초급 영상처리

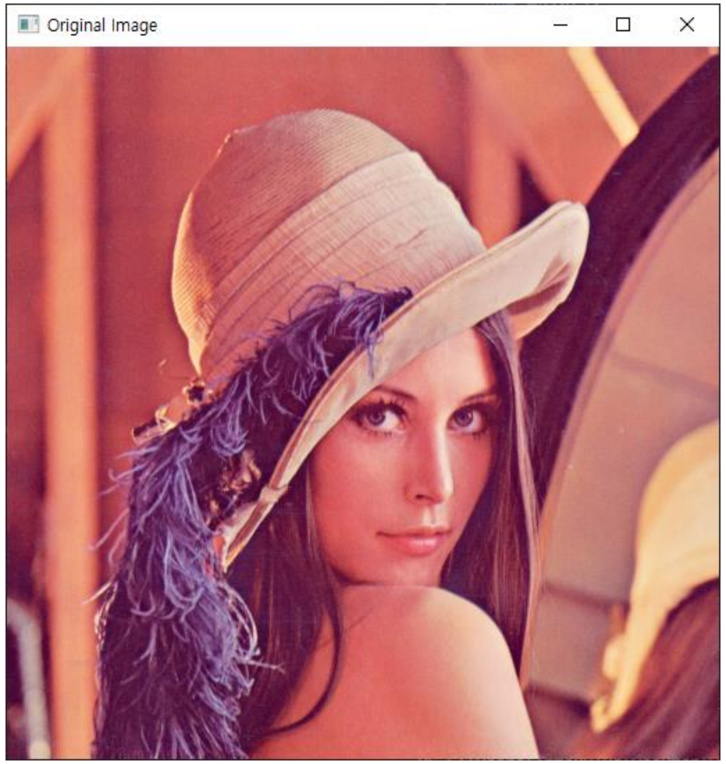
(나만의 Opencv 구현하기)

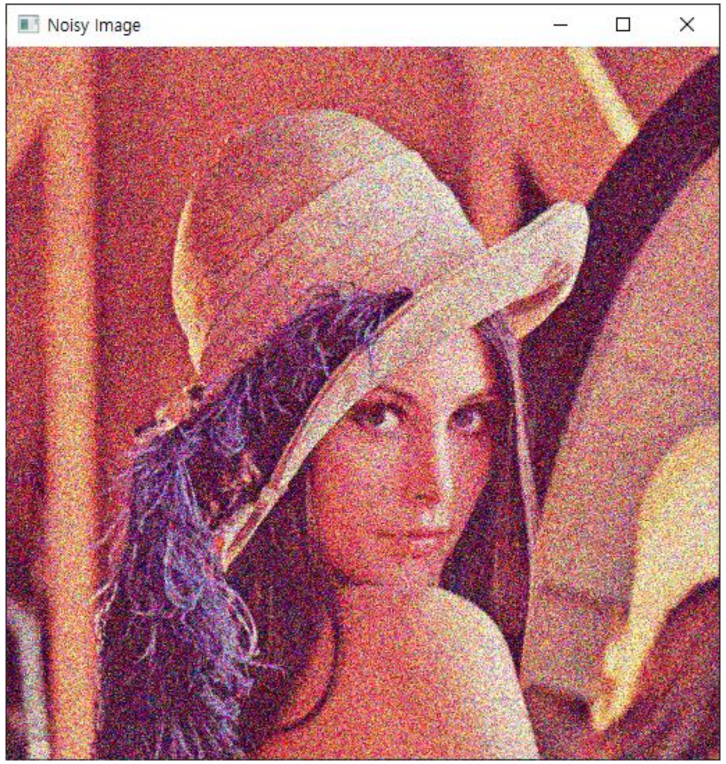
박화종

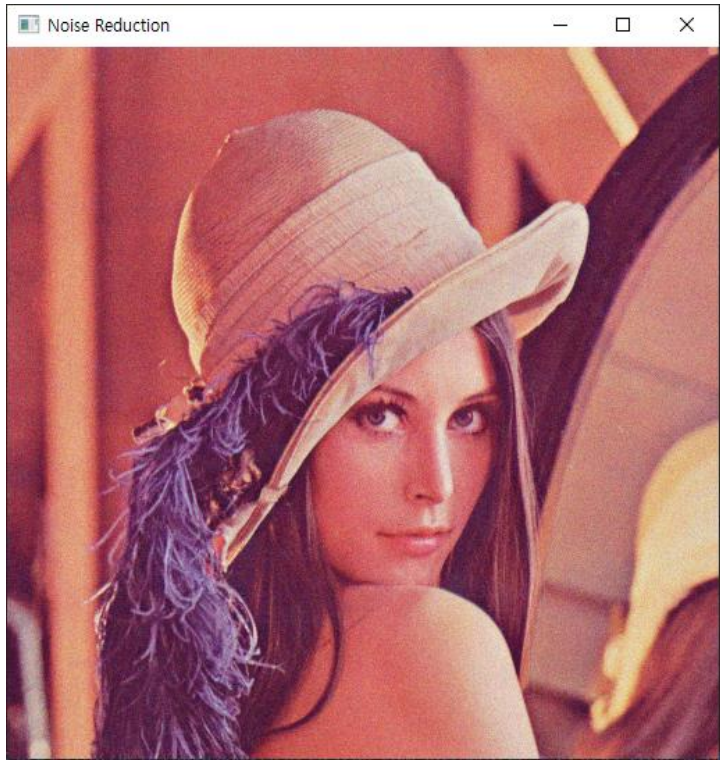
INDEX

저번 주 과제(IP4_test1)

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

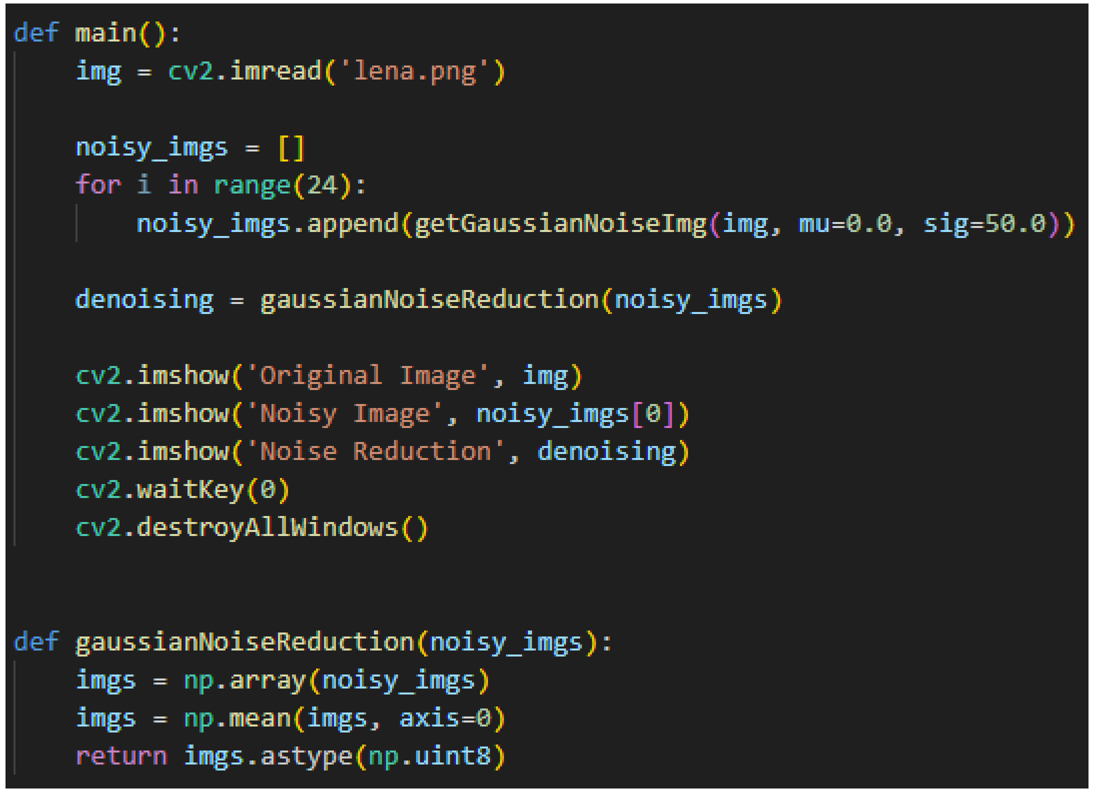






저번 주 과제(IP4_test1)

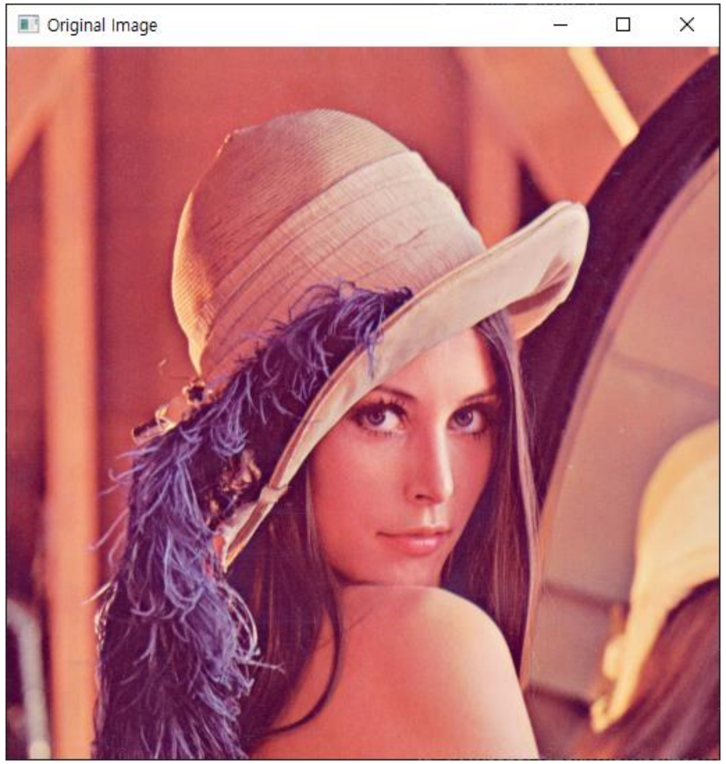
- Noise Reduction1
  - 여러 장의 noise 영상이 있는 경우 평균을 사용하여 noise 제거

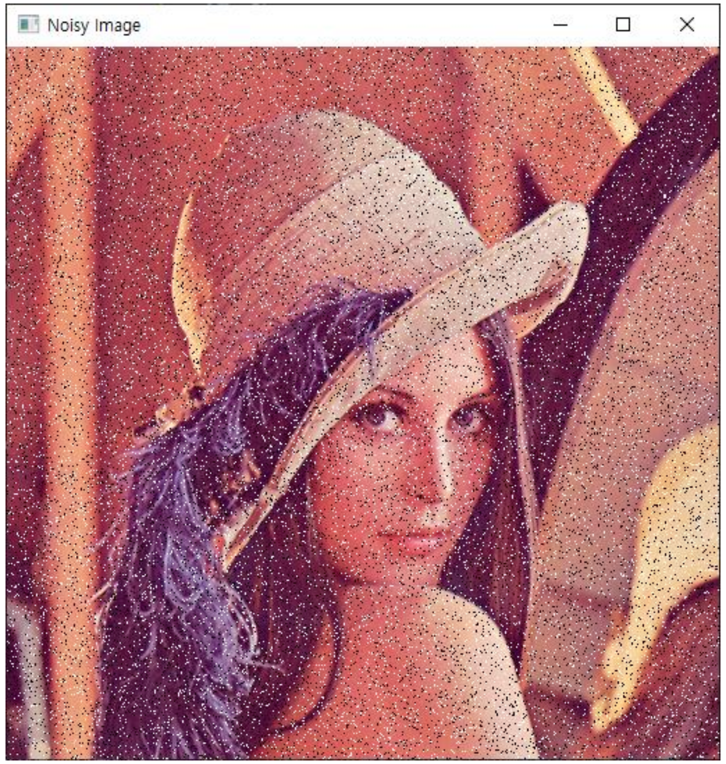


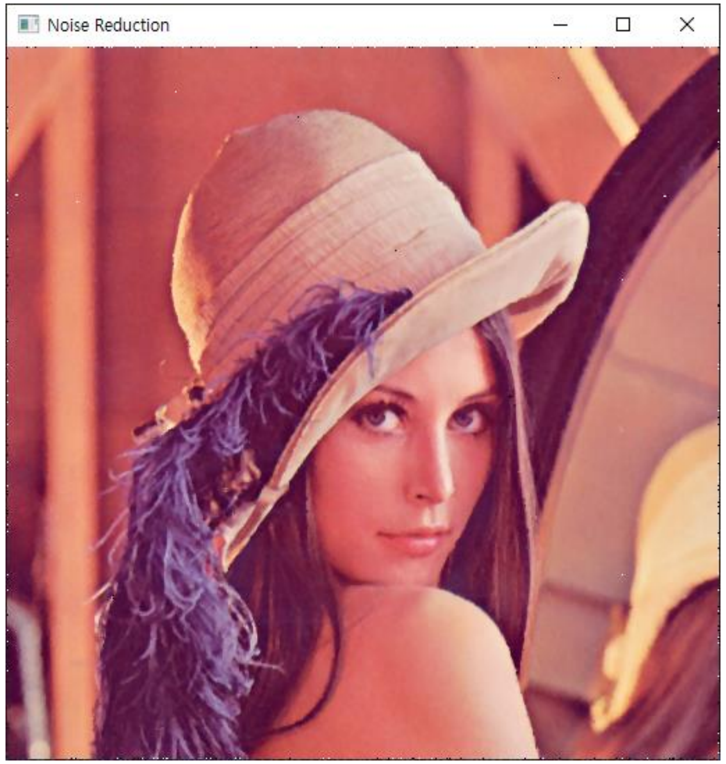
```python
def main():
    img = cv2.imread('lena.png')

    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)
    imgs = np.mean(imgs, axis=0)
    return imgs.astype(np.uint8)
```

저번 주 과제(IP4_test2)

- Noise Reduction2
  - Median filter를 활용하여 noise 제거

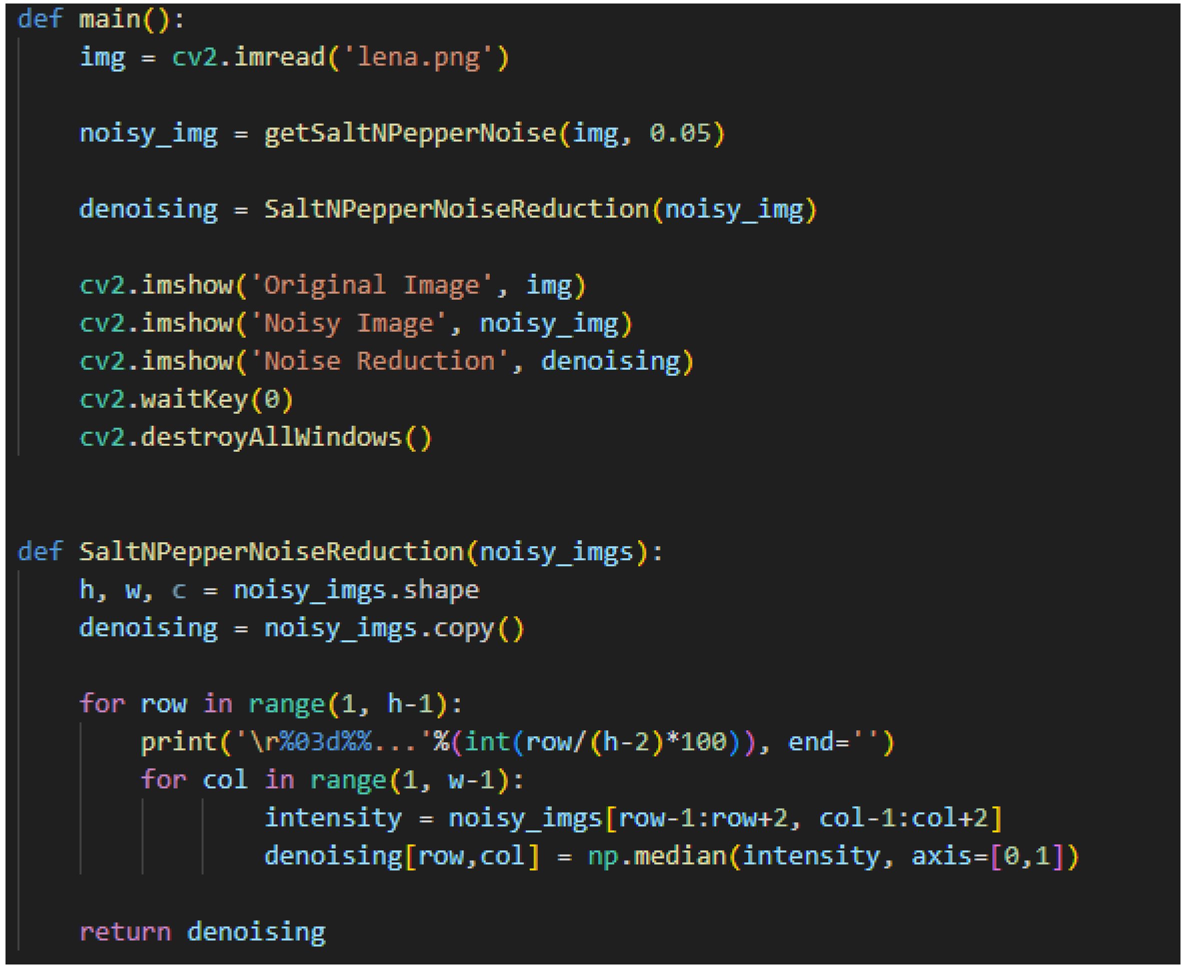






저번 주 과제(IP4_test2)

- Noise Reduction2
  - Median filter를 활용하여 noise 제거



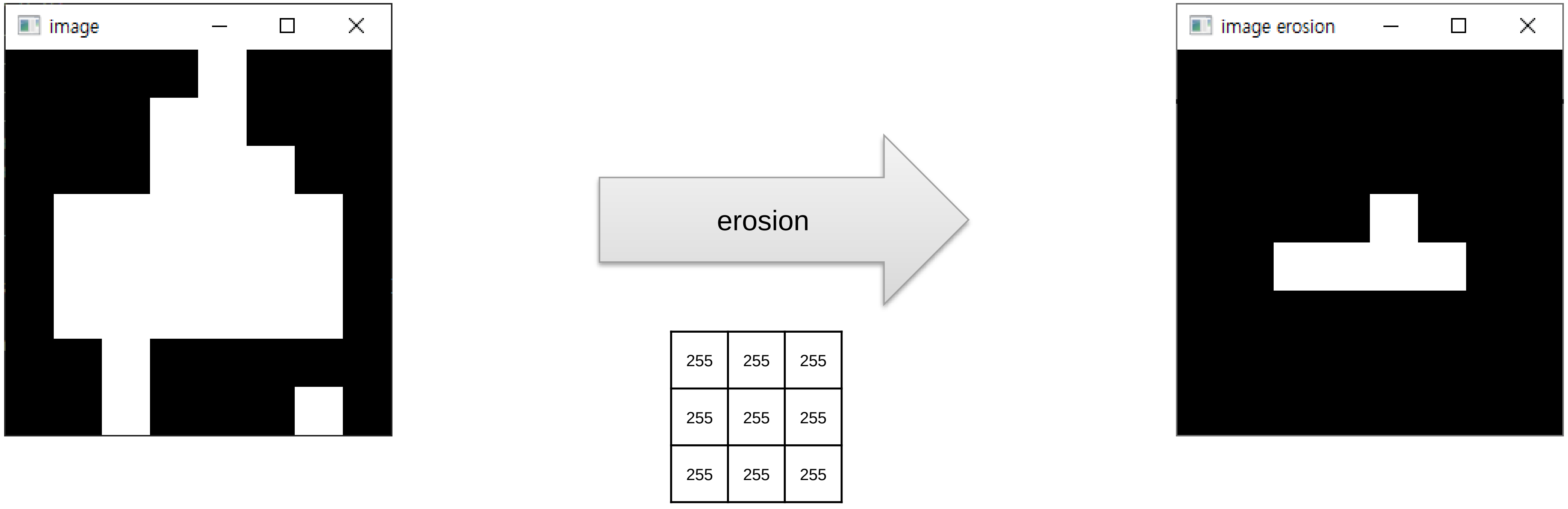
```python
def main():
    img = cv2.imread('lena.png')

    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):
                intensity = noisy_imgs[row-1:row+2, col-1:col+2]
                denoising[row,col] = np.median(intensity, axis=[0,1])

    return denoising
```

Morphology

- Erosion
- Dilation
- Opening
- Closing

Morphology

- Erosion
  뜻 : 부식, 침식



255	255	255
255	255	255
255	255	255

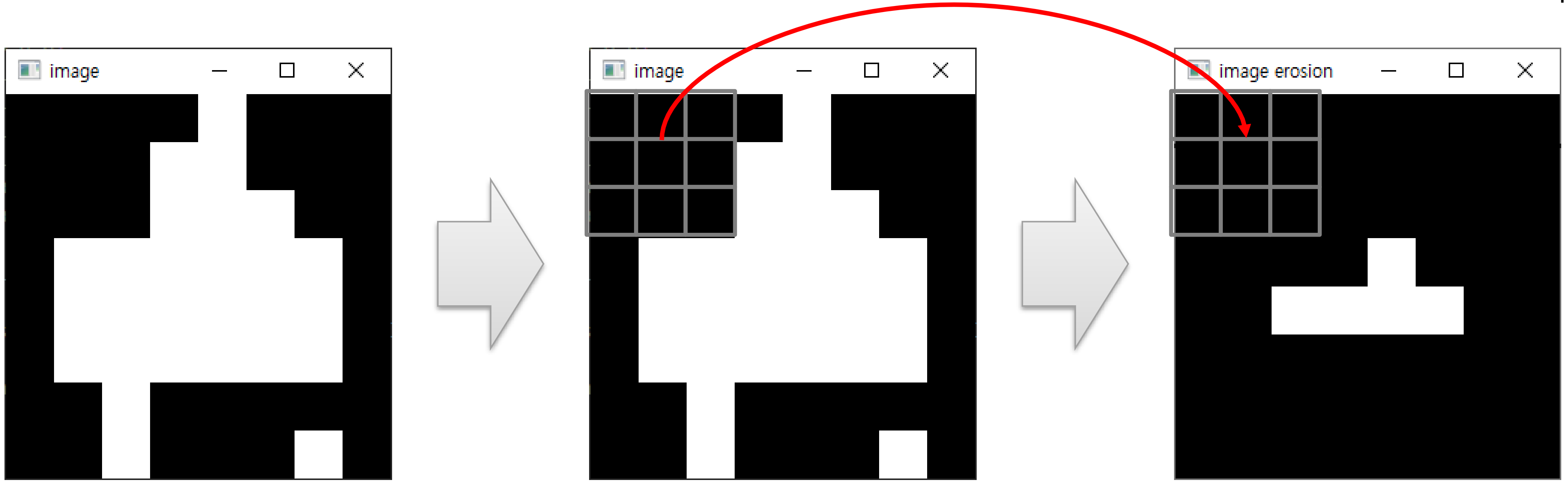
kernel

Morphology

- Erosion
  뜻 : 부식, 침식

255	255	255
255	255	255
255	255	255

kernel

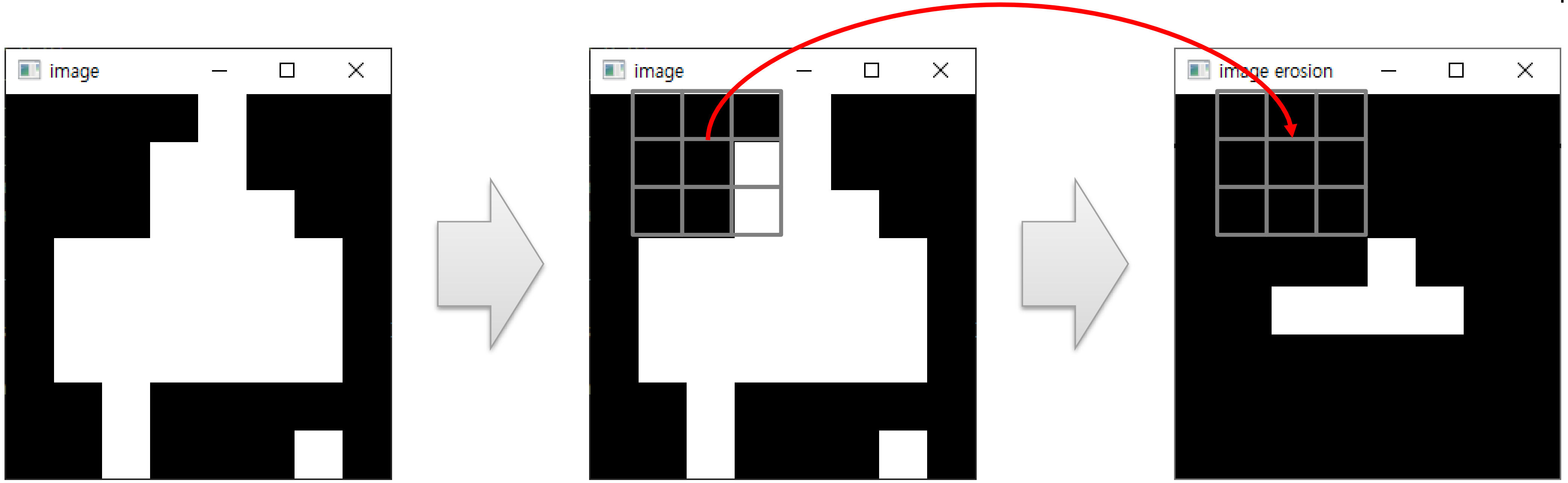


Morphology

- Erosion
  뜻 : 부식, 침식

255	255	255
255	255	255
255	255	255

kernel



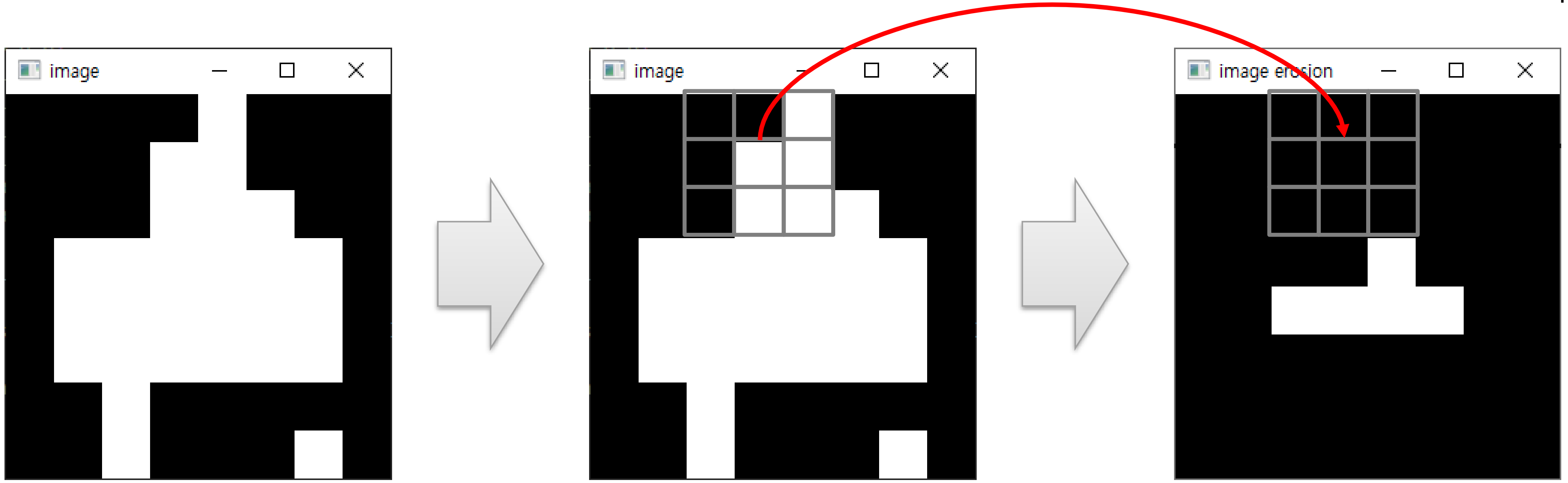
Morphology

- Erosion

  뜻 : 부식, 침식

255	255	255
255	255	255
255	255	255

kernel

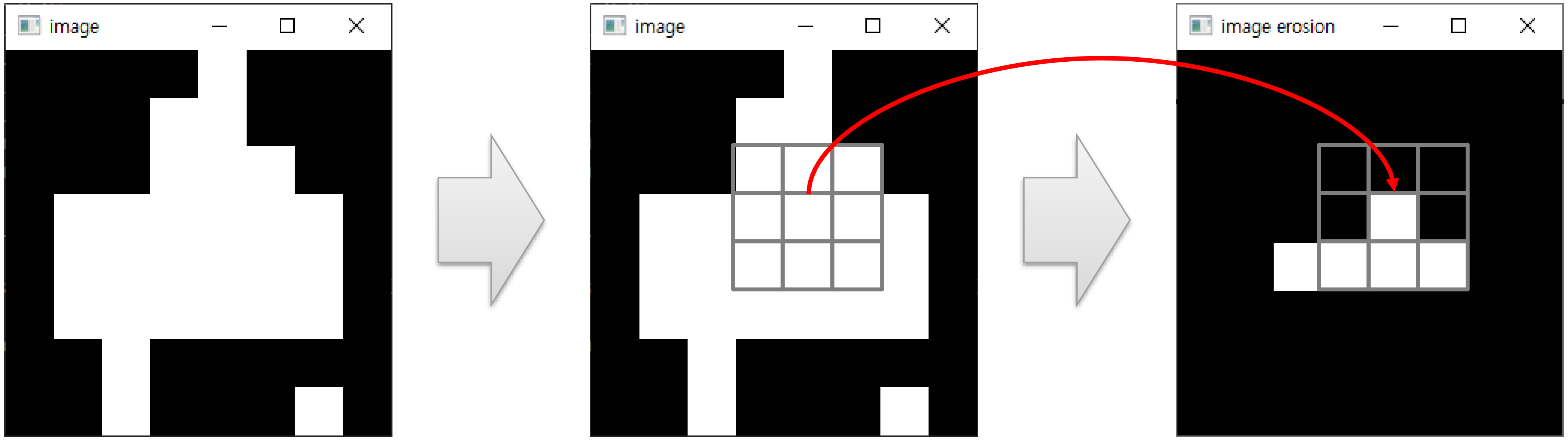


Morphology

- Erosion
  뜻 : 부식, 침식

255	255	255
255	255	255
255	255	255

kernel

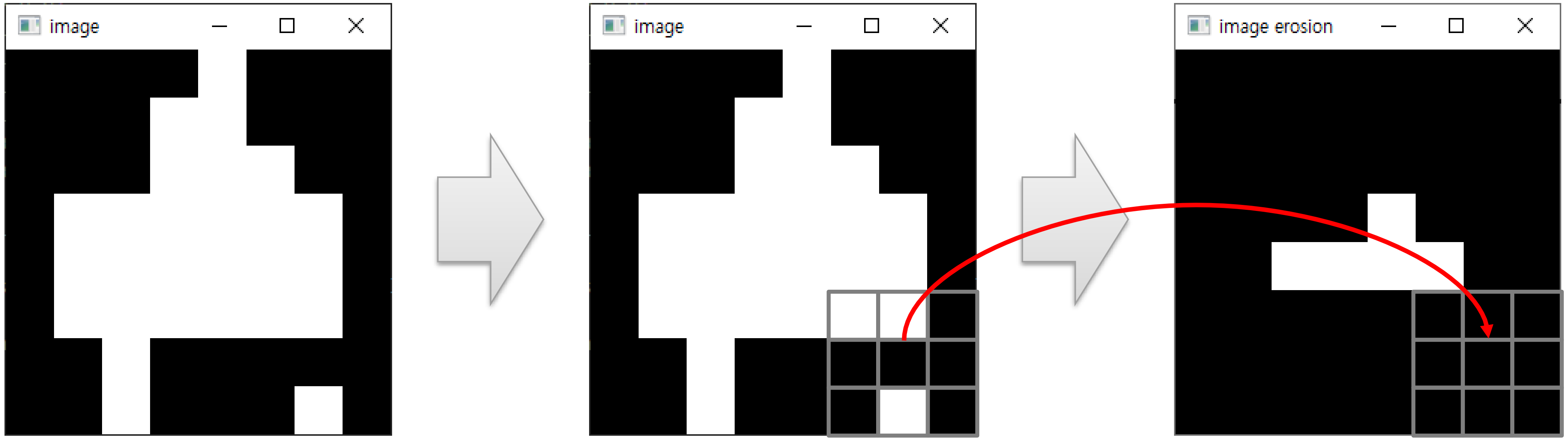


Morphology

- Erosion
  뜻 : 부식, 침식

255	255	255
255	255	255
255	255	255

kernel

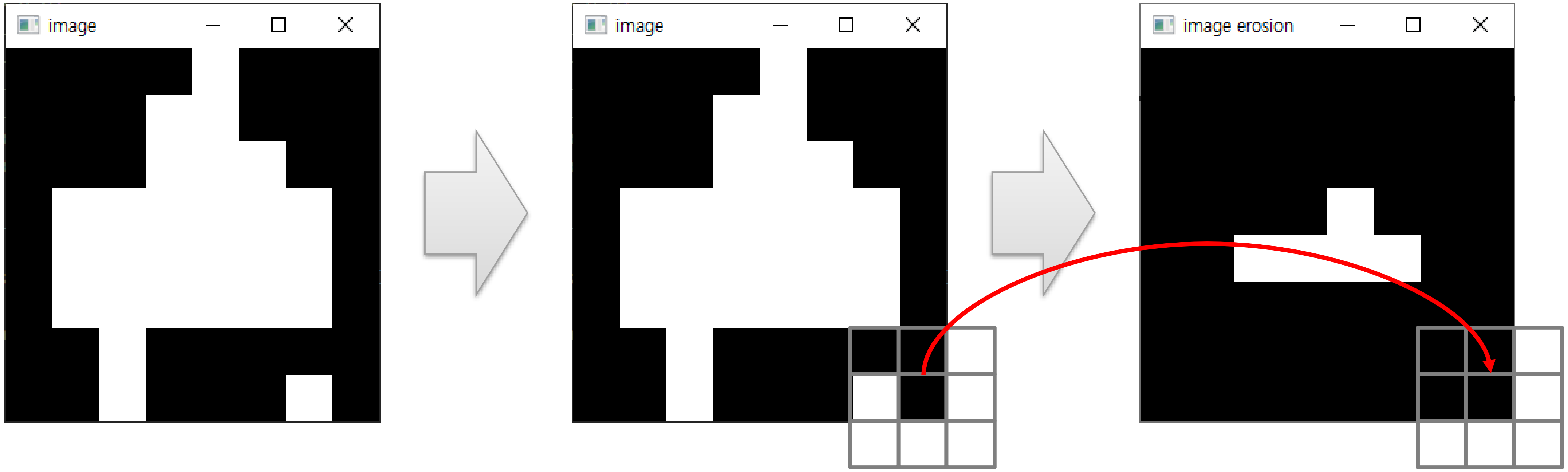


Morphology

- Erosion
  뜻 : 부식, 침식

255	255	255
255	255	255
255	255	255

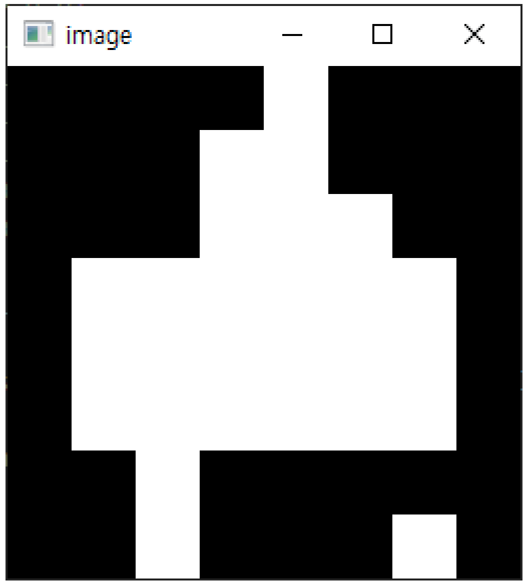
kernel



* 가장자리는 어떻게 할까?
  - (일단 지금은)가장자리는 하지 않는다

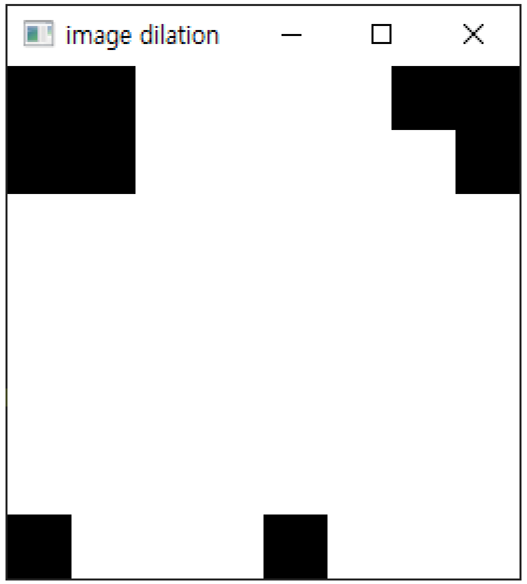
Morphology

- Dilation
  뜻 : 팽창



255	255	255
255	255	255
255	255	255

kernel

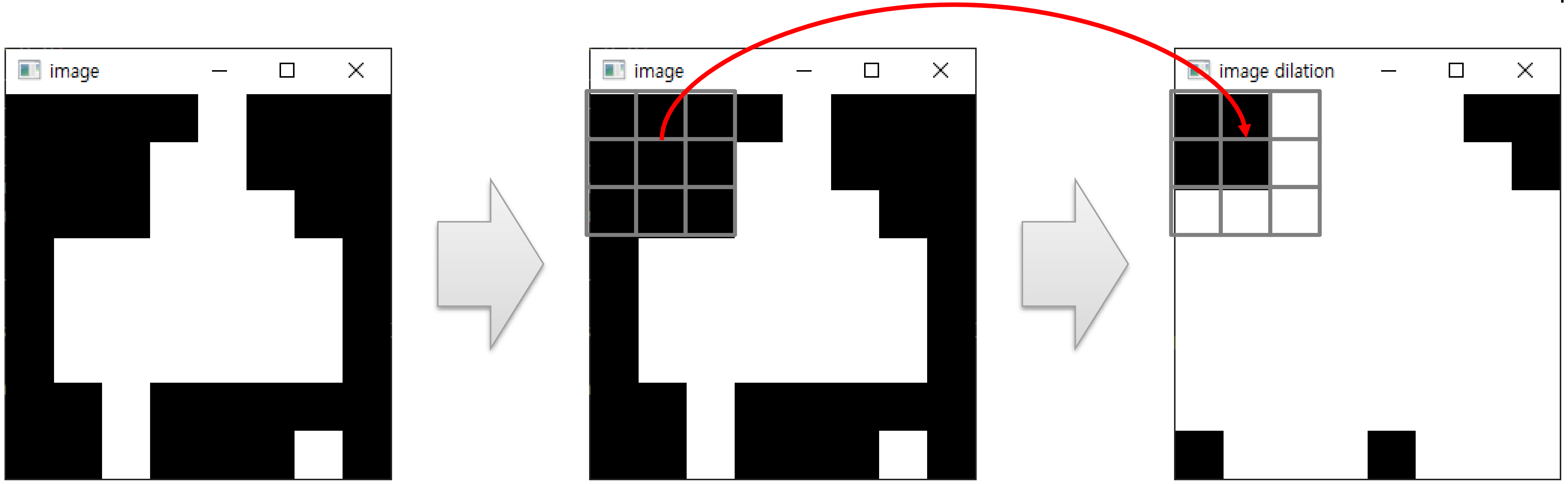


Morphology

- Dilation
  뜻 : 팽창

255	255	255
255	255	255
255	255	255

kernel

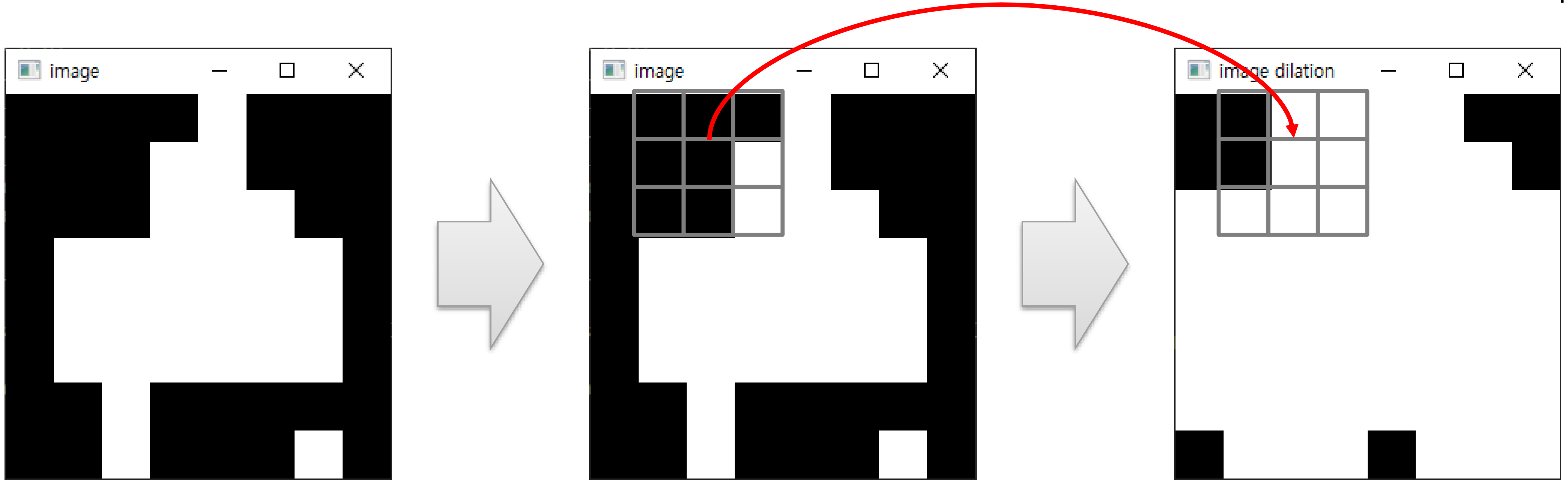


Morphology

- Dilation
  뜻 : 팽창

255	255	255
255	255	255
255	255	255

kernel



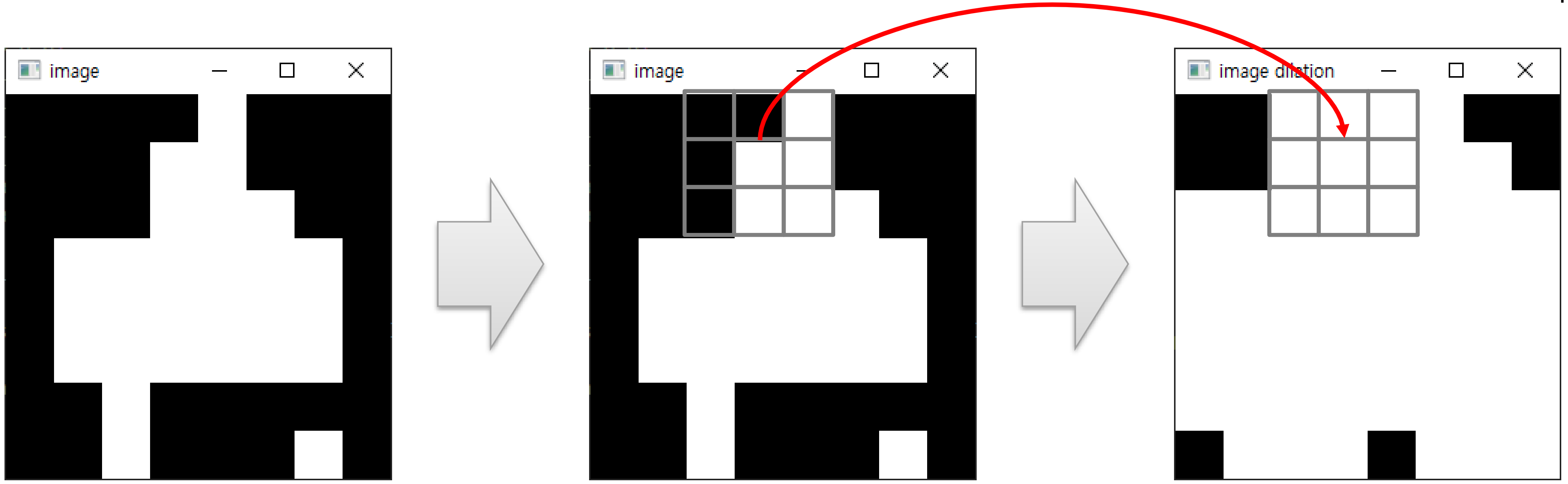
Morphology

- Dilation

  뜻 : 팽창

255	255	255
255	255	255
255	255	255

kernel

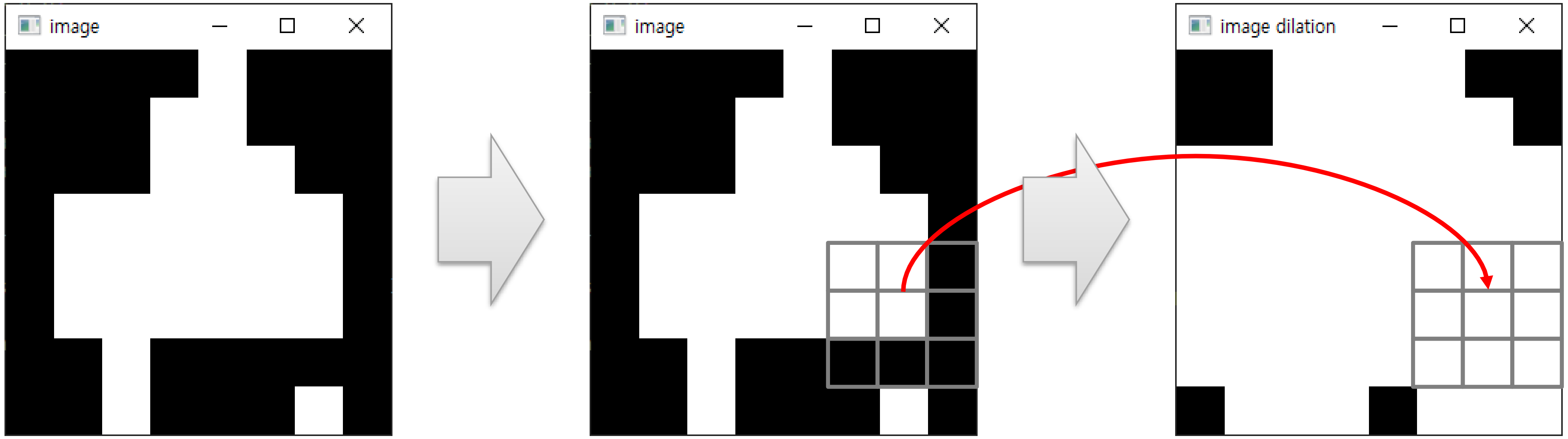


Morphology

- Dilation
  뜻 : 팽창

255	255	255
255	255	255
255	255	255

kernel

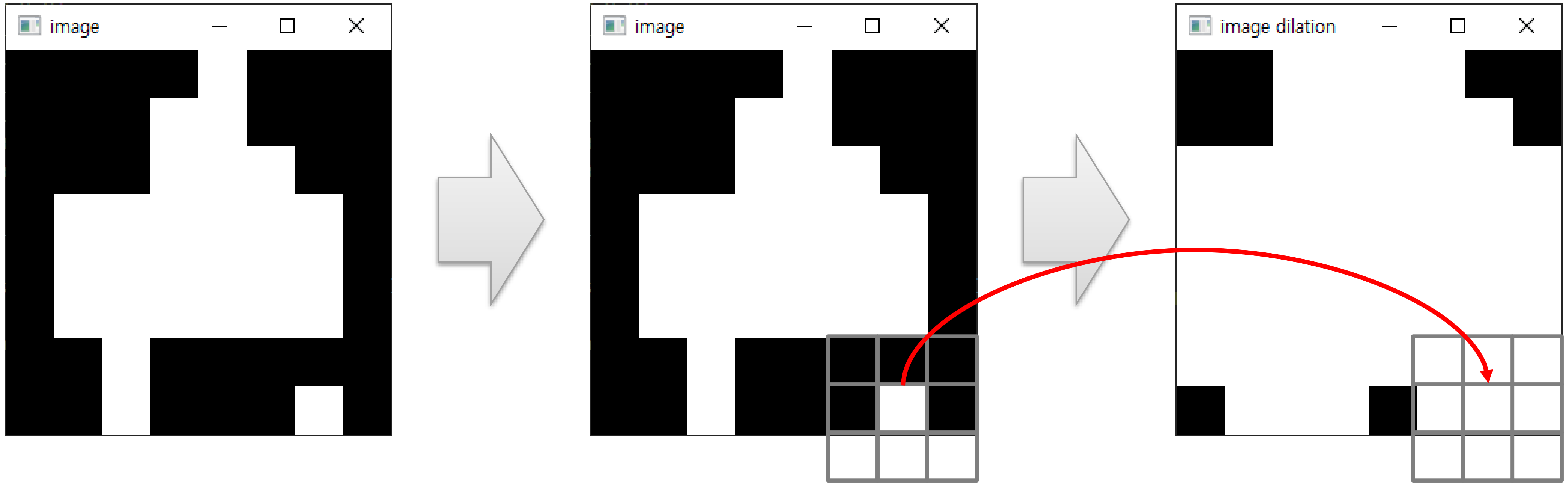


Morphology

- Dilation
  뜻 : 팽창

255	255	255
255	255	255
255	255	255

kernel



* 가장자리는 어떻게 할까?
- (일단 지금은)가장자리는 하지 않는다

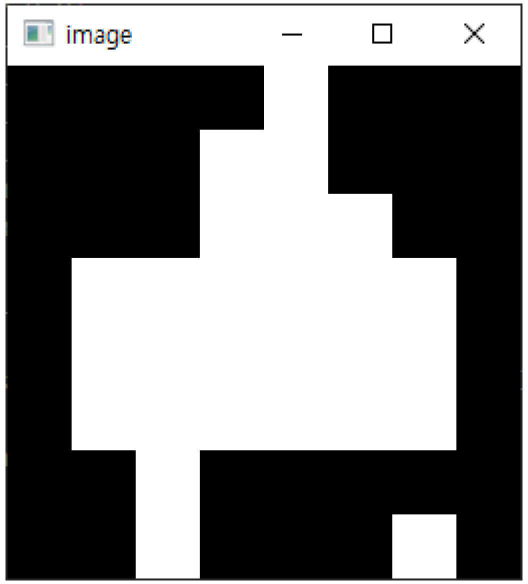
Morphology

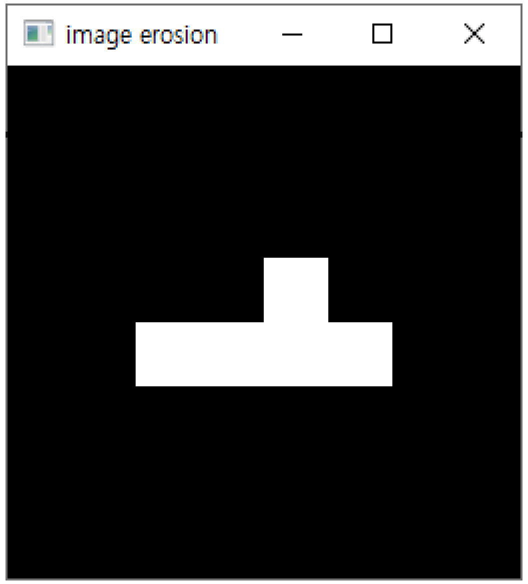
- Opening

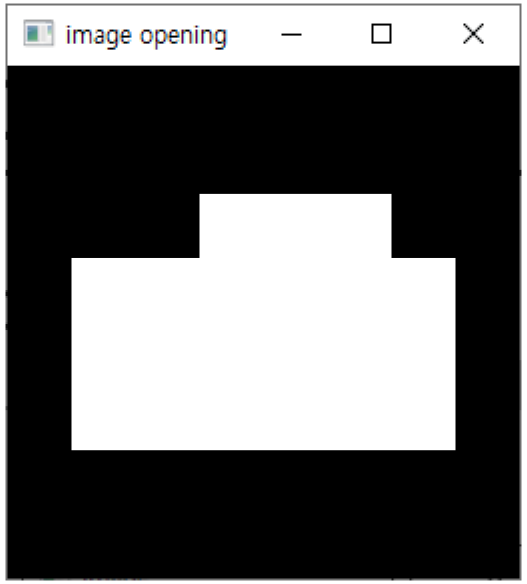
  Erosion 적용 후 Dilation 적용

255	255	255
255	255	255
255	255	255

kernel





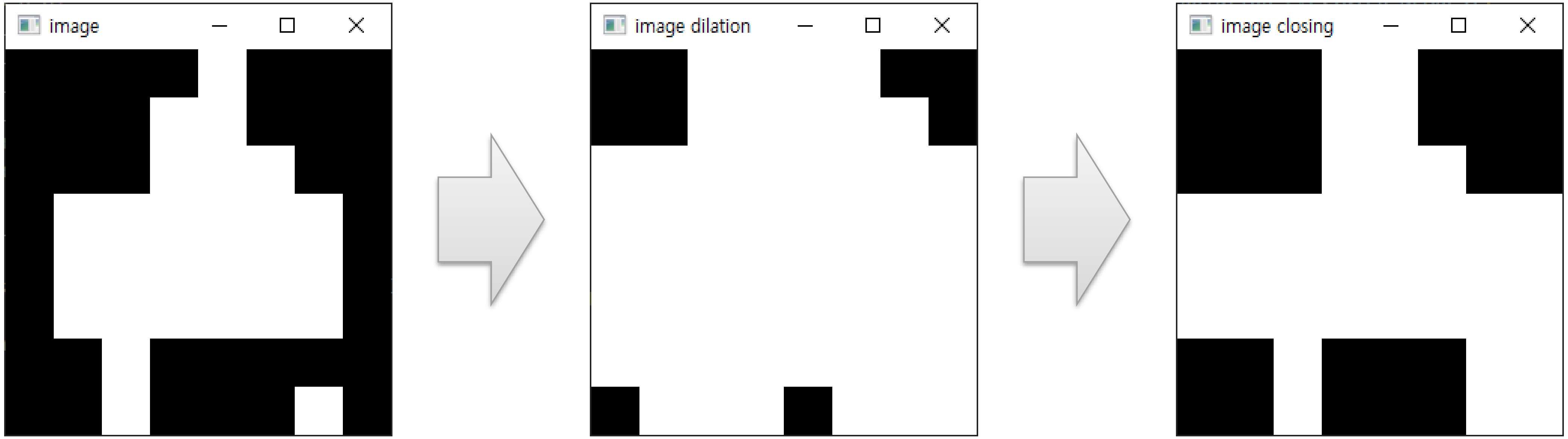


Morphology

- Closing
  Dilation 적용 후 Erosion 적용

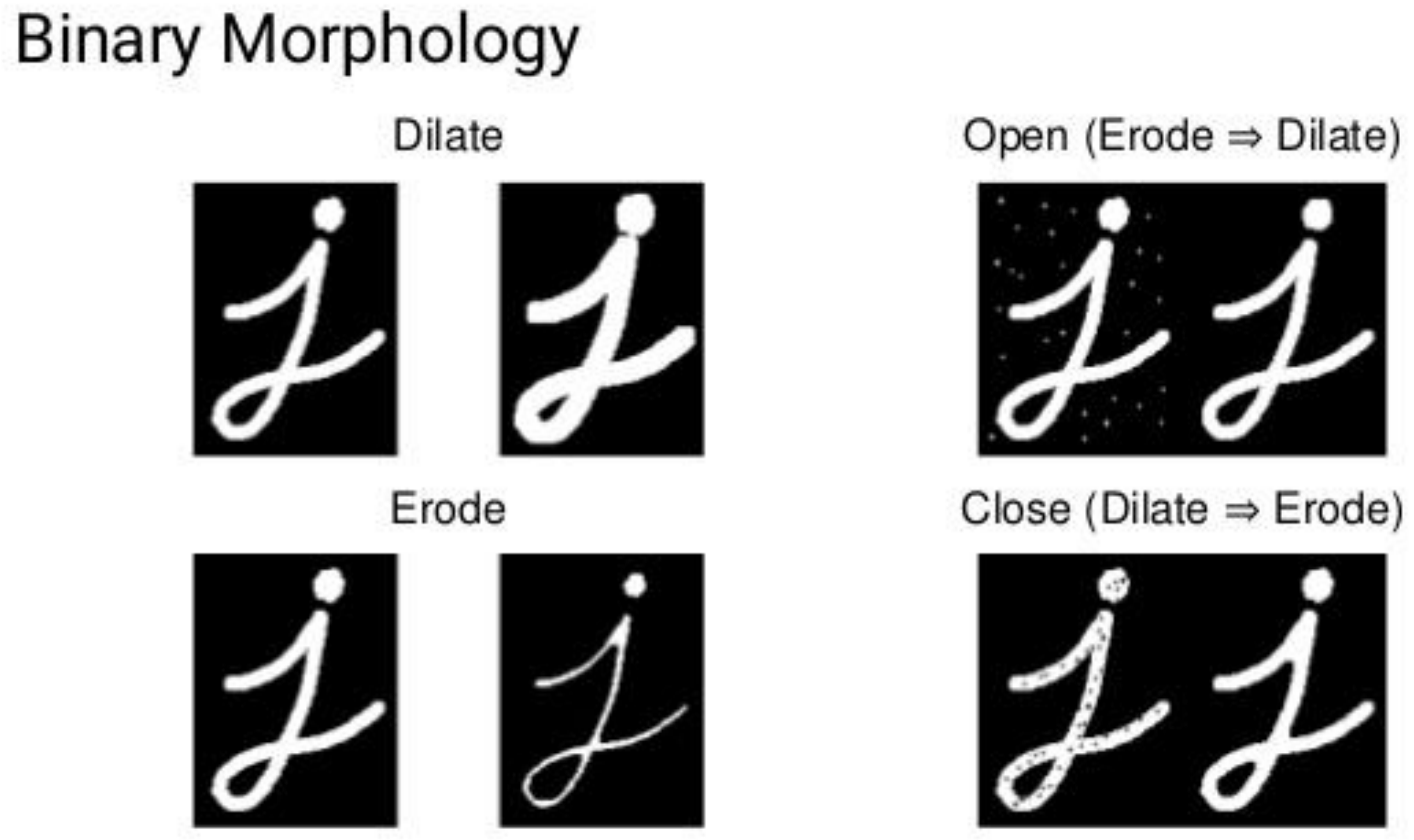
255	255	255
255	255	255
255	255	255

kernel



Morphology

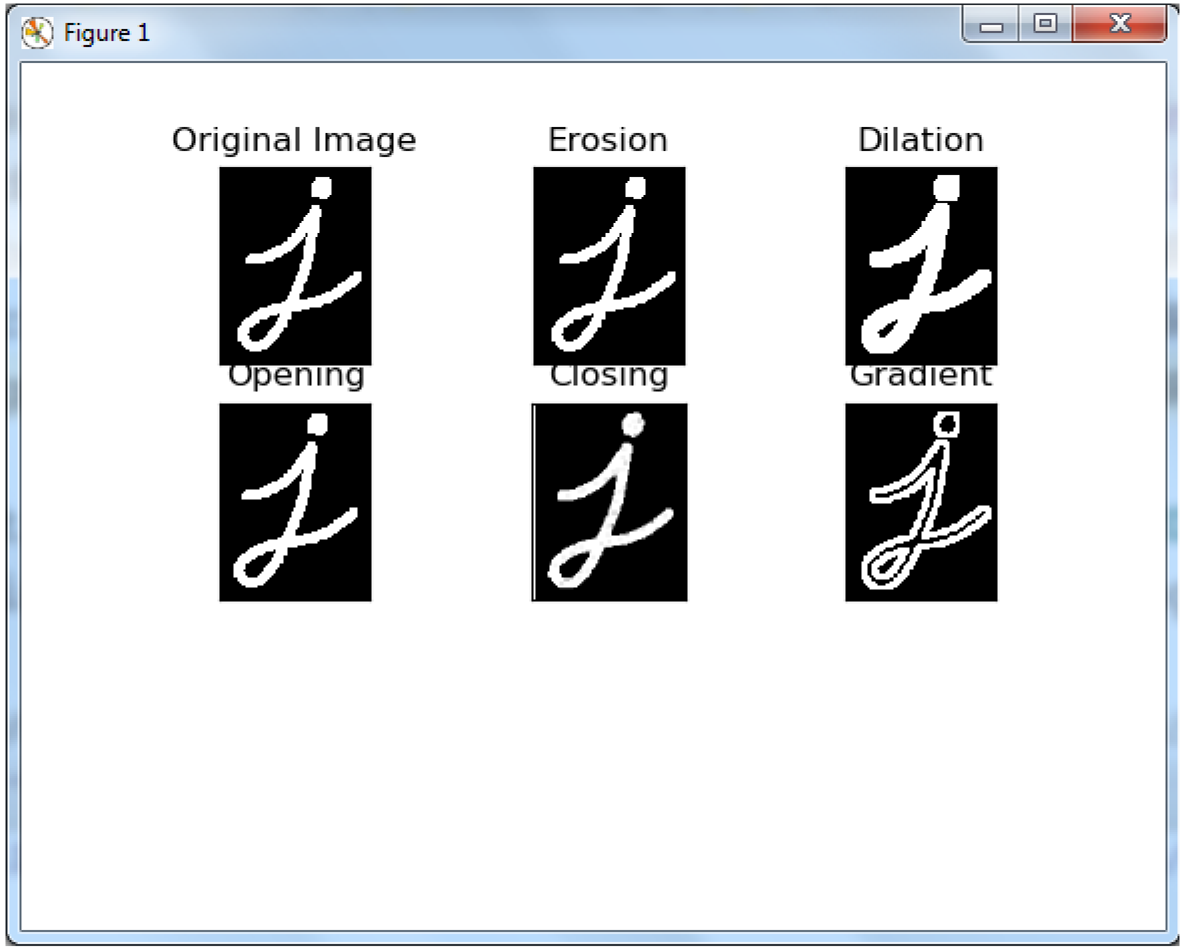
- 어디에 사용할 수 있을까?



출처 : https://lapina.tistory.com/84

Morphology

- 어디에 사용할 수 있을까?



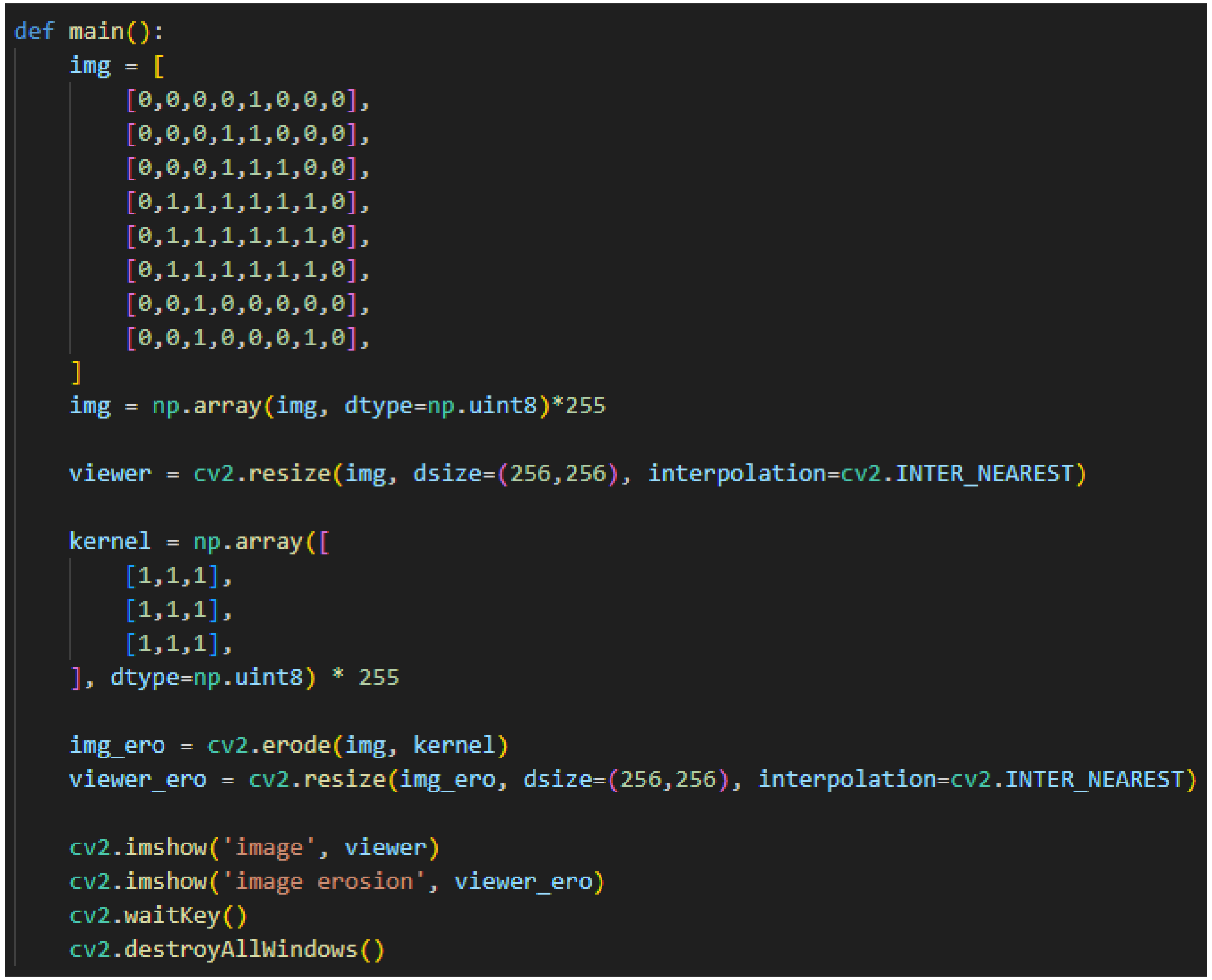
출처 : https://codeloop.org/opencv-python-morphological-transformations/

실습 및 과제

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

실습(IP5_1)

- Erosion

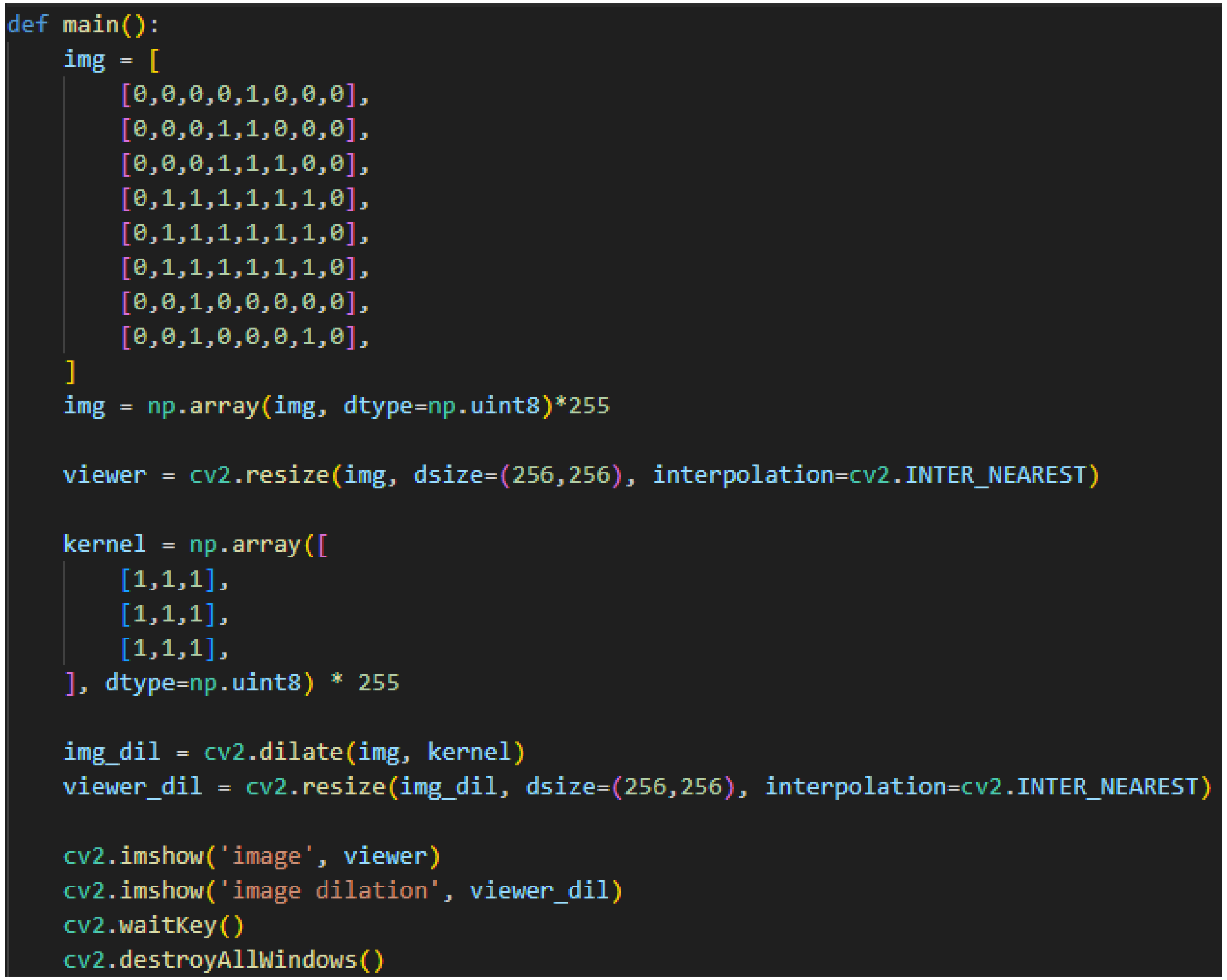


```
def main():
    img = [
        [0,0,0,0,1,0,0,0],
        [0,0,0,1,1,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,0,1,0,0,0,0,0],
        [0,0,1,0,0,0,1,0],
    ]
    img = np.array(img, dtype=np.uint8)*255

    viewer = cv2.resize(img, dsize=(256,256), interpolation=cv2.INTER_NEAREST)

    kernel = np.array([
        [1,1,1],
        [1,1,1],
        [1,1,1],
    ], dtype=np.uint8) * 255

    img_ero = cv2.erode(img, kernel)
    viewer_ero = cv2.resize(img_ero, dsize=(256,256), interpolation=cv2.INTER_NEAREST)

    cv2.imshow('image', viewer)
    cv2.imshow('image erosion', viewer_ero)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

```
img = [
        [0,0,0,0,1,0,0,0],
        [0,0,0,1,1,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,0,1,0,0,0,0,0],
        [0,0,1,0,0,0,1,0],
   ]
```

실습(IP5_2)

- Dilation

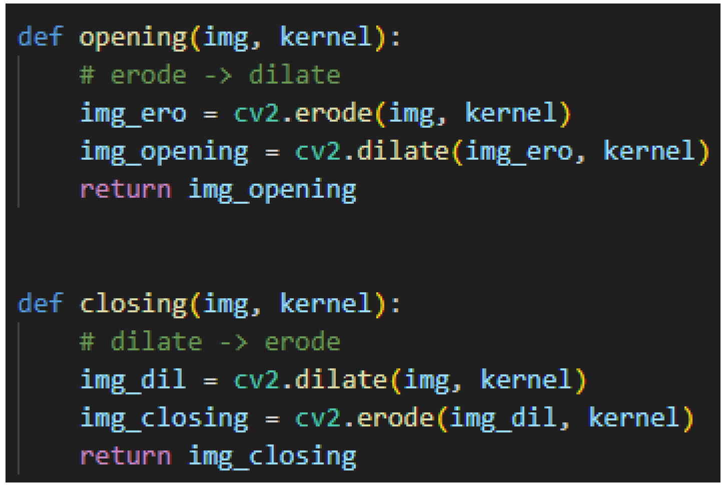


```
def main():
    img = [
        [0,0,0,0,1,0,0,0],
        [0,0,0,1,1,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,0,1,0,0,0,0,0],
        [0,0,1,0,0,0,1,0],
    ]
    img = np.array(img, dtype=np.uint8)*255

    viewer = cv2.resize(img, dsize=(256,256), interpolation=cv2.INTER_NEAREST)

    kernel = np.array([
        [1,1,1],
        [1,1,1],
        [1,1,1],
    ], dtype=np.uint8) * 255

    img_dil = cv2.dilate(img, kernel)
    viewer_dil = cv2.resize(img_dil, dsize=(256,256), interpolation=cv2.INTER_NEAREST)

    cv2.imshow('image', viewer)
    cv2.imshow('image dilation', viewer_dil)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

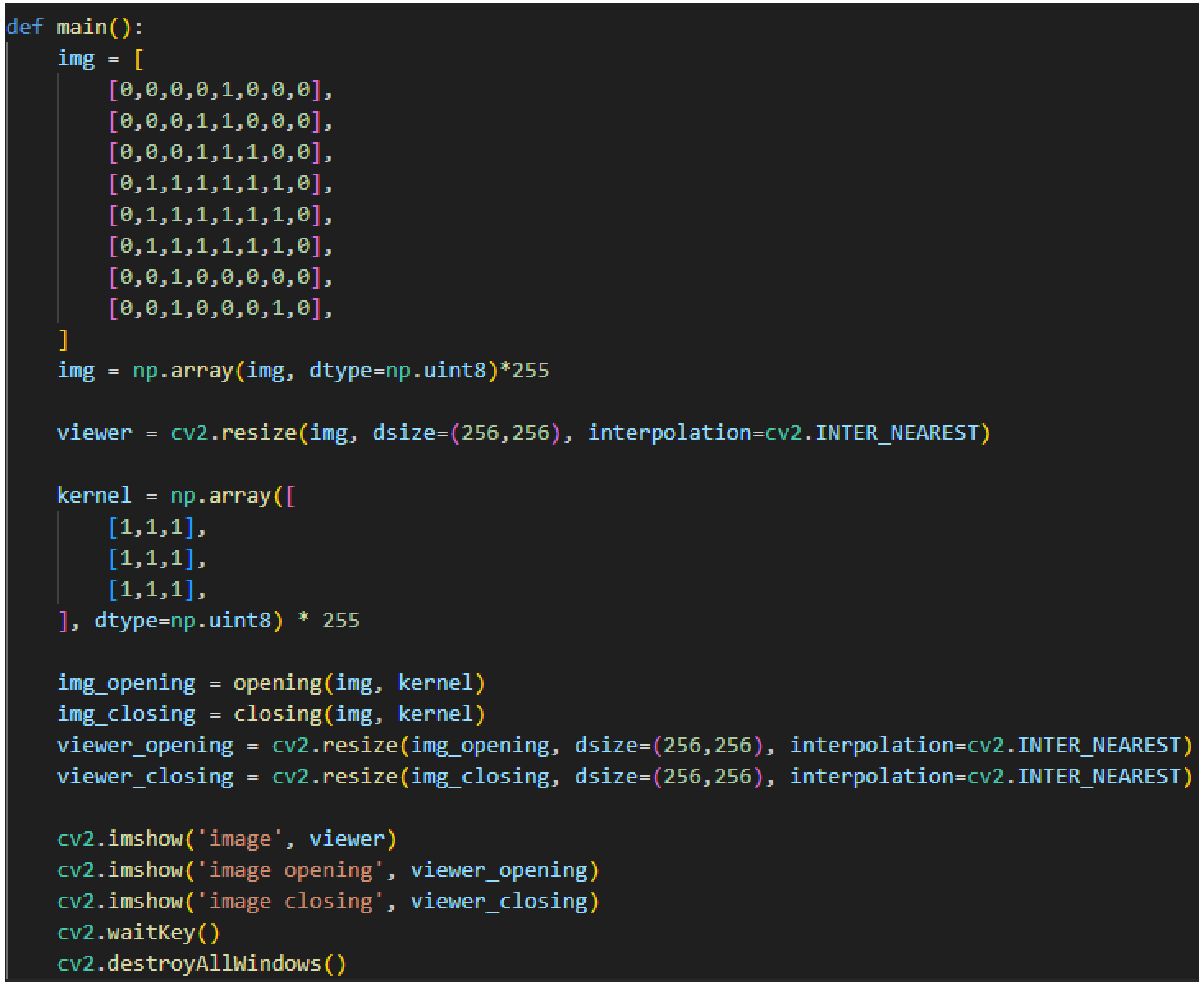
```
img = [
        [0,0,0,0,1,0,0,0],
        [0,0,0,1,1,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,0,1,0,0,0,0,0],
        [0,0,1,0,0,0,1,0],
    ]
```

실습(IP5_3)

- Opening & closing



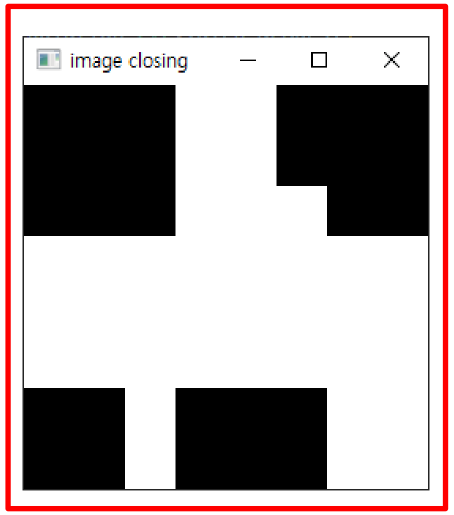
```python
def opening(img, kernel):
    # erode -> dilate
    img_ero = cv2.erode(img, kernel)
    img_opening = cv2.dilate(img_ero, kernel)
    return img_opening


def closing(img, kernel):
    # dilate -> erode
    img_dil = cv2.dilate(img, kernel)
    img_closing = cv2.erode(img_dil, kernel)
    return img_closing
```



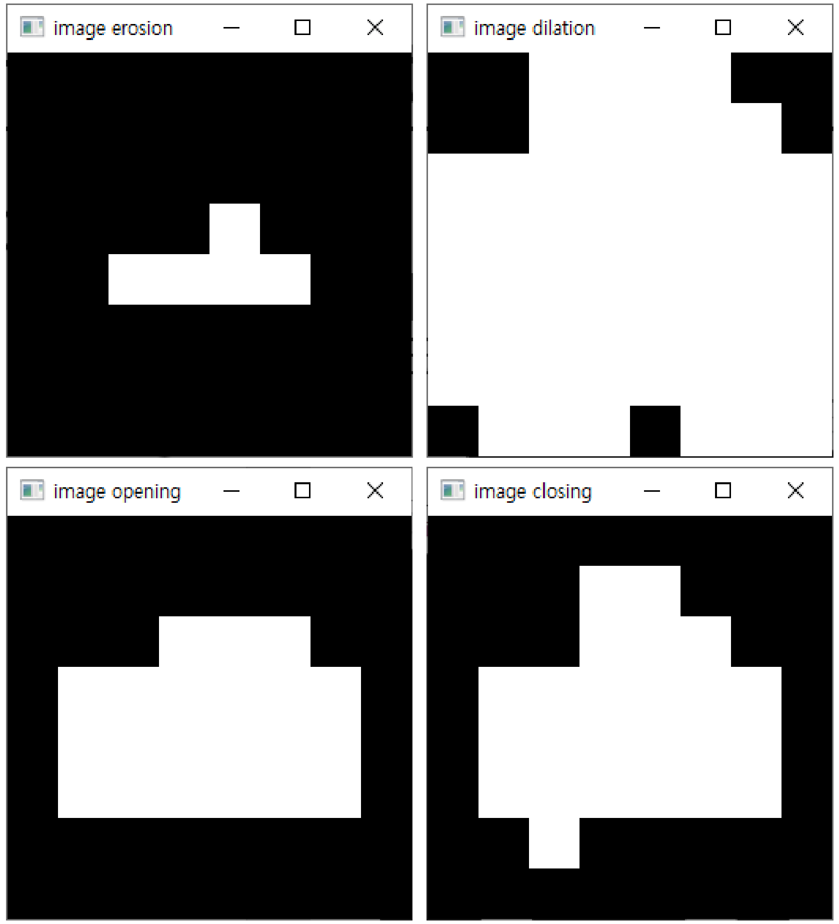
```python
def main():
    img = [
        [0,0,0,0,1,0,0,0],
        [0,0,0,1,1,0,0,0],
        [0,0,0,1,1,1,0,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,1,1,1,1,1,1,0],
        [0,0,1,0,0,0,0,0],
        [0,0,1,0,0,0,1,0],
    ]
    img = np.array(img, dtype=np.uint8)*255

    viewer = cv2.resize(img, dsize=(256,256), interpolation=cv2.INTER_NEAREST)

    kernel = np.array([
        [1,1,1],
        [1,1,1],
        [1,1,1],
    ], dtype=np.uint8) * 255

    img_opening = opening(img, kernel)
    img_closing = closing(img, kernel)
    viewer_opening = cv2.resize(img_opening, dsize=(256,256), interpolation=cv2.INTER_NEAREST)
    viewer_closing = cv2.resize(img_closing, dsize=(256,256), interpolation=cv2.INTER_NEAREST)

    cv2.imshow('image', viewer)
    cv2.imshow('image opening', viewer_opening)
    cv2.imshow('image closing', viewer_closing)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

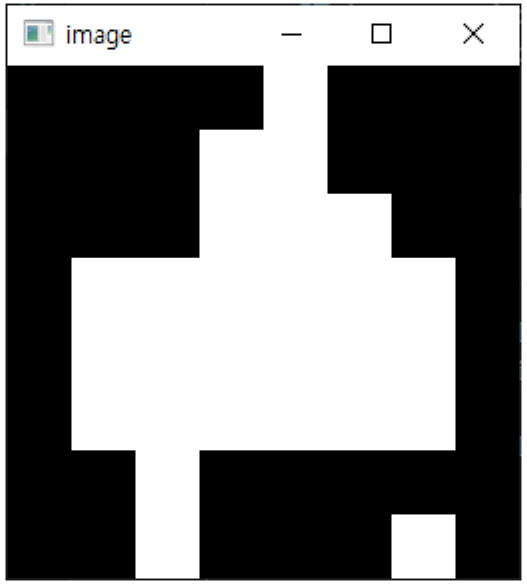
과제(IP5_test1)

- Opening & Closing 구현하기
  - 가장자리는 처리하지 않아도 됨
  - 원본과 똑같은 크기의 이미지가 생성되도록 하기



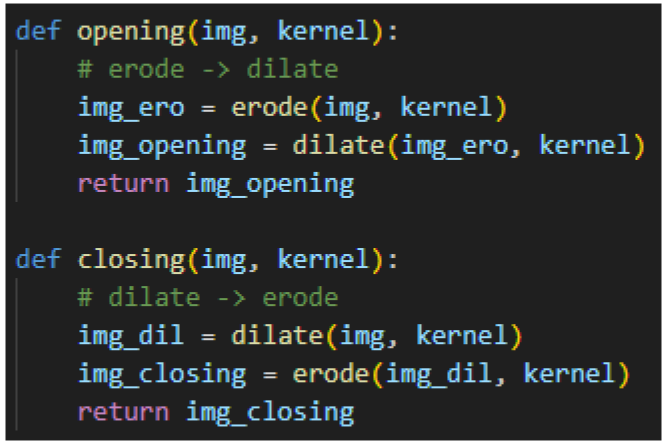
Opencv Closing 결과.
위와 동일해도 상관 없음



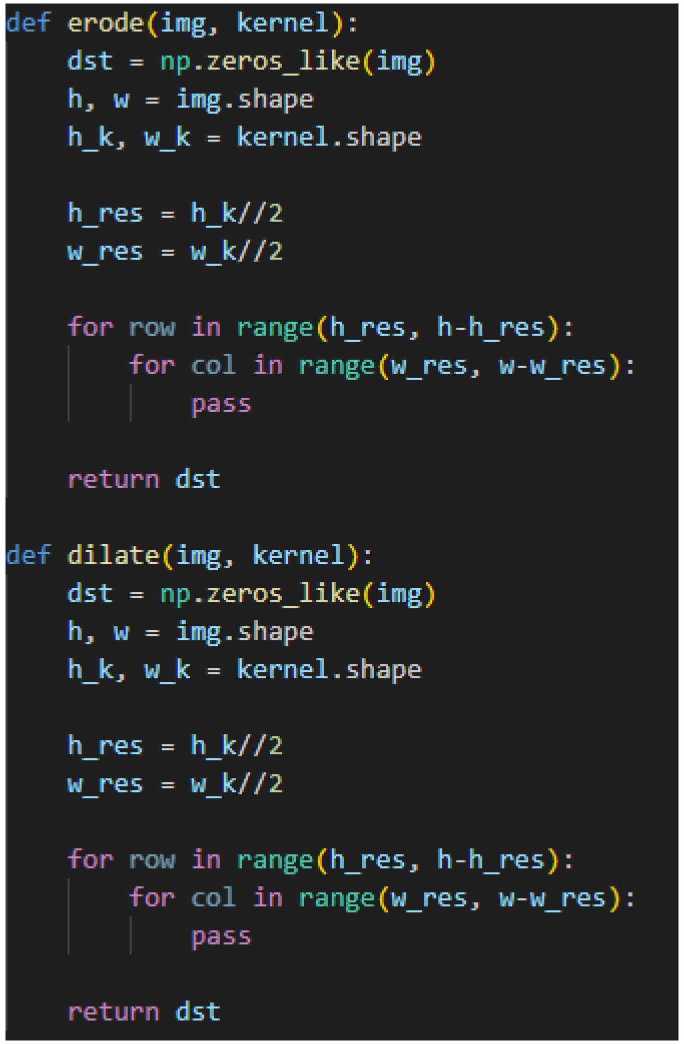


과제(IP5_test1)

- Opening & Closing 구현하기



```python
def opening(img, kernel):
    # erode -> dilate
    img_ero = erode(img, kernel)
    img_opening = dilate(img_ero, kernel)
    return img_opening

def closing(img, kernel):
    # dilate -> erode
    img_dil = dilate(img, kernel)
    img_closing = erode(img_dil, kernel)
    return img_closing
```



```python
def erode(img, kernel):
    dst = np.zeros_like(img)
    h, w = img.shape
    h_k, w_k = kernel.shape

    h_res = h_k//2
    w_res = w_k//2

    for row in range(h_res, h-h_res):
        for col in range(w_res, w-w_res):
            pass

    return dst

def dilate(img, kernel):
    dst = np.zeros_like(img)
    h, w = img.shape
    h_k, w_k = kernel.shape

    h_res = h_k//2
    w_res = w_k//2

    for row in range(h_res, h-h_res):
        for col in range(w_res, w-w_res):
            pass

    return dst
```

QnA