초급 영상처리

(나만의 Opencv 구현하기)

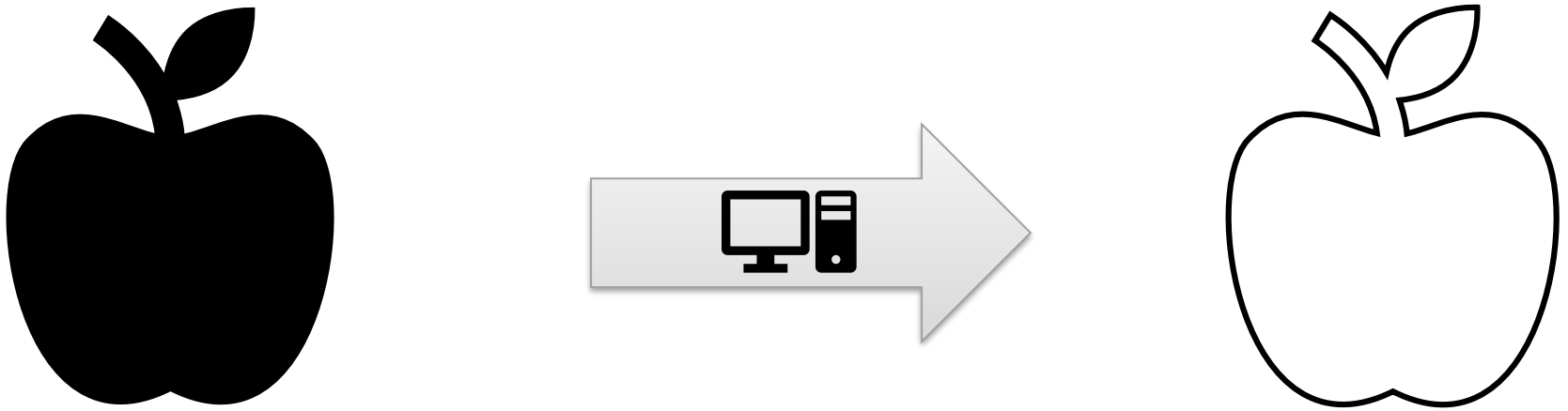
박화종

INDEX

- 영상처리 기초
- 실습
- 과제

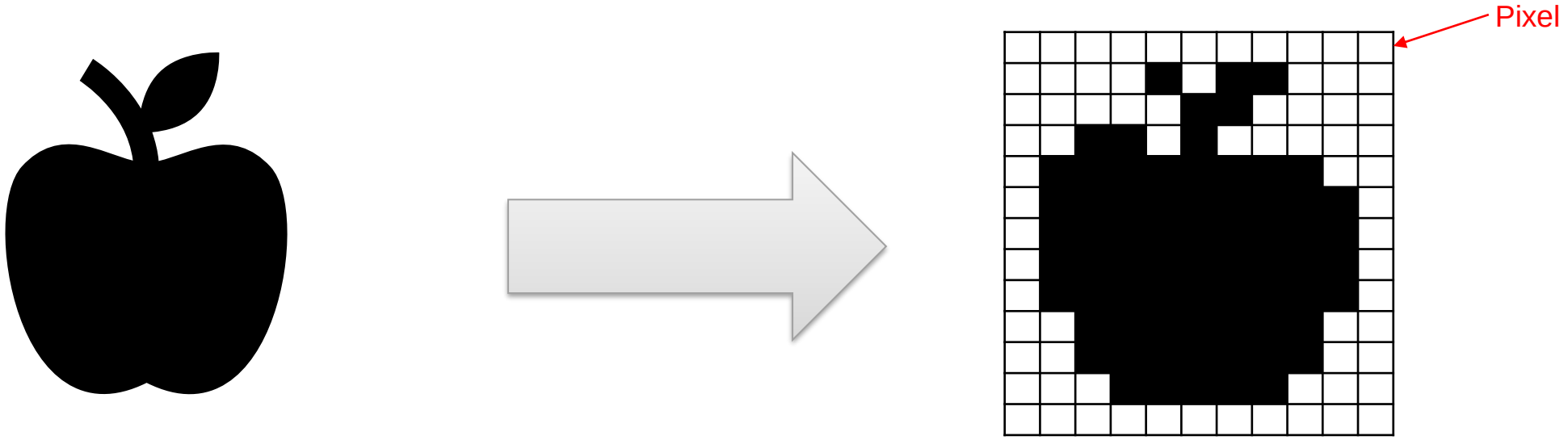
영상처리 기초

- 영상처리란?
  - 영상(Image)을 분석하여 유용한 정보를 도출하는 것



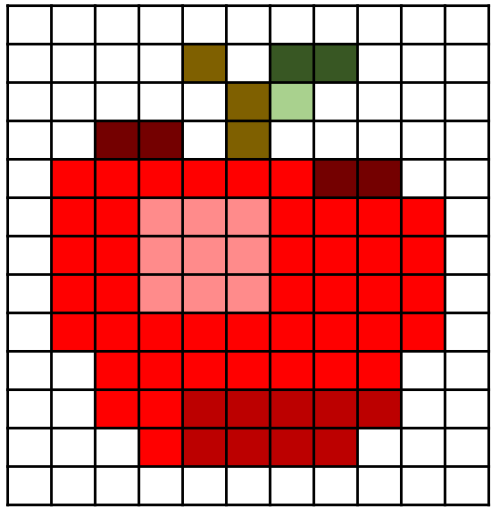
영상처리 기초

- Sampling
  - 연속적인 아날로그 데이터(현실 데이터)를 디지털화 하는 방법



영상처리 기초

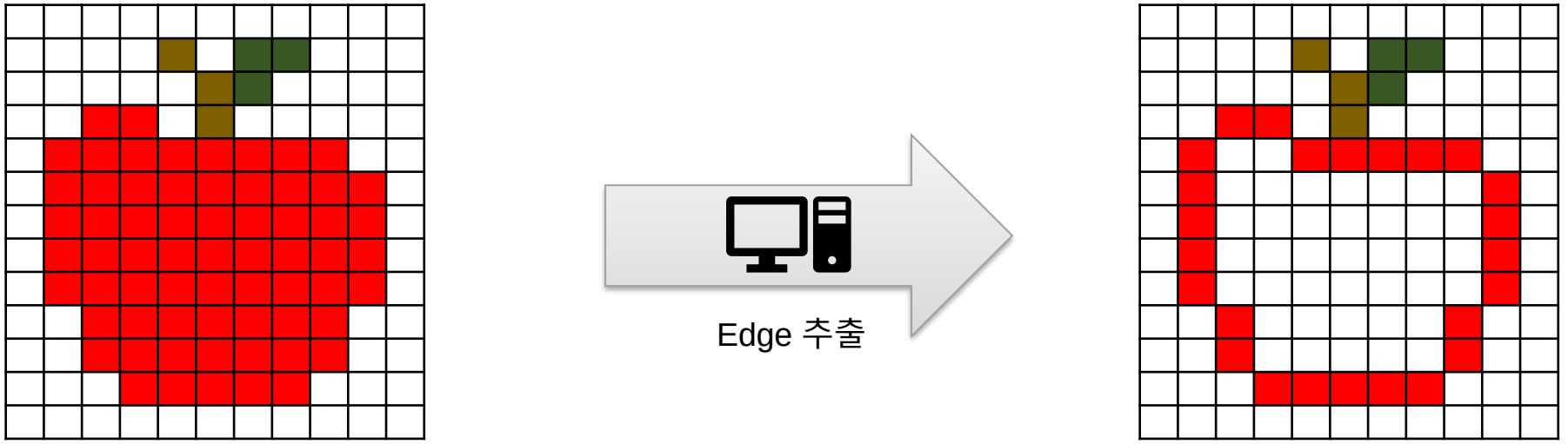
- Quantization
  - Sampling된 각 pixel에 들어갈 값을 근사화 시키는 방법
  - 일반적으로 0~255



출처 : https://pixabay.com/ko/

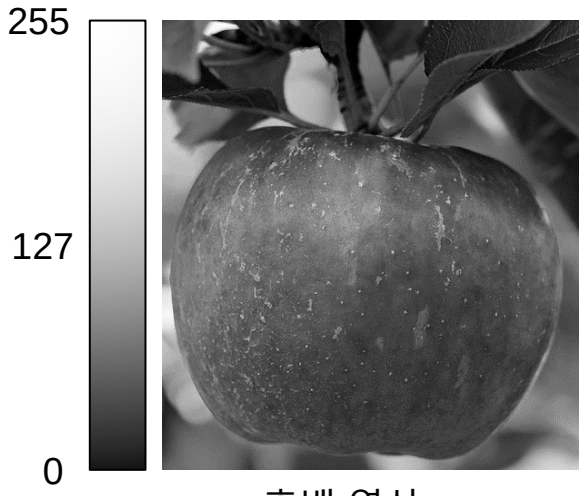
영상처리 기초

- Quantization
  - Sampling된 각 pixel에 들어갈 값을 근사화 시키는 방법
  - 일반적으로 0~255



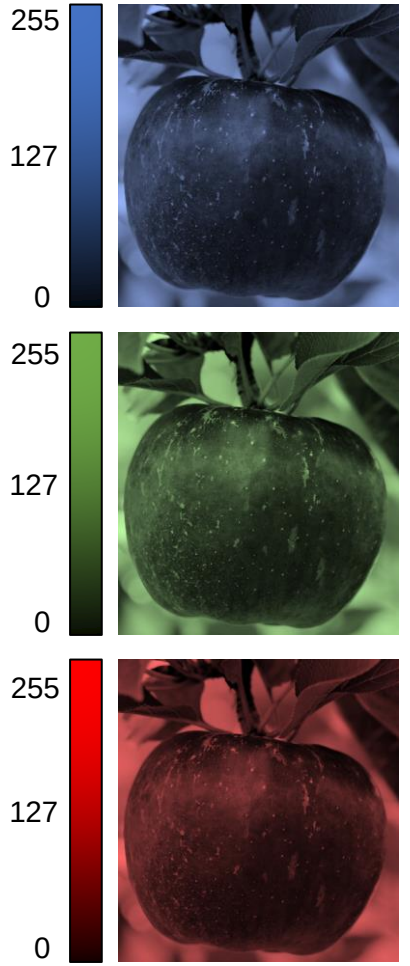
영상처리 기초

- 흑백 영상 vs 컬러 영상(BGR)



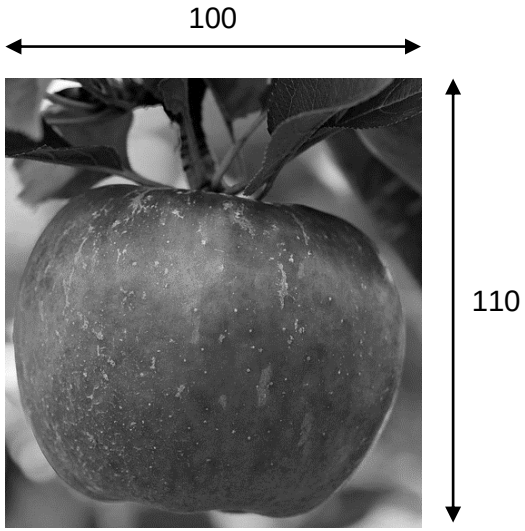
흑백 영상

컬러 영상

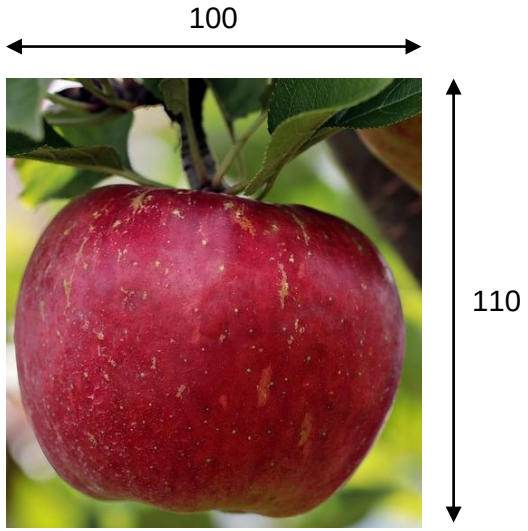


영상처리 기초

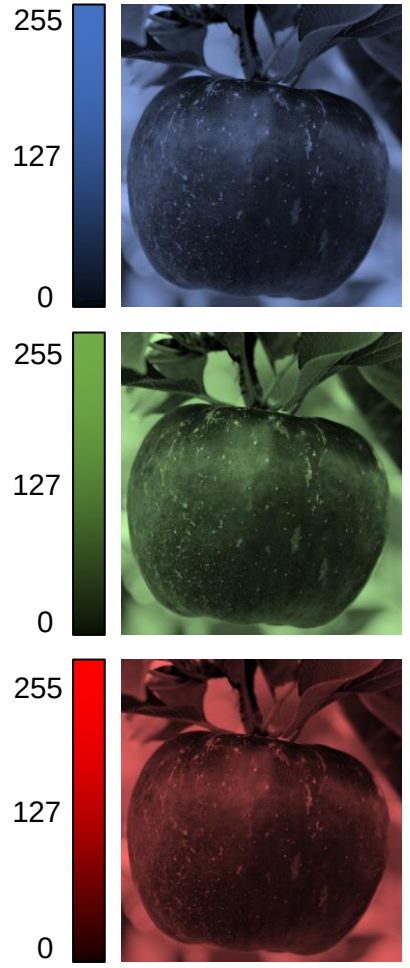
- 흑백 영상 vs 컬러 영상(BGR)



흑백 영상
(110 x 100)



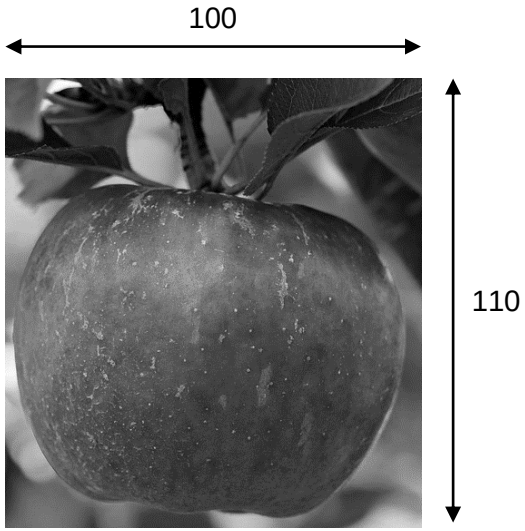
컬러 영상
(110 x 100 x 3)



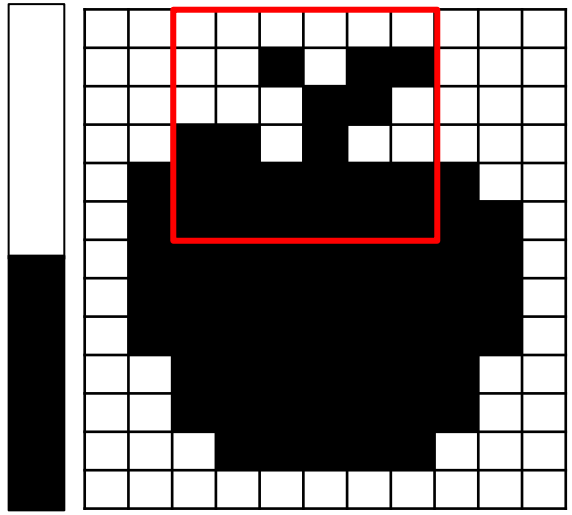
영상처리 기초

- 흑백 영상
  - 강의 대부분 흑백영상 기준으로 설명 및 실습 진행

255	255	255	255	255	255
255	255	0	255	0	0
255	255	255	0	0	255
0	0	255	0	255	255
0	0	0	0	0	0
0	0	0	0	0	0



흑백 영상
(110 x 100)

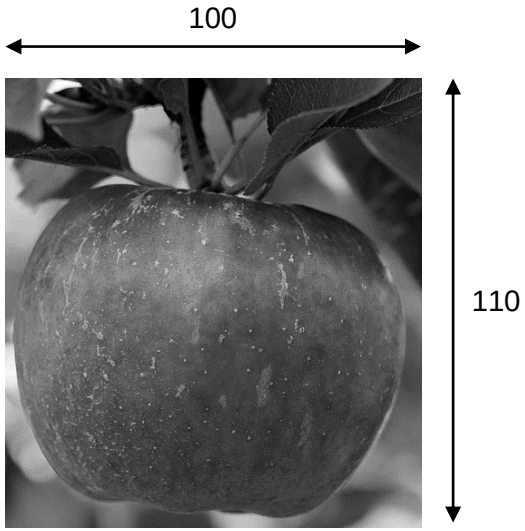


흑백 영상
(13 x 11)

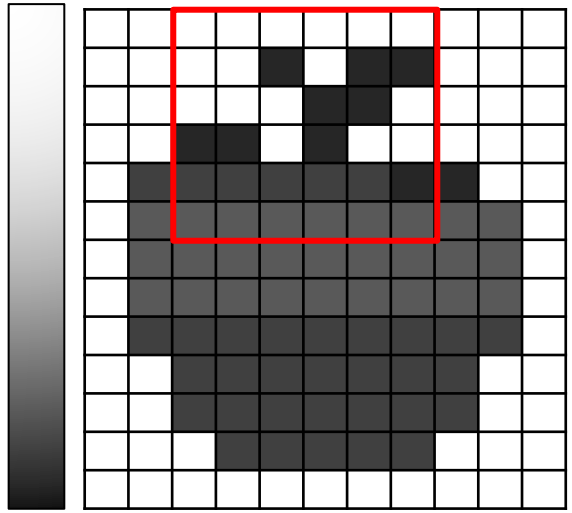
영상처리 기초

- 흑백 영상
  - 강의 대부분 흑백영상 기준으로 설명 및 실습 진행

255	255	255	255	255	255
255	255	23	255	19	21
255	255	255	22	20	255
19	11	255	20	255	255
50	53	52	49	50	9
63	62	60	63	61	58



흑백 영상
(110 x 100)



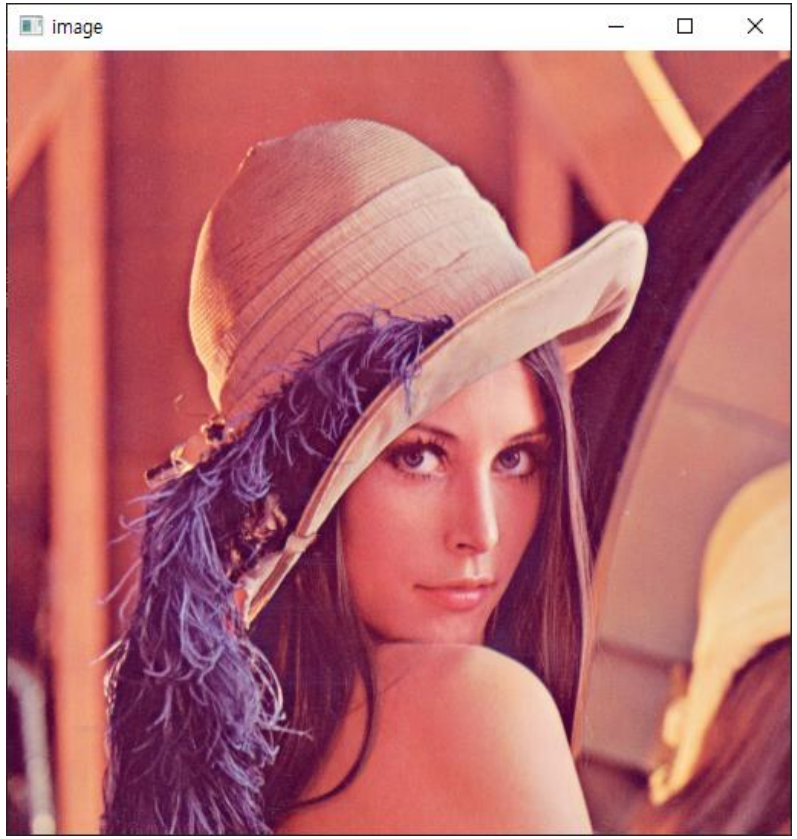
흑백 영상
(13 x 11)

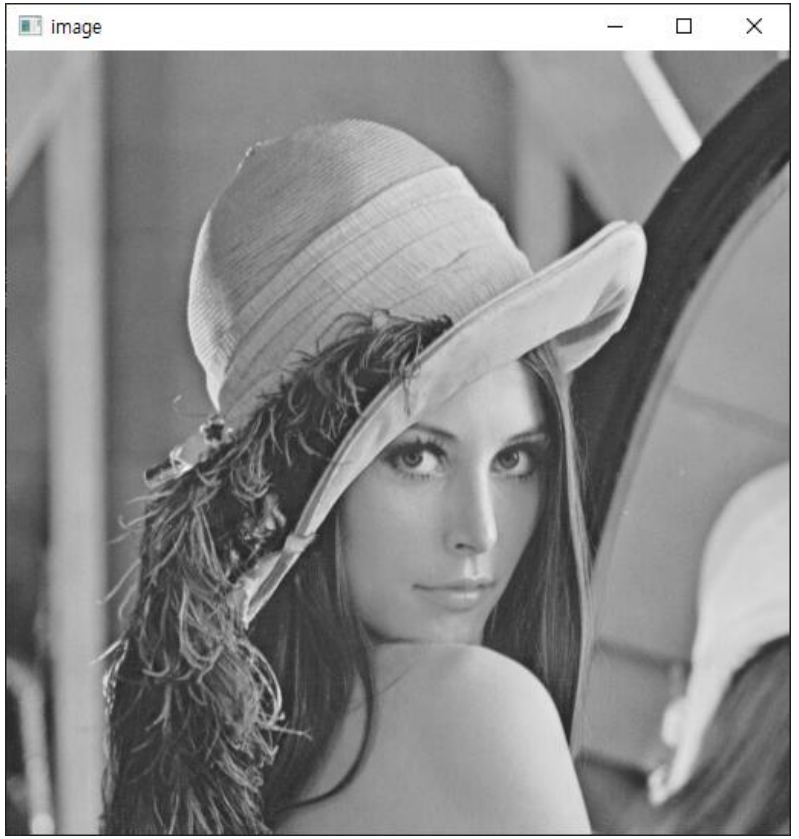
영상처리 기초

- OpenCV 흑백영상
  - 2차원 : (높이, 너비)
  - 0 ~ 255사이의 값을 가짐
  - Data type : uint8
    - Overflow & Underflow 문제 발생 가능

실습(IP1_1)

- 컬러영상 읽기 및 화면에 출력하기
- 흑백영상 읽기 및 화면에 출력하기





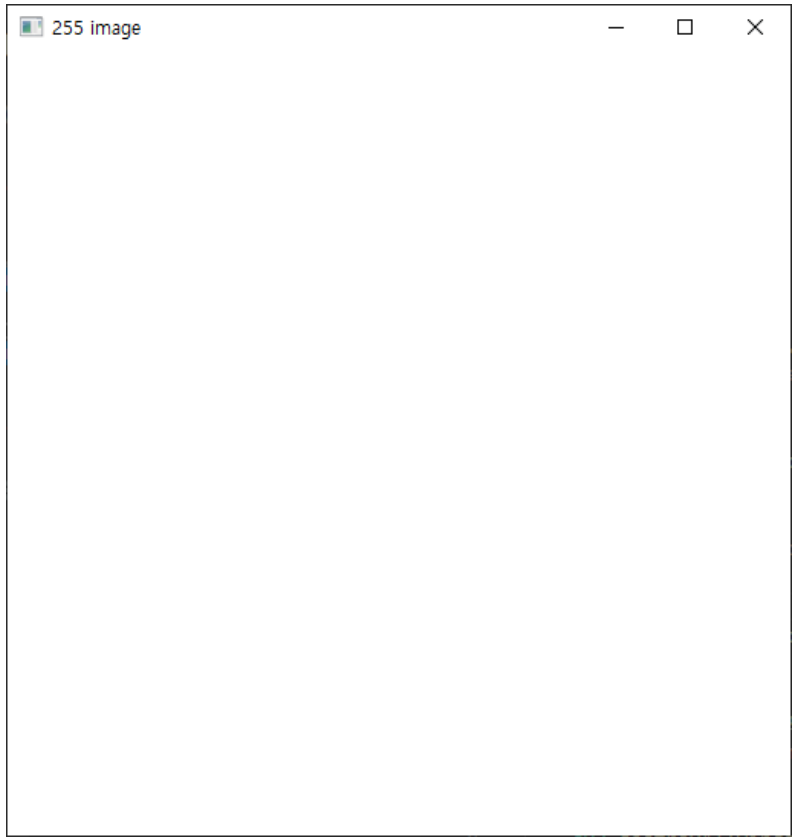
실습(IP1_1)

- 컬러영상 읽기 및 화면에 출력하기
- 흑백영상 읽기 및 화면에 출력하기

```python
import cv2

def main():
    #img = cv2.imread('lena.png')
    img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE)

    cv2.imshow('image', img)
    cv2.waitKey()
    cv2.destroyAllWindows()


if __name__ =='__main__':
    main()
```

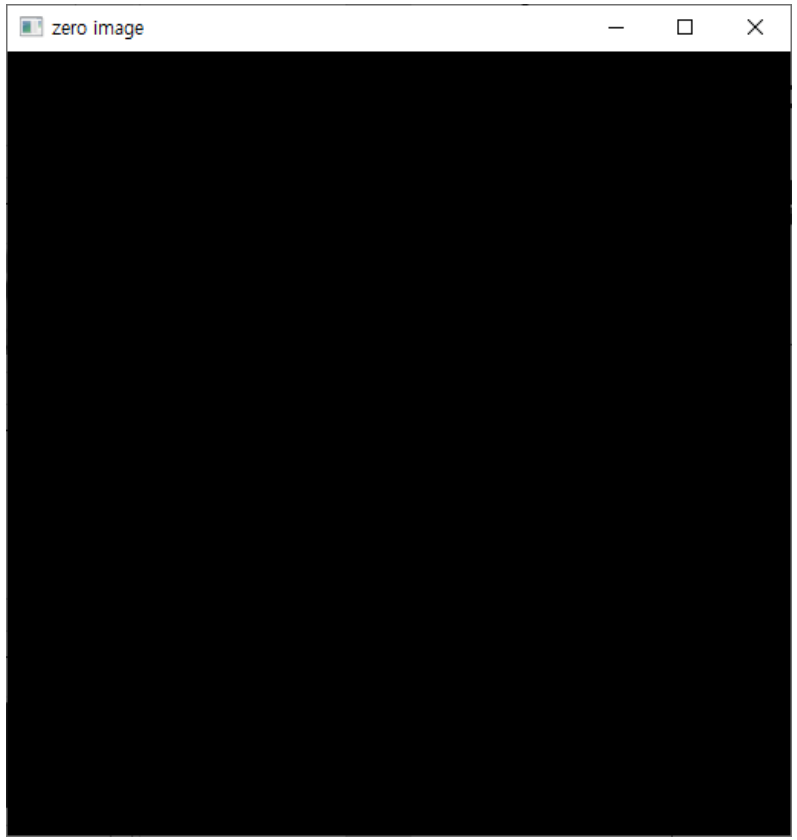
실습(IP1_2)

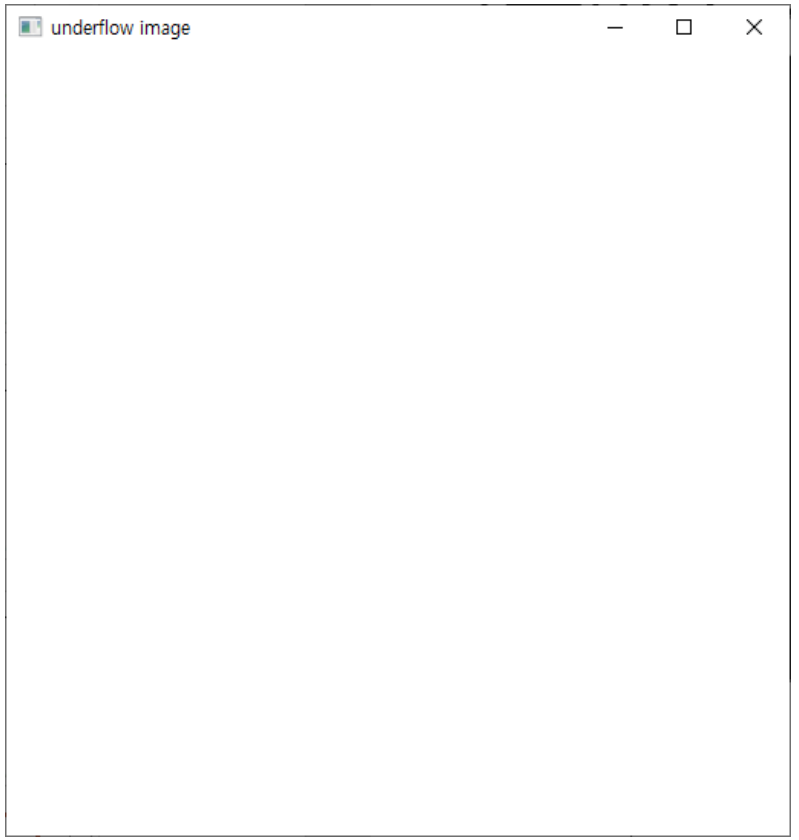
- Overflow 및 Underflow 확인해보기



실습(IP1_2)

- Overflow 및 Underflow 확인해보기





실습(IP1_2)

- Overflow 및 Underflow 확인해보기

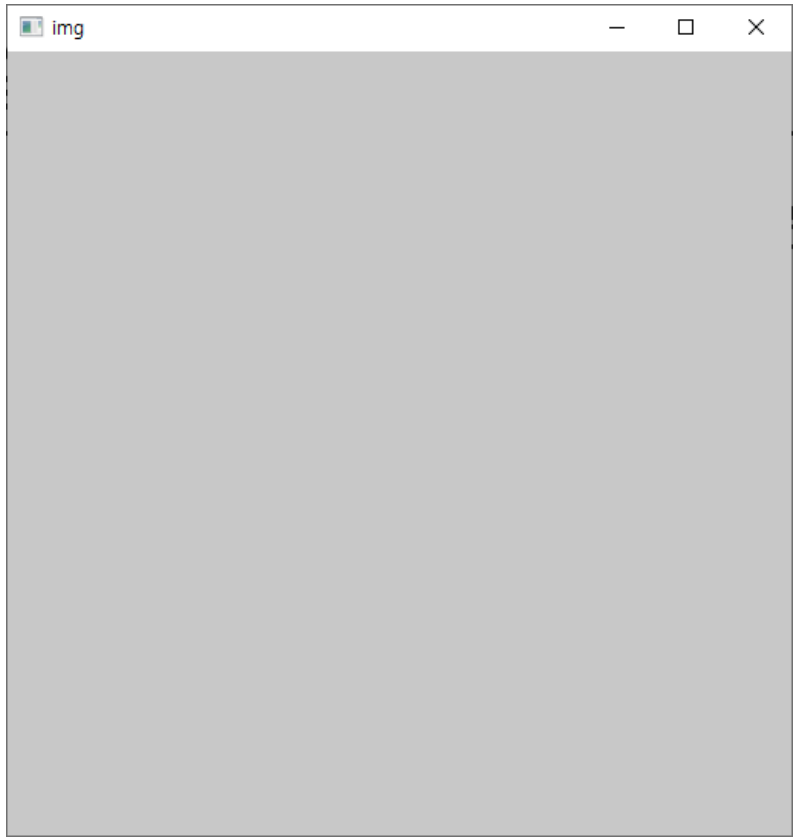
```python
import cv2
import numpy as np

def main():
    showOverflow()
    showUnderflow()

def showOverflow():
    img = np.full((512,512), 255, dtype=np.uint8)
    cv2.imshow('255 image', img)

    cv2.waitKey()
    cv2.destroyAllWindows()

    img_over = img + 1
    cv2.imshow('overflow image', img_over)

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

```python
def showUnderflow():
    img = np.zeros((512,512), dtype=np.uint8)
    cv2.imshow('zero image', img)

    cv2.waitKey()
    cv2.destroyAllWindows()

    img_under = img - 1
    cv2.imshow('underflow image', img_under)

    cv2.waitKey()
    cv2.destroyAllWindows()



if __name__ =='__main__':
    main()
```

실습(IP1_3)

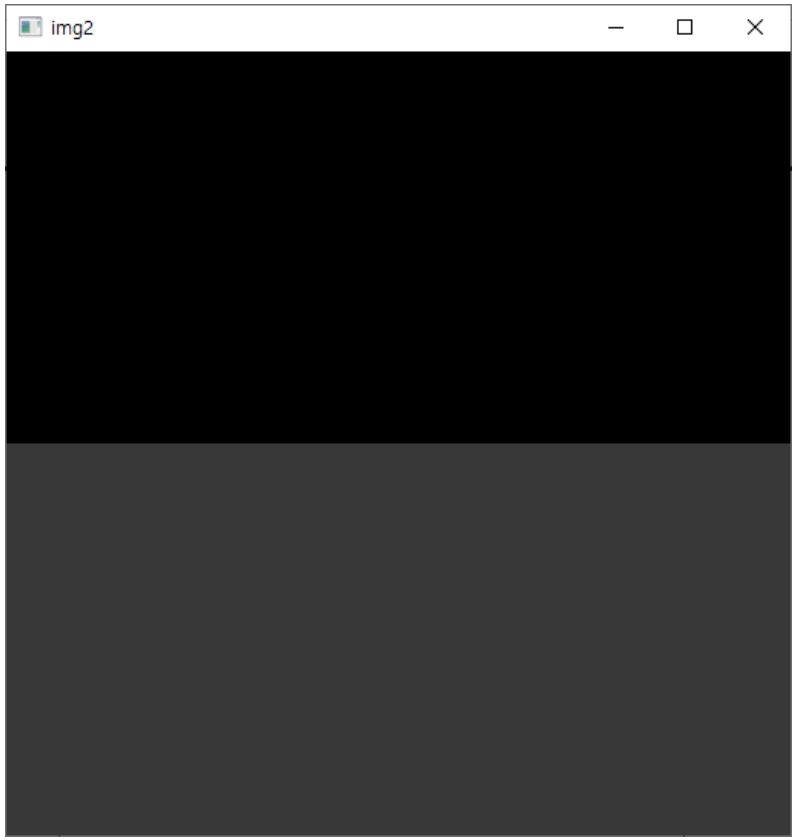
- Overflow 및 Underflow 예방하기

전부 200



0

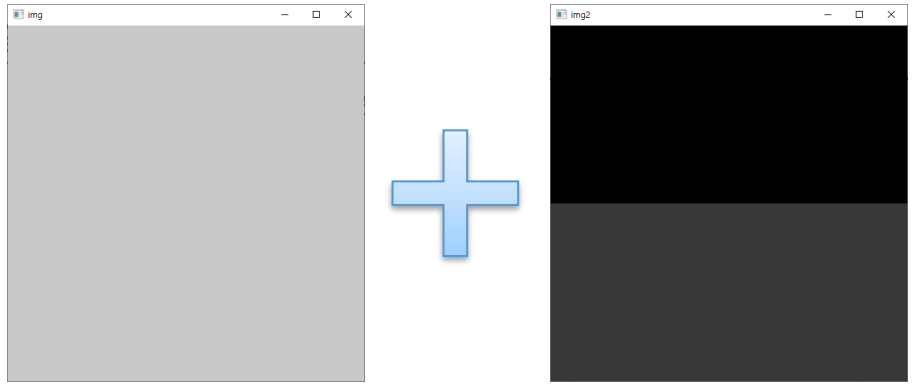
56

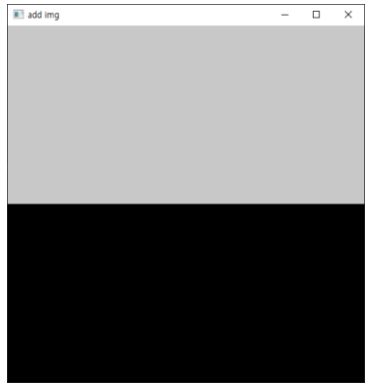


실습(IP1_3)

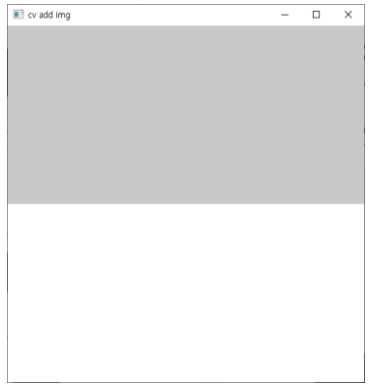
- Overflow 및 Underflow 예방하기

```python
def main():
    addTest()

def addTest():
    img = np.full((512,512), 200, dtype=np.uint8)

    img2 = np.zeros_like(img)
    img2[256:] = 56

    cv2.imshow('img', img)
    cv2.imshow('img2', img2)
    cv2.waitKey()
    cv2.destroyAllWindows()

    img_add = img + img2
    cv2.imshow('add img', img_add)
    cv2.waitKey()
    cv2.destroyAllWindows()

    img_cvadd = cv2.add(img, img2)
    cv2.imshow('cv add img', img_cvadd)
    cv2.waitKey()
    cv2.destroyAllWindows()
```





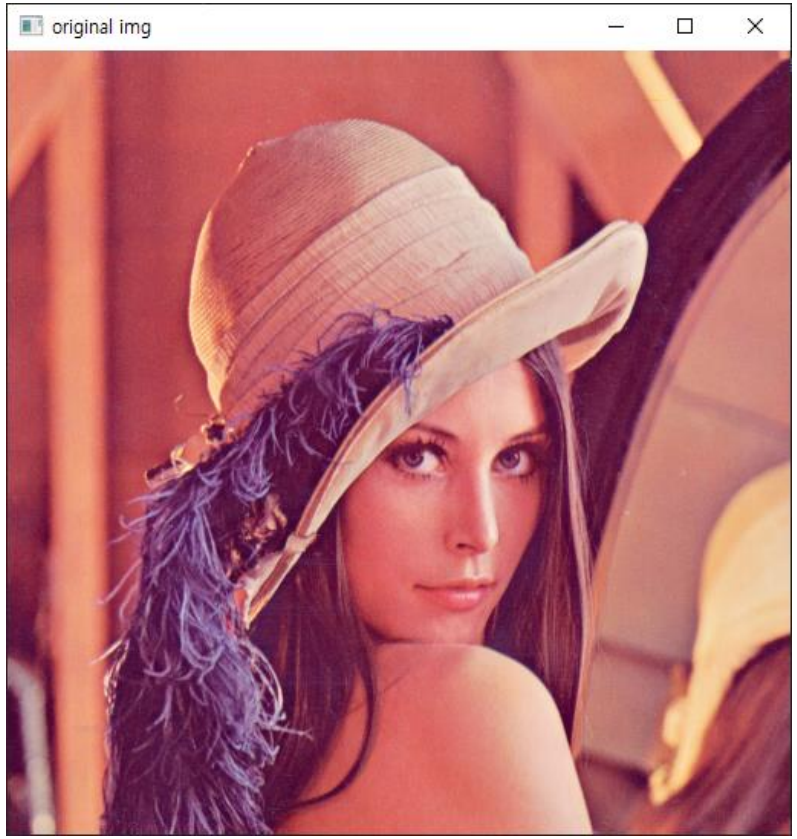
Img_add



Img_cvadd

실습(IP1_4)

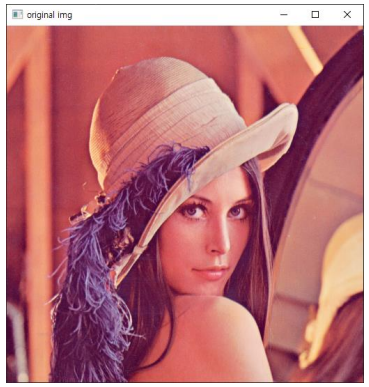
- 컬러 영상을 흑백 영상으로 변환하기

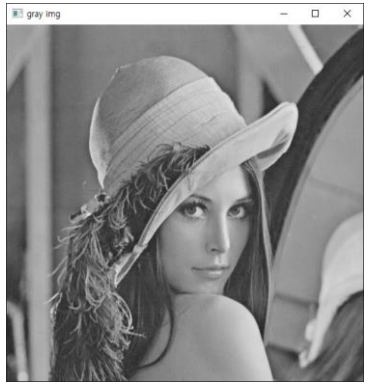


실습(IP1_4)

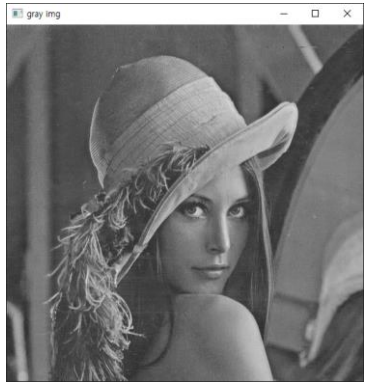
- 컬러 영상 변환하기

```
def main():
    img = cv2.imread('lena.png')
    cv2.imshow('original img', img)

    gray = BGR2Gray1(img)
    cv2.imshow('gray img', gray.astype(np.uint8))

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



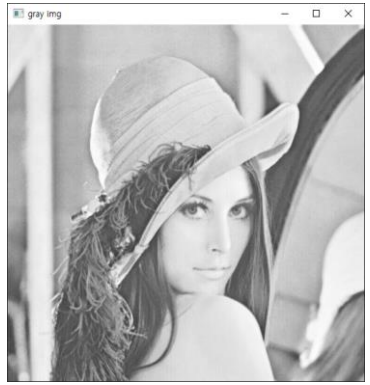



```
def BGR2Gray1(img):
    img = img.astype(np.float32)
    gray = (img[..., 0] + img[..., 1] + img[..., 2]) / 3
    return gray
```




```
def BGR2Gray2(img):
    gray = img[..., 0]
    return gray
```


```
def BGR2Gray3(img):
    gray = img[..., 1]
    return gray
```

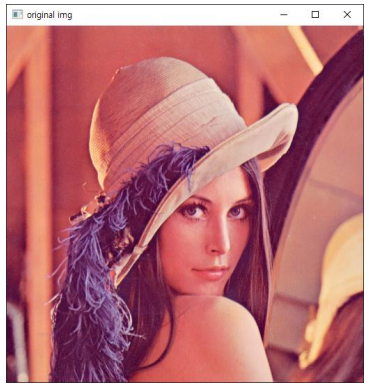


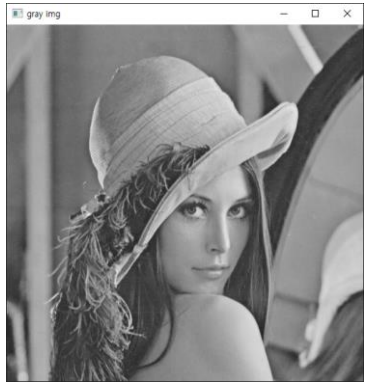

```
def BGR2Gray4(img):
    gray = img[..., 2]
    return gray
```

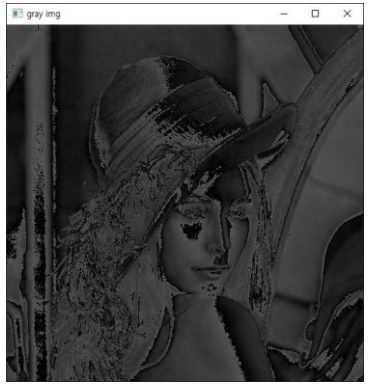
실습(IP1_4)

- 컬러 영상 변환하기

```
def main():
    img = cv2.imread('lena.png')
    cv2.imshow('original img', img)

    gray = BGR2Gray1(img)
    cv2.imshow('gray img', gray.astype(np.uint8))

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





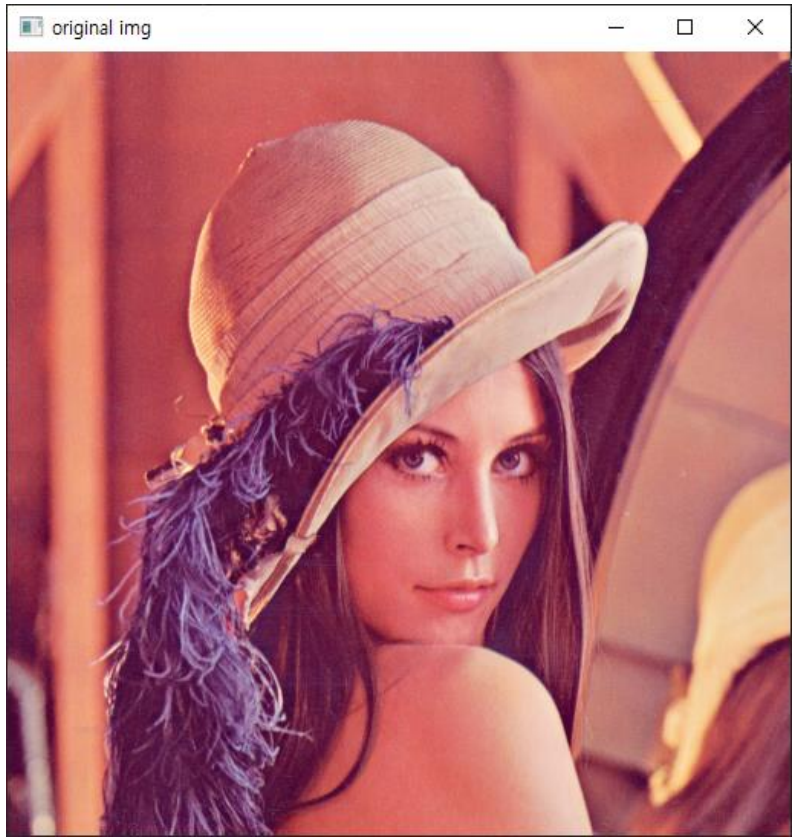



```
def BGR2Gray1(img):
    img = img.astype(np.float32)
    gray = (img[..., 0] + img[..., 1] + img[..., 2]) / 3
    return gray
```

```
def BGR2Gray1(img):
    #img = img.astype(np.float32)
    gray = (img[..., 0] + img[..., 1] + img[..., 2]) / 3
    return gray
```

과제(IP1_test1)

- 컬러 영상 변환하기



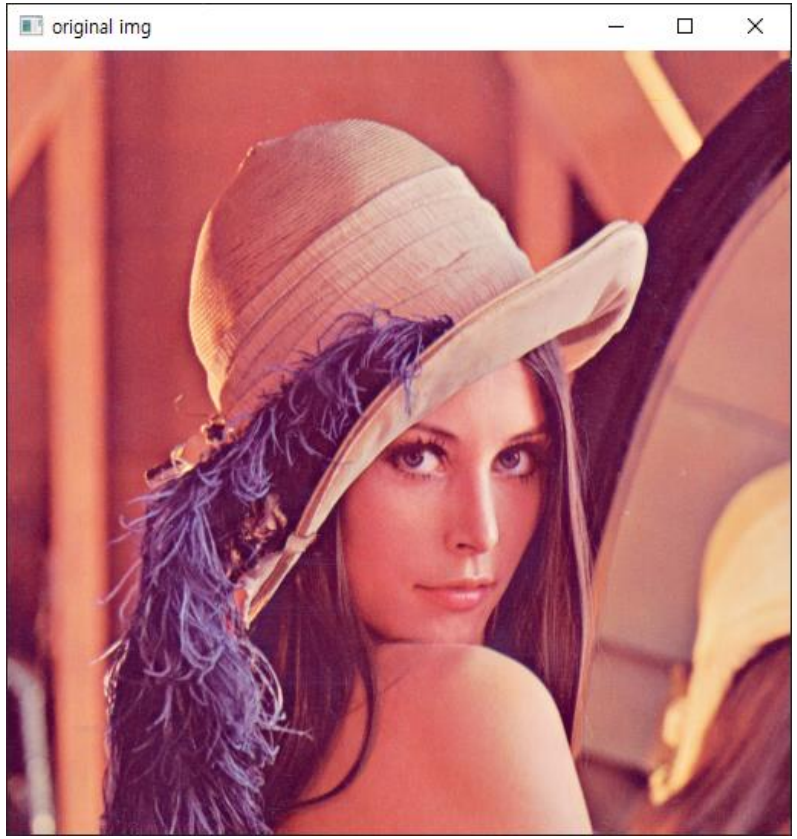
과제(IP1_test1)

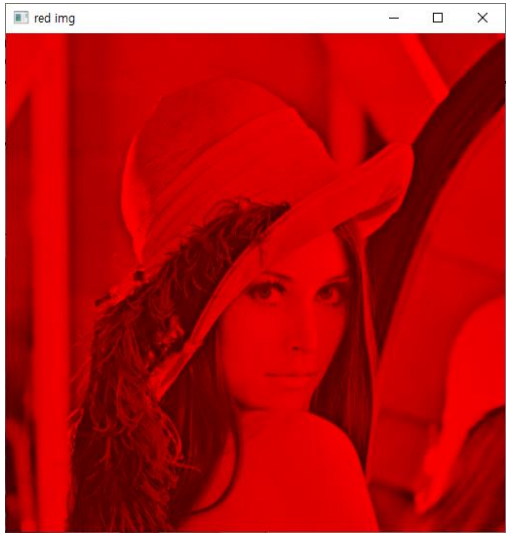
- 컬러 영상 변환하기
  - 과연 B, G, R 각각 1/3씩 사용하는 것이 맞는가?

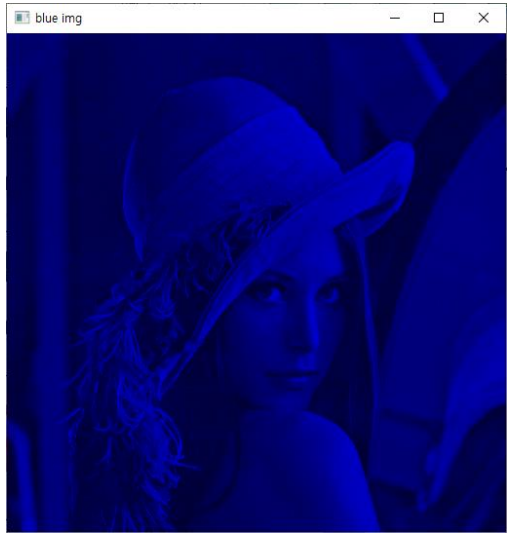
```python
def main():
    img = cv2.imread('lena.png')
    cv2.imshow('original img', img)

    gray = BGR2Gray(img)
    cv2.imshow('gray img', gray.astype(np.uint8))

    cv2.waitKey()
    cv2.destroyAllWindows()

def BGR2Gray(img):
    """
    [input]
    img : color image

    [output]
    gray : gray image
    """

    return gray
```

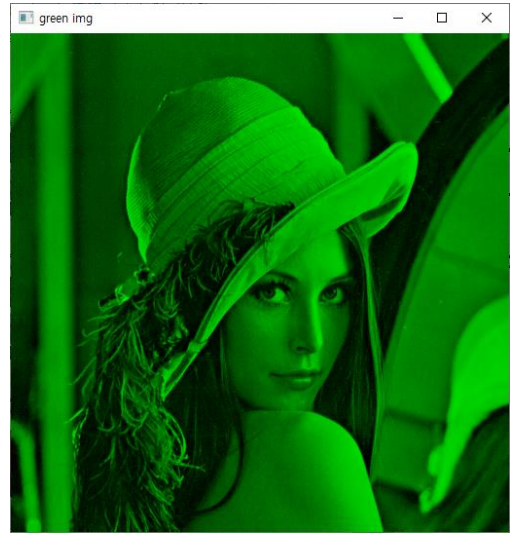
과제(IP1_test2)

- 컬러 영상 변환하기2









과제(IP1_test2)

- 컬러 영상 변환하기2
  - .copy() 붙인 이유도 찾아보기

```python
def main():
    img = cv2.imread('lena.png')
    cv2.imshow('original img', img)

    red = BGR2Red(img.copy())
    cv2.imshow('red img', red)

    green = BGR2Green(img.copy())
    cv2.imshow('green img', green)

    blue = BGR2Blue(img.copy())
    cv2.imshow('blue img', blue)

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

```python
def BGR2Red(img):
    """
    [input]
    img : color image

    [output]
    img : red image
    """

    return img
```

```python
def BGR2Green(img):
    """
    [input]
    img : color image

    [output]
    img : green image
    """

    return img
```

```python
def BGR2Blue(img):
    """
    [input]
    img : color image

    [output]
    img : blue image
    """

    return img
```

QnA