초급 영상처리

(나만의 Opencv 구현하기)

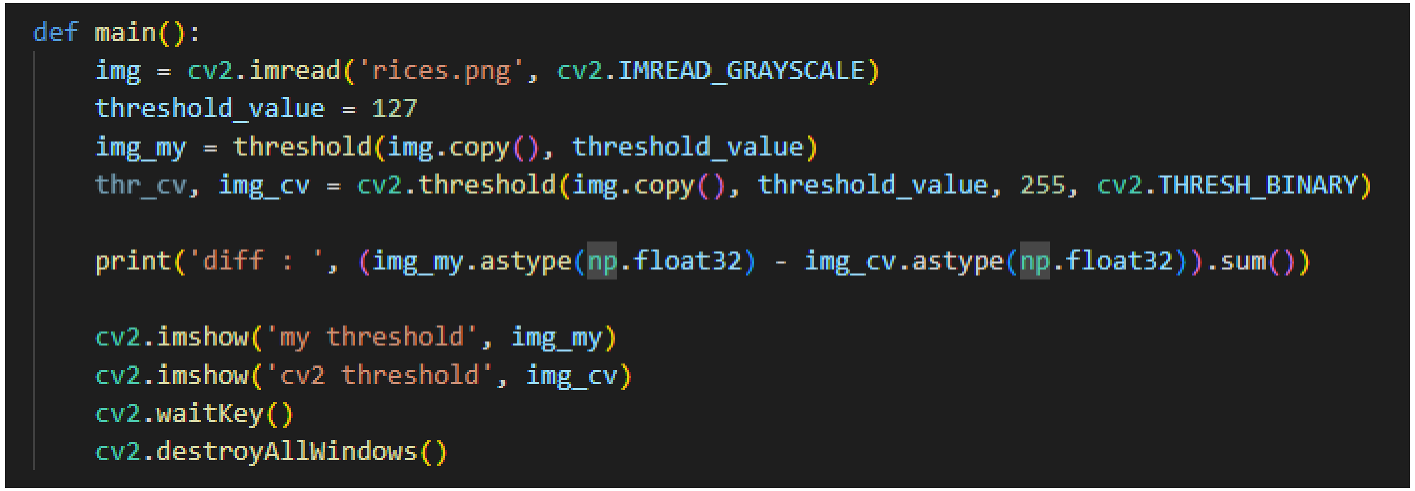
박화종

INDEX

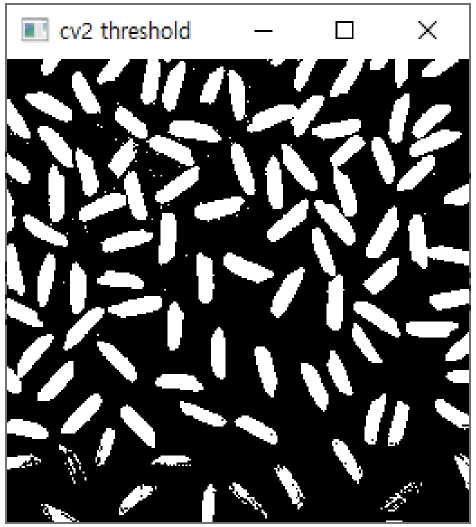
- 저번 주 과제 정답
- Noise
- 실습
- 과제

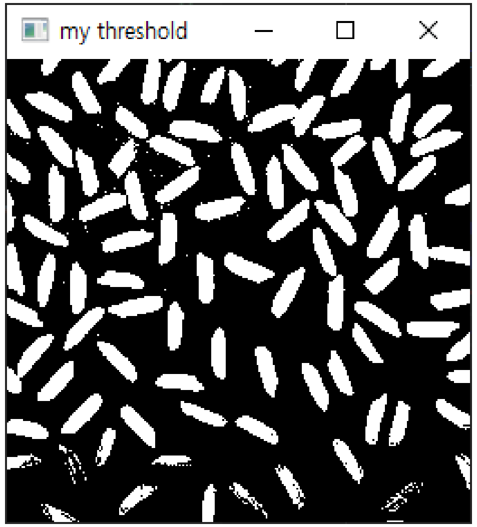
저번 주 과제(IP3_test1)

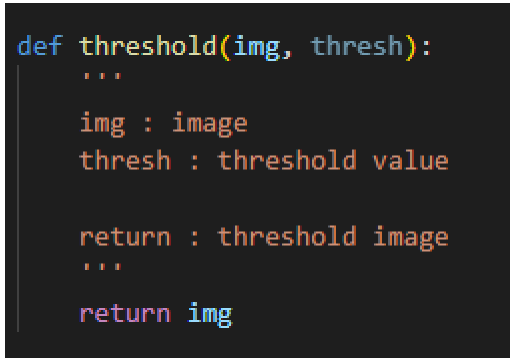
- Threshold 함수 직접 구현하기



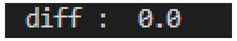
```python
def main():
    img = cv2.imread('rices.png', cv2.IMREAD_GRAYSCALE)
    threshold_value = 127
    img_my = threshold(img.copy(), threshold_value)
    thr_cv, img_cv = cv2.threshold(img.copy(), threshold_value, 255, cv2.THRESH_BINARY)

    print('diff : ', (img_my.astype(np.float32) - img_cv.astype(np.float32)).sum())

    cv2.imshow('my threshold', img_my)
    cv2.imshow('cv2 threshold', img_cv)
    cv2.waitKey()
    cv2.destroyAllWindows()
```







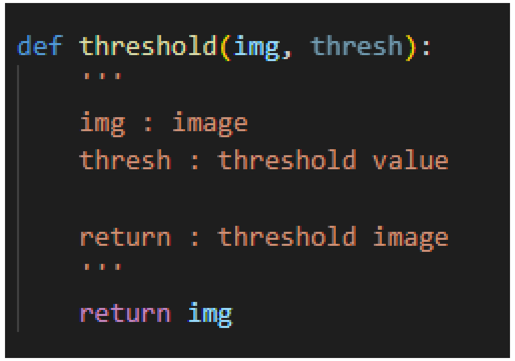
```python
def threshold(img, thresh):
    '''
    img : image
    thresh : threshold value

    return : threshold image
    '''
    return img
```



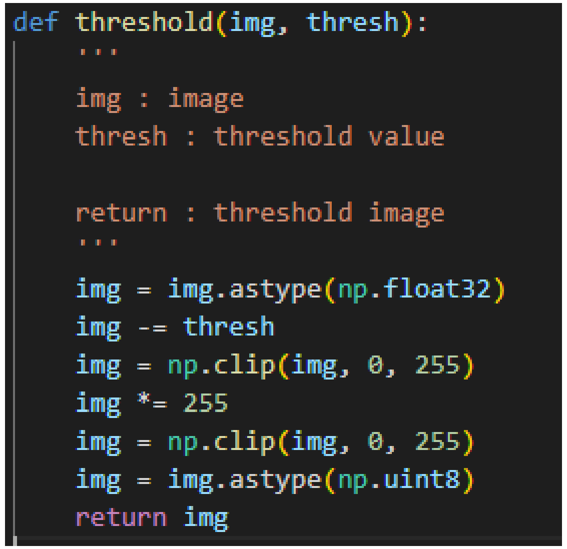
```
diff :  0.0
```

저번 주 과제(IP3_test1)

- Threshold 함수 직접 구현하기



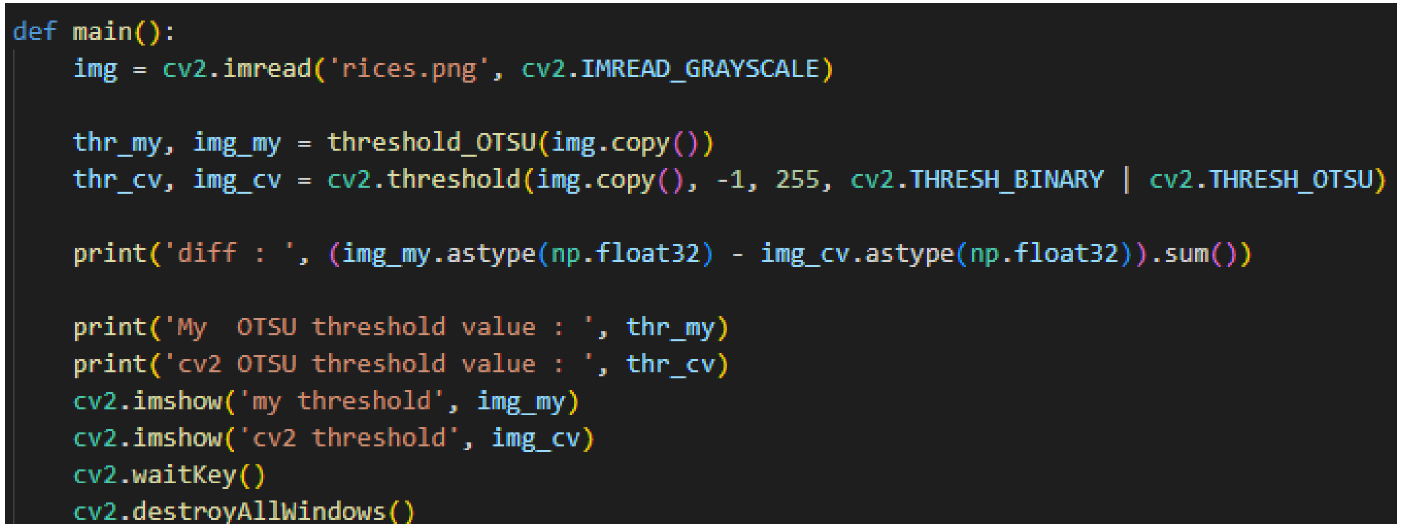
```python
def threshold(img, thresh):
    '''
    img : image
    thresh : threshold value

    return : threshold image
    '''
    return img
```



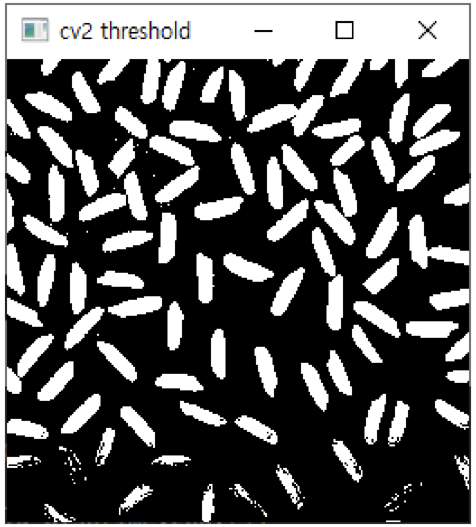
```python
def threshold(img, thresh):
    '''
    img : image
    thresh : threshold value

    return : threshold image
    '''
    img = img.astype(np.float32)
    img -= thresh
    img = np.clip(img, 0, 255)
    img *= 255
    img = np.clip(img, 0, 255)
    img = img.astype(np.uint8)
    return img
```

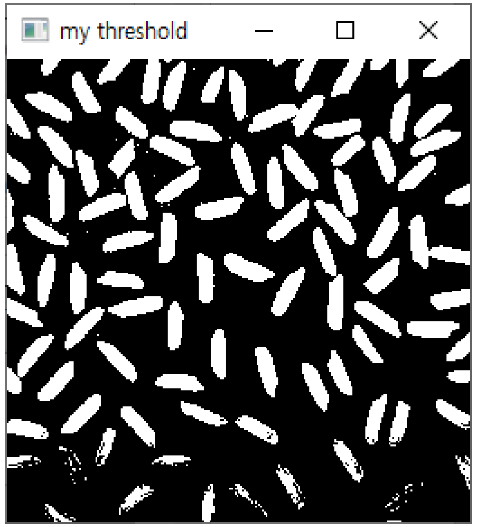
저번 주 과제(IP3_test2)

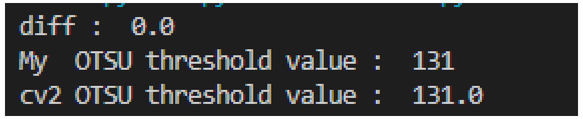
- OTSU's method 직접 구현하기



```
def main():
    img = cv2.imread('rices.png', cv2.IMREAD_GRAYSCALE)

    thr_my, img_my = threshold_OTSU(img.copy())
    thr_cv, img_cv = cv2.threshold(img.copy(), -1, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)

    print('diff : ', (img_my.astype(np.float32) - img_cv.astype(np.float32)).sum())

    print('My  OTSU threshold value : ', thr_my)
    print('cv2 OTSU threshold value : ', thr_cv)
    cv2.imshow('my threshold', img_my)
    cv2.imshow('cv2 threshold', img_cv)
    cv2.waitKey()
    cv2.destroyAllWindows()
```



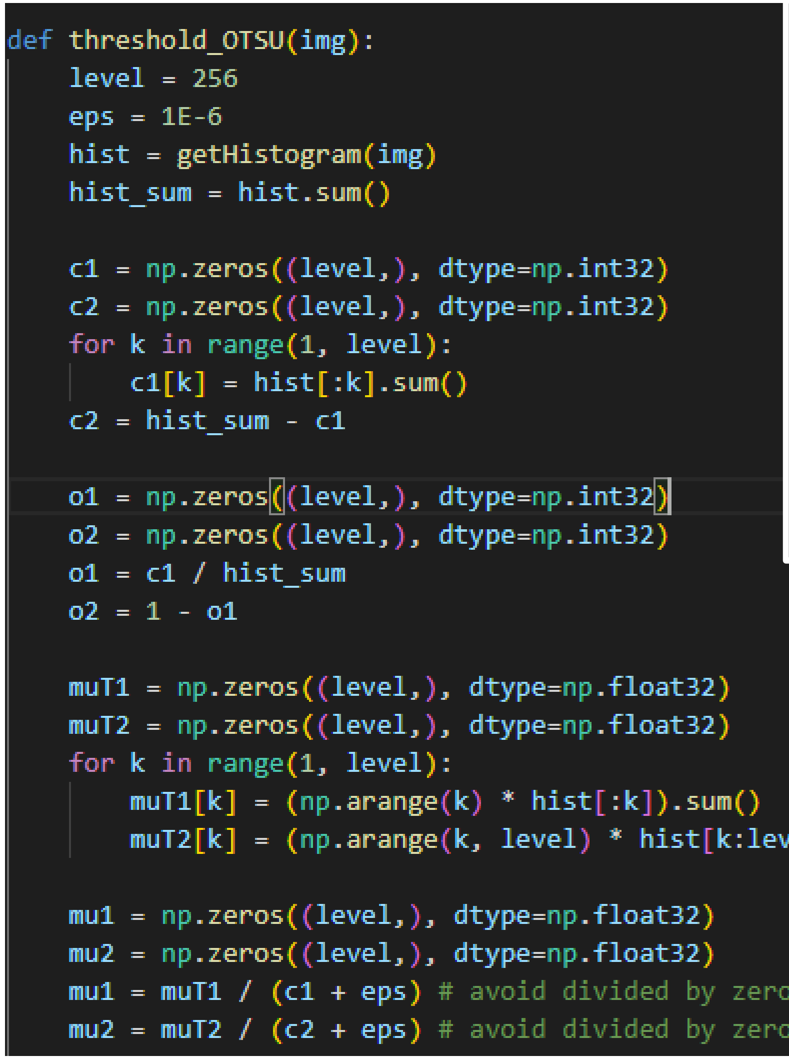




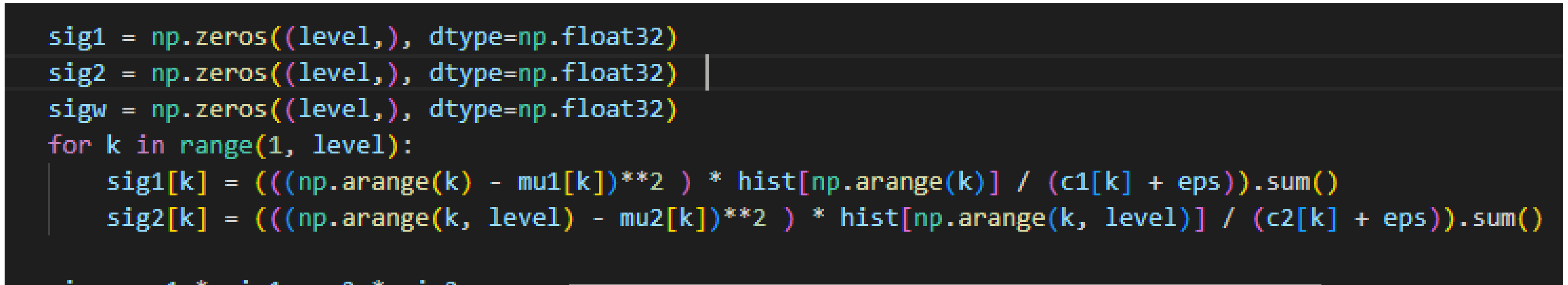
```
diff :  0.0
My  OTSU threshold value :  131
cv2 OTSU threshold value :  131.0
```

저번 주 과제(IP3_test2)

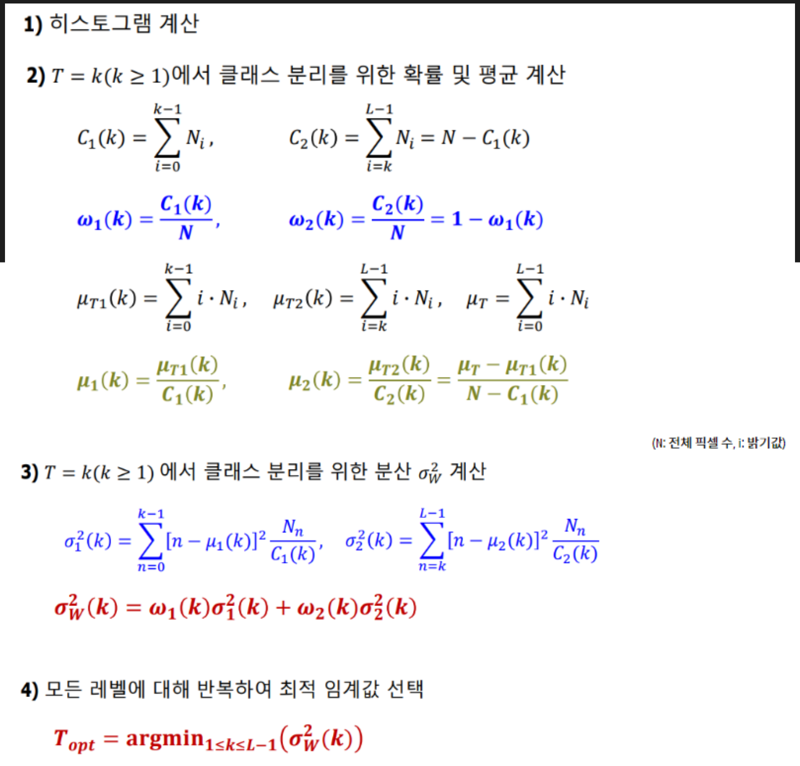
- OTSU's method 직접 구현하기



```python
def threshold_OTSU(img):
    level = 256
    eps = 1E-6
    hist = getHistogram(img)
    hist_sum = hist.sum()

    c1 = np.zeros((level,), dtype=np.int32)
    c2 = np.zeros((level,), dtype=np.int32)
    for k in range(1, level):
        c1[k] = hist[:k].sum()
    c2 = hist_sum - c1

    o1 = np.zeros((level,), dtype=np.int32)
    o2 = np.zeros((level,), dtype=np.int32)
    o1 = c1 / hist_sum
    o2 = 1 - o1

    muT1 = np.zeros((level,), dtype=np.float32)
    muT2 = np.zeros((level,), dtype=np.float32)
    for k in range(1, level):
        muT1[k] = (np.arange(k) * hist[:k]).sum()
        muT2[k] = (np.arange(k, level) * hist[k:level]).sum()

    mu1 = np.zeros((level,), dtype=np.float32)
    mu2 = np.zeros((level,), dtype=np.float32)
    mu1 = muT1 / (c1 + eps) # avoid divided by zero
    mu2 = muT2 / (c2 + eps) # avoid divided by zero
```



```python
    sig1 = np.zeros((level,), dtype=np.float32)
    sig2 = np.zeros((level,), dtype=np.float32)
    sigw = np.zeros((level,), dtype=np.float32)
    for k in range(1, level):
        sig1[k] = (((np.arange(k) - mu1[k])**2 ) * hist[np.arange(k)] / (c1[k] + eps)).sum()
        sig2[k] = (((np.arange(k, level) - mu2[k])**2 ) * hist[np.arange(k, level)] / (c2[k] + eps)).sum()

    sigw = o1 * sig1 + o2 * sig2

    thr = sigw[1:].argmin()

    img = threshold(img, thr)

    return thr, img
```



1) 히스토그램 계산

2) $T = k(k \geq 1)$에서 클래스 분리를 위한 확률 및 평균 계산

$$C_1(k) = \sum_{i=0}^{k-1} N_i\,, \qquad C_2(k) = \sum_{i=k}^{L-1} N_i = N - C_1(k)$$

$$\boldsymbol{\omega_1(k) = \frac{C_1(k)}{N}}, \qquad \boldsymbol{\omega_2(k) = \frac{C_2(k)}{N} = 1 - \omega_1(k)}$$

$$\mu_{T1}(k) = \sum_{i=0}^{k-1} i \cdot N_i\,, \quad \mu_{T2}(k) = \sum_{i=k}^{L-1} i \cdot N_i\,, \quad \mu_T = \sum_{i=0}^{L-1} i \cdot N_i$$

$$\boldsymbol{\mu_1(k) = \frac{\mu_{T1}(k)}{C_1(k)}}, \qquad \boldsymbol{\mu_2(k) = \frac{\mu_{T2}(k)}{C_2(k)} = \frac{\mu_T - \mu_{T1}(k)}{N - C_1(k)}}$$

(N: 전체 픽셀 수, i: 밝기값)

3) $T = k(k \geq 1)$ 에서 클래스 분리를 위한 분산 σ_W^2 계산

$$\sigma_1^2(k) = \sum_{n=0}^{k-1} [n - \mu_1(k)]^2 \frac{N_n}{C_1(k)}, \quad \sigma_2^2(k) = \sum_{n=k}^{L-1} [n - \mu_2(k)]^2 \frac{N_n}{C_2(k)}$$

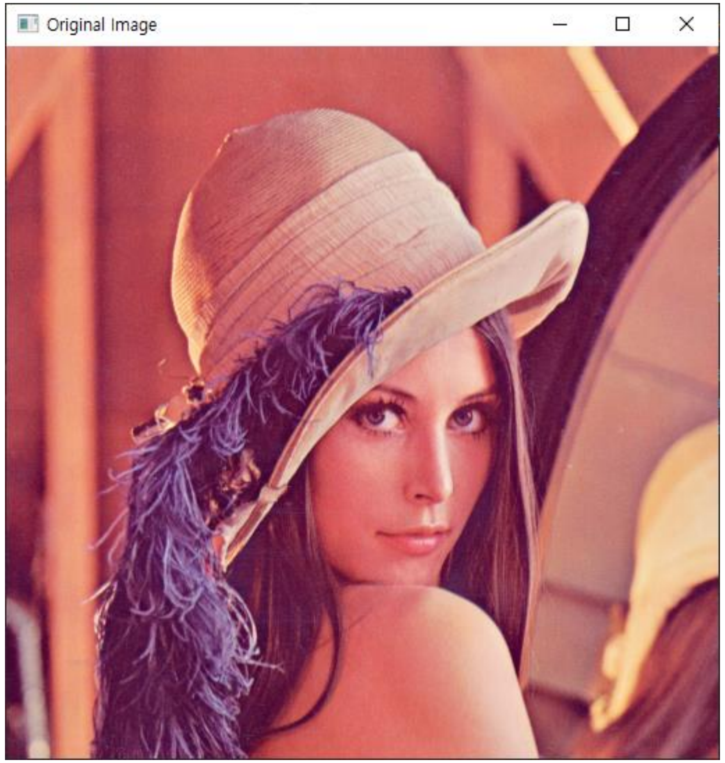
$$\boldsymbol{\sigma_W^2(k) = \omega_1(k)\sigma_1^2(k) + \omega_2(k)\sigma_2^2(k)}$$

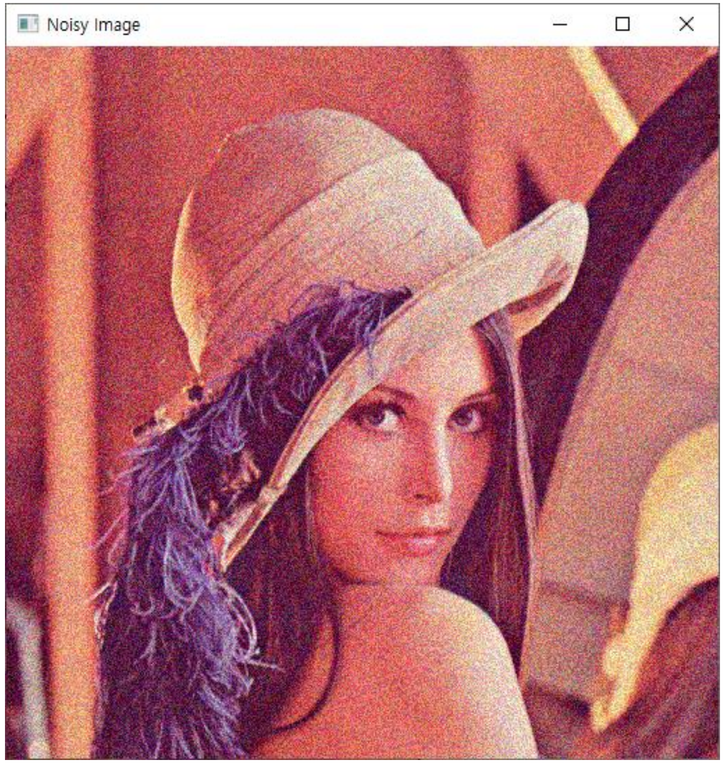
4) 모든 레벨에 대해 반복하여 최적 임계값 선택

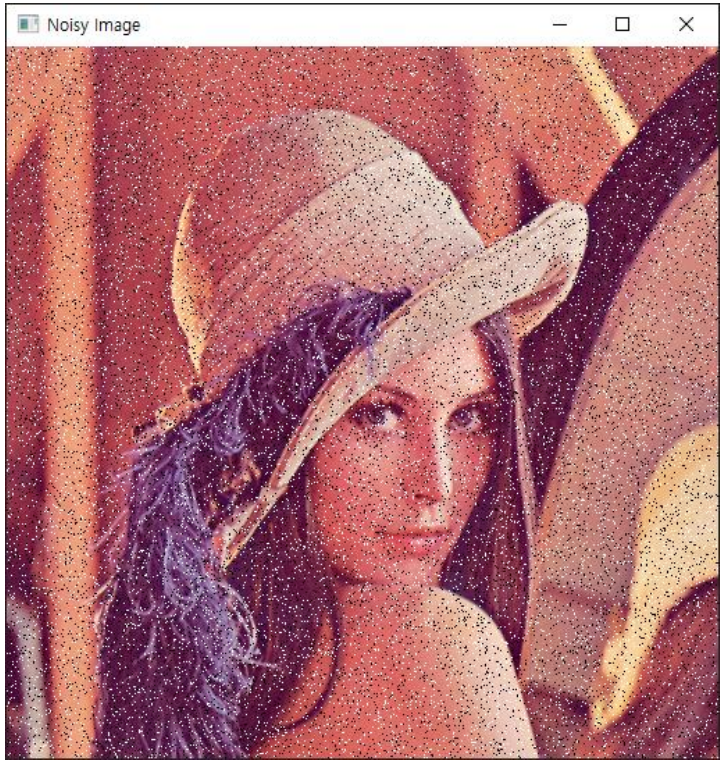
$$\boldsymbol{T_{opt} = \mathrm{argmin}_{1 \leq k \leq L-1}\left(\sigma_W^2(k)\right)}$$

Noise

- Noise : 이미지의 밝기 또는 색상 정보의 무작위 변화







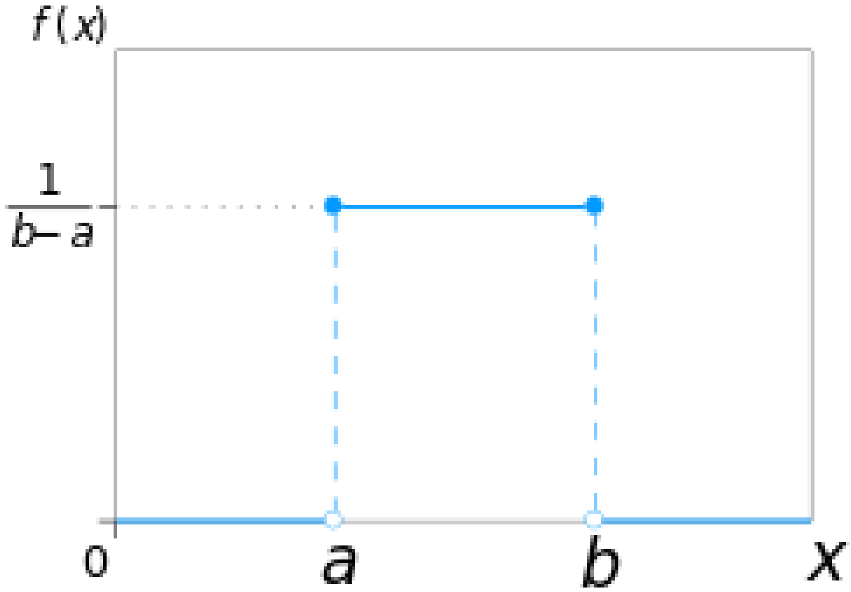
Noise

- Noise의 종류
  - Uniform noise
  - Gaussian noise
  - Salt&Pepper noise

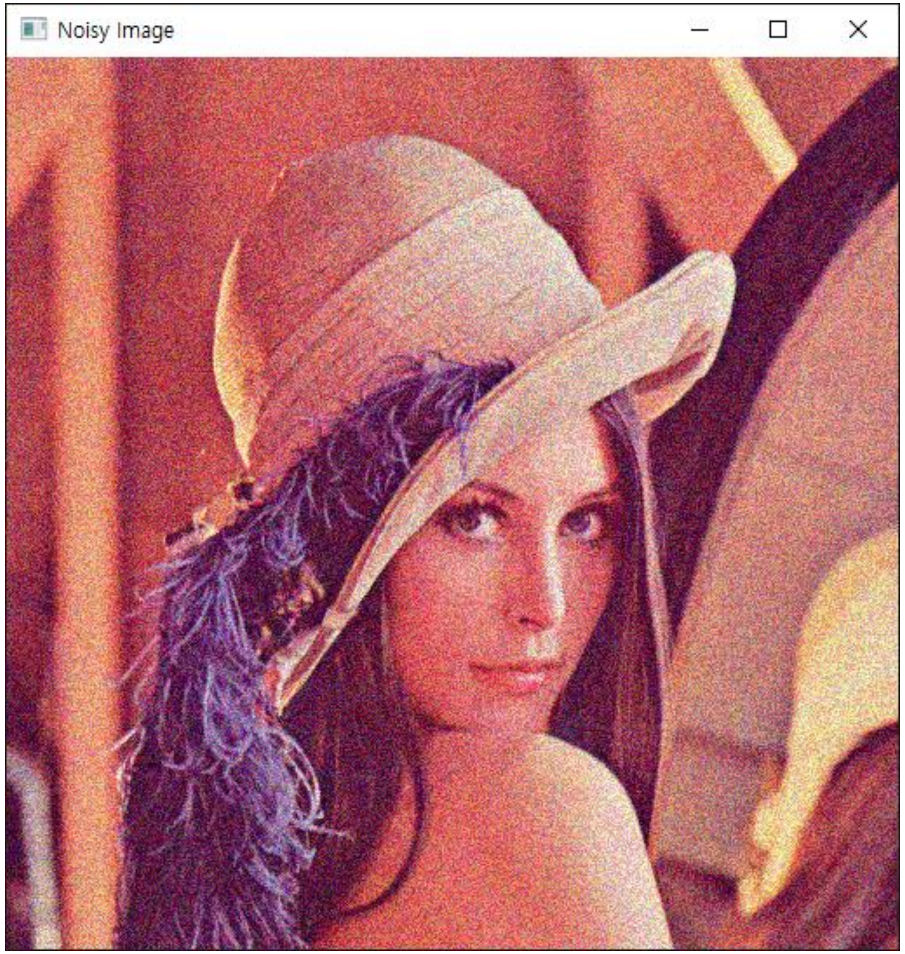
참고 : https://everyday-image-processing.tistory.com/168

Noise

- Uniform noise
  - Uniform distribution(균일 분포)을 따르는 noise



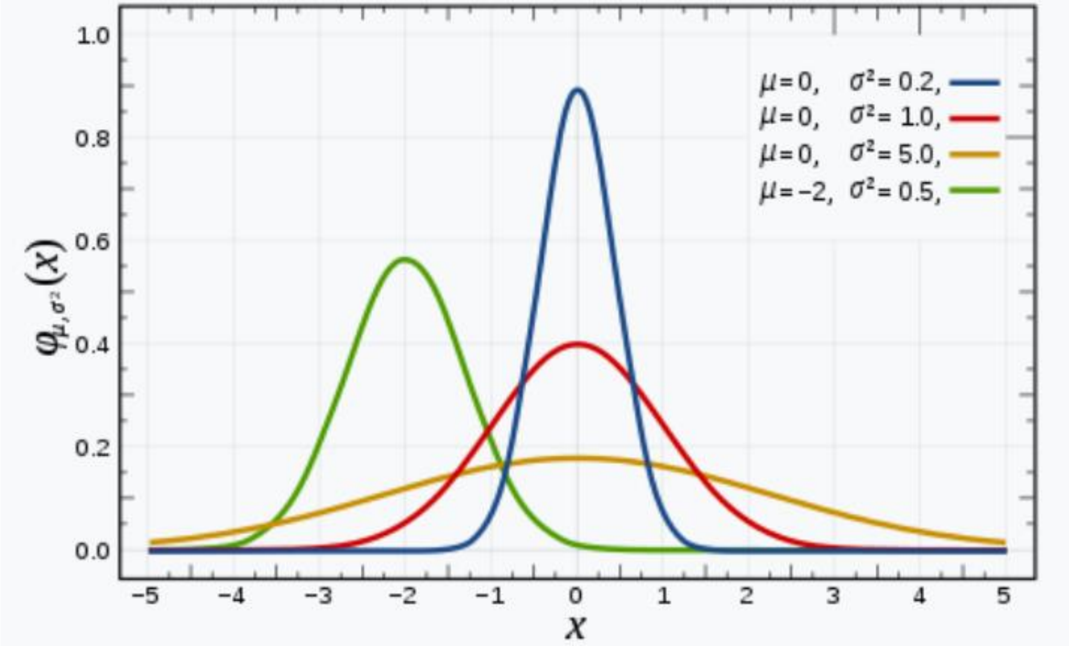
Uniform distribution



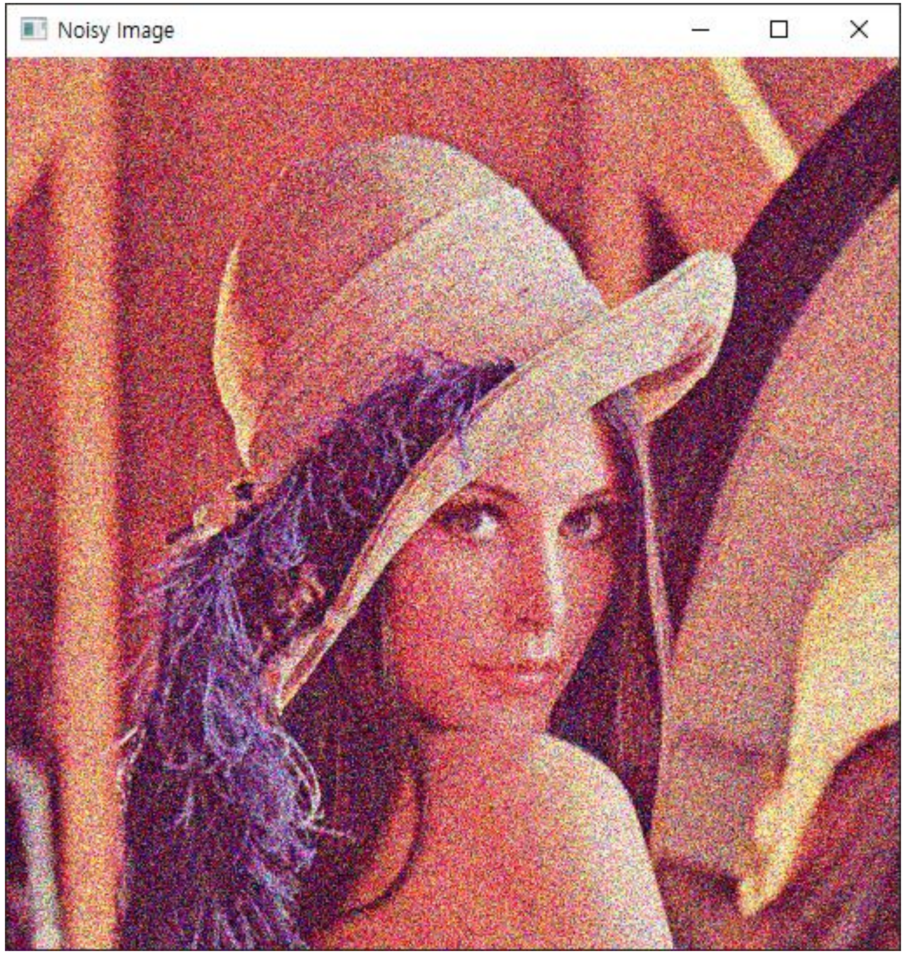
출처 : https://en.wikipedia.org/wiki/Continuous_uniform_distribution

Noise

- Gaussian noise
  - Gaussian distribution을 따르는 noise



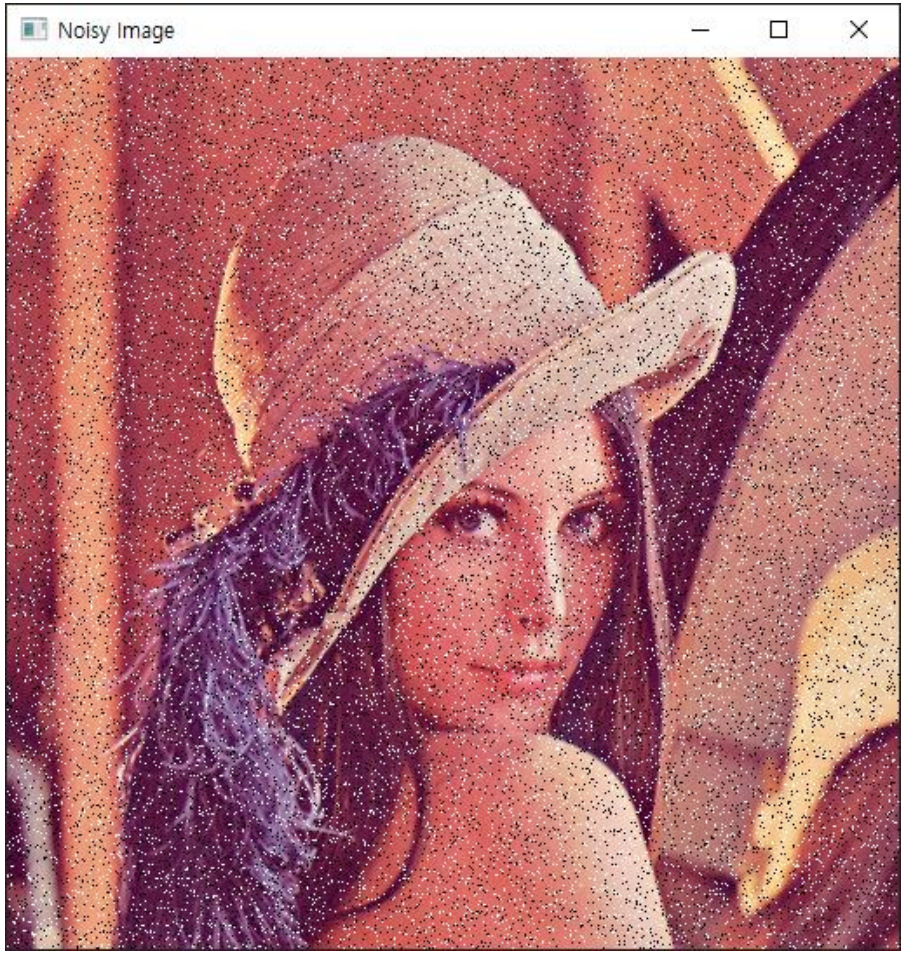
Gaussian distribution



출처 : https://en.wikipedia.org/wiki/Normal_distribution

Noise

- Salt&Pepper noise
  - 무작위로 생긴 흰색과 검은색 noise

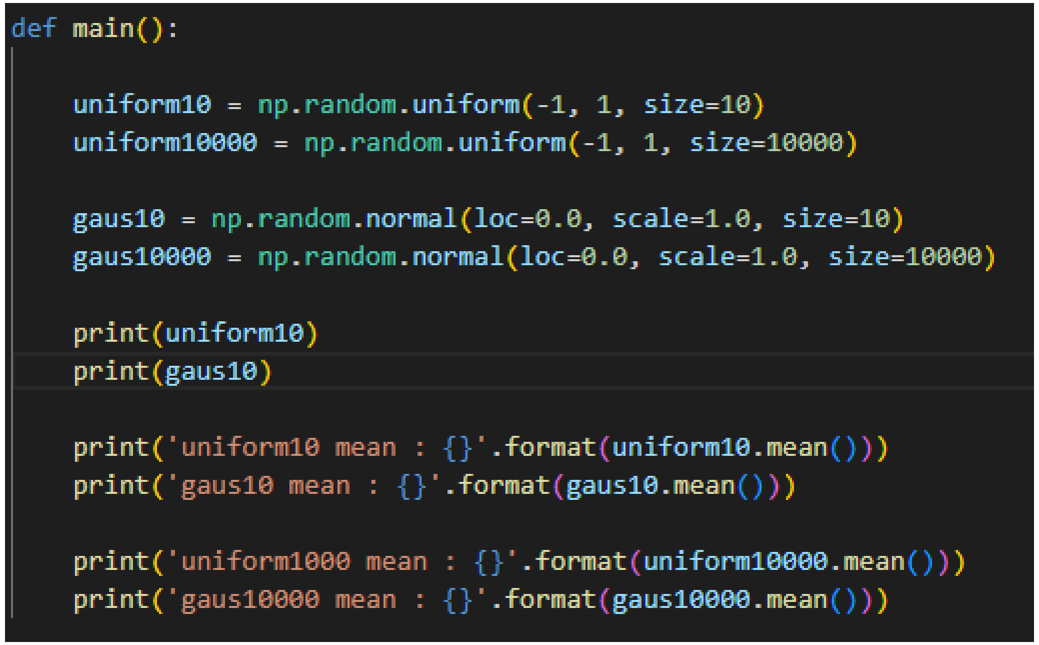


실습 및 과제

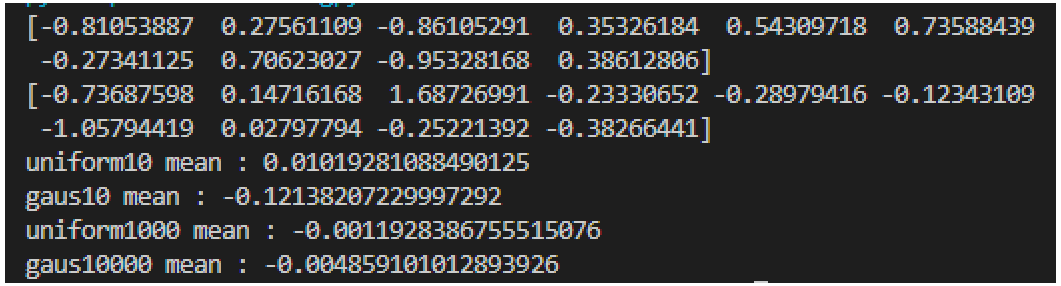
- Github : Hwa-Jong/MyOpenCV: study Opencv (github.com)

실습(IP4_1)

- Uniform vs gaussian noise



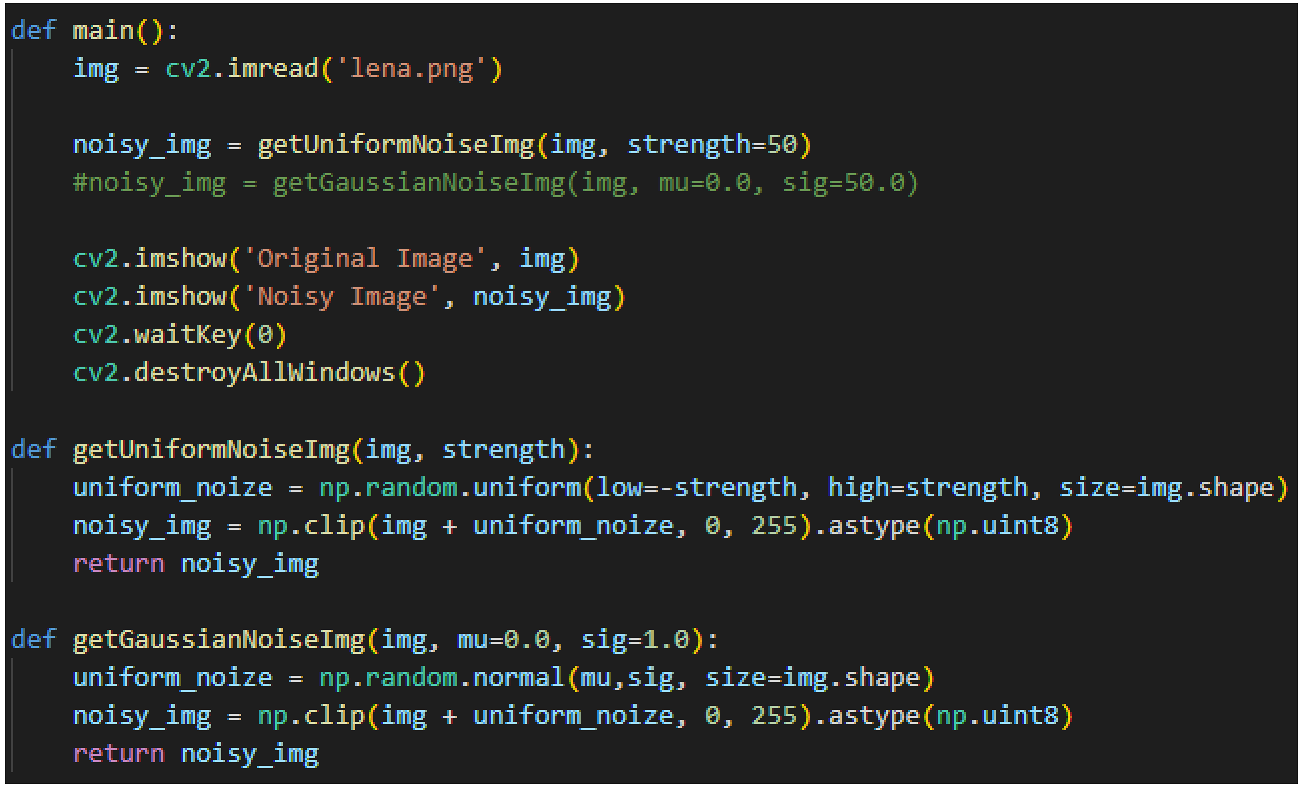
```
def main():

    uniform10 = np.random.uniform(-1, 1, size=10)
    uniform10000 = np.random.uniform(-1, 1, size=10000)

    gaus10 = np.random.normal(loc=0.0, scale=1.0, size=10)
    gaus10000 = np.random.normal(loc=0.0, scale=1.0, size=10000)

    print(uniform10)
    print(gaus10)

    print('uniform10 mean : {}'.format(uniform10.mean()))
    print('gaus10 mean : {}'.format(gaus10.mean()))

    print('uniform1000 mean : {}'.format(uniform10000.mean()))
    print('gaus10000 mean : {}'.format(gaus10000.mean()))
```



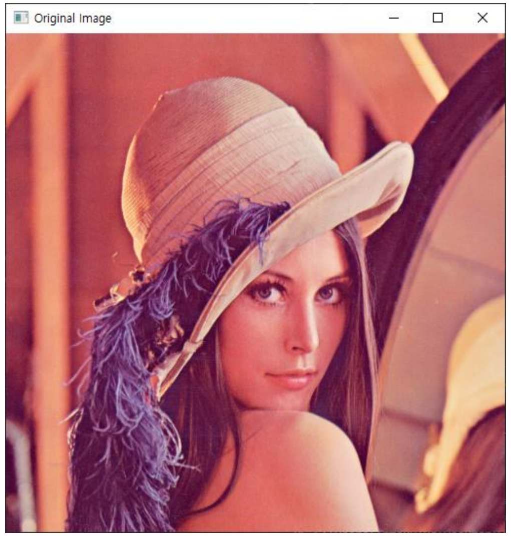
```
[-0.81053887  0.27561109 -0.86105291  0.35326184  0.54309718  0.73588439
 -0.27341125  0.70623027 -0.95328168  0.38612806]
[-0.73687598  0.14716168  1.68726991 -0.23330652 -0.28979416 -0.12343109
 -1.05794419  0.02797794 -0.25221392 -0.38266441]
uniform10 mean : 0.010192810884490125
gaus10 mean : -0.12138207229997292
uniform1000 mean : -0.0011928386755515076
gaus10000 mean : -0.004859101012893926
```

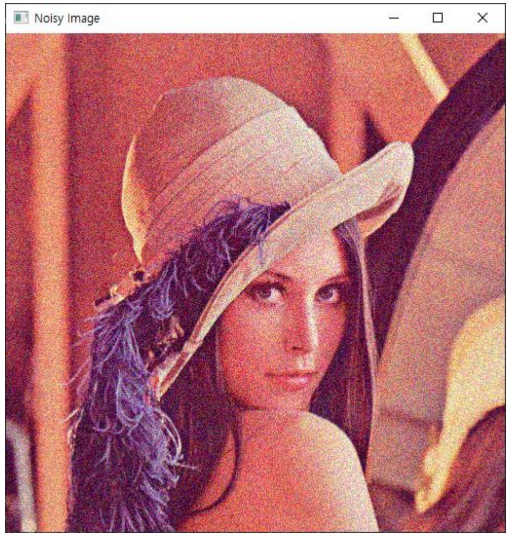
실습(IP4_2)

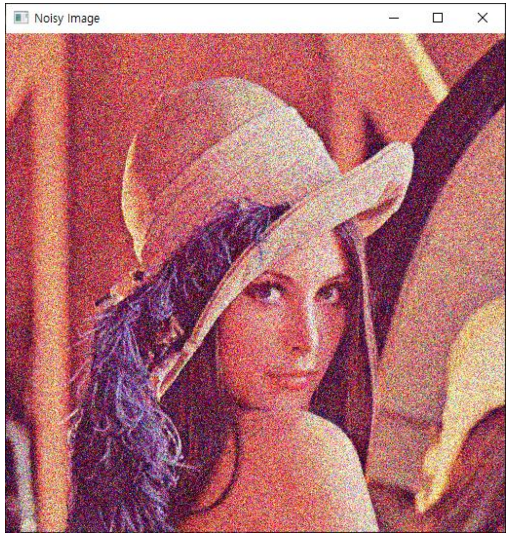
- Uniform vs gaussian noise 적용



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

    noisy_img = getUniformNoiseImg(img, strength=50)
    #noisy_img = getGaussianNoiseImg(img, mu=0.0, sig=50.0)

    cv2.imshow('Original Image', img)
    cv2.imshow('Noisy Image', noisy_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def getUniformNoiseImg(img, strength):
    uniform_noize = np.random.uniform(low=-strength, high=strength, size=img.shape)
    noisy_img = np.clip(img + uniform_noize, 0, 255).astype(np.uint8)
    return noisy_img

def getGaussianNoiseImg(img, mu=0.0, sig=1.0):
    uniform_noize = np.random.normal(mu,sig, size=img.shape)
    noisy_img = np.clip(img + uniform_noize, 0, 255).astype(np.uint8)
    return noisy_img
```

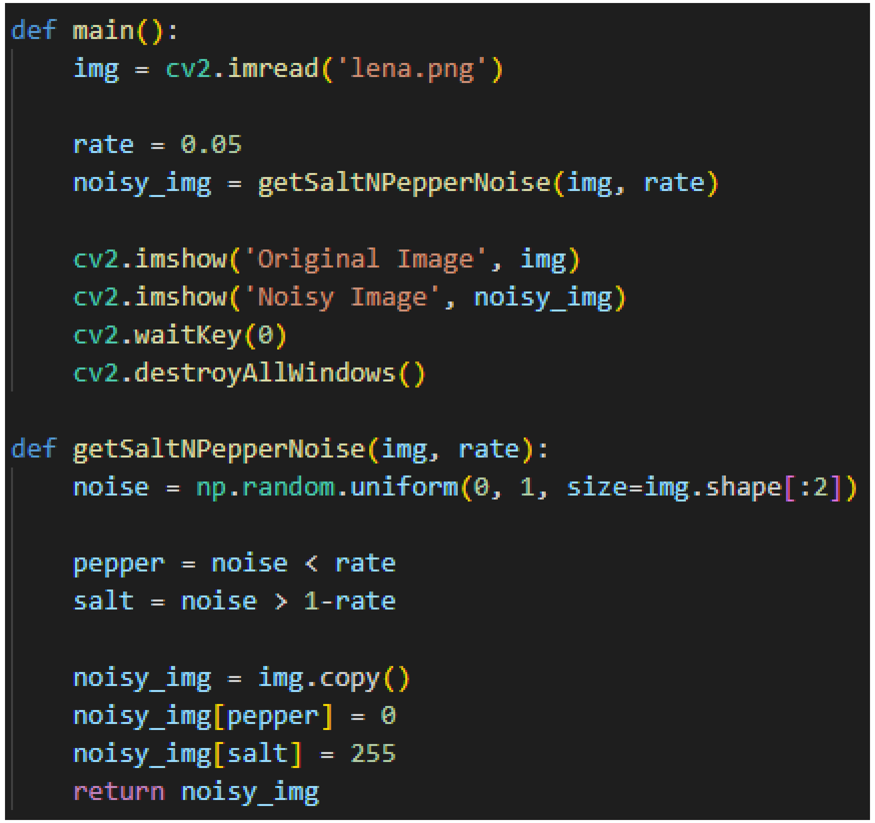




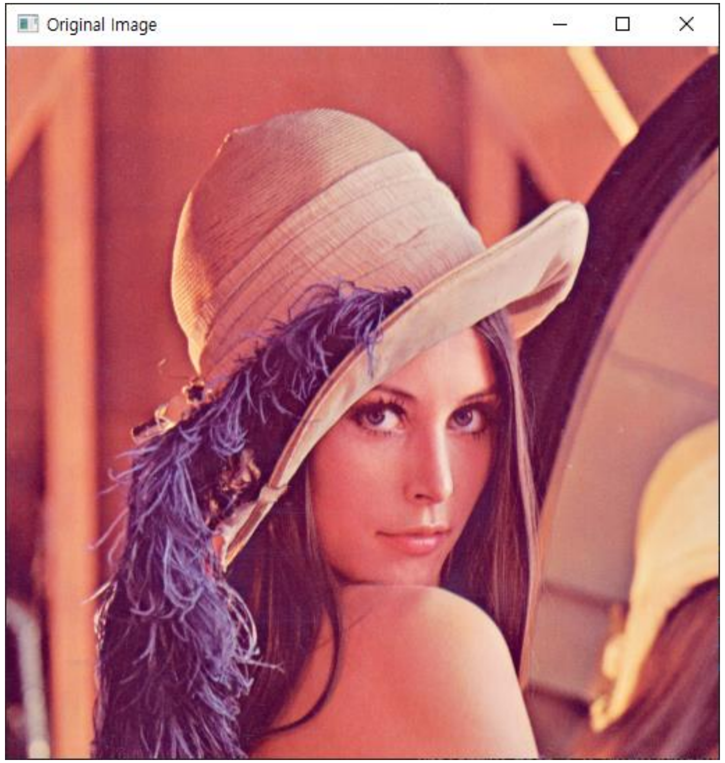


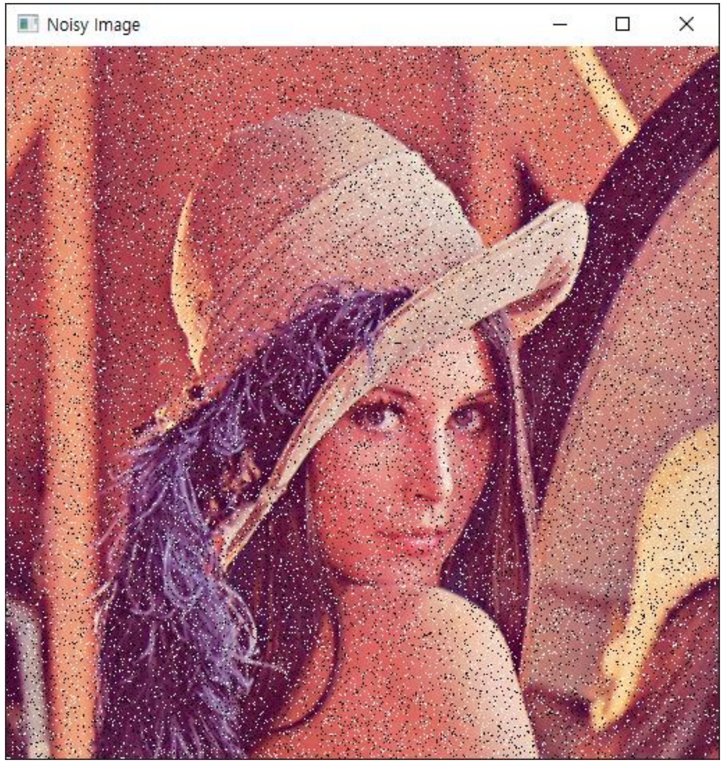
실습(IP4_3)

- Salt & pepper noise 적용



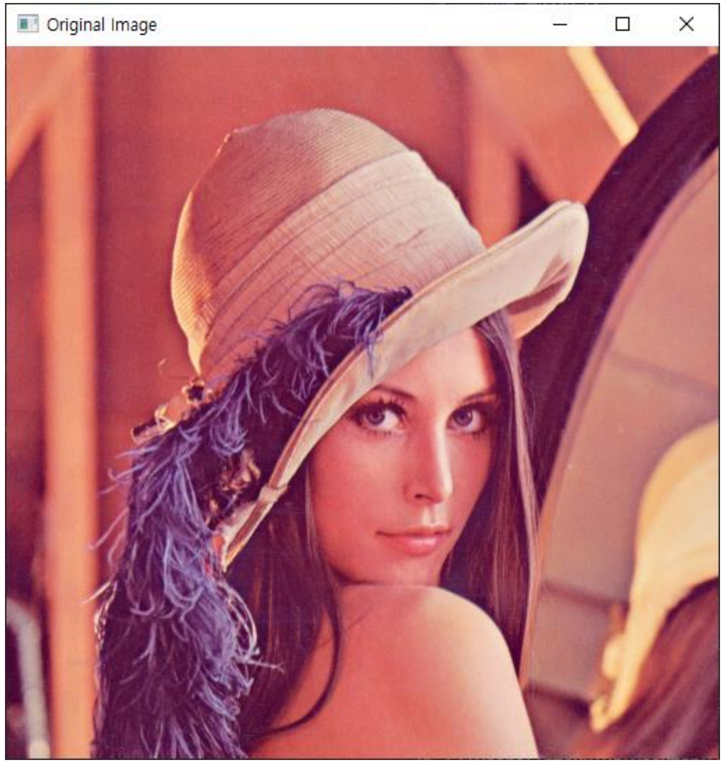
```
def main():
    img = cv2.imread('lena.png')

    rate = 0.05
    noisy_img = getSaltNPepperNoise(img, rate)

    cv2.imshow('Original Image', img)
    cv2.imshow('Noisy Image', noisy_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def getSaltNPepperNoise(img, rate):
    noise = np.random.uniform(0, 1, size=img.shape[:2])

    pepper = noise < rate
    salt = noise > 1-rate

    noisy_img = img.copy()
    noisy_img[pepper] = 0
    noisy_img[salt] = 255
    return noisy_img
```

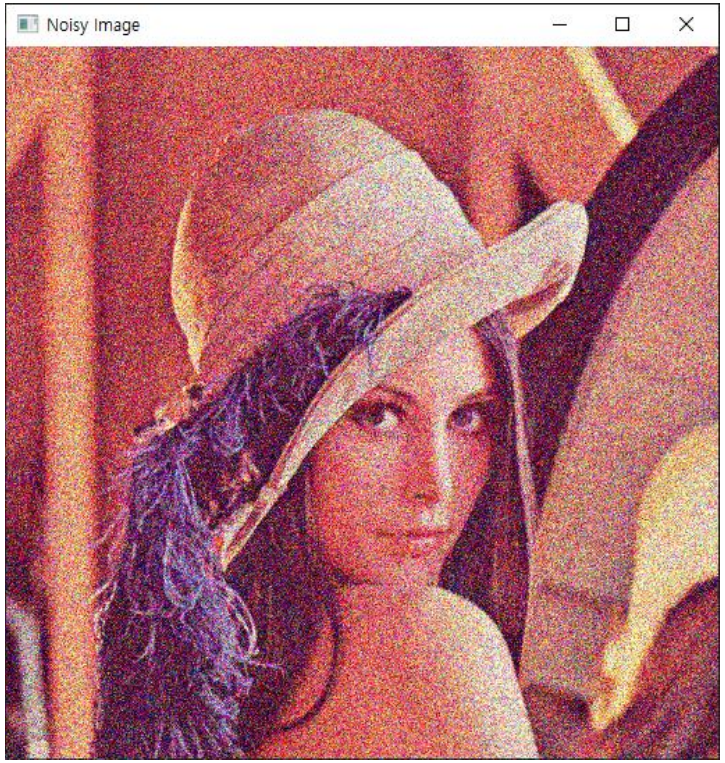


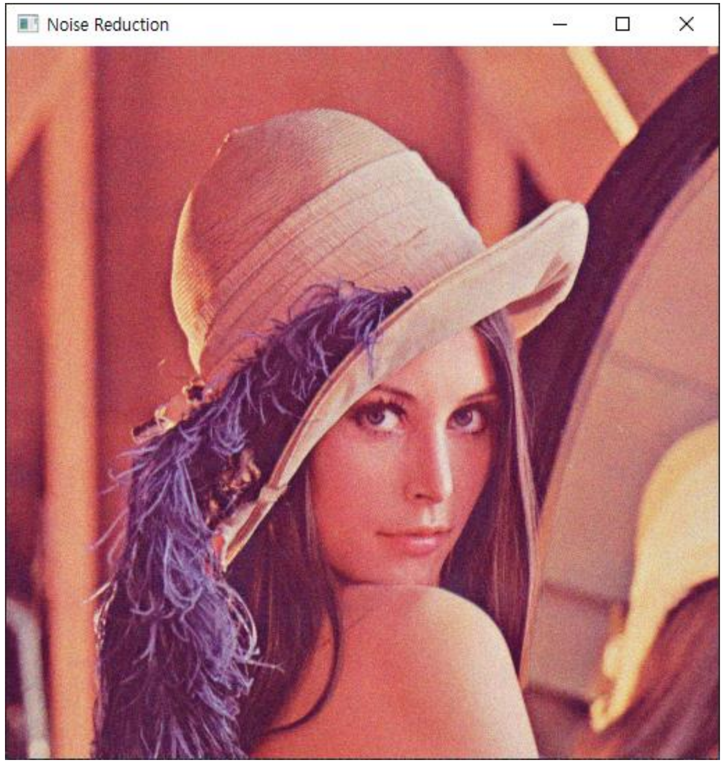


과제(IP4_test1)

- Noise Reduction1
  - 여러 장의 noise 영상이 있는 경우 평균을 사용하여 noise 제거

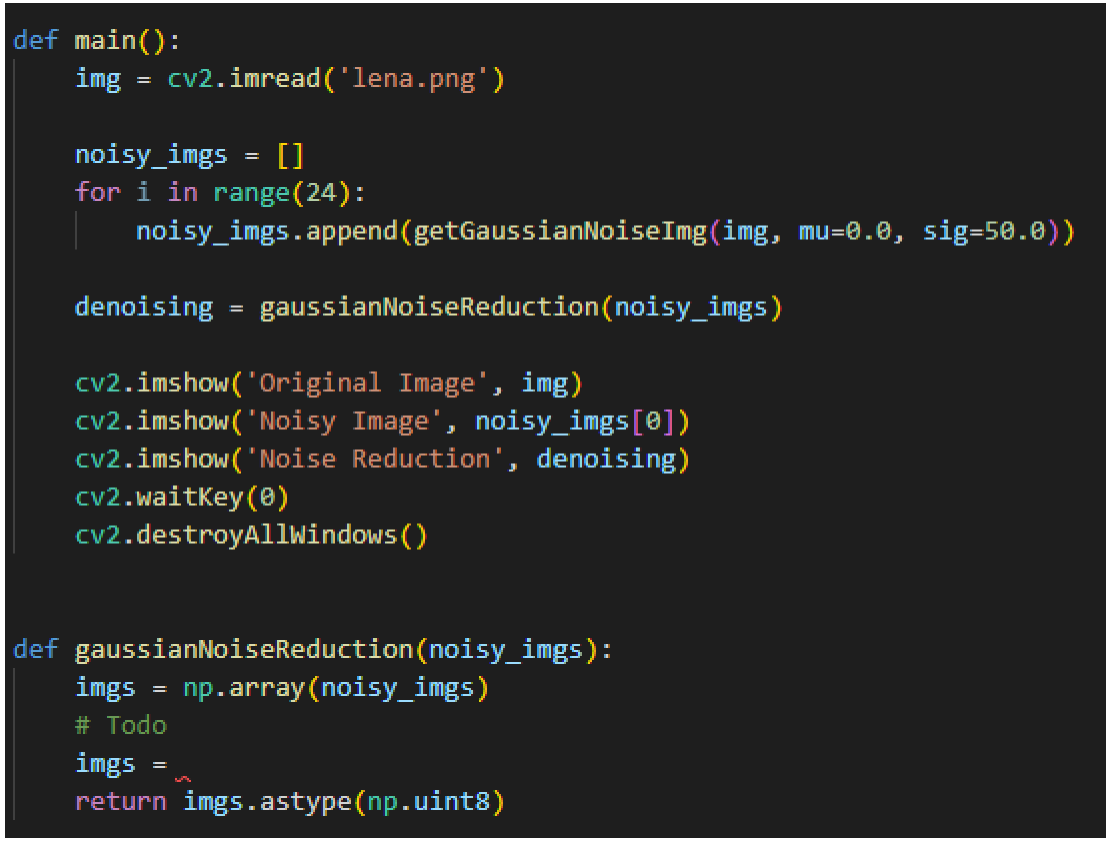






과제(IP4_test1)

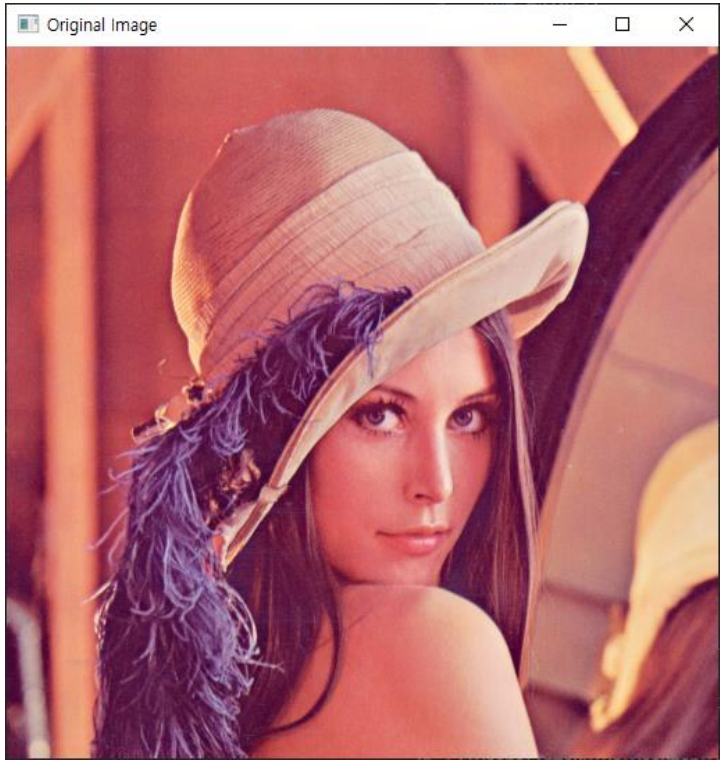
- Noise Reduction1
  - 여러 장의 noise 영상이 있는 경우 평균을 사용하여 noise 제거

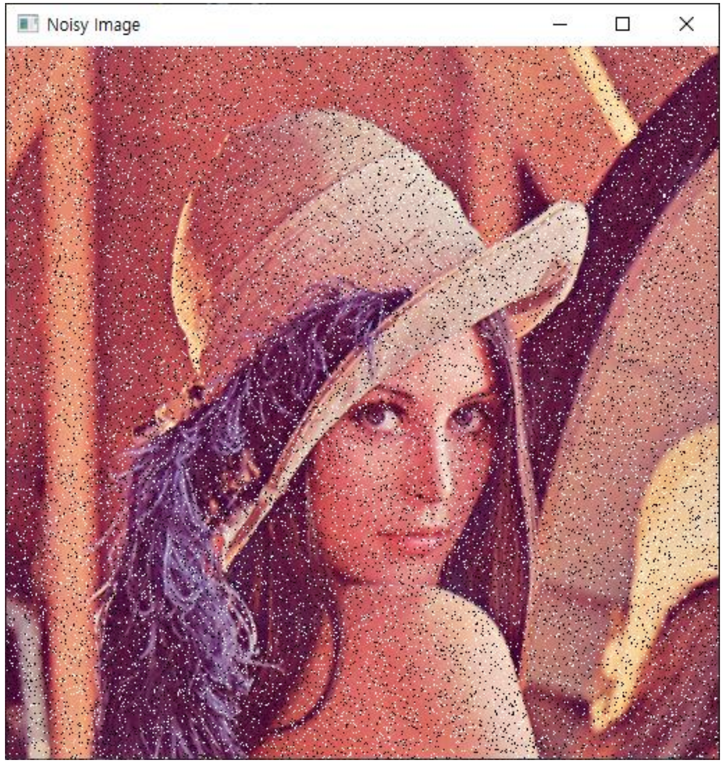


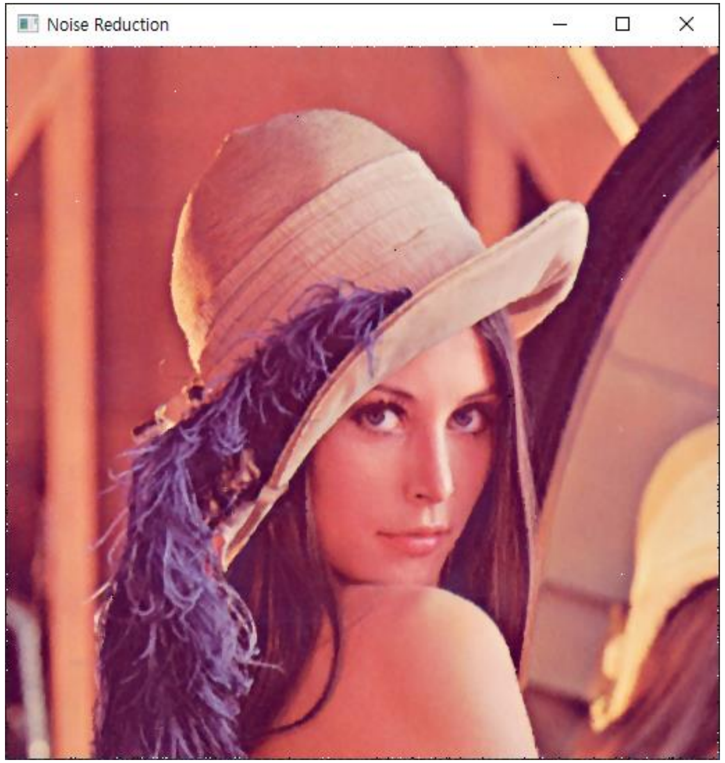
```
def main():
    img = cv2.imread('lena.png')

    noisy_imgs = []
    for i in range(24):
        noisy_imgs.append(getGaussianNoiseImg(img, mu=0.0, sig=50.0))

    denoising = gaussianNoiseReduction(noisy_imgs)

    cv2.imshow('Original Image', img)
    cv2.imshow('Noisy Image', noisy_imgs[0])
    cv2.imshow('Noise Reduction', denoising)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


def gaussianNoiseReduction(noisy_imgs):
    imgs = np.array(noisy_imgs)
    # Todo
    imgs = 
    return imgs.astype(np.uint8)
```

과제(IP4_test2)

- Noise Reduction2
  - Median filter를 활용하여 noise 제거

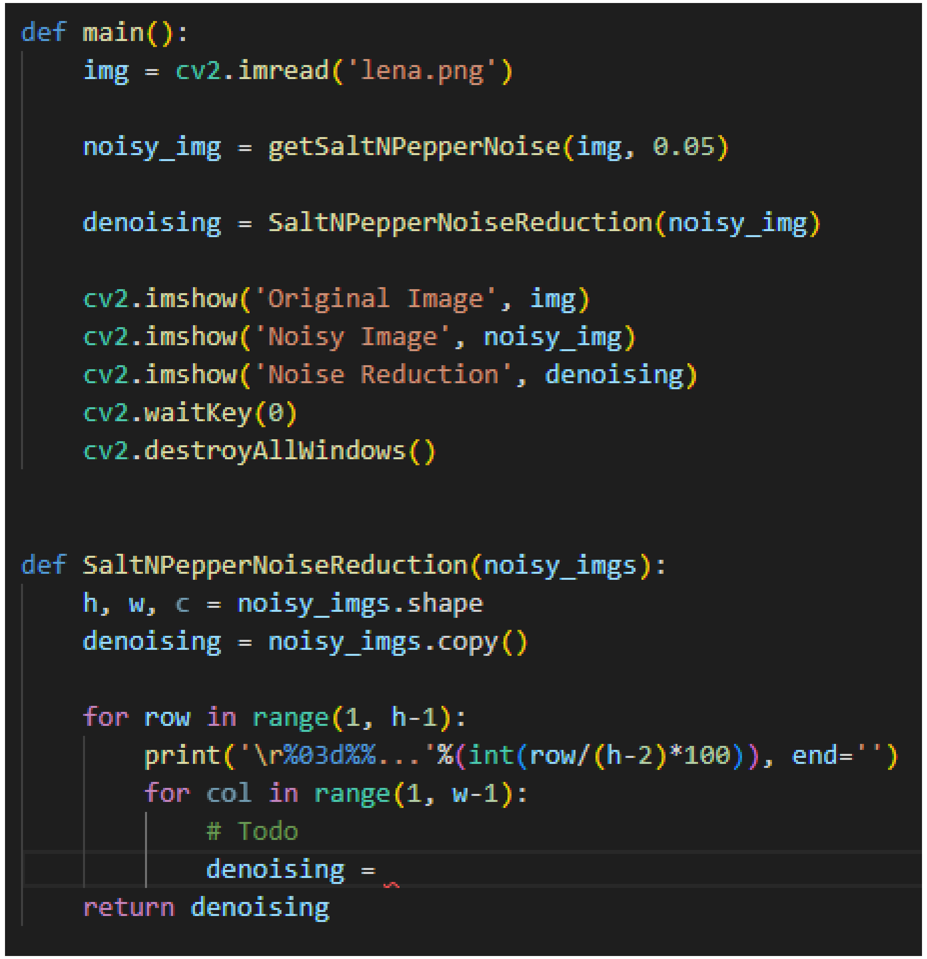






과제(IP4_test2)

- Noise Reduction2
  - Median filter를 활용하여 noise 제거



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

    noisy_img = getSaltNPepperNoise(img, 0.05)

    denoising = SaltNPepperNoiseReduction(noisy_img)

    cv2.imshow('Original Image', img)
    cv2.imshow('Noisy Image', noisy_img)
    cv2.imshow('Noise Reduction', denoising)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


def SaltNPepperNoiseReduction(noisy_imgs):
    h, w, c = noisy_imgs.shape
    denoising = noisy_imgs.copy()

    for row in range(1, h-1):
        print('\r%03d%%...'%(int(row/(h-2)*100)), end='')
        for col in range(1, w-1):
            # Todo
            denoising = 
    return denoising
```

QnA