNCHANGED)
8
9 cv2.imshow('Original', original)
10 cv2.imshow('Gray', gray)
11 cv2.imshow('Unchange', unchange)
12
13 cv2.waitKey(0)
14 cv2.destroyAllWindows()
```



jpg 파일일 경우 unchange.shape[2] 는 3 이지만
투명화소 정보가 있는 png 파일일 경우 4임(alpha값 포함)

이미지 저장하기

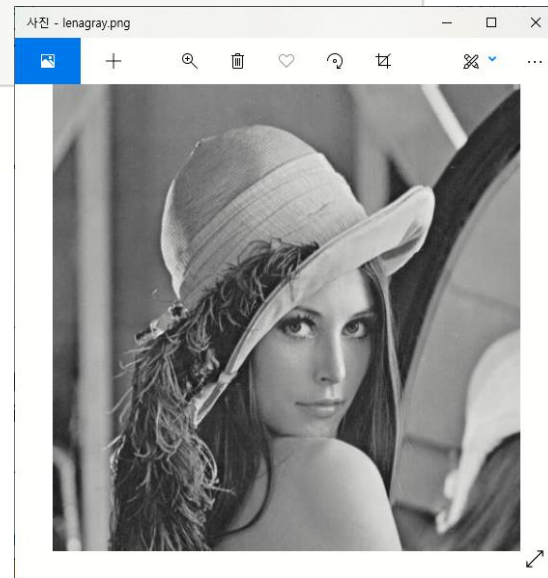
1장. 영상처리 개요 / 2절. OpenCV와 이미지 읽기

`cv2.imwrite(fileName, image) -> retval`

- ▶ `fileName (str)` – 저장될 파일명
`image` – 저장할 이미지

```
1 import cv2
2
3 img = cv2.imread('lena.jpg', cv2.IMREAD_GRAYSCALE)
4 cv2.imwrite('lenagray.png', img)
```

True



Matplotlib으로 이미지 출력하기

1장. 영상처리 개요 / 2절. OpenCV와 이미지 읽기

plt.imshow(image)

- ▶ Matplotlib는 다양한 plot기능을 가진 Python Plot Library입니다.
- ▶ 이미지를 zoom하거나 하나의 화면에 여러개의 이미지를 보고자 할 때 유용합니다.

```
1 #-*- coding:utf-8 -*-
2 import cv2
3 from matplotlib import pyplot as plt
4 %matplotlib inline
5
6 img = cv2.imread('lena.jpg', cv2.IMREAD_COLOR)
7
8 plt.imshow(img)
9 plt.xticks([]) # x축 눈금
10 plt.yticks([]) # y축 눈금
11 plt.show()
```



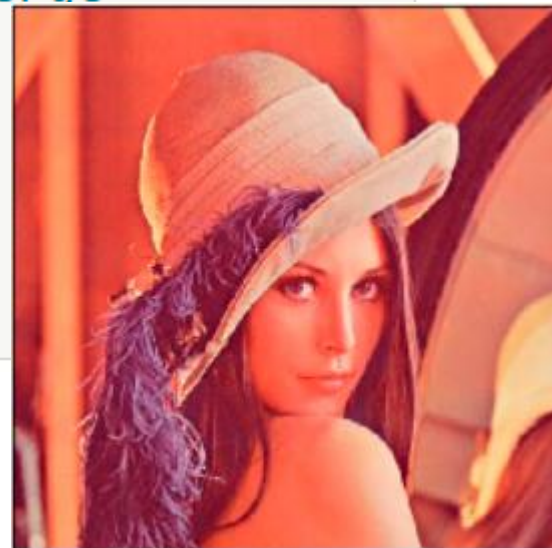
원본은 붉은색 계열인데, 결과는 파란색 계열로 나타납니다.

이유는 openCV는 BGR로 사용하지만, Matplotlib는 RGB로 이미지를 보여주기 때문입니다.
즉 결과 값은 3차원 배열의 값 중 첫 번째와 세 번째 배열값을 서로 바꿔 주어야 합니다.

Matplotlib으로 이미지 출력하기

1장. 영상처리 개요 / 2절. OpenCV와 이미지 읽기

```
1  #-*- coding:utf-8 -*-
2  import cv2
3  from matplotlib import pyplot as plt
4
5  img = cv2.imread('lena.jpg', cv2.IMREAD_COLOR)
6
7  b, g, r = cv2.split(img)    # img파일을 b,g,r로 분리
8  img2 = cv2.merge([r,g,b])  # b, r을 바꿔서 Merae
9
10 plt.imshow(img2)
11 plt.xticks([]) # x축 눈금
12 plt.yticks([]) # y축 눈금
13 plt.show()
```



디지털 이미지 유형 – Binary Image

1장. 영상처리 개요 / 3절. 디지털 이미지와 컬러

Binary Image는 pixel당 1bit로 표현하는 영상을 의미합니다.

즉 흰색과 검은색으로만 표현이 되는 영상입니다.

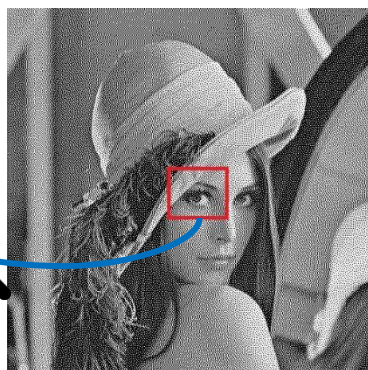
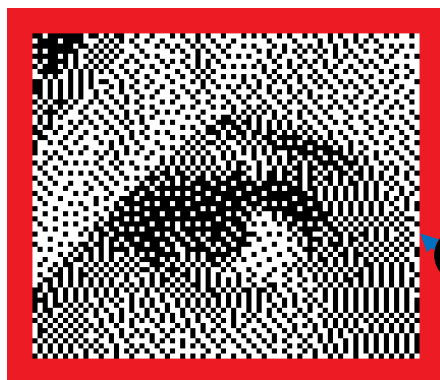
회색조 이미지



2진 이미지



원본 이미지를
thresholding 처리를
하여 binary
image로 변환한 결과
(threshold=120)



dithering : 밀도를
조절하여 밝기를 표현
하는 방법

회색조 화상

1장. 영상처리 개요 / 3절. 디지털 이미지와 컬러

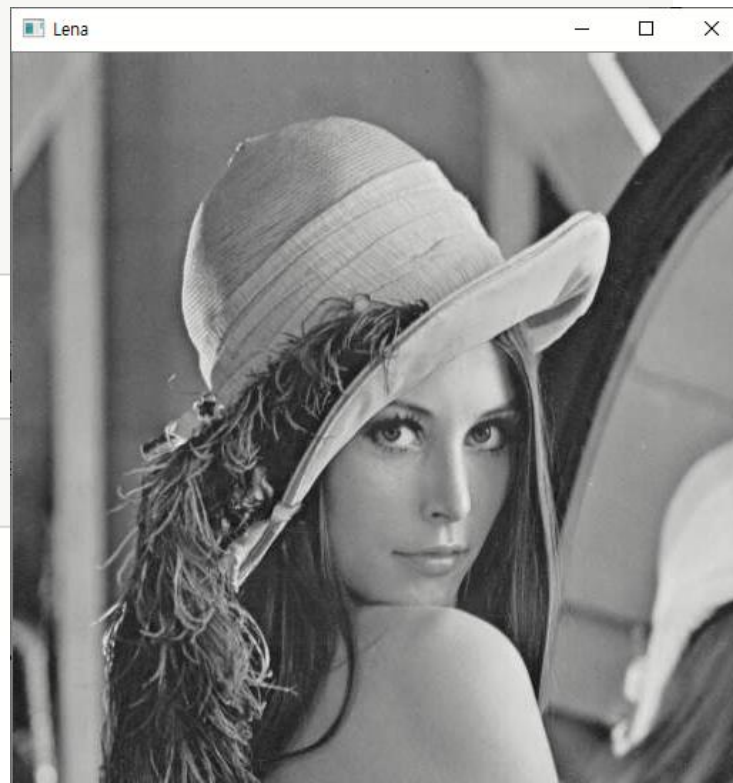
Pixel당 8bit, 즉 256단계의 명암(빛의 세기)을 표현할 수 있는 이미지입니다.

```
1 lena_grey = cv2.imread("lena.jpg", cv2.IMREAD_GRAYSCALE)
2 print(lena_grey.shape)
3 cv2.imshow("Lena", lena_grey)
4 cv2.waitKey()
5 cv2.destroyAllWindows()
```

(512, 512)

```
1 lena_grey[0][0]
```

162



컬러 화상

1장. 영상처리 개요 / 3절. 디지털 이미지와 컬러

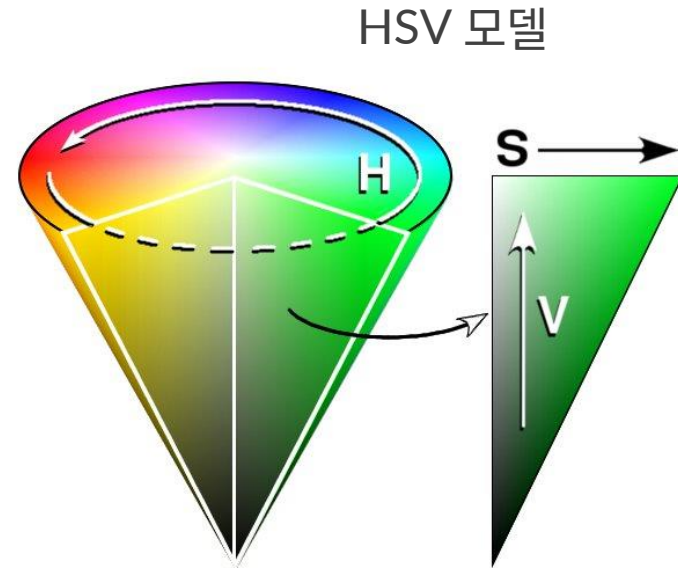
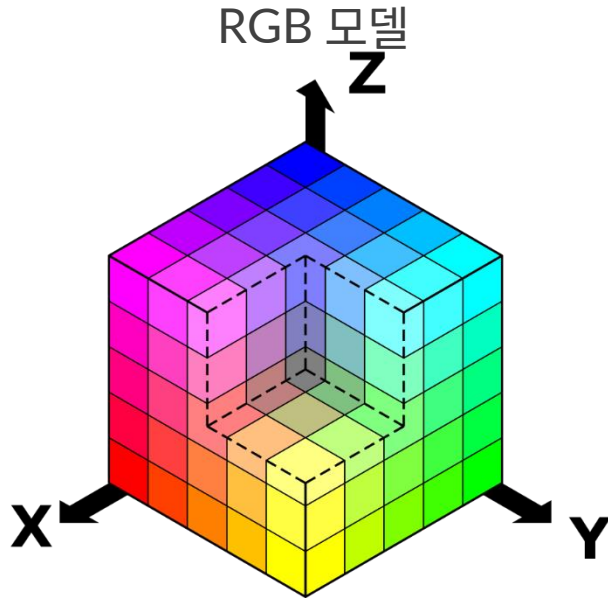


Color Image

- ▶ Color 이미지는 pixel의 색을 표현하기 위해서 pixel당 24bit를 사용합니다. 총 16,777,216 가지의 색을 표현할 수 있습니다.
- ▶ 이것을 일반적으로 True color image라고 합니다. pixel은 RGB 각각을 위해서 8bit를 사용하게 됩니다.
- ▶ OpenCV에서는 BGR로 표현을 하기 때문에 Blue- $\rightarrow(255,0,0)$, Green- $\rightarrow(0,255,0)$, Red- $\rightarrow(0,0,255)$, White- $\rightarrow(255,255,255)$, Black- $\rightarrow(0,0,0)$ 으로 표현할 수 있습니다.
- ▶ 각 pixel당 3Byte를 사용하기 때문에 용량이 큼니다. 이를 해결하기 위해서 lookup table을 사용하여, 해당 pixel에는 index만 저장하기도 합니다.

RGB 모델 vs. HSV 모델

1장. 영상처리 개요 / 3절. 디지털 이미지와 컬러



HSV의 의미는 다음과 같습니다.

- ▶ Hue : 색상. 일반적인 색을 의미함. 원추모형에서 각도로 표현이 됨. (0: Red, 120도 : Green, 240: Blue)
- ▶ Saturation : 채도. 색의 순수성을 의미하며 일반적으로 '짙다', '흐리다'로 표현이 됨. 중심에서 바깥쪽으로 이동하면 채도가 높음.
- ▶ Value : 명도. 색의 밝고 어두운 정도. 수직축의 깊이로 표현. 어둡다 밝다로 표현이 됨.

By Wapcaplet - From en wiki, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=308191>



색상 공간 변환

1장. 영상처리 개요 / 3절. 디지털 이미지와 컬러

cv2.cvtColor(src, flag) -> dst

OpenCV에는 150여가지 변환 방법이 있습니다. 아래는 변환 방법을 확인하는 코드입니다.

```
flags = [flag for flag in dir(cv2) if flag.startswith('COLOR_')]
print(flags)
```

['COLOR_BAYER_BG2BGR', 'COLOR_BAYER_BG2BGRA', 'COLOR_BAYER_BG2BGR_EA', 'COLOR_BAYER_BG2BGR_VNG', 'COLOR_BAYER_BG2GRAY', 'COLOR_BAYER_BG2RGB', 'COLOR_BAYER_BG2RGBA', 'COLOR_BAYER_BG2RGB_EA', 'COLOR_BAYER'

밝기 정보(Y)만을 따로 분리해서 전송하고, 색깔 정보(U,V)는 따로 보내는 방법을 사용

컬러 영상을 Gray 영상으로 변환하는 공식은 YPbPr, YCbCr, YIQ등이 있음

$$Y_{PrPb} : Y = Red * 0.299 + Green * 0.587 + Blue * 0.114$$

$$Y_{CrCb} : Y = Red * 0.2126 + Green * 0.7152 + Blue * 0.0722$$

HSL, HSV : http://en.wikipedia.org/wiki/HSL_and_HSV

YCrCb : <http://en.wikipedia.org/wiki/YCbCr>

색상 공간 변환

1장. 영상처리 개요 / 3절. 디지털 이미지와 컬러

```
1 lena_img = cv2.imread("lena.jpg")
2 B, G, R = lena_img[0][0]
3 print("컬러 화소", lena_img[0][0])
4 print("YPrPb", 0.114*B + 0.587*G + 0.299*R)
5 print("YCrCb", 0.0722*B + 0.7152*G + 0.2116*R)
6 lena_gray = cv2.imread("lena.jpg", cv2.IMREAD_GRAYSCALE)
7 print("그레이스케일 화소", lena_gray[0][0])
```

컬러 화소 [127 136 226]


YPrPb 161.884

YCrCb 154.2582

그레이스케일 화소 162

선 그리기

1장. 영상처리 개요 / 6절. 도형 그리기



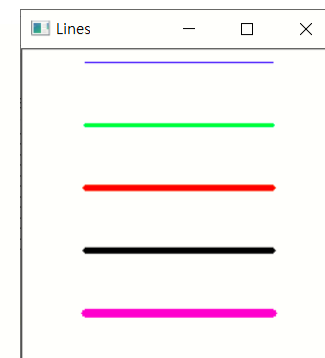
```
cv2.line(img, start, end, color, thickness=1)
```

- ▶ `img` - 도형을 그릴 이미지 객체입니다.
- ▶ `start` - 선의 시작 좌표입니다. (x, y) 형식으로 지정합니다.
- ▶ `end` - 선의 끝 좌표입니다. (x, y) 형식으로 지정합니다.
- ▶ `color` - BGR 형식의 색상값입니다. (B, G, R) 형식이며 (255, 0, 0)이면 Blue입니다.
- ▶ `thickness` - 선의 두께를 픽셀단위 정수로 지정합니다

선 그리기


1장. 영상처리 개요 / 6절. 도형 그리기

```
1 import cv2
2 import numpy as np
3
4 img = np.full((250, 250, 3), 255, dtype=np.uint8)
5
6 cv2.line(img, (50, 10), (200, 10), (255, 0, 0), 1)
7 cv2.line(img, (50, 60), (200, 60), (0, 255, 0), 2)
8 cv2.line(img, (50, 110), (200, 110), (0, 0, 255), 3)
9 cv2.line(img, (50, 160), (200, 160), (0, 0, 0), 4)
10 cv2.line(img, (50, 210), (200, 210), (255, 0, 255), 5)
11
12 cv2.imshow('Lines',img)
13 cv2.waitKey(0)
14 cv2.destroyAllWindows()
```



사각형 그리기

1장. 영상처리 개요 / 6절. 도형 그리기



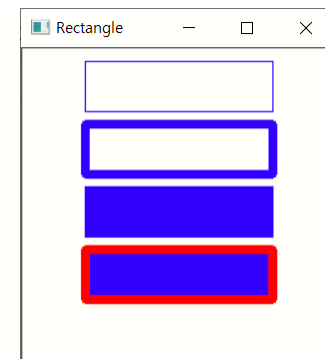
```
cv2.rectangle(img, start, end, color, thickness=1)
```

- ▶ `img` - 도형을 그릴 이미지 객체입니다.
- ▶ `start` - 사각형의 왼쪽 위 좌표입니다. (x, y) 형식으로 지정합니다.
- ▶ `end` - 사각형의 오른쪽 아래 좌표입니다. (x, y) 형식으로 지정합니다.
- ▶ `color` - BGR 형식의 색상값입니다. (B, G, R) 형식이며 (255, 0, 0)이면 Blue입니다.
- ▶ `thickness` - 선의 두께를 픽셀단위 정수로 지정합니다. -1이면 사각형의 안쪽을 채웁니다.

사각형 그리기


1장. 영상처리 개요 / 6절. 도형 그리기

```
1 import cv2
2 import numpy as np
3
4 img = np.full((250, 250, 3), 255, dtype=np.uint8)
5
6 cv2.rectangle(img, (50, 10), (200, 50), (255, 0, 0))
7 cv2.rectangle(img, (50, 60), (200, 100), (255, 0, 0), 5)
8 cv2.rectangle(img, (50, 110), (200, 150), (255, 0, 0), -1)
9 cv2.rectangle(img, (50, 160), (200, 200), (255, 0, 0), -1)
10 cv2.rectangle(img, (50, 160), (200, 200), (0, 0, 255), 5)
11
12 cv2.imshow('Rectangle',img)
13 cv2.waitKey(0)
14 cv2.destroyAllWindows()
```



원 그리기

1장. 영상처리 개요 / 6절. 도형 그리기



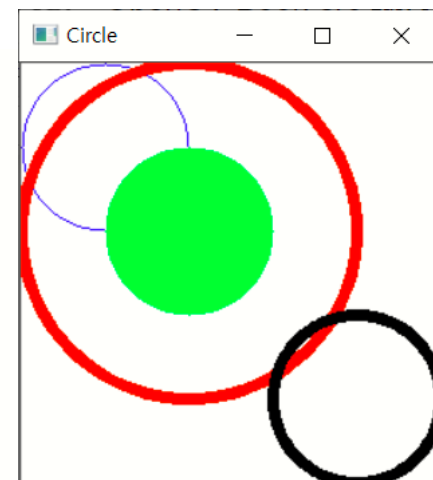
```
cv2.circle(img, center, radius, color, thickness=1)
```

- ▶ `img` - 도형을 그릴 이미지 객체입니다.
- ▶ `center` - 원의 중심 좌표입니다. (x, y) 형식으로 지정합니다.
- ▶ `radius` - 반지름을 픽셀단위 정수로 지정합니다.
- ▶ `color` - BGR 형식의 색상값입니다. (B, G, R) 형식이며 $(255, 0, 0)$ 이면 Blue입니다.
- ▶ `thickness` - 선의 두께를 픽셀단위 정수로 지정합니다. -1 이면 원의 안쪽을 채웁니다.

원 그리기


1장. 영상처리 개요 / 6절. 도형 그리기

```
1 import cv2
2 import numpy as np
3
4 img = np.full((250, 250, 3), 255, dtype=np.uint8)
5
6 cv2.circle(img, (50, 50), 50, (255, 0, 0))
7 cv2.circle(img, (100, 100), 50, (0, 255, 0), -1)
8 cv2.circle(img, (100, 100), 100, (0, 0, 255), 5)
9 cv2.circle(img, (200, 200), 50, (0, 0, 0), 5)
10
11 cv2.imshow('Circle',img)
12 cv2.waitKey(0)
13 cv2.destroyAllWindows()
```



타원과 호 그리기

1장. 영상처리 개요 / 6절. 도형 그리기

 `cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color, thickness=1)`

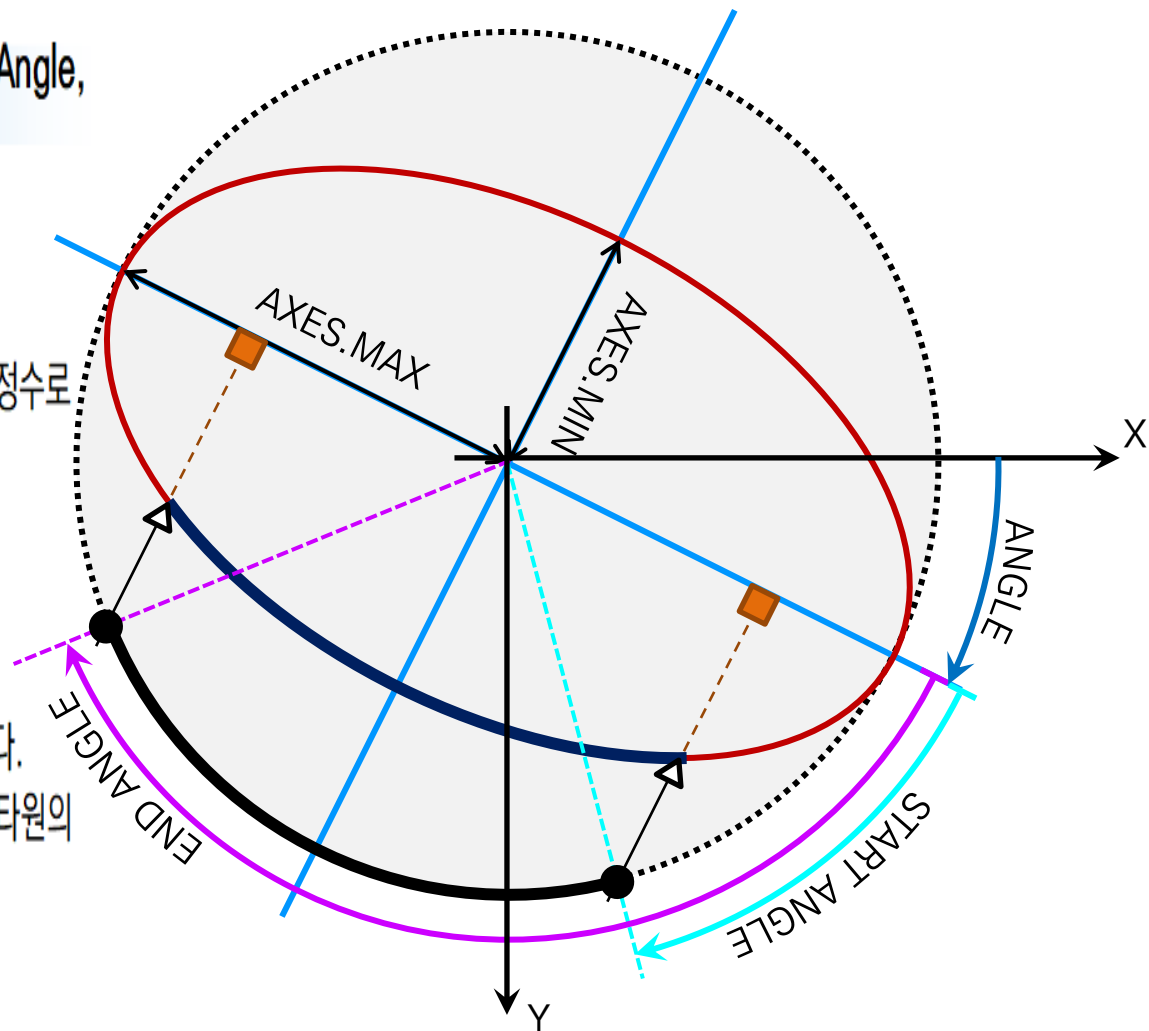
- ▶ `img` - 도형을 그릴 이미지 객체입니다.
- ▶ `center` - 타원의 중심 좌표입니다. (x, y) 형식으로 지정합니다.
- ▶ `axes` - 타원의 중심에서 가장 큰 거리와 작은 거리를 픽셀단위 정수로 지정합니다. (max, min) 형식으로 지정합니다.
- ▶ `angle` - 타원의 기울기 각도입니다.
- ▶ `startAngle` - 호의 시작 각도입니다. 아래 그림을 참고하세요.
- ▶ `endAngle` - 호의 끝 각도입니다. 아래 그림을 참고하세요.
- ▶ `color` - (B,G,R) 형식의 색상값입니다. (255,0,0)이면 Blue입니다.
- ▶ `thickness` - 선의 두께를 픽셀단위 정수로 지정합니다. -1이면 타원의 안쪽을 채웁니다.

Ellipse 속성

1장. 영상처리 개요 / 6절. 도형 그리기

`cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color, thickness=1)`

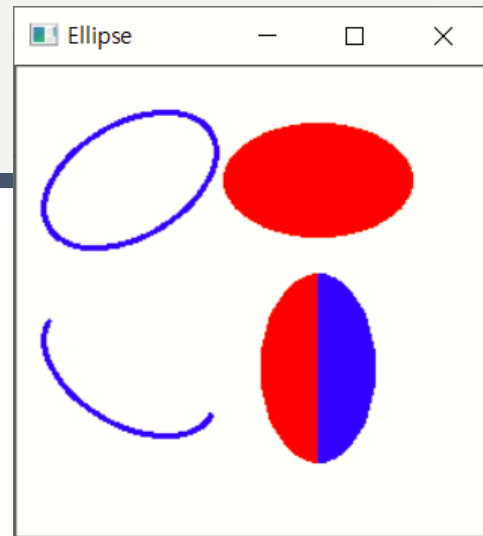
- ▶ `img` - 도형을 그릴 이미지 객체입니다.
- ▶ `center` - 타원의 중심 좌표입니다. (x, y) 형식으로 지정합니다.
- ▶ `axes` - 타원의 중심에서 가장 큰 거리와 작은 거리를 픽셀단위 정수로 지정합니다. (max, min) 형식으로 지정합니다.
- ▶ `angle` - 타원의 기울기 각도입니다.
- ▶ `startAngle` - 호의 시작 각도입니다. 아래 그림을 참고하세요.
- ▶ `endAngle` - 호의 끝 각도입니다. 아래 그림을 참고하세요.
- ▶ `color` - (B,G,R) 형식의 색상값입니다. (255,0,0)이면 Blue입니다.
- ▶ `thickness` - 선의 두께를 픽셀단위 정수로 지정합니다. -1이면 타원의 안쪽을 채웁니다.



타원과 호 그리기


1장. 영상처리 개요 / 6절. 도형 그리기

```
1 import cv2
2 import numpy as np
3
4 img = np.full((250, 250, 3), 255, dtype=np.uint8)
5
6 cv2.ellipse(img, (60,60), (50,30), -30, 0, 360, (255,0,0), 2)
7 cv2.ellipse(img, (160,60), (50,30), 0, 0, 360, (0,0,255), -1)
8 cv2.ellipse(img, (60,160), (50,30), 30, 0, 180, (255,0,0), 2)
9 cv2.ellipse(img, (160,160), (50,30), 90, 0, 180, (0,0,255), -1)
10 cv2.ellipse(img, (160,160), (50,30), 90, 180, 360, (255,0,0), -1)
11
12 cv2.imshow('Ellipse',img)
13 cv2.waitKey(0)
14 cv2.destroyAllWindows()
```



다각형 그리기

1장. 영상처리 개요 / 6절. 도형 그리기



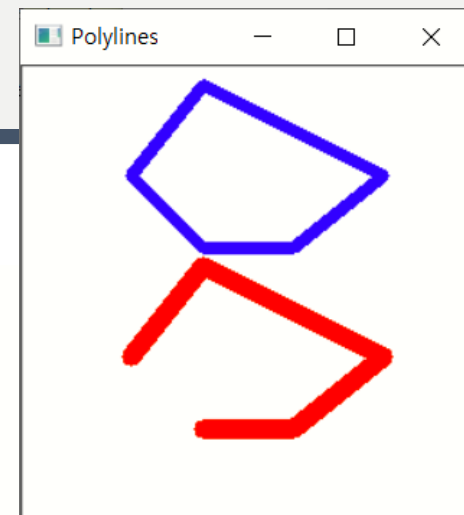
```
cv2.polylines(img, points, isClosed, color, thickness)
```

- ▶ `img` - 도형을 그릴 이미지 객체입니다.
- ▶ `points` - 다각형 꼭지점의 좌표들입니다. `[[[x1,y1], [x2,y2]....]]` 형식으로 지정합니다.
- ▶ `isClosed` - `True`이면 닫힌 도형이 됩니다.
- ▶ `color` - BGR 형식의 색상값입니다. (B, G,R) 형식이며 (255,0,0)이면 Blue입니다.
- ▶ `thickness` - 선의 두께를 픽셀단위 정수로 지정합니다. 기본값은 1입니다. 음수는 사용할 수 없습니다.

다각형 그리기


1장. 영상처리 개요 / 6절. 도형 그리기

```
1 import cv2
2 import numpy as np
3
4 img = np.full((250, 250, 3), 255, dtype=np.uint8)
5 points = np.array([[60,60], [100,10], [200,60], [150,100],
6                    [100, 100]]], dtype=np.int32)
7
8 cv2.polylines(img, points, True, (255, 0, 0), 5)
9 cv2.polylines(img, points+[0,100], False, (0, 0, 255), 10)
10
11 cv2.imshow('Polylines',img)
12 cv2.waitKey(0)
13 cv2.destroyAllWindows()
```



문자열 표시하기

1장. 영상처리 개요 / 6절. 도형 그리기



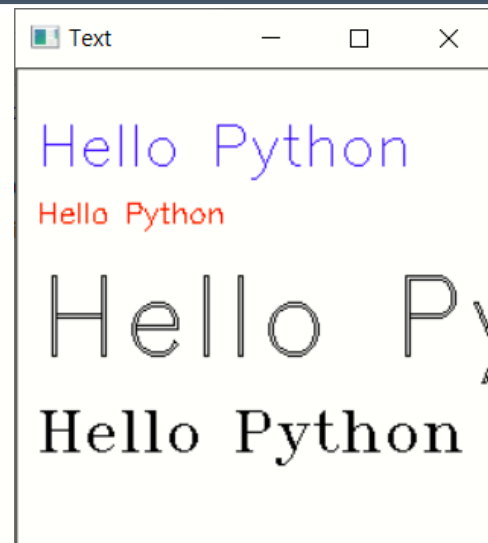
```
cv2.putText(img, text, org, font, fontScale, color)
```

- ▶ `img` - 도형을 그릴 이미지 객체입니다.
- ▶ `text` - 표시할 문자열입니다.
- ▶ `org` - 문자열이 표시될 위치입니다. 문자열의 왼쪽 아래 모서리 지점이 기준입니다.
- ▶ `font` - 폰트를 지정합니다. `cv2.FONT_XXX` 형식의 속성으로 지정합니다.
- ▶ `fontScale` - 폰트의 크기를 지정합니다.
- ▶ `color` - BGR 형식의 색상값입니다. (B, G,R) 형식이며 (255,0,0)이면 Blue입니다.

문자열 표시하기

1장. 영상처리 개요 / 6절. 도형 그리기

```
1 import cv2
2 import numpy as np
3
4 img = np.full((250, 250, 3), 255, dtype=np.uint8)
5
6 cv2.putText(img, "Hello Python", (10,50),
              cv2.FONT_ITALIC, 1, (255,0,0))
7 cv2.putText(img, "Hello Python", (10,80),
              cv2.FONT_ITALIC, 0.5, (0,0,255))
8 cv2.putText(img, "Hello Python", (10,150),
              cv2.FONT_HERSHEY_DUPLEX, 2, (0,0,0))
9 cv2.putText(img, "Hello Python", (10,200),
              cv2.FONT_HERSHEY_TRIPLEX, 1, (0,0,0))
10
11 cv2.imshow('Text',img)
12 cv2.waitKey(0)
13 cv2.destroyAllWindows()
```



2장. 영상 기본 연산

영상 기본 연산에 대해 설명합니다.

- 1절. 영상 기본 정보
- 2절. 기본 연산
- 3절. 히스토그램



영상 기본 속성

2장. 영상 기본 연산 / 1절. 영상 기본 정보

OpenCV의 imread() 함수를 이용해서 이미지를 읽으면 넘파이 배열로 반환함

영상처리를 잘 하려면 넘파이 패키지를 잘 다뤄야 함

```
1 img = cv2.imread("lena.jpg")
```

```
1 print(img.shape)      512x512 크기 채널 3개인 이미지
```

(512, 512, 3)

```
1 print(img[0,0])      (0, 0)위치의 화소 값(BGR 순서)
```

[127 136 226]

```
1 print(img.size)
```

786432

512x512x3 = 786,432

```
1 print(img.dtype)
```

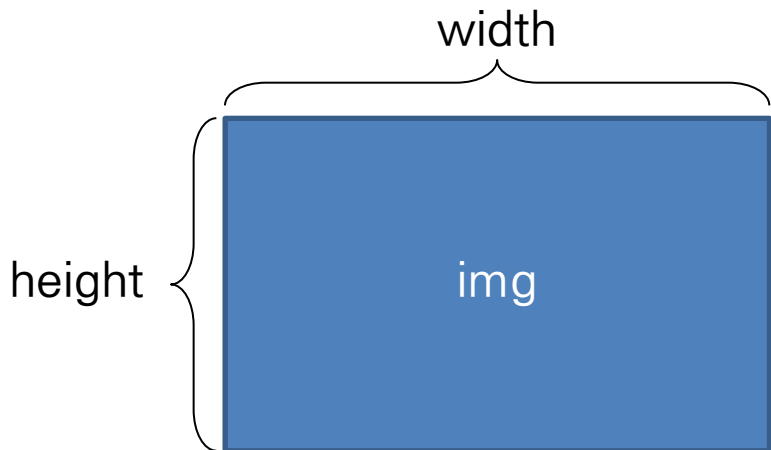
uint8

화소의 자료형은 np.uint8(Unsigned int)
부호를 갖지 않은 8비트 정수이므로 0~255표현

이미지 화소 처리하기

2장. 영상 기본 연산 / 1절. 영상 기본 정보

height, width = img.shape



반복문 실행 시 화소의 좌표는 `img[y, x]`

셀 실행시간 확인은 아래 명령으로 주피터 노트북
익스텐션 설치 후 ExecuteTime 선택
`pip install jupyter_contrib_nbextensions`
`jupyter contrib nbextension install`

짝수 행을 검은 선으로 처리하는 구문의 예

```
1 height, width = img.shape
```

```
1 for y in range(height) :  
2     for x in range(width) :  
3         if(y%2==0) :  
4             img[y,x] = 0
```

executed in 57ms, finished 09:42:33 2020-06-18

```
1 for y in range(height) :  
2     if(y%2==0) :  
3         for x in range(width) :  
4             img[y,x] = 0
```

executed in 27ms, finished 09:42:33 2020-06-18

```
1 for y in range(height) :  
2     if(y%2==0) :  
3         img[y,:] = 0
```

executed in 4ms, finished 09:42:34 2020-06-18

함수 만들어 사용하기

2장. 영상 기본 연산 / 1절. 영상 기본 정보

```
1 import numpy as np
2
3
4 def black_lines(img):
5     height, width = img.shape
6     img_ = np.zeros(img.shape, dtype=np.uint8)
7     for y in range(height):
8         if(y % 2 == 0):
9             img_[y, :] = 0
10        else:
11            img_[y, :] = img[y, :]
12    return img_
```



```
1 def img_pro(func, img, *args, show=True, **kwargs):
2     img_ = func(img, *args, **kwargs)
3     if show:
4         cv2.imshow("Image", img_)
5         cv2.waitKey()
6         cv2.destroyAllWindows()
7     else:
8         return img_
```

```
1 img = cv2.imread("./images/lena.jpg", cv2.IMREAD_GRAYSCALE)
2 img_pro(black_lines, img)
```

```
1 t = img_pro(black_lines, img, show=False)
```

```
1 cv2.imshow("Lena", t)
2 cv2.waitKey()
3 cv2.destroyAllWindows()
```

Matplotlib을 이용해 출력 함수 만들어 사용하기

2장. 영상 기본 연산 / 1절. 영상 기본 정보

```
1 import cv2
2 import matplotlib.pyplot as plt
3
4 def img_pro2(func, img, *args, output=True, win=False, **kwargs):
5     img_ = func(img, *args, **kwargs)
6     if output:
7         if win:
8             cv2.imshow("Image", img_)
9             cv2.waitKey()
10            cv2.destroyAllWindows()
11        else:
12            fig, axes = plt.subplots(1,2)
13            axes[0].imshow(img, cmap="gray")
14            axes[0].axis("off"); axes[0].set_title("origin")
15            axes[1].imshow(img_, cmap="gray", interpolation=None)
16            axes[1].axis("off"); axes[1].set_title("target")
17    else:
18        return img_
```

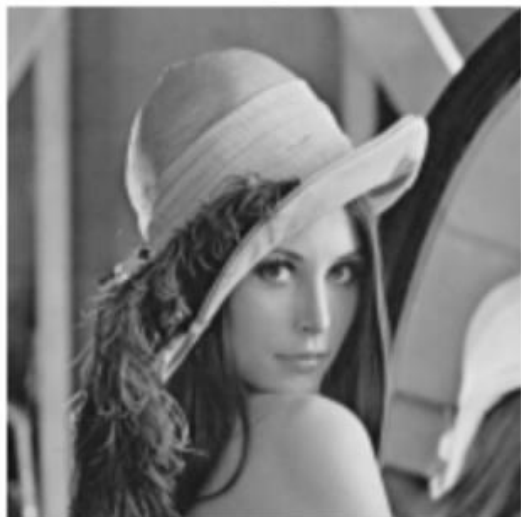


Matplotlib을 이용한 출력 함수 사용

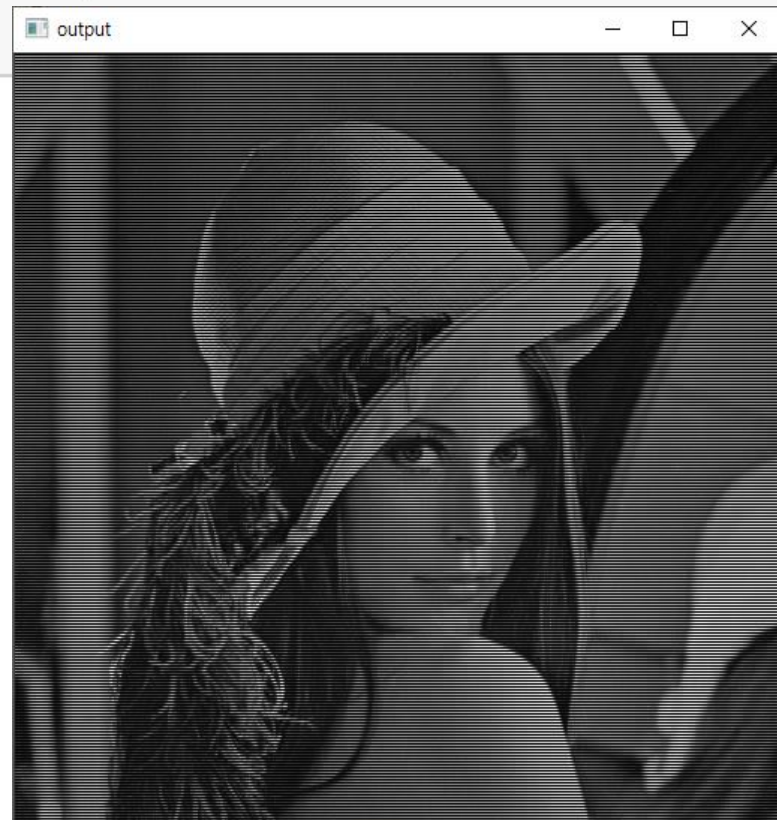
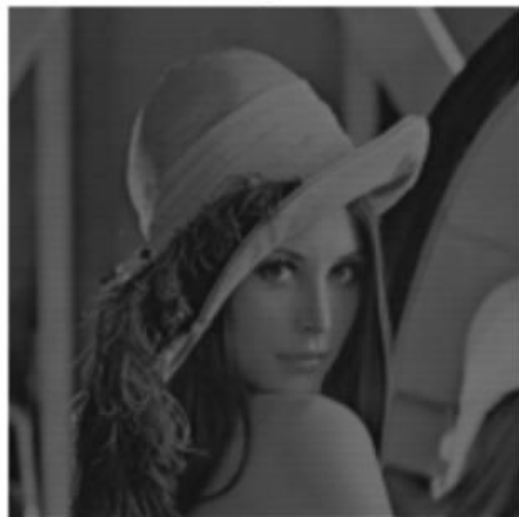
2장. 영상 기본 연산 / 1절. 영상 기본 정보

```
1 img = cv2.imread("lena.jpg", cv2.IMREAD_GRAYSCALE)
2 img_pro2(black_lines, img)
```

Origin



Target



```
1 img = cv2.imread("lena.jpg", cv2.IMREAD_GRAYSCALE)
2 img_pro2(black_lines, img, win=True)
```

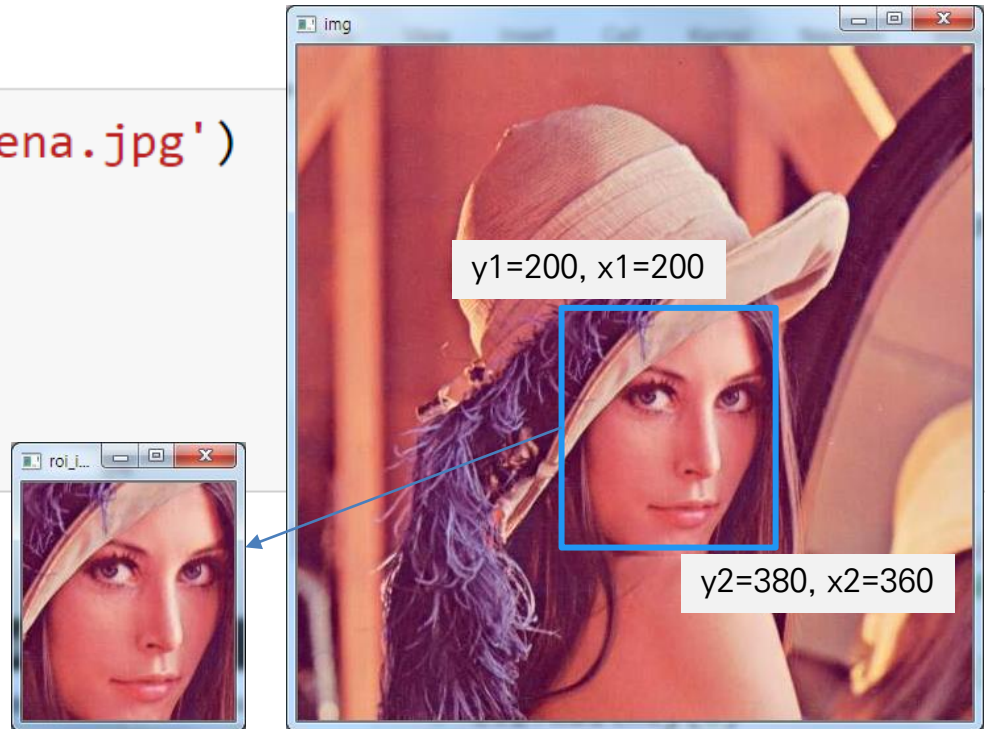

Region of Image(ROI)

2장. 영상 기본 연산 / 1절. 영상 기본 정보

이미지 작업시에는 특정 pixel단위 보다는 특정 영역단위로 작업을 하게 됩니다.

ROI 설정은 Numpy의 indexing방법을 사용합니다.

```
1 img = cv2.imread('./images/lena.jpg')
2 temp = img[200:380, 200:360]
3 cv2.imshow('img', img)
4 cv2.imshow('roi_img', temp)
5 cv2.waitKey(0)
6 cv2.destroyAllWindows()
```



이미지 채널

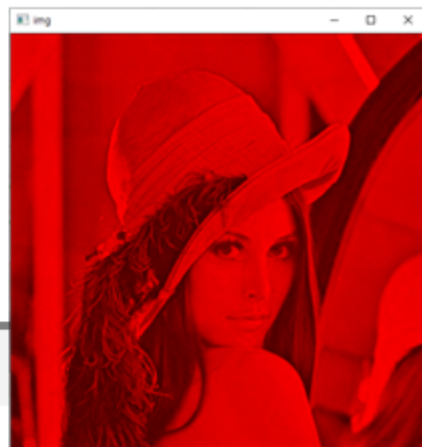
2장. 영상 기본 연산 / 2절. 기본 연산

Color Image는 3개의 채널 B,G,R로 구성이 되어 있습니다. 이것을 각 채널별로 분리할 수 있습니다.

```
1 img = cv2.imread('lena.jpg')
2 b, g, r = cv2.split(img)
3 print(np.mean(b), np.mean(g), np.mean(r), sep=", ")
```

(105.54556274414062, 98.9676513671875, 180.31714248657227)

```
1 img = cv2.imread('lena.jpg')
2 img[:, :, [0,1]] = 0
3
4 cv2.imshow('img', img)
5 cv2.waitKey(0)
6 cv2.destroyAllWindows()
```



`cv2.split()` 함수는 비용이 많이 드는 함수입니다. 가능하다면 Numpy indexing 방법을 사용하는 것이 더 효율적입니다.

이미지 합치기

2장. 영상 기본 연산 / 2절. 기본 연산

이미지를 더하는 방법은 OpenCV의 `cv2.add()` 함수를 사용하는 방법과

Numpy 연산, 즉 `img1 + img2`로 하는 방법이 있습니다.



OpenCV의 `cv2.add()` 는 Saturation 연산을 하고, Numpy는 Modulo 연산을 합니다.

- ▶ Saturation연산은 한계값을 정하고 그 값을 벗어나는 경우는 모두 특정 값으로 계산하는 방식입니다. 이미지에서는 0 이하는 모두 0, 255 이상은 모두 255로 표현하는 것입니다.
- ▶ Modulo연산은 'a와 b는 n으로 나눈 나머지 값이 같다'라는 의미입니다. 시계를 예로 들면 2와 14는 12로 나눈 나머지가 2로 동일합니다. 이미지에서는 연산의 결과가 256보다 큰 경우는 256으로 나눈 나머지 값으로 결정을 합니다.

OpenCV의 add()와 Numpy의 add()

2장. 영상 기본 연산 / 2절. 기본 연산

Original 1



Original 2



OpenCV Add
(Saturation)



Numpy Add
(Modulo)

