Chap. 4 OpenCV를 이용한 기본 영상처리 기법들-모폴로지

► 수업 내용

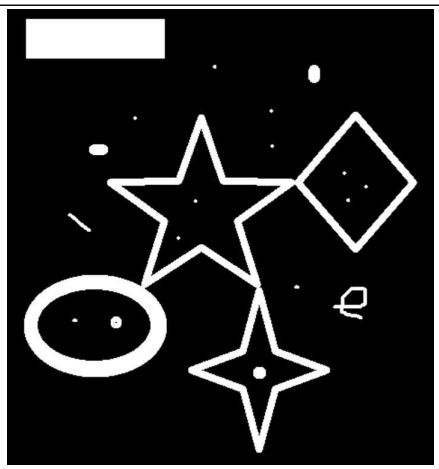
- Morphology 이용한 기본 영상처리 기법들
  : Erosion, Dilation, Opening, Closing, Morpholigical Gradient 연산
- 기하학적 변환

1. 모폴로지 변환(Morphology)

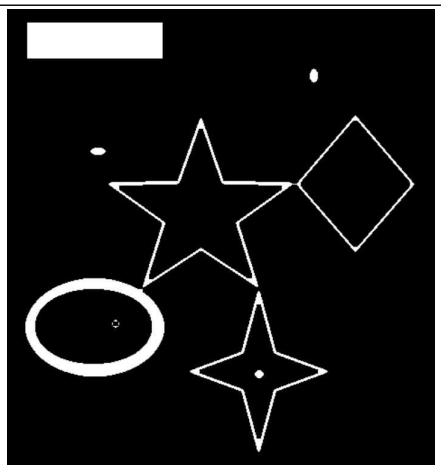
- 입력영상에 대하여 주어진 커널을 이용하여 침식(Erosion), 팽창(Dilation), 열림(Opening), 닫힘(Closing) 등의 연산을 통하여 입력영상을 변환하는 것
- Convolution처럼 단순한 알고리즘으로 영상에 대한 변환이 가능

(1) Erosion

- Kernel부분 영역내의 모든 픽셀이 일정 조건을 갖출때만 해당 픽셀을 남기고, 나머지 경우에는 해당 픽셀을 지우므로써 Object의 경계 부분을 깎아내는 연산
- 작은 크기의 잡음을 지우거나 연결된 오브젝트들을 떼어 내는데 사용
- Erosion후 오브젝트는 작아질 수 있음



입력영상



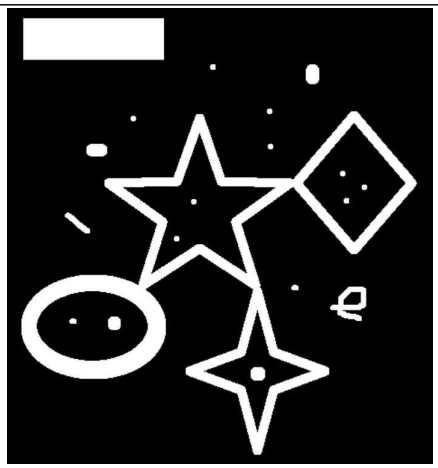
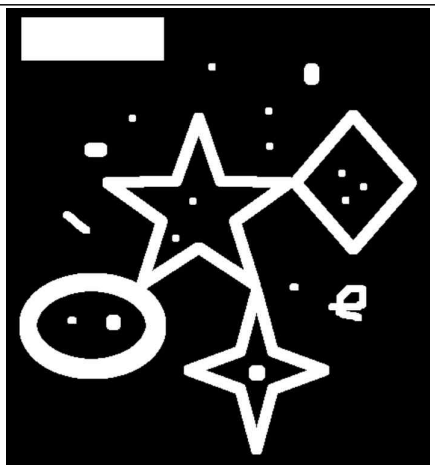
Erosion 2번한 결과

- 코딩의 예

```
img = cv2.imread('test.png',0)
kernel = np.ones((3,3),np.uint8)
erosion = cv2.erode(img, kernel, iterations = 1)
```

(2) Dilation

- Kernel부분 영역내의 한 개 이상의 픽셀이 일정 조건을 갖추면 해당 픽셀을 남김으로써 Object의 경계 부분을 확대시키는 연산
- 떨어진 오브젝트들을 붙이는데 사용
- Dilation후 오브젝트는 커질 수 있음

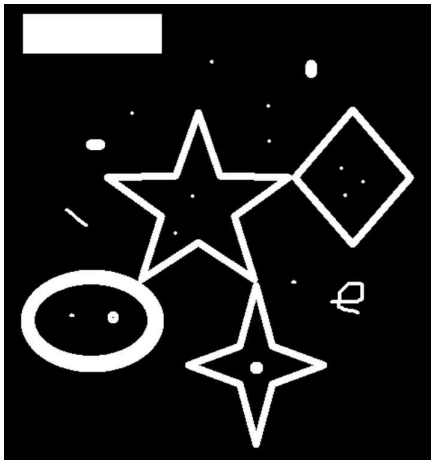
	
Dilation 한번한 결과	Dilation 3번한 결과

- 코딩의 예

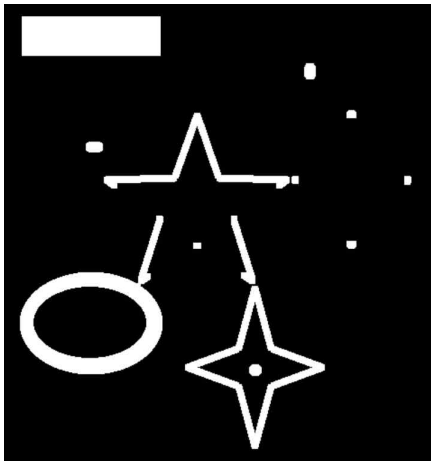
```
import numpy as np

img = cv2.imread('test.png',0)
kernel = np.ones((3,3),np.uint8)
dilation = cv.2dilate(img, kernel, iterations = 1)
```

(3) Opening

- Erosion후 Dilation 하는 연산
- 잡음제거와 붙은 오브젝트를 분리하는데 사용



입력영상



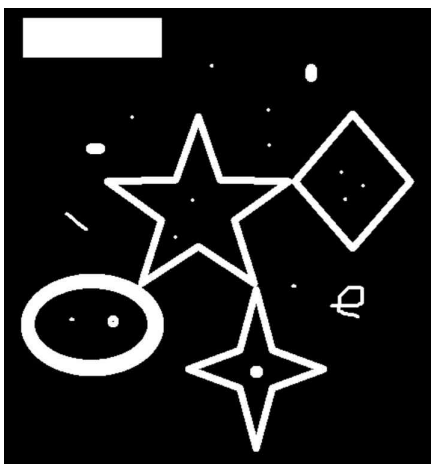
Opening 3번한 결과

- 코딩의 예

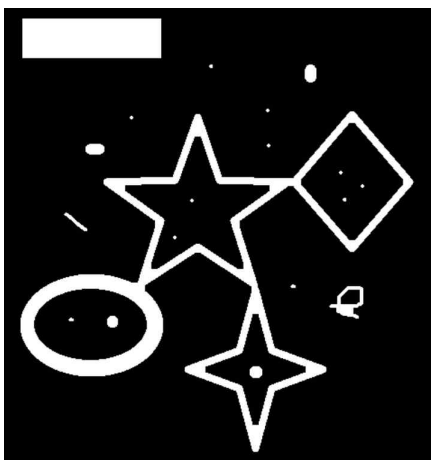
```
import numpy as np

img = cv2.imread('test.png',0)
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel, iterations = 3)
```

(4) Closing

- Dilation후 Erosion 하는 연산. Opening의 반대 연산
- 오브젝트상의 작은 홀을 메우거나 오브젝트를 연결하는데 사용



입력영상



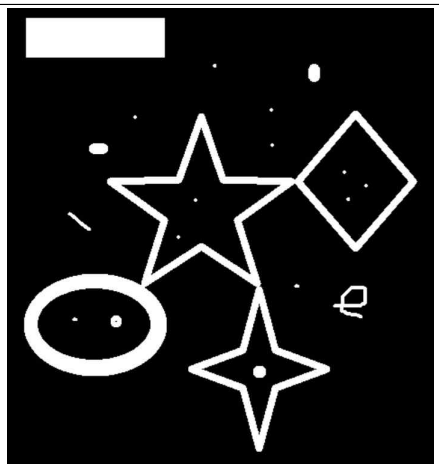
Closing 3번한 결과

- 코딩의 예

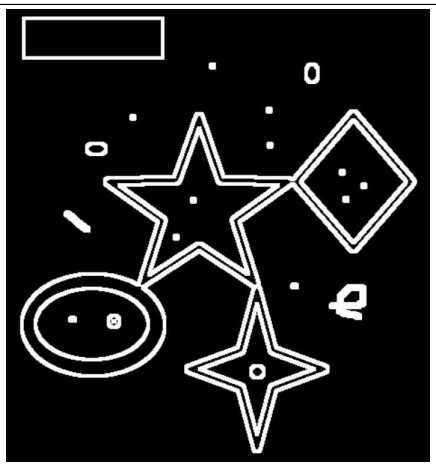
```
import numpy as np

img = cv2.imread('test.png',0)
kernel = np.ones((3,3),np.uint8)
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

(5) Morphological Gradient

- Dilation한 영상으로부터 Erosion 한 영상을 빼는 연산.
- 오브젝트의 윤곽을 추출할 수 있음



입력영상



Closing 5번한 결과

- 코딩의 예

```
import numpy as np

img = cv2.imread('test.png',0)
kernel = np.ones((3,3),np.uint8)
gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)
```

(6) Structuring Element

- kernel의 모양을 직사각형 외에 타원이나 십자(+)모양으로 설정 가능
- cv2.getStructuringElement() 함수 사용

```
import numpy as np

img = cv2.imread('test.png',0)
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)

# Rectangular Kernel
kernel = cv2.getStructuringElement(cv.MORPH_RECT,(5,5))
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]], dtype=uint8)
# Elliptical Kernel
>>> cv2.getStructuringElement(cv.MORPH_ELLIPSE,(5,5))
array([[0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0]], dtype=uint8)
# Cross-shaped Kernel
>>> cv2.getStructuringElement(cv.MORPH_CROSS,(5,5))
array([[0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0]], dtype=uint8)
```

2. 기하학적 변환 (Geometric Transformation)

(1) Scaling (크기 변환)

- 영상을 확대하거나 축소
- 코딩의 예

```
img = cv2.imread('sample.png',0)
   # 축소
shrinking = cv2.resize(img, None, fx=0.5, fy=0.5, interpolation = cv2.INTER_AREA)
   # 확대
enlarzing = cv2.resize(img, None, fx=2, fy=2, interpolation = cv2.INTER_LINEAR)
```

(2) Translation (위치 이동)

- 영상의 위치를 이동 (sliding)

Mask : $$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

- 코딩의 예

```
img = cv2.imread('sample.png',0)
   # Translation
rows, cols = img.shape
Mask = np.float32([[1,0,100],[0,1,50]])
dst = cv2.warpAffine(img, Mask, (cols,rows))  # cols, row, 영상의 크기
cv2.imshow('img',dst)
```

(3) Rotation (회전)

- 영상을 임의 각도로 회전

Mask : $$M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

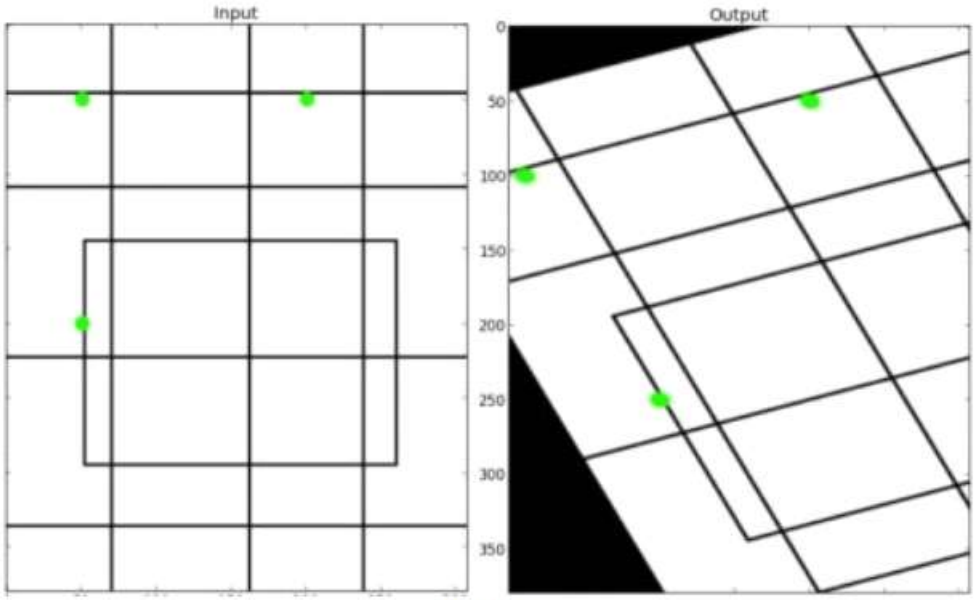
- 코딩의 예

```
img = cv2.imread('sample.png',0)
   # Rotation
rows, cols = img.shape
   # cols-1 and rows-1 are the coordinate limits.
Mask = cv2.getRotationMatrix2D(((cols-1)/2.0,(rows-1)/2.0), 90, 1)
dst = cv2.warpAffine(img, Mask, (cols,rows))
cv2.imshow('img',dst)
```

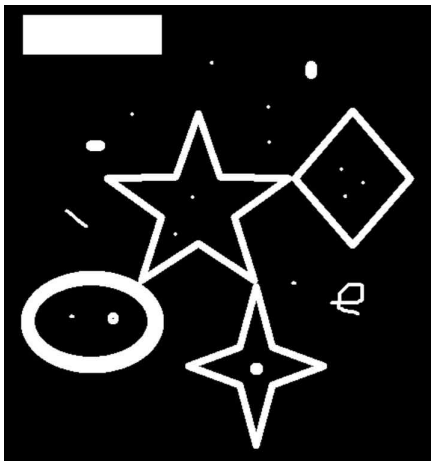
(4) Affine Transformation (선형 변환)

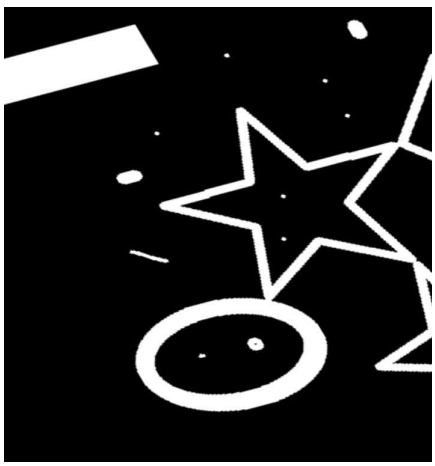
- 입력 영상과 선형관계를 유지하는 선형변환

- 입력과 타겟 영상의 관계를 나타내기 위하여 각각 3점의 좌표값 사용



image


입력영상

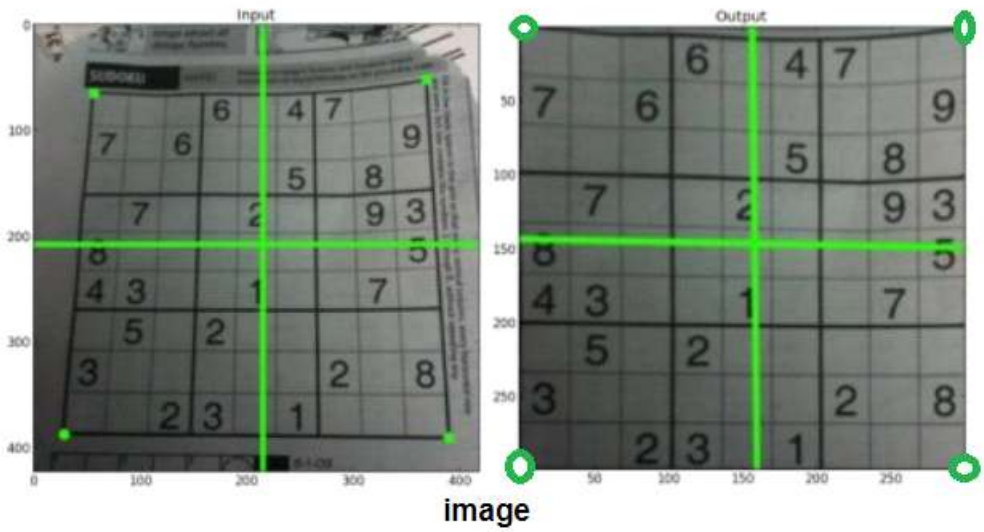

Affine 변환된 영상

- 코딩의 예

```
img = cv2.imread('sample.png',0)
    # Affine Tr.
pts1 = np.float32([[50,50],[200,50],[50,200]])       # 입력 영상의 점
pts2 = np.float32([[10,100],[200,50],[100,250]])     # 타겟 영상의 대응점
M = cv2.getAffineTransform(pts1,pts2)
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow('img',dst)
```

(5) Perspective Transformation (관점 변환)

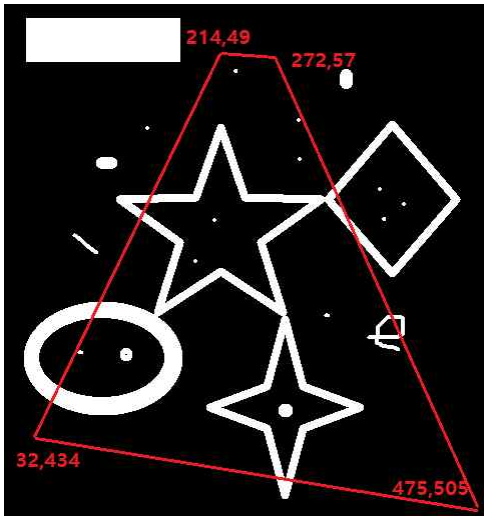
- 보는 관점에 른 기하학적 변환을 나타냄
- 입력과 타겟 영상의 관계를 나타내기 위하여 각각 4점의 좌표값 사용

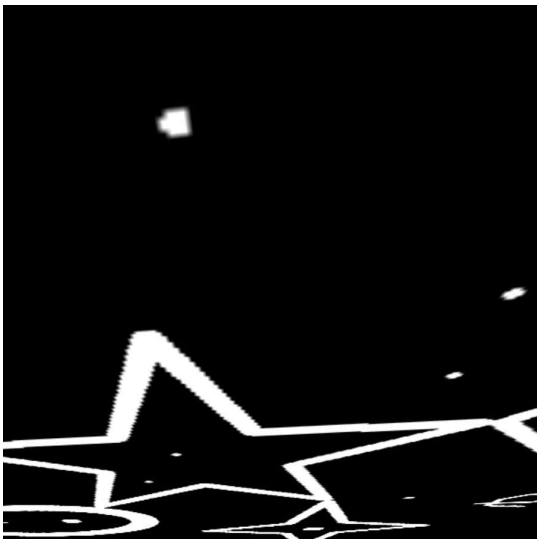


image

- 코딩의 예

```
img = cv2.imread('sample.png',0)
    # Perspective Tr.
pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])
pts2 = np.float32([[0,0],[300,0],[0,300],[300,300]])
M = cv2.getPerspectiveTransform(pts1,pts2)
dst = cv2.warpPerspective(img, M, (300,300))

cv2.imshow('img',dst)
```



pts1= [[214,49],[272,57],[32,434],[475,505]] pts2=[[0,0],[600,0],[0,600],[600, 600]]	pts1= [[32,434],[475,505],[214,49],[272,57]] pts2= [[0,0],[600,0],[0,600],[600, 600]]
	

3. 실습

(ip) d:\ip>python iprocessing_basic2.py