KMITL BEYOND THE LIMIT

Computer Visions

01416500
Course Code

Image Reconstruction

Overview

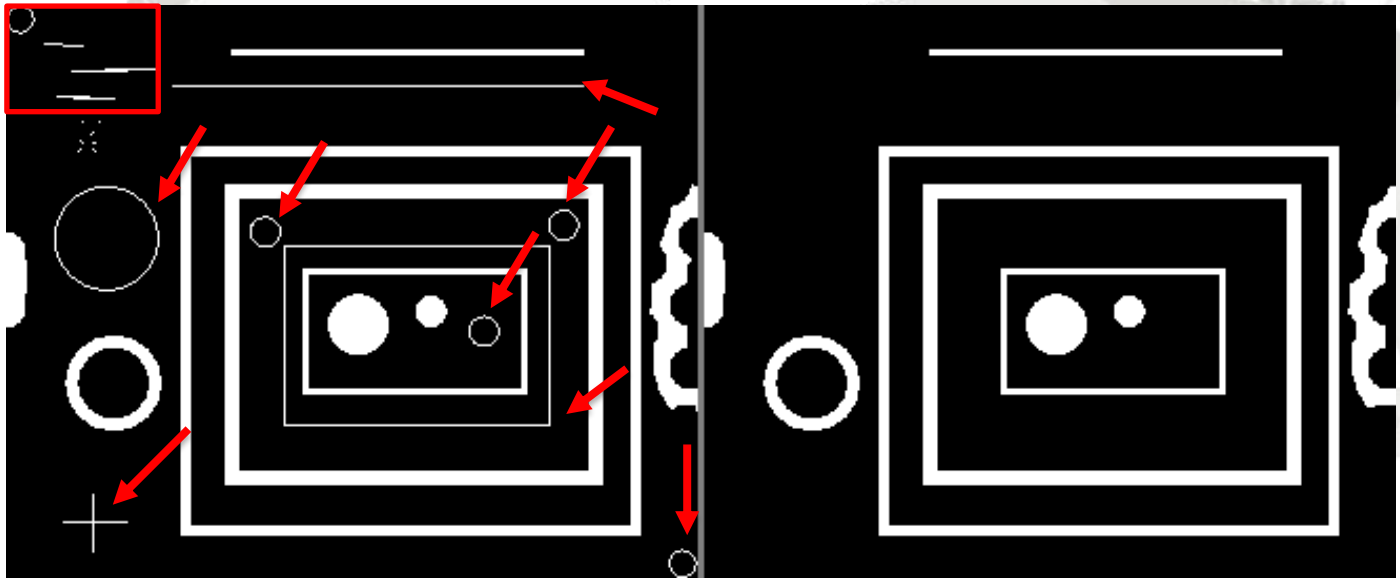
1. Morphological reconstruction
2. Hough Line Transforms
3. Hough Circle Transforms

Asst. Prof. Dr. Anakkapon Saenthon
King Mongkut's Institute of Technology Ladkrabang

1. Morphological Theory

Morphology is a broad set of image processing operations that process images based on shapes. ***Morphological operations (kernel)*** apply **a structuring element** to an input image (***gray image***), creating an output image of the same size.

Example **Applied the opening operator**

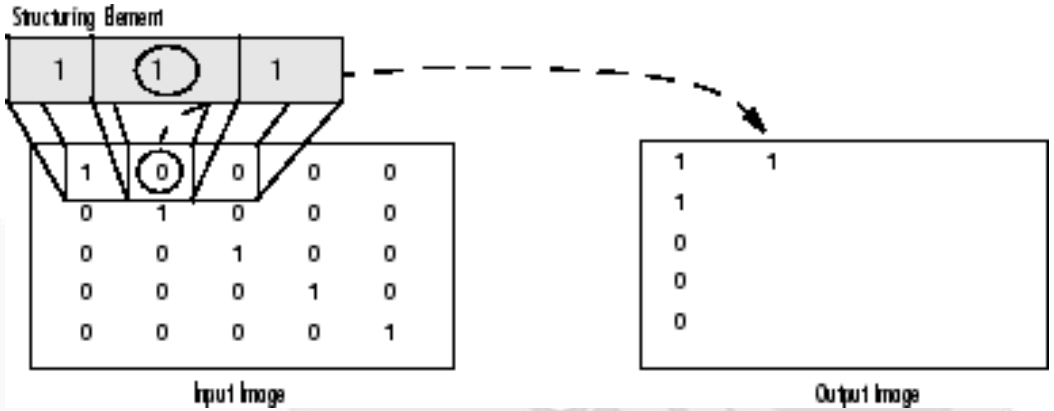


input image **output image**

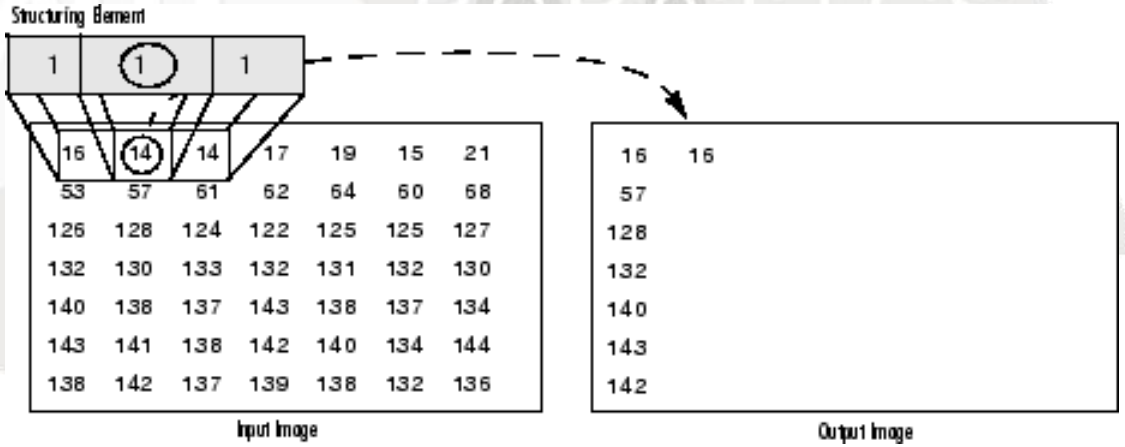
1. Morphological Theory

1. Dilation
2. Erosion
3. Opening
4. Closing
5. Gradient
6. Top Hat
7. Black Hat

Morphological *Dilation* of a Binary Image



Morphological *Dilation* of a Grayscale Image



1. Dilation (increase border)

A pixel element is “1” if **at least one pixel** under **the kernel** is “1”.

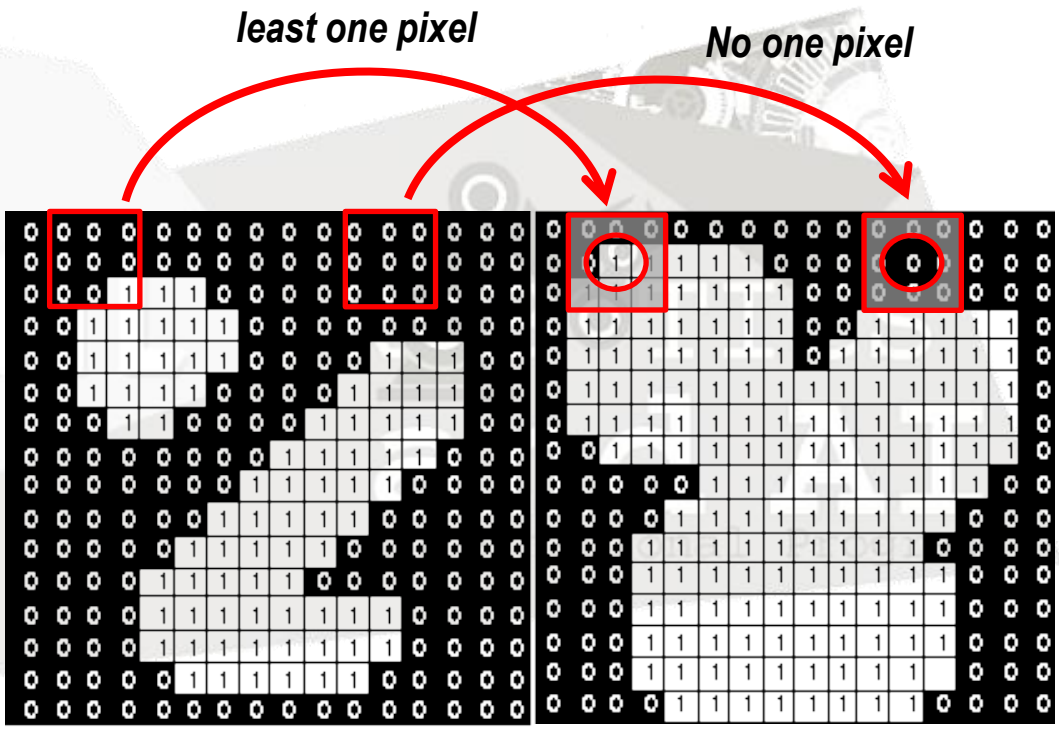
So it ***increases*** the white region in the image or size of foreground object increases.

Dilation

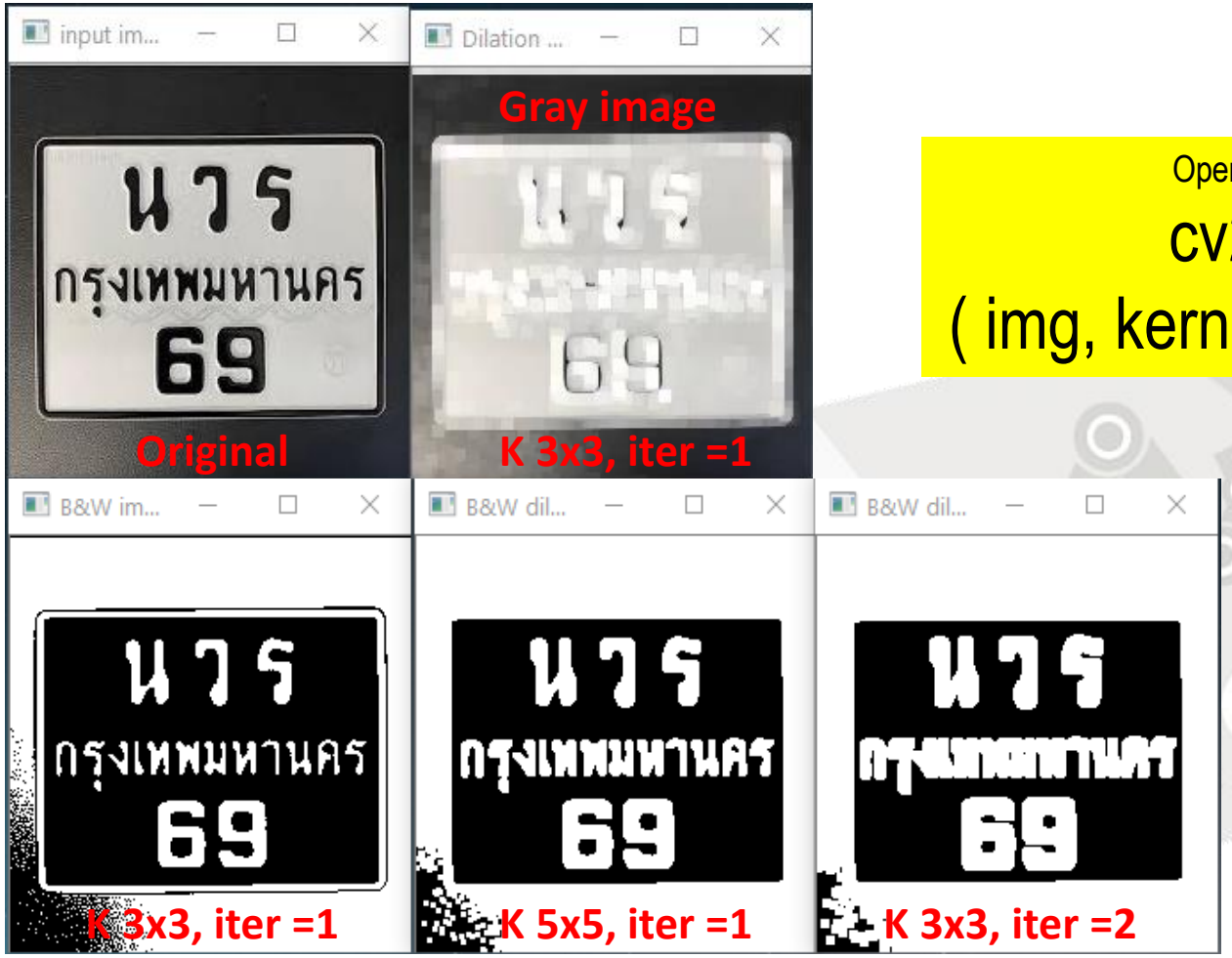
kernel size [3x3]

1	1	1
1	1	1
1	1	1

Example dilation



1. Dilation (increase border)



OpenCV function :

cv2.dilate

(img, kernel, iterations=1)

2. Erosion (decrease border)

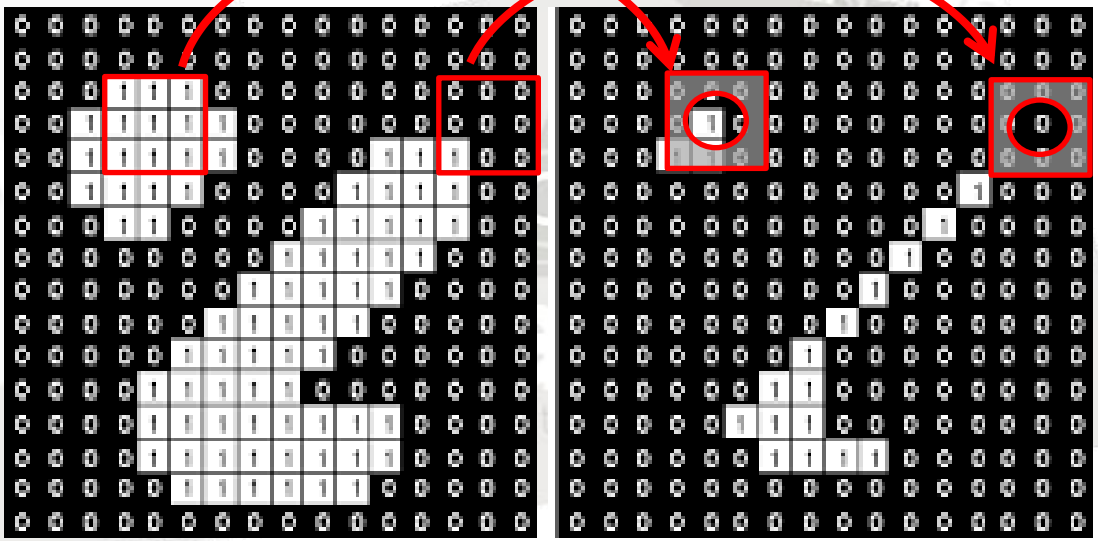
The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only **if all the pixels** under the kernel is 1, otherwise it is eroded (made to zero).

Erosion

kernel size [3x3]

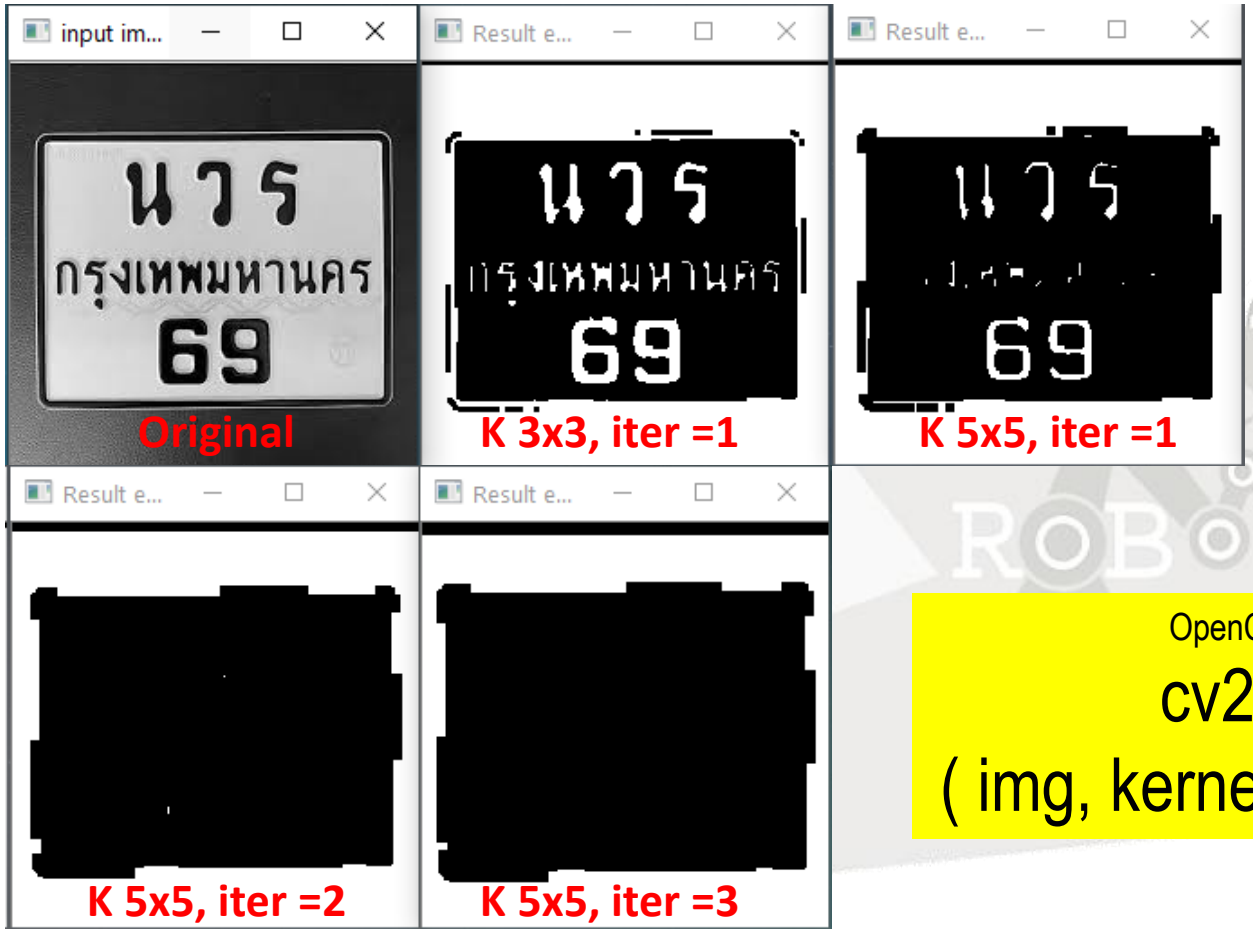
1	1	1
1	1	1
1	1	1

Erosion



Input image output image

2. Erosion (decrease border)



OpenCV function :

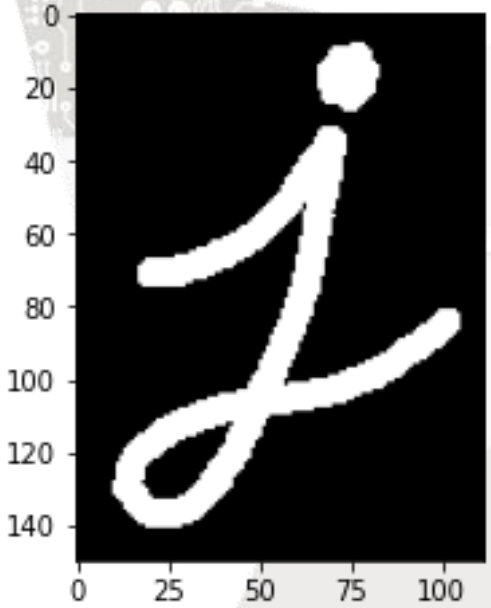
cv2.erode

(img, kernel, iterations=1)

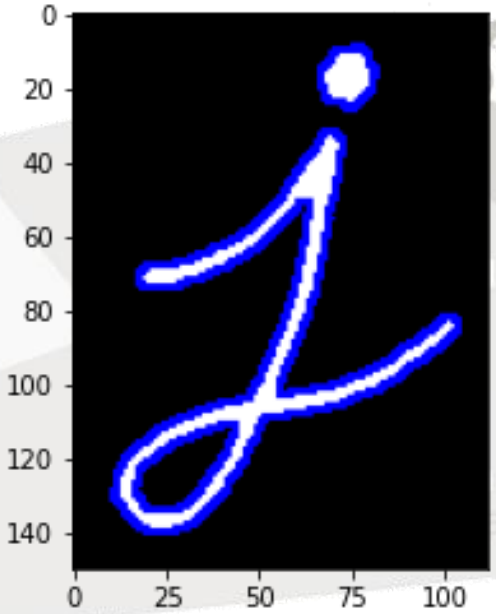
Quiz#1 (15 min)

1. Let's show the blue border inside object.

Original



Output



***** Border size 2 pixels**

3. Opening (reduce noise)

Opening is just another name of ***erosion followed by dilation***

Opening

kernel size [3x3]

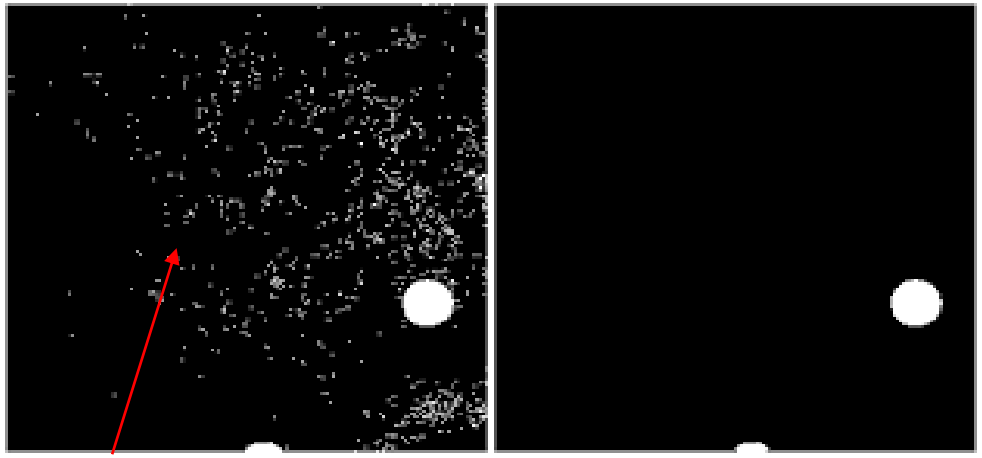
1	1	1
1	1	1
1	1	1

OpenCV function :

cv2.morphologyEx()
cv2.MORPH_OPEN

Example Opening

Input image output image



noise

4. Closing (Fill holes)

Closing is reverse of Opening, **Dilation followed by Erosion**. It is useful in closing small holes inside the foreground objects, or small black points on the object.

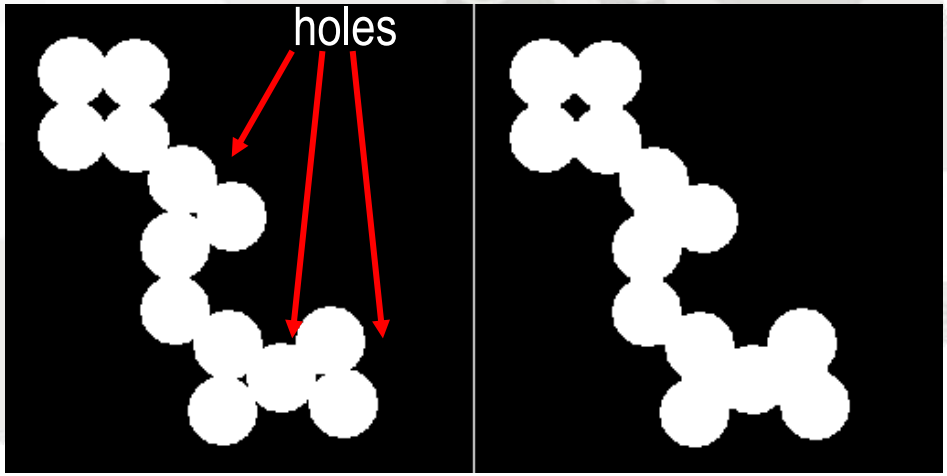
OpenCV function :

cv2.morphologyEx()
cv2.MORPH_CLOSE

Closing

kernel size [3x3]

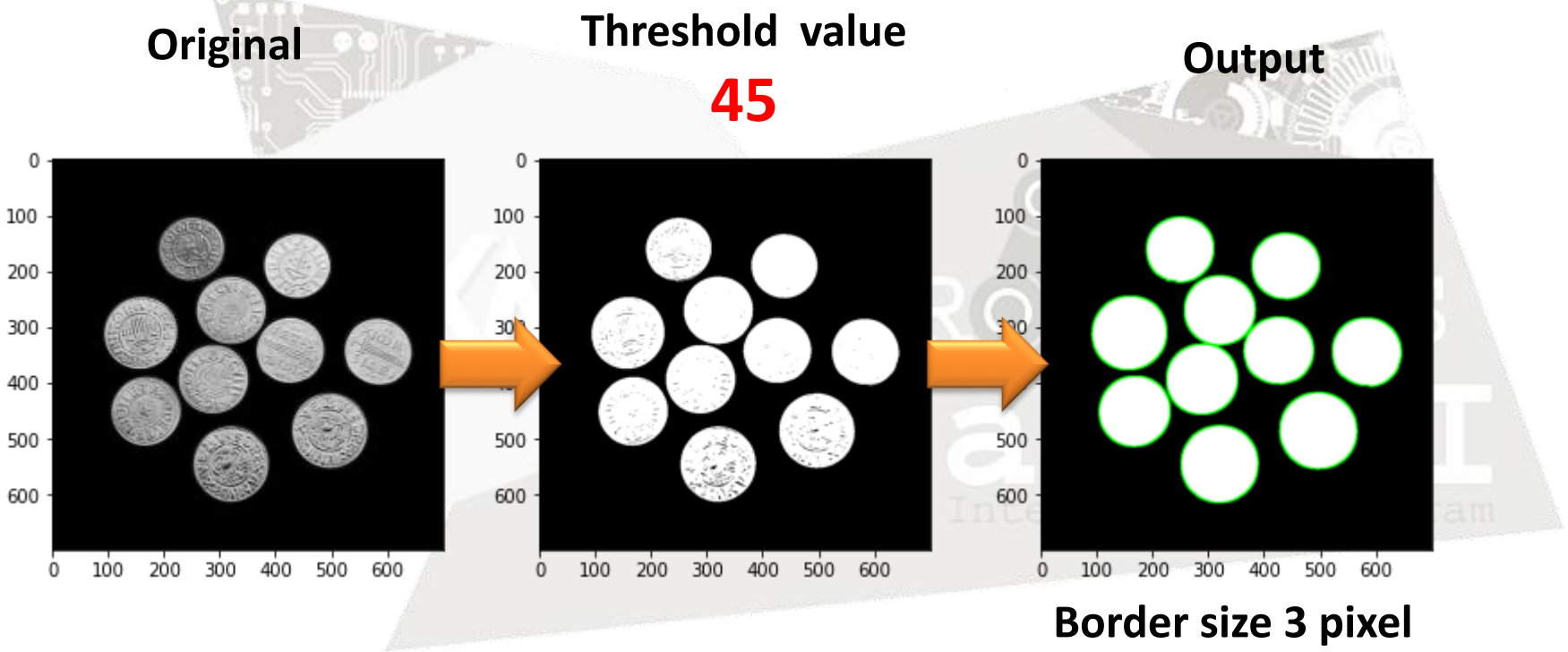
1	1	1
1	1	1
1	1	1



Example Closing

Quiz#2 (15 min)

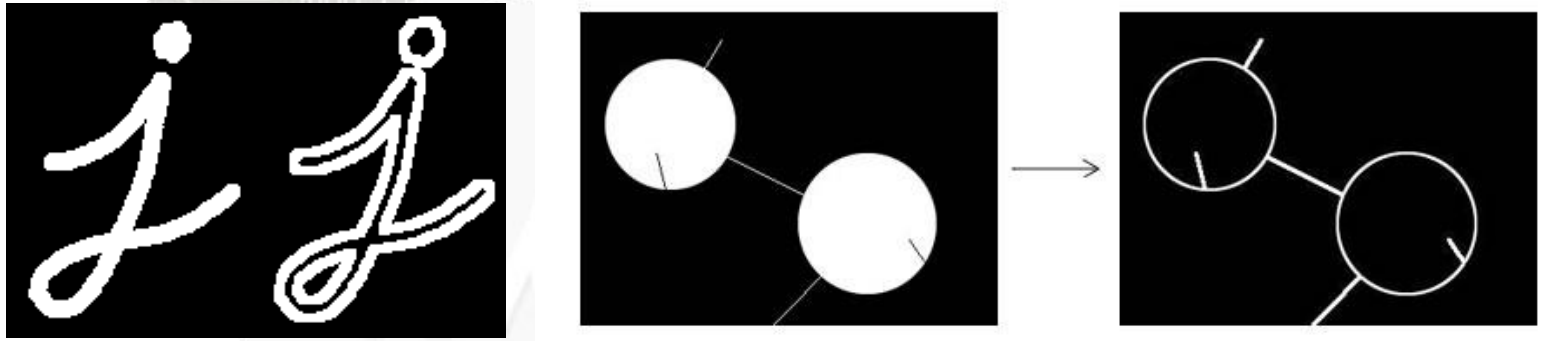
1. Let's show the code for draw the green border outside object.



5. Morphological Gradient

It is the difference between dilation and erosion of an image.

Below example is done for a ***9x9 kernel***



Morphological Gradient

OpenCV function :

cv2.morphologyEx()
cv2.MORPH_CLOSE

5. Morphological Gradient

Example Gradient

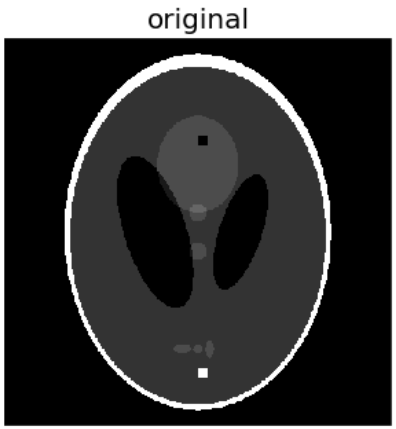
Original | K3x3, iter =1 | K5x5, iter =1 | K3x3, iter=2 | K5x5, iter=2

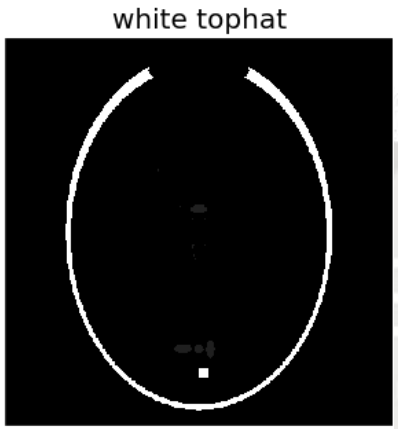
```
img = cv2.imread("dataset/69.jpg",0)
# threshold
_,bw = cv2.threshold(img,180,255,cv2.THRESH_BINARY)
gra1 = cv2.morphologyEx(bw,cv2.MORPH_GRADIENT,np.ones((3,3),np.uint8))
gra2 = cv2.morphologyEx(bw,cv2.MORPH_GRADIENT,np.ones((5,5),np.uint8))
gra3 = cv2.morphologyEx(bw,cv2.MORPH_GRADIENT,np.ones((3,3),np.uint8),iterations=2)
gra4 = cv2.morphologyEx(bw,cv2.MORPH_GRADIENT,np.ones((5,5),np.uint8),iterations=2)
```

6. Top Hat (noise object)

It is the **difference** between **input image** and **Opening** of the image..

Example is done for a ***9x9 kernel***





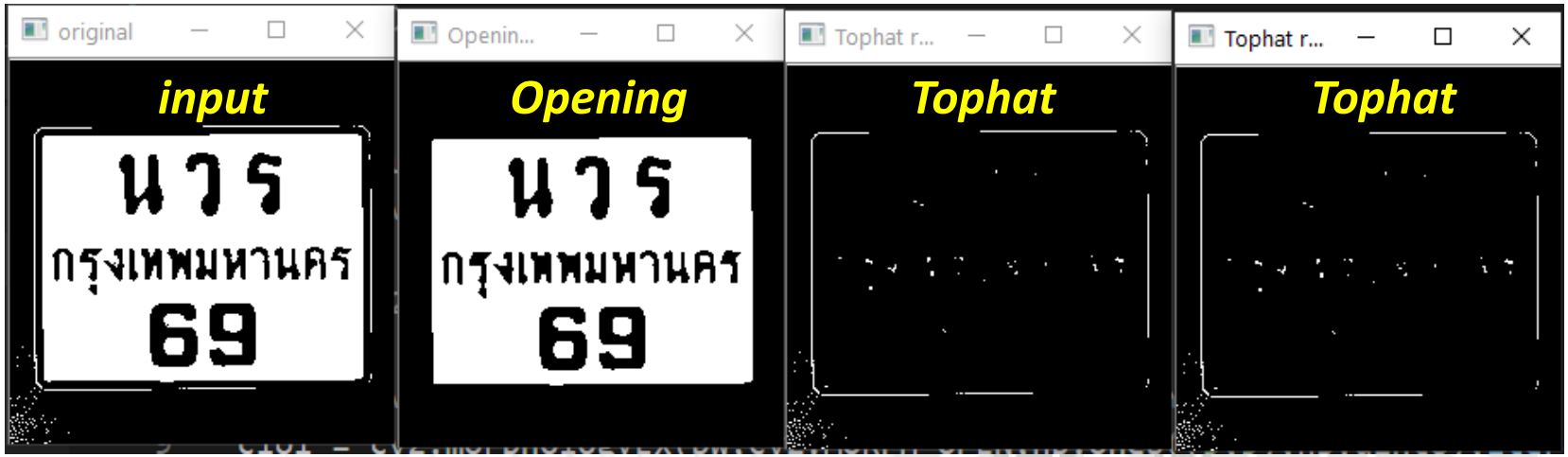
OpenCV function :

cv2.morphologyEx()
cv2.MORPH_TOPHAT

6. Top Hat (noise object)

Example Tophat

Input-opening ***morphologyEx***

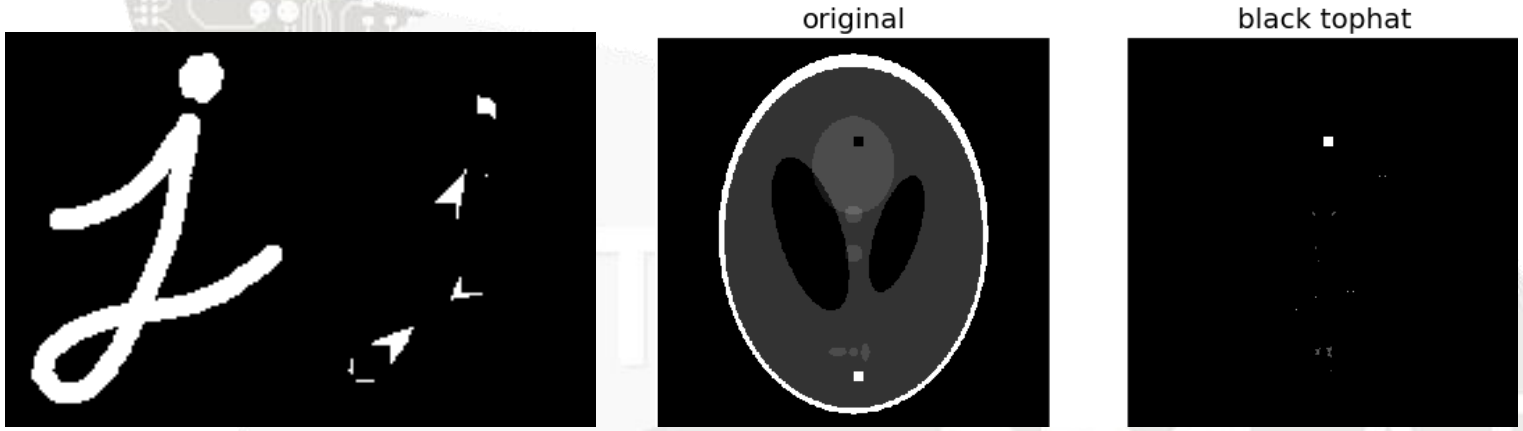



```
img = cv2.imread("dataset/69.jpg",0)
# tophat  = input - closing
_,bw = cv2.threshold(img,150,255,cv2.THRESH_BINARY)
clo1 = cv2.morphologyEx(bw,cv2.MORPH_OPEN,np.ones((3,3),np.uint8),iterations=1)

tophat = cv2.subtract(bw,clo1)
tophat2 = cv2.morphologyEx(bw,cv2.MORPH_TOPHAT,np.ones((3,3),np.uint8),iterations=1)
```

7. Black Hat (Hole object)

It is the **difference** between the **closing** of the input image and **input image**.

Example is done for a ***9x9 kernel***



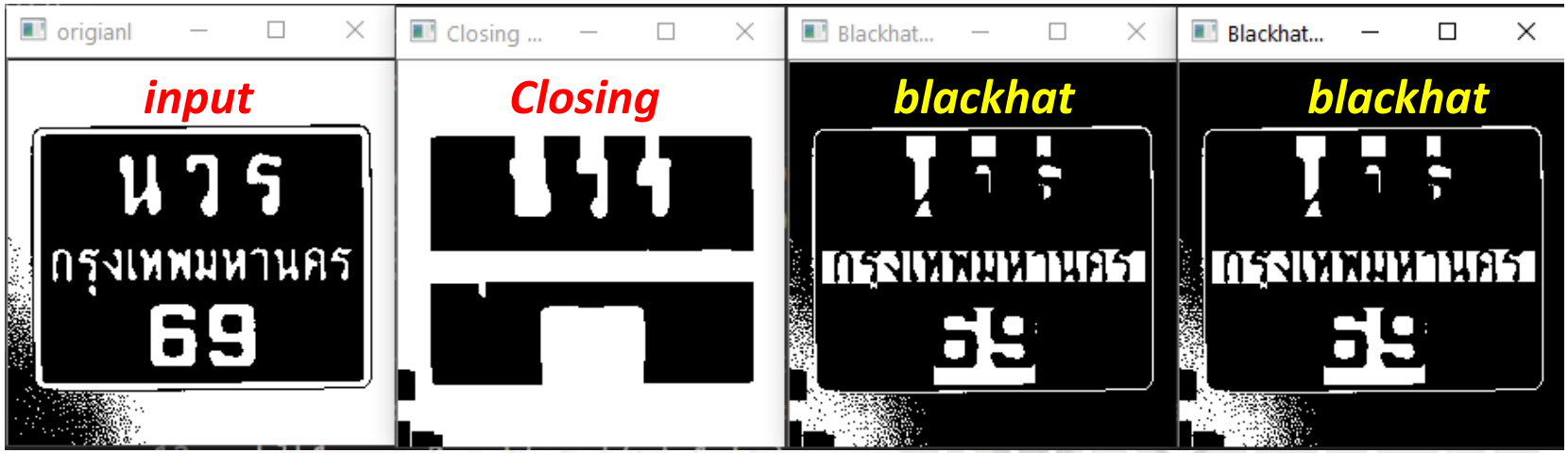
OpenCV function :

cv2.morphologyEx()
cv2.MORPH_BLACKHAT

Example Blackhat

Closing-Input

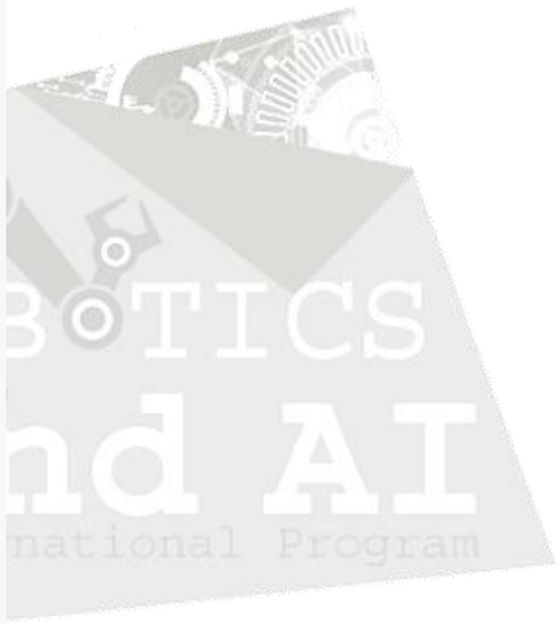
morphologyEx




```
img = cv2.imread("dataset/69.jpg",0)
_,bw = cv2.threshold(img,100,255,cv2.THRESH_BINARY_INV)

# blackhat = closing - input
cls1 = cv2.morphologyEx(bw,cv2.MORPH_CLOSE,np.ones((5,5),np.uint8),iterations=3)
blk1 = cv2.subtract(cls1,bw)
blk2 = cv2.morphologyEx(bw,cv2.MORPH_BLACKHAT,np.ones((5,5),np.uint8),iterations=3)
```

II. Structuring Element

```
# Rectangular Kernel
>>> cv2.getStructuringElement(cv2.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(cv2.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(cv2.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)
```



KMITL BEYOND THE LIMIT

COMPUTER VISIONS

01416500
Course Code

Image Reconstruction

2. Hough Line Transforms

Asst. Prof. Dr. Anakkapon Saenthon
King Mongkut's Institute of Technology Ladkrabang

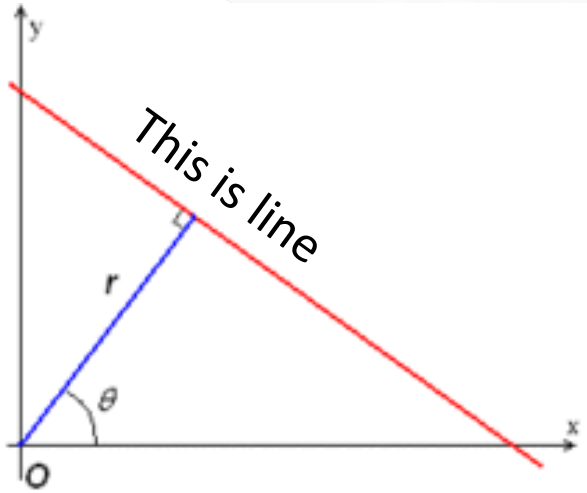
Hough Line Transforms

The **Hough transform** is a feature extraction technique used in image analysis, computer vision, and digital image processing. he purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure.

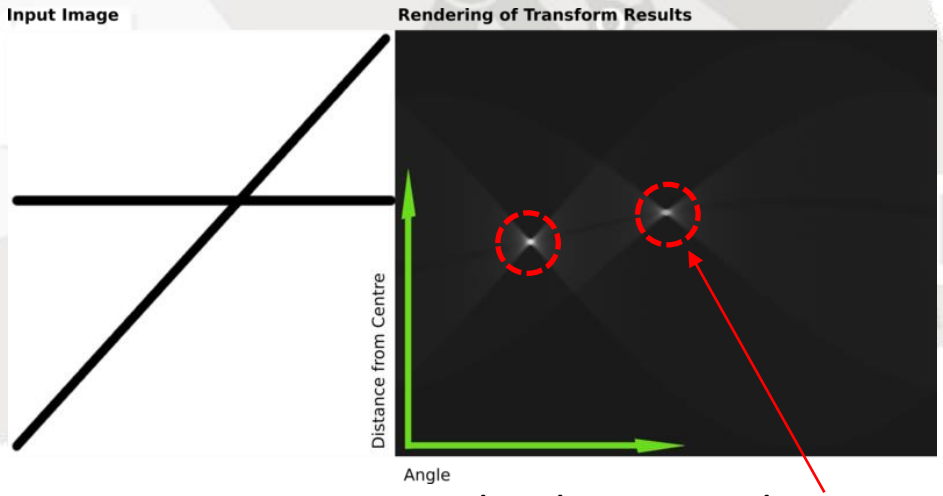
Pixel
Coordinate (x,y)

→ Hough transform →

Polar coordinate system
rho, theta



All pixel have same rho,theta



A peak value mean distance of line

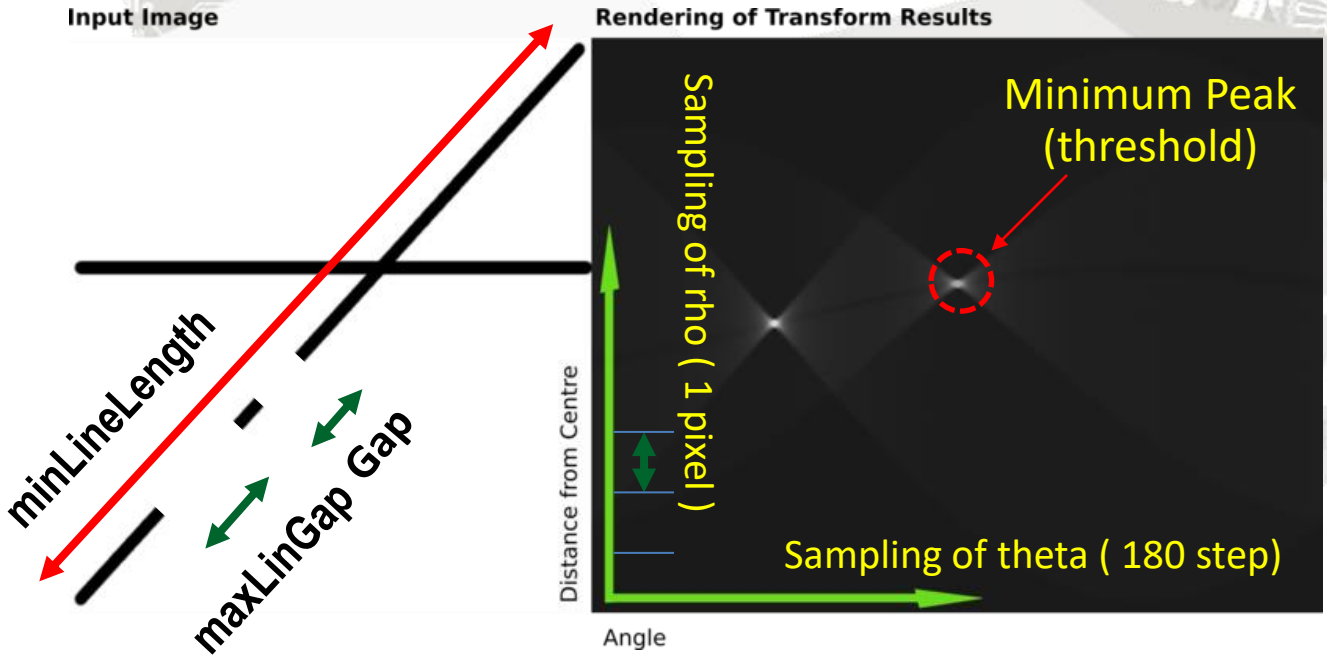
Hough Line Transforms

Hough line Parameters

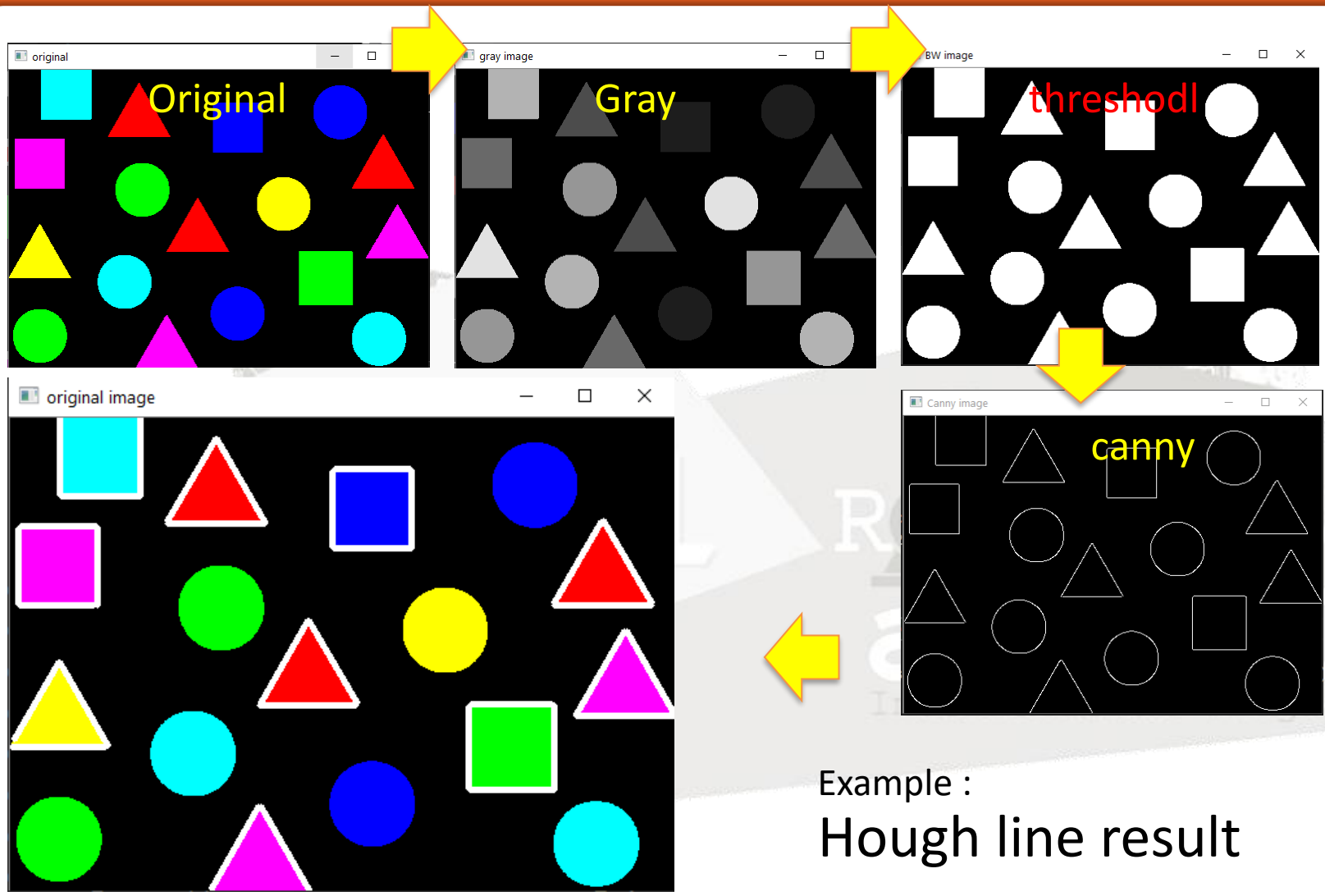
OpenCV function :

cv2.HoughLinesP

(dst, rho, theta, threshold, minLineLength, maxLineGap)



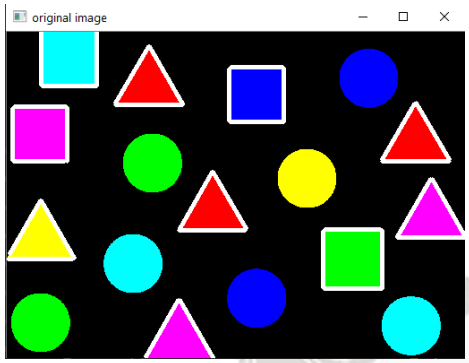
Hough Line Transforms



Example :

Hough line result

Hough Line Transforms



OpenCV function :

cv2.HoughLinesP()

```
img = cv2.imread("dataset/colorobject.png")
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
# threshold
_,thr = cv2.threshold(gray,10,255,cv2.THRESH_BINARY)
# Canny edge detection
res = cv2.Canny(thr,10,200)
# hough line
lines = cv2.HoughLinesP(res,1,np.pi/360,50,minLineLength=20,maxLineGap=10)
print(len(lines))

for i in range(len(lines)):
    for x1,y1,x2,y2 in lines[i]:
        cv2.line(img,(x1,y1),(x2,y2),(255,255,255),thickness=3) # change color to white
```

KMITL BEYOND THE LIMIT

COMPUTER VISIONS

01416500
Course Code

KMITL ROBOTICS and AI International Program

3. Hough Circle Transforms

Asst. Prof. Dr. Anakkapon Saenthon
King Mongkut's Institute of Technology Ladkrabang

Hough Circle Transforms

In a two-dimensional space, a circle can be described by:

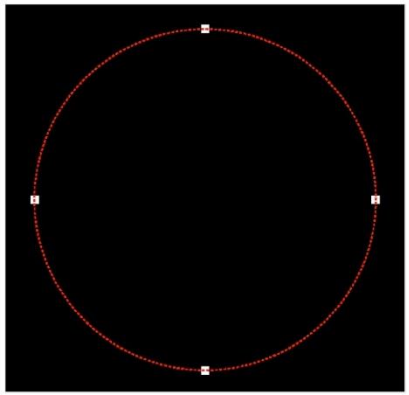
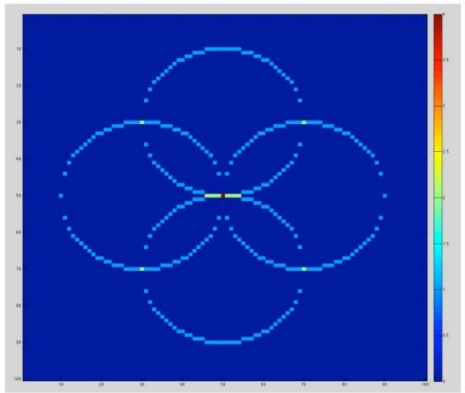
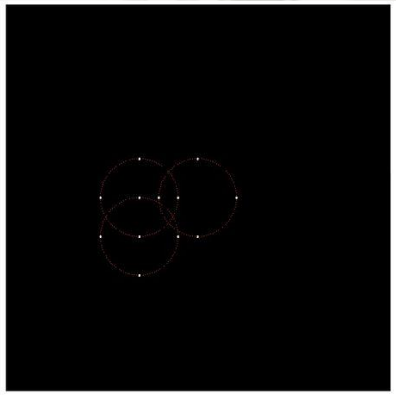
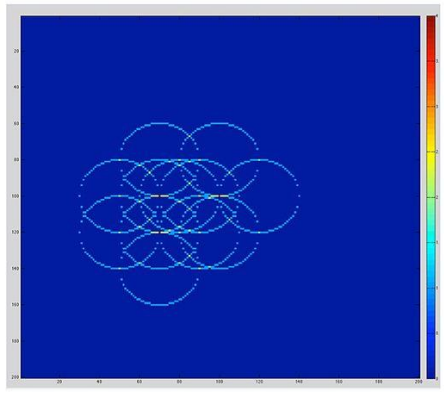
$$(x-a)^2 + (y-b)^2 = r^2$$

Pixel
Coordinate (x,y)

→ Hough Circle transform →

a,b,r

In practice, an accumulator matrix is introduced to find the intersection point in the parameter space

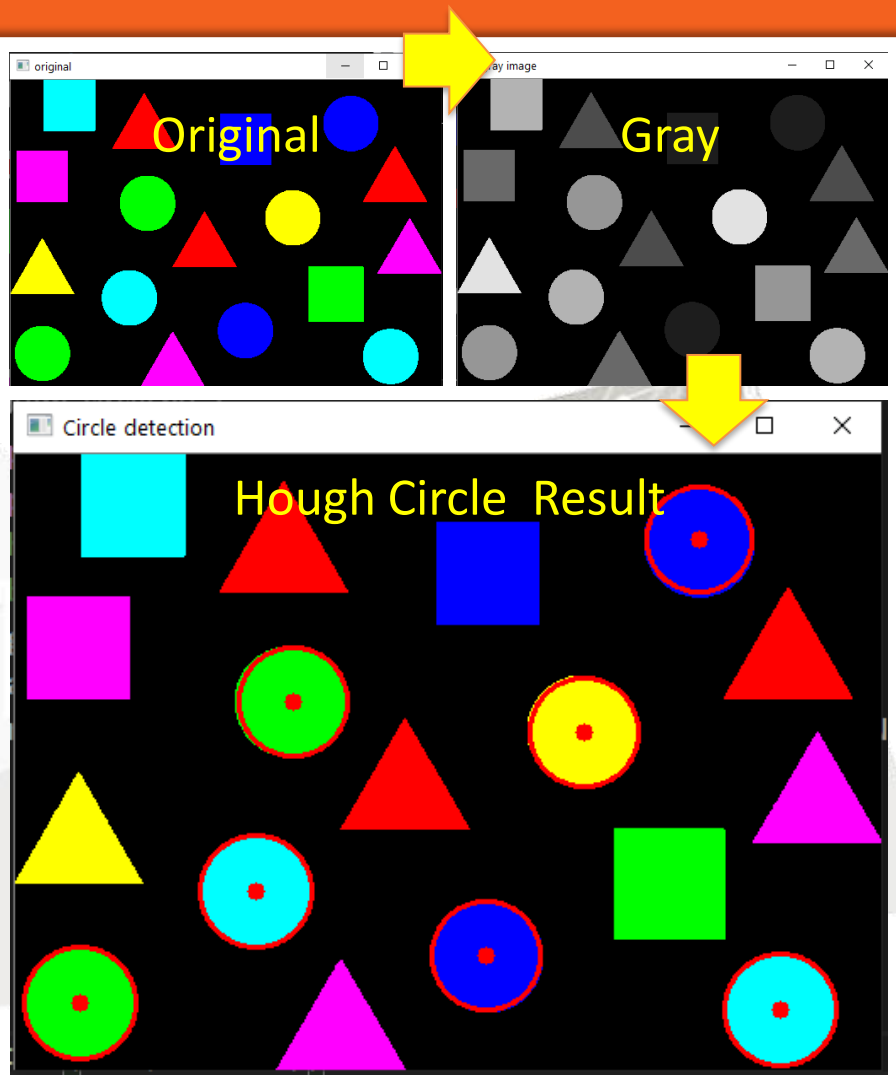





Hough Circle Transforms

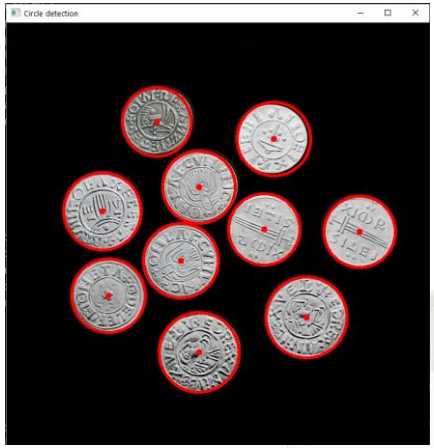
Example

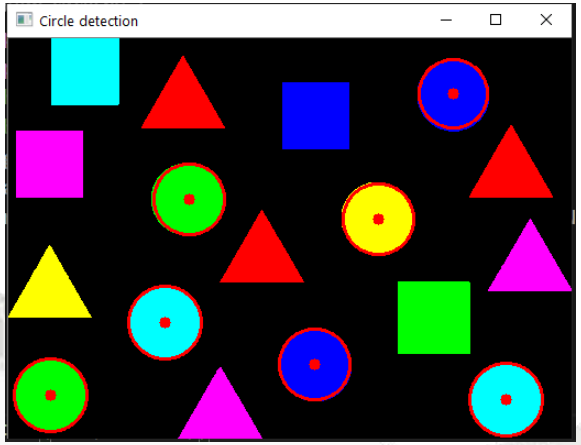
Circle detection 3 step :

1. BGR
2. Convert to Gray
3. HoughCircles()



Hough Circle Transforms





OpenCV function :

cv2.HoughCircles()

```
circles = cv2.HoughCircles(gray,cv2.HOUGH_GRADIENT,
                           dp=1.0,minDist=20,
                           param1=10,param2=10,
                           minRadius=30,maxRadius=40)
print(len(circles))
print("Befor around : ",circles[0])
circles = np.uint16(np.around(circles))
print("After around : ",circles[0])
for i in circles[0]:
    # index 0 : x-center , index 1 : y-center, index 2 : radius
    cv2.circle(img,(i[0],i[1]),i[2],(0,0,255),thickness=2)
    cv2.circle(img,(i[0],i[1]),5,(0,0,255),thickness=-1) # thickness -1 mean fill color
```

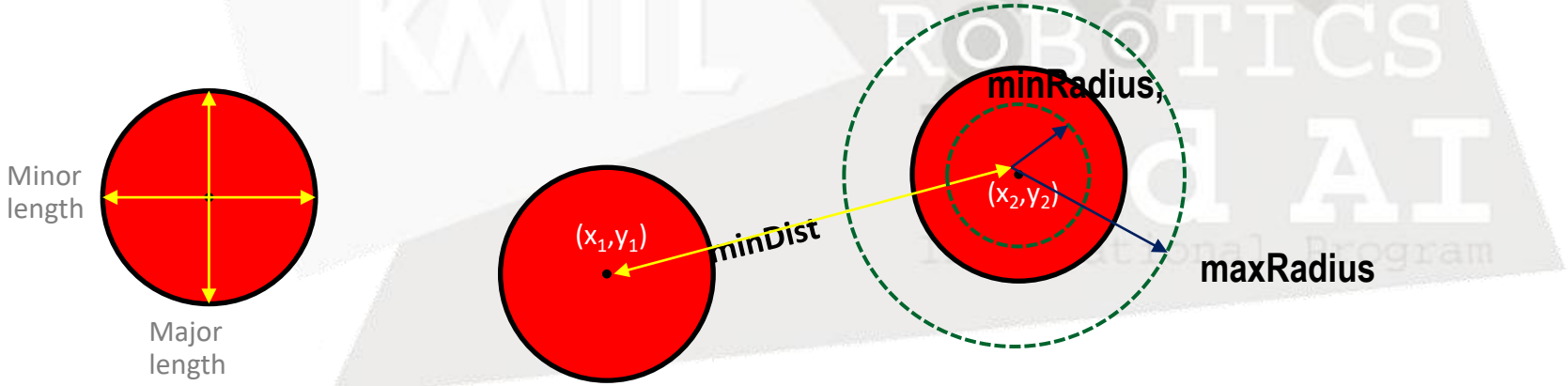
Hough Circle Transforms

OpenCV function :

```
cv2. cv2.HoughCircles(
gray, cv2.HOUGH_GRADIENT,
dp, minDist, param1, param2, minRadius, maxRadius)
```

$$dp = \frac{Major}{Minor}$$

param_1 = 200: Upper threshold for the internal Canny edge detector.
param_2 = 100*: Threshold for center detection.



KMITL BEYOND THE LIMIT

COMPUTER VISIONS

01416500
Course Code

Image Reconstruction

Q & A

Next : Segmentation

Asst. Prof. Dr. Anakkapon Saenthon
King Mongkut's Institute of Technology Ladkrabang