Morphological Transformations

김수환

https://www.soohwan.kim


학습목표

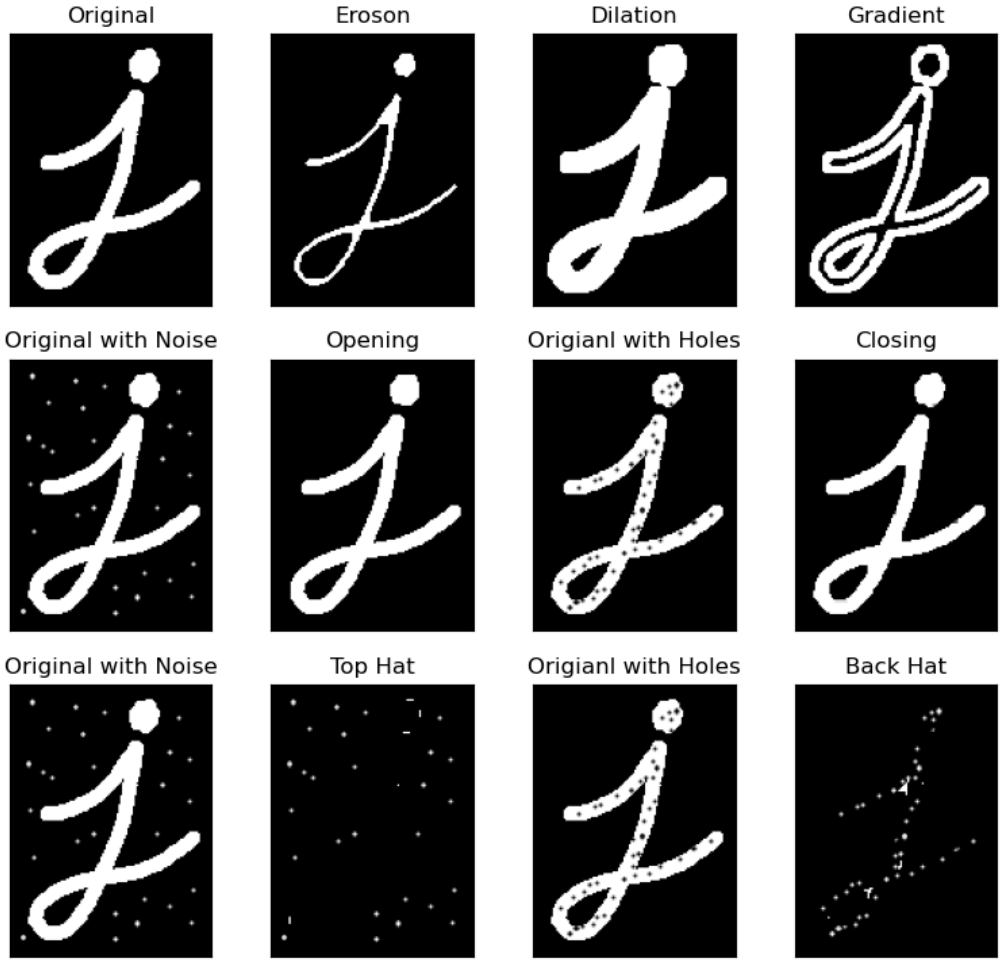
학습목표

1. Image Processing
    1. Image Thresholding
    2. Image Blending
    3. Image Filtering
    4. Morphological Transformations

Morphological Transformations

- Image의 형태를 바꾸는 연산
- Binary image에 주로 적용
- Two basic morphological operators
  - Erosion
  - Dilation
- Variants
  - Opening
  - Closing
  - Gradient
  - Top Hat
  - Black Hat
  - Hit or Miss

Morphological Transformations



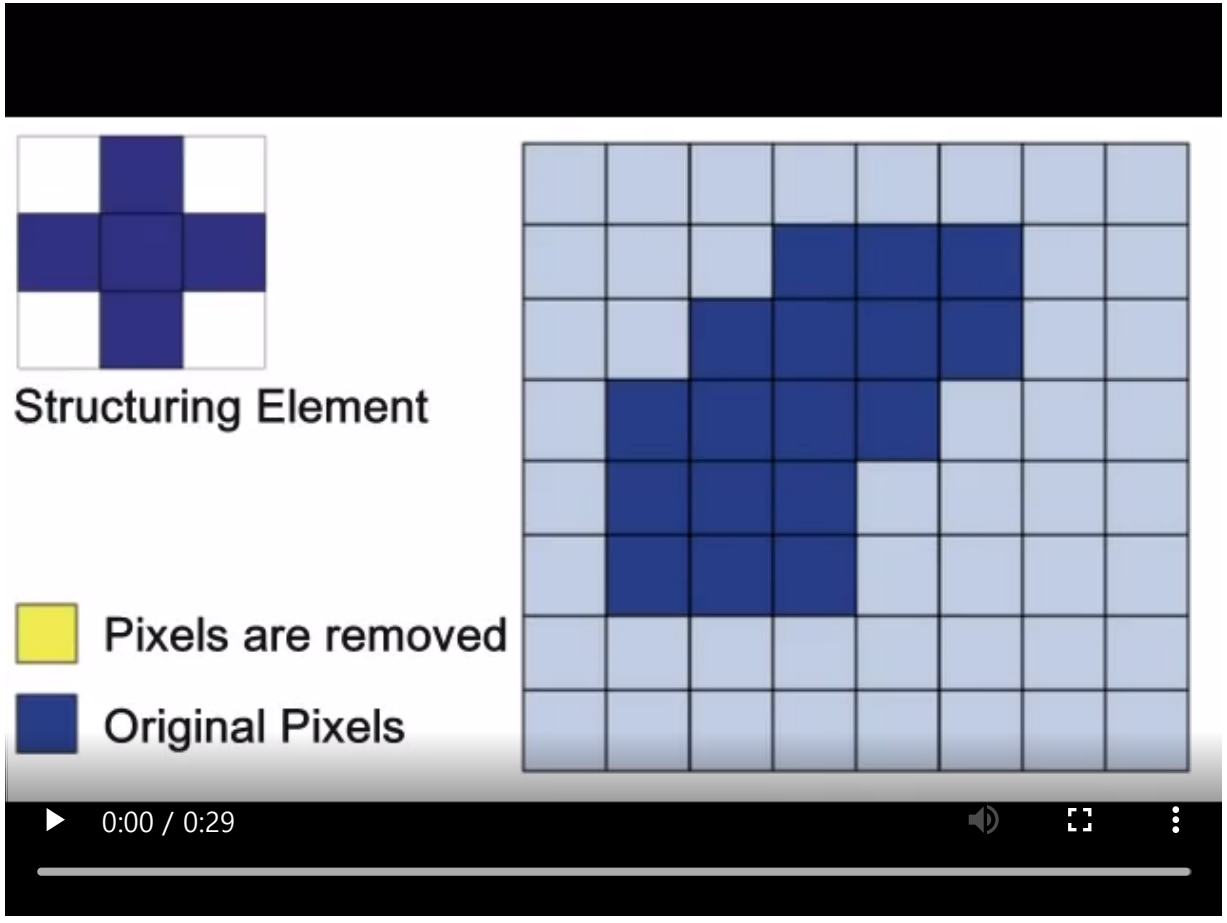
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

Erosion (침식)

Erosion (침식)

- 토양이 침식되듯 foreground object의 boundaries가 침식됨
- Convolution처럼 kernel이 image를 slide하면서 연산을 수행
- 각 pixel에 대해서 kernel 안에 있는 모든 pixel이 1이면 해당 pixel 값이 1, 아니면 0.
- 결과적으로 모든 foreground object boundary가 침식됨
- 작은 white noise를 제거하는데 효과적임
- 두 개의 connected objects를 떨어뜨리는데 효과적임

Erosion (침식)



https://gfycat.com/ko/finekeybream-computer-science-field-of-study

`cv.erode()`[1]

```
dst = cv.erode(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]])
```

- Image를 특별한 structuring element로 침식시킨다.
  - `src`: input image; the number of channels can be arbitrary, but the depth should be one of `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or `CV_64F`)
  - `dst`: output array of the same size and type as `src`
  - `kernel`: structuring element used for erosion
  - `anchor`: anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1, -1) means that the anchor is at the kernel center
  - `iterations`: number of times erosion is applied
  - `borderType`: pixel extrapolation method
  - `borderValue`: border value in case of a constant border

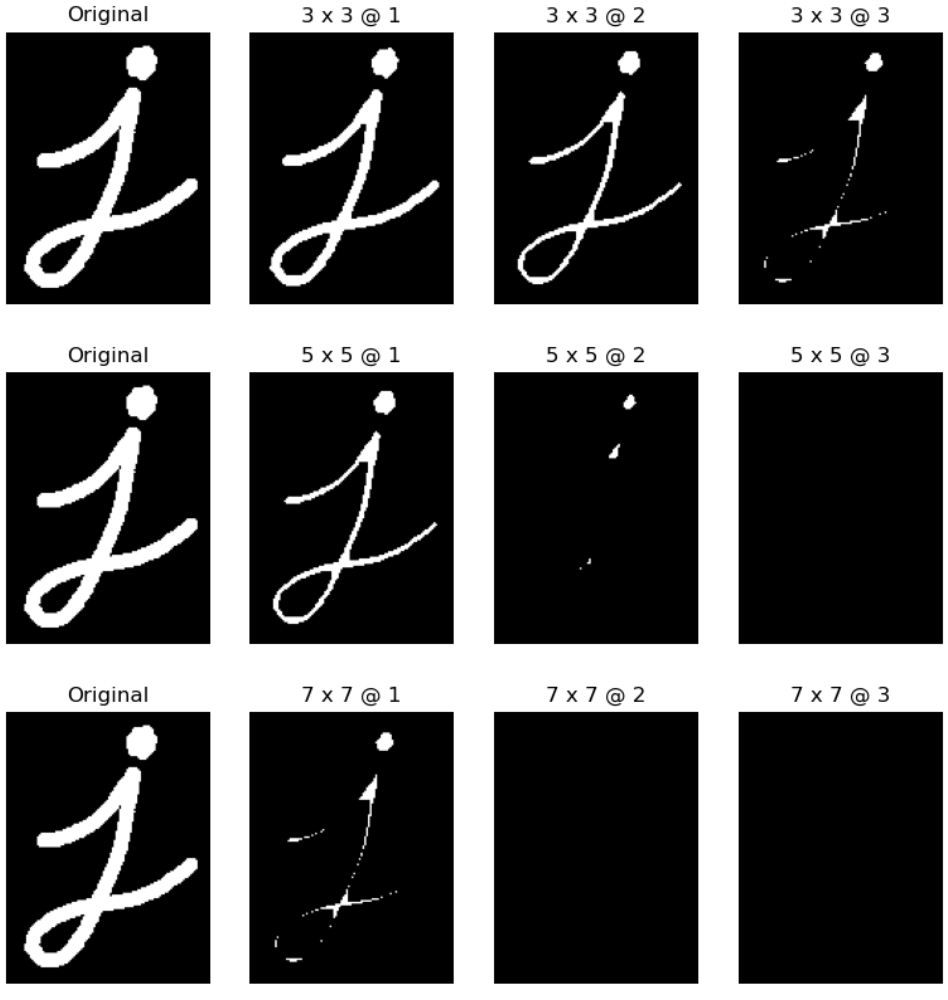
$$\texttt{dst}(x,y) = \min_{(x',y'):\texttt{element}(x',y')\neq 0} \texttt{src}(x+x', y+y')$$

1. https://docs.opencv.org/4.4.0/d4/d86/group__imgproc__filter.html#gaeb1e0c1033e3f6b891a25d0511362aeb

Erosion: Code

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load grayscale images
img_orig = cv2.imread('j.png', 0)

index = 1
for kernel_size in [3,5,7]:
    for iterations in [1,2,3]:
        # Kernel
        kernel = np.ones((kernel_size, kernel_size), np.uint8)

        # Erosion
        img_res = cv2.erode(img_orig, kernel, iterations=iterations)

        # Display results
        plt.subplot(3, 3, index)
        index += 1
        plt.imshow(img_res, 'gray')
        plt.title(str(kernel_size) + ' x ' + str(kernel_size) + ' @ ' + str(iterations))
        plt.axis('off')
```

Erosion: Result

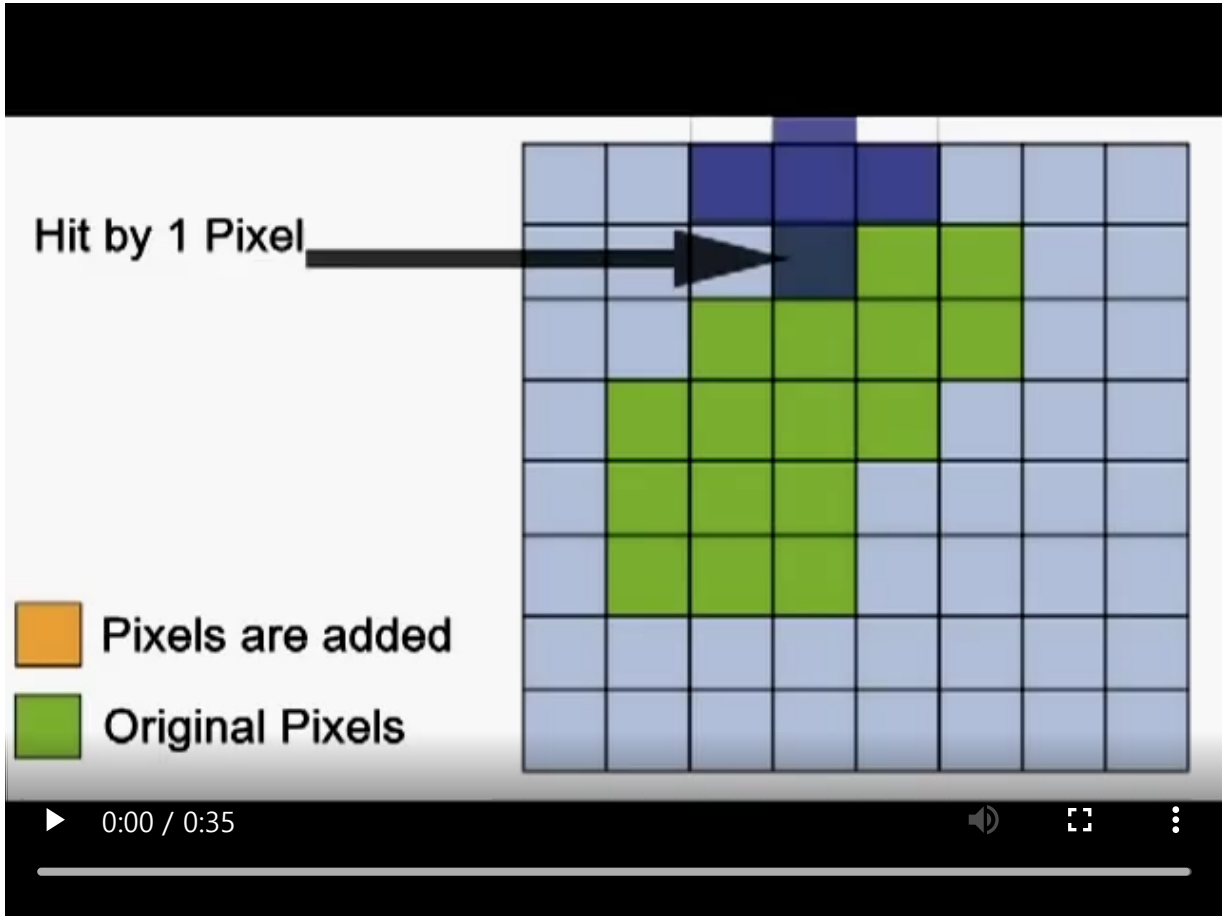


Dilation (팽창)

Dilation (팽창)

- 토양이 퇴적되듯 foreground object의 boundary가 팽창됨
- 각 pixel에 대해서 kernel 안의 pixel 중에 하나라도 1이면 해당 pixel 값이 1, 아니면 0.
- Noise 제거를 위해서 erosion 후 작아진 object를 다시 키우기 위해서 dilation을 수행.
- 두 개의 broken parts를 합치는데 효과적임

Dilation (팽창)



https://gfycat.com/ko/sourpaltryblackwidowspider-smart-e-learning-processing-university

`cv.dilate()`[1]

```
dst = cv.dilate(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]])
```

- Image를 특별한 structuring element로 팽창시킨다.
  - `src`: input image; the number of channels can be arbitrary, but the depth should be one of `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or `CV_64F`)
  - `dst`: output array of the same size and type as `src`
  - `kernel`: structuring element used for dilation
  - `anchor`: anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1, -1) means that the anchor is at the kernel center
  - `iterations`: number of times erosion is applied
  - `borderType`: pixel extrapolation method
  - `borderValue`: border value in case of a constant border

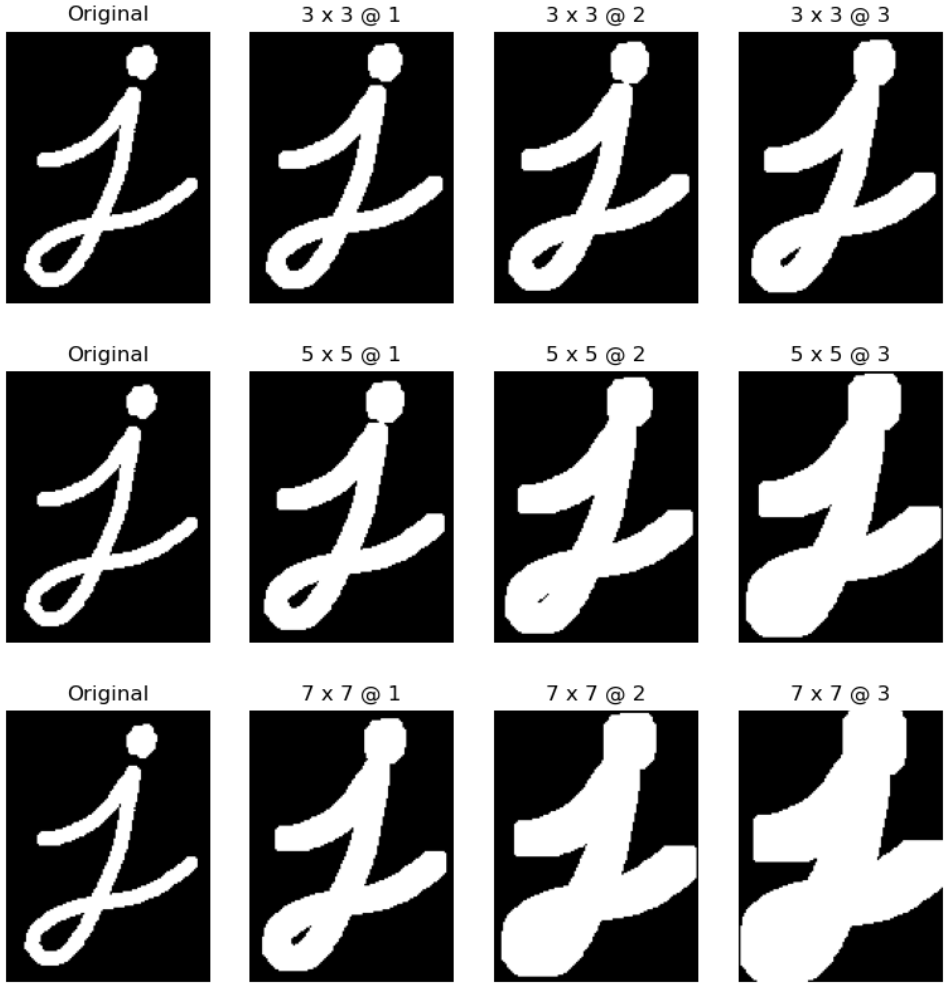
$$\texttt{dst}(x,y) = \max_{(x',y'):\,\texttt{element}(x',y')\neq 0} \texttt{src}(x+x',y+y')$$

1. https://docs.opencv.org/4.4.0/d4/d86/group__imgproc__filter.html#ga4ff0f3318642c4f469d0e11f242f3b6c

Dilation: Code

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load grayscale images
img_orig = cv2.imread('j.png', 0)

index = 1
for kernel_size in [3,5,7]:
    for iterations in [1,2,3]:
        # Kernel
        kernel = np.ones((kernel_size, kernel_size), np.uint8)

        # Erosion
        img_res = cv2.dilate(img_orig, kernel, iterations=iterations)

        # Display results
        plt.subplot(3, 3, index)
        index += 1
        plt.imshow(img_res, 'gray')
        plt.title(str(kernel_size) + ' x ' + str(kernel_size) + ' @ ' + str(iterations))
        plt.axis('off')
```

Dilation: Result



Structuring Elements

`cv.getStructuringElement()`[1]

```
retval = cv.getStructuringElement(shape, ksize[, anchor])
```

- Morphological operation을 하기 위해 특정한 모양과 크기의 structuring element를 만든다.
  - `shape`: element shape that could be one of `MorphShapes`
  - `kernel`: size of structuring element
  - `anchor`: anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1, -1) means that the anchor is at the kernel center

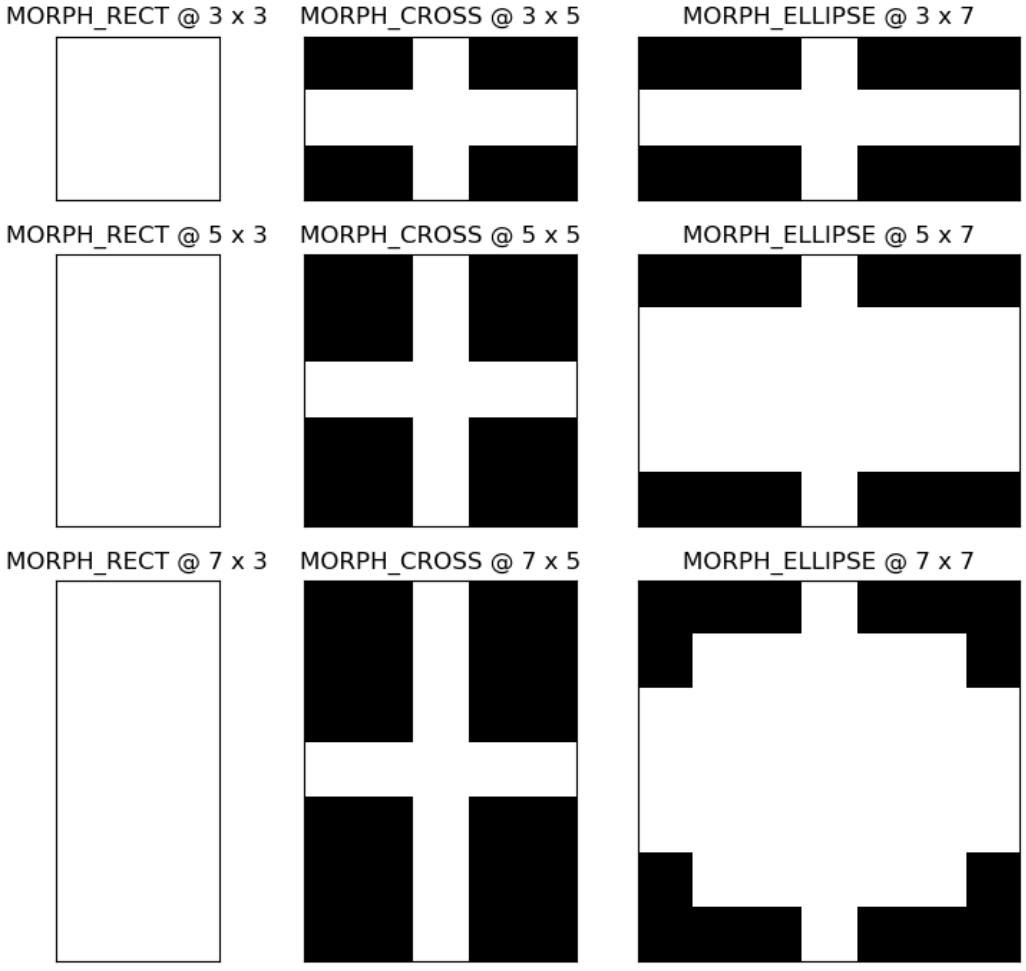
`MorphTypes`[2]	Operation
`cv.MORPH_RECT`	$E_{ij} = 1$
`cv.MORPH_CROSS`	$E_{ij} = \begin{cases} 1, & \text{if } i = \text{anchor.y or } j = \text{anchor.x} \\ 0, & \text{otherwise} \end{cases}$
`cv.MORPH_ELLIPSE`	Elliptic structuring element

1. https://docs.opencv.org/4.4.0/d4/d86/group__imgproc__filter.html#gac342a1bb6eabf6f55c803b09268e36dc
2. https://docs.opencv.org/4.4.0/d4/d86/group__imgproc__filter.html#ga7be549266bad7b2e6a04db49827f9f32

Structuring Elements: Code #1

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

fig = plt.figure(constrained_layout=True)
rows = cols = [3, 5, 7]
spec = fig.add_gridspec(ncols=len(cols), nrows=len(rows), width_ratios=cols,
height_ratios=rows)

kernel_shape = [cv2.MORPH_RECT, cv2.MORPH_CROSS, cv2.MORPH_ELLIPSE]
kernel_shape_str = ['MORPH_RECT', 'MORPH_CROSS', 'MORPH_ELLIPSE']

for r in range(len(rows)):
    for c in range(len(cols)):
        # Create a kernel
        kernel = cv2.getStructuringElement(kernel_shape[c], (cols[c], rows[r]))

        # Display results
        ax = fig.add_subplot(spec[r, c])
        plt.imshow(kernel, 'gray', vmin=0, vmax=1)
        plt.title(kernel_shape_str[c] + ' @ ' + str(rows[r]) + ' x ' + str(cols[c]))
        plt.xticks([]), plt.yticks([])
```

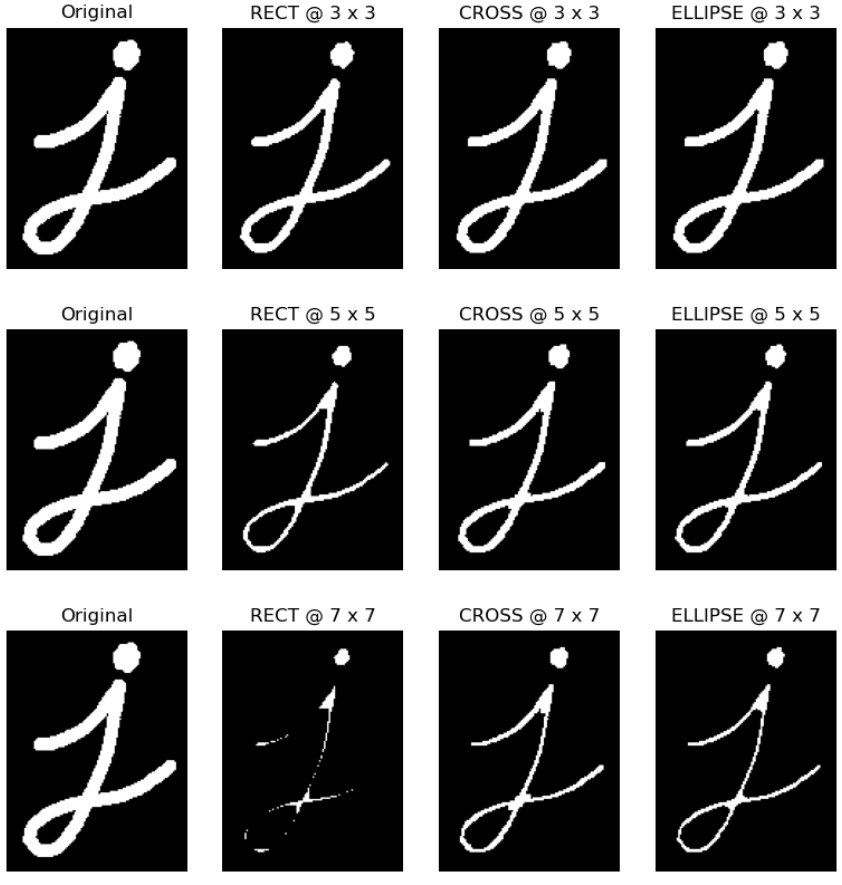
Structuring Elements: Result #1



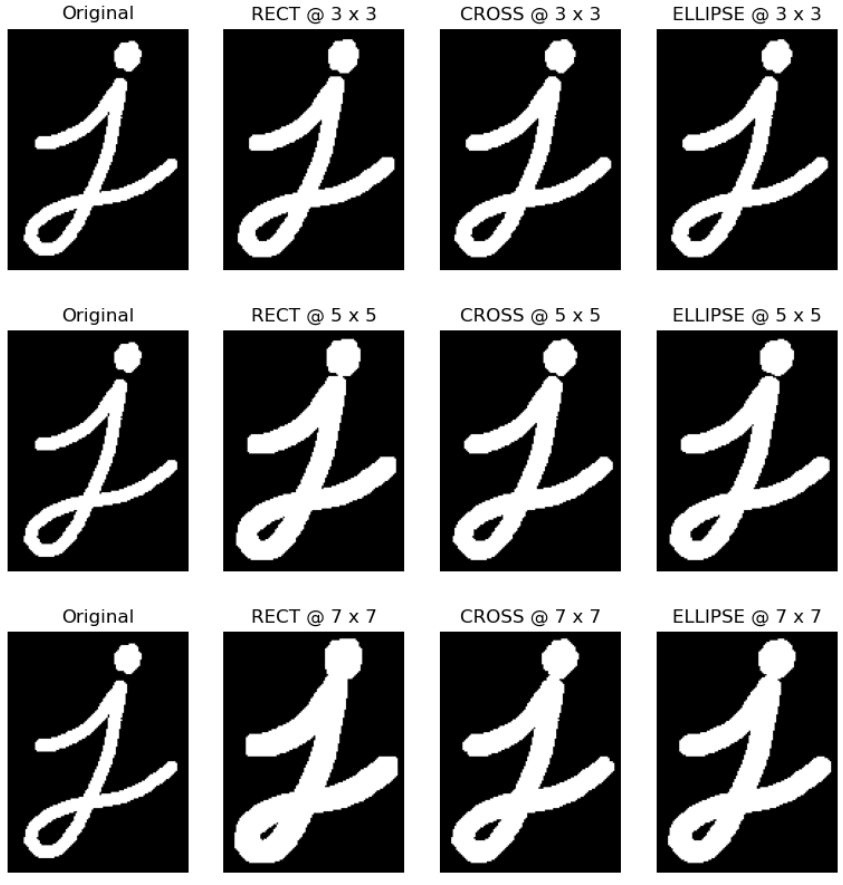
Structuring Elements: Code #2

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load grayscale images
img_orig = cv2.imread('j.png', 0)

# Settings
kernel_sizes = [3, 5, 7]
kernel_shapes = ['RECT', 'CROSS', 'ELLIPSE']
index = 1
for size in kernel_sizes:
    # Display original
    plt.subplot(3, 4, index); index += 1
    plt.imshow(img_orig, 'gray')
    plt.title('Original')
    plt.axis('off')

    for shape in kernel_shapes:
        # Kernel
        kernel = cv2.getStructuringElement(getattr(cv2, 'MORPH_' + shape), (size, size))
```

Structuring Elements: Result #2



Erosion



Dilation

Opening and Closing

Opening vs. Closing vs. Gradient

- Opening
  - Erosion 다음 Dilation
  - Noise를 제거하는데 효과적임

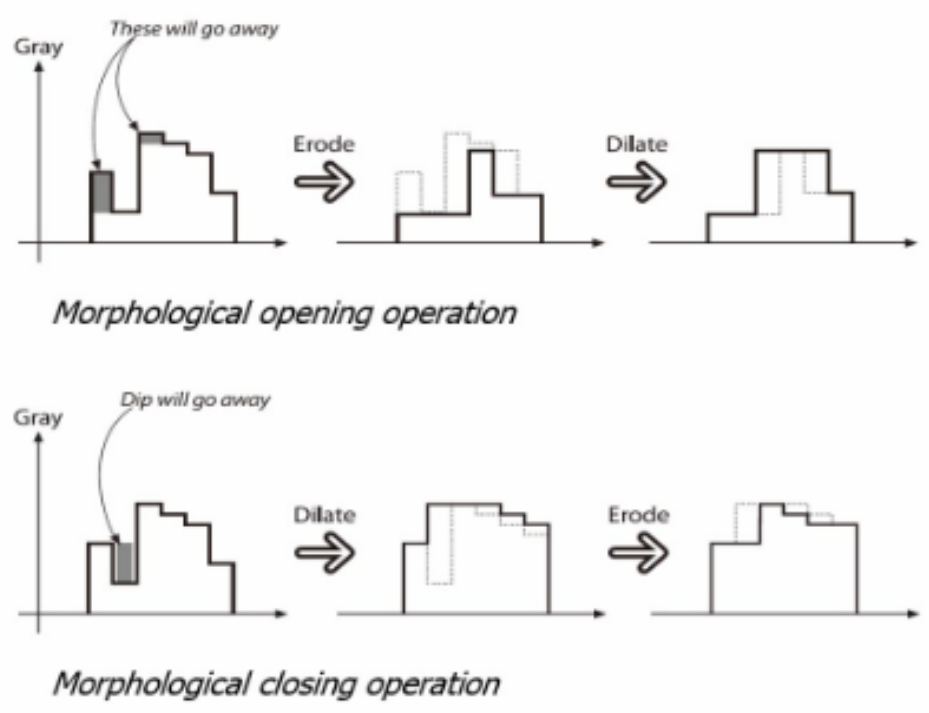
$$\mathrm{open}(\mathtt{src}) = \mathrm{dilate}(\mathrm{erode}(\mathtt{src}))$$

- Closing
  - Dilation 다음 Erosion
  - 작은 hole들을 닫는데 효과적임

$$\mathrm{close}(\mathtt{src}) = \mathrm{erode}(\mathrm{dilate}(\mathtt{src}))$$

- Gradient
  - Dilation과 Erosion의 차영상
  - 물체의 윤곽선을 찾는데 효과적임

$$\mathrm{morph\_grad}(\mathtt{src}) = \mathrm{dilate}(\mathtt{src}) - \mathrm{erode}(\mathtt{src})$$



https://opencv-python.readthedocs.io/en/latest/doc/12.imageMorphological/imageMorphological.html

Top Hat vs. Black Hat

- Top Hat
  - Input image와 Opening의 차영상

$$\mathrm{tophat}(\mathtt{src}) = \mathtt{src} - \mathrm{open}(\mathtt{src})$$

  - $\mathrm{open}(\mathtt{src})$: noise가 제거된 image
  - $\mathrm{tophat}(\mathtt{src})$: opening으로 제거된 noise

- Black Hat
  - Closing과 input image의의 차영상

$$\mathrm{blackhat}(\mathtt{src}) = \mathrm{close}(\mathtt{src}) - \mathtt{src}$$

  - $\mathrm{close}(\mathtt{src})$: 작은 holes가 제거된 image
  - $\mathrm{blackhat}(\mathtt{src})$: closing으로 제거된 holes

https://076923.github.io/posts/Python-opencv-27/

`cv.morphologyEx()`[1]

```
dst = cv.morphologyEx(src, op, kernel[, dst[, anchor[, iterations[, borderType[,
borderValue]]]]])
```

- Advanced morphology transformations을 수행한다.
  - `src`: input image; the number of channels can be arbitrary, but the depth should be one of `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or `CV_64F`)
  - `dst`: output array of the same size and type as `src`
  - `op`: type of a morphological operation
  - `kernel`: structuring element used for dilation
  - `anchor`: anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1, -1) means that the anchor is at the kernel center
  - `iterations`: number of times erosion is applied
  - `borderType`: pixel extrapolation method
  - `borderValue`: border value in case of a constant border

1. https://docs.opencv.org/4.4.0/d4/d86/group__imgproc__filter.html#ga67493776e3ad1a3df63883829375201f

`MorphTypes`[1]

`MorphTypes`	Operation
`cv.MORPH_ERODE`	erode
`cv.MORPH_DILATE`	dilate
`cv.MORPH_OPEN`	$\texttt{dst} = \mathrm{open}(\texttt{src}, \texttt{element}) = \mathrm{dilate}(\mathrm{erode}(\texttt{src}, \texttt{element}))$
`cv.MORPH_CLOSE`	$\texttt{dst} = \mathrm{close}(\texttt{src}, \texttt{element}) = \mathrm{erode}(\mathrm{dilate}(\texttt{src}, \texttt{element}))$
`cv.MORPH_GRADIENT`	$\texttt{dst} = \mathrm{morph\_grad}(\texttt{src}, \texttt{element}) = \mathrm{dilate}(\texttt{src}, \texttt{element}) - \mathrm{erode}(\texttt{src}, \texttt{element})$
`cv.MORPH_TOPHAT`	$\texttt{dst} = \mathrm{tophat}(\texttt{src}, \texttt{element}) = \texttt{src} - \mathrm{open}(\texttt{src}, \texttt{element})$
`cv.MORPH_BLACKHAT`	$\texttt{dst} = \mathrm{blackhat}(\texttt{src}, \texttt{element}) = \mathrm{close}(\texttt{src}, \texttt{element}) - \texttt{src}$
`cv.MORPH_HITMISS`	"hit or miss"; Only supported for CV_8UC1 binary images.

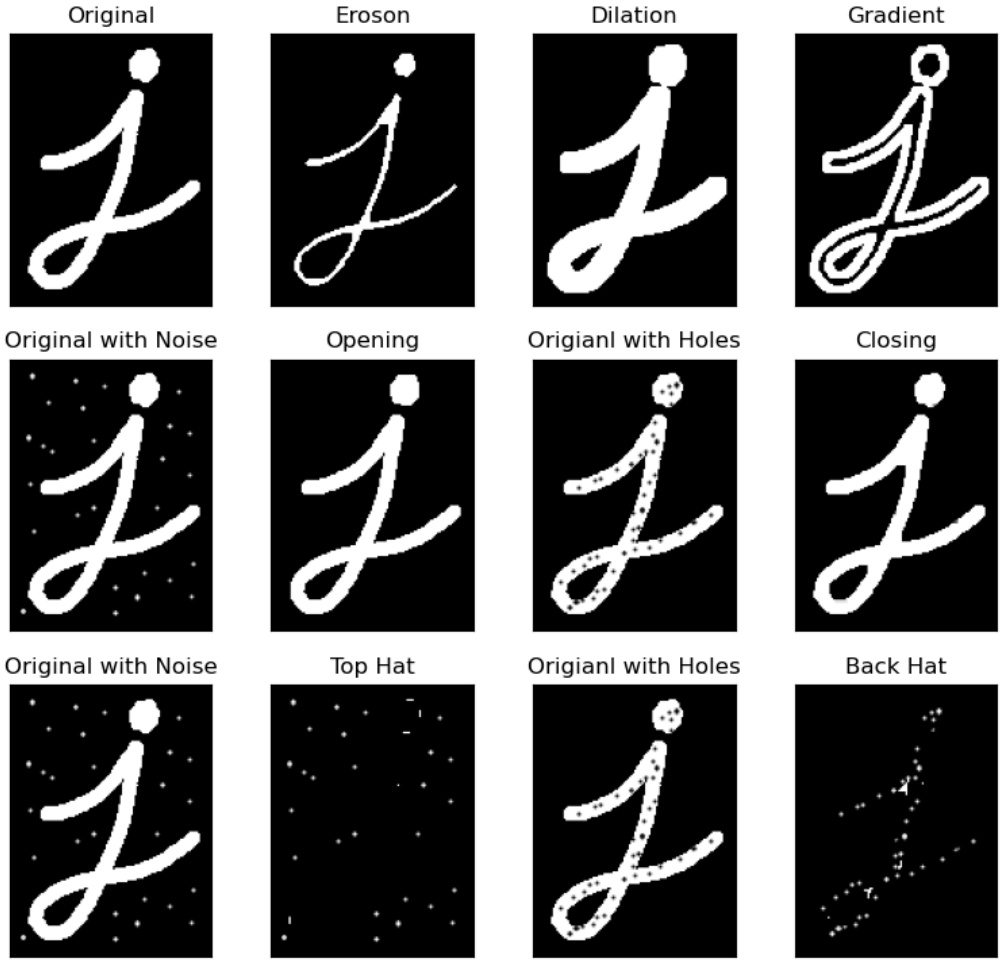
1. https://docs.opencv.org/4.4.0/d4/d86/group__imgproc__filter.html#ga7be549266bad7b2e6a04db49827f9f32

Morphological Transformations: Code

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load grayscale images
img_orig  = cv2.imread('j.png',       0)
img_noise = cv2.imread('j_noise.png', 0)
img_holes = cv2.imread('j_holes.png', 0)

# Kernel for erosion and dilation
kernel = np.ones((5, 5), np.uint8)

# Erosion
res_erosion = cv2.erode(img_orig, kernel, iterations=1)

# Dilation
res_dilation = cv2.dilate(img_orig, kernel, iterations=1)

# Morphological gradient
res_gradient = cv2.morphologyEx(img_orig, cv2.MORPH_GRADIENT, kernel)
```

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering

Morphological Transformations: Results



https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

Hit-or-Miss

Hit-or-Miss

- Binary image에서 지정된 패턴을 찾는 방법.
- Image A에 대해서 첫번째 structuring element인 B_1에 맞고(hit) 두번째 structuring element인 B_2의 모양에는 맞지 않는(miss) 픽셀들을 찾아준다.

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

  1. Erode image A with structuring element B_1.
  2. Erode the complement of image A (A^c) with structuring element B_2.
  3. AND results from step 1 and step 2.

- Hit-or-Miss 혹은 Hit-and-Miss라고 불린다.

https://docs.opencv.org/master/db/d06/tutorial_hitOrMiss.html

Hit-or-Miss

- Structuring Elements B_1 and B_2 can be combined into a single element B.

0	1	0
1	0	1
0	1	0

kernel to 'hit'

0	0	0
0	1	0
0	0	0

kernel to 'miss'

0	1	0
1	-1	1
0	1	0

combined kernel

https://docs.opencv.org/master/db/d06/tutorial_hitOrMiss.html

Hit-or-Miss

- Input binary image

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

- Output binary image

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

https://docs.opencv.org/master/db/d06/tutorial_hitOrMiss.html

Hit-or-Miss

- Input binary image

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

- Output binary image

0	-1	-1
1	1	-1
0	1	0

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

https://docs.opencv.org/master/db/d06/tutorial_hitOrMiss.html

Hit-or-Miss

- Input binary image

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

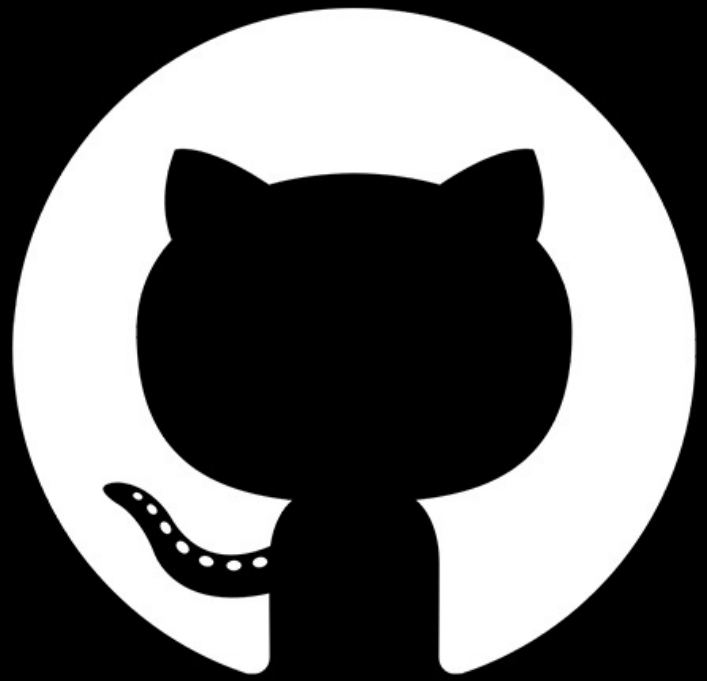
- Output binary image

-1	-1	0
-1	1	0
-1	-1	0

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

https://docs.opencv.org/master/db/d06/tutorial_hitOrMiss.html

Push Code to GitHub



References

References

- OpenCV Python Tutorials
  - Core Operations
    - Basic Operations on Images
    - Arithmetic Operations on Images
  - Image Processing
    - Image Thresholding
    - Smoothing Images
    - Morphological Transformations