Morphological Operation

Morphological Operation

- Morphological operation as a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries and skeletons
- Morphological operations are
  - Reflection
  - Translation
  - Erosion
  - Dilation
  - Opening
  - Closing
  - Hit-or-Miss Transform
  - Thinning
  - Thickening

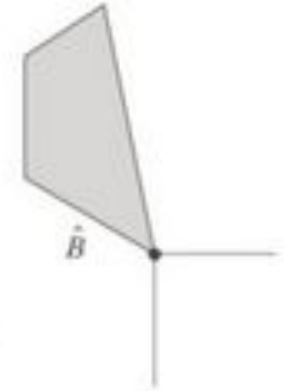
Preliminaries

- Binary image can be represented as a set
- Sets in mathematical morphology represent objects in an image.
- For Example:
  - Set of all white pixels in a binary image is a complete morphological description of the image

Reflection

- Reflection - 180 degree of rotation of an object with respect to the origin

Ref(B) = { - b | b ε B}

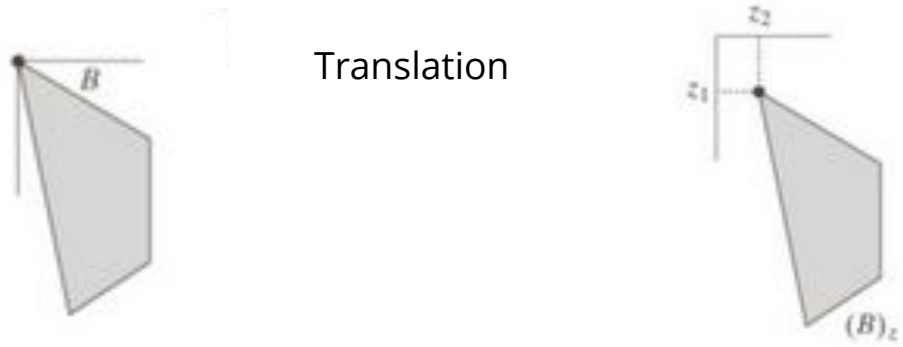


Reflection

Translation

- The translation of a set B by point $z = (z_1, z_2)$ denoted $(B)_z$, is defined as

  $(B)_z = \{ b + z \mid b \in B \}$



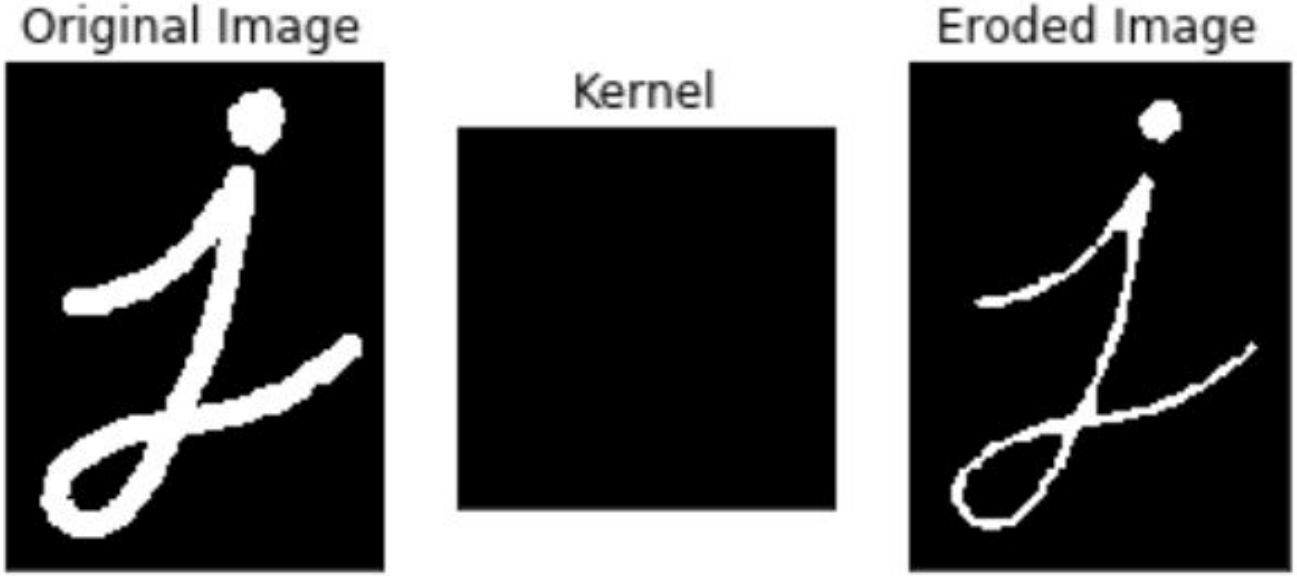
Erosion

The erosion of A by B, denoted $A \ominus B$, is defined by

$A \ominus B = \{ z \mid (B)_z \subseteq A \}$

- The erosion of A by B is the set of all points z such that B, translated by z, is contained in A.
- B is structuring element / Kernel

Erosion



Cont..

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('j.png',0)

kernel = np.ones((5,5),np.uint8)

erosion = cv2.erode(img,kernel,iterations = 1)
```

Cont..

```
plt.subplot(131),plt.imshow(img,cmap = 'gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(132),plt.imshow(kernel,cmap = 'gray')

plt.title('Kernel'), plt.xticks([]), plt.yticks([])

plt.subplot(133),plt.imshow(erosion,cmap = 'gray')

plt.title('Eroded Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

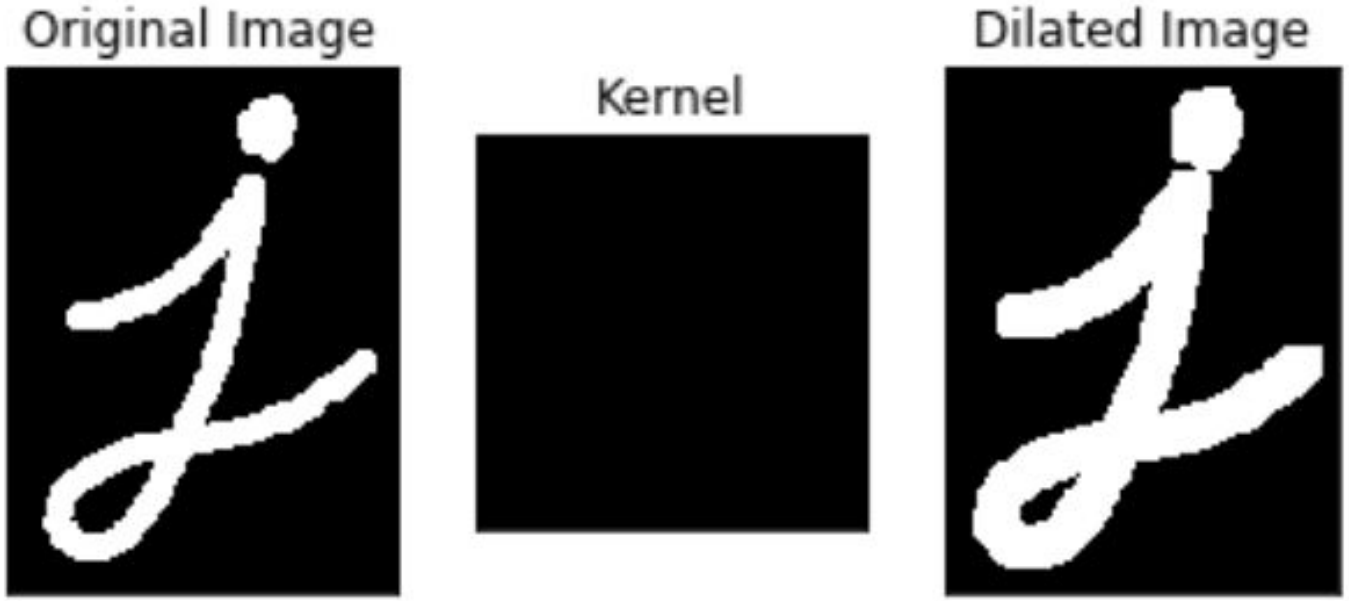
Dilation

The dilation of A by B, denoted $A \oplus B$, is defined by

$A \oplus B = \{ z \mid (Ref(B))_z \cap A \neq \varnothing \}$

- The dilation of A by B is a set of all displacements z such that Ref(B) and A overlap by at least one element

Dilation



Cont..

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('j.png',0)

kernel = np.ones((5,5),np.uint8)

dilation = cv2.dilate(img,kernel,iterations = 1)
```

Cont..

```
plt.subplot(131),plt.imshow(img,cmap = 'gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(132),plt.imshow(kernel,cmap = 'gray')

plt.title('Kernel'), plt.xticks([]), plt.yticks([])

plt.subplot(133),plt.imshow(dilation,cmap = 'gray')

plt.title('Dilated Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

Opening

The opening of a set A by the kernel B, denoted $A \circ B$, is defined as

$$A \circ B = (A \ominus B) \oplus B$$

- Thus opening A by B is the erosion of A by B, followed by a dilation of the result by B

Opening

Original Image

Kernel

erosion followed by dilation in Image

Cont..

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('jforopening.png',0)

kernel = np.ones((5,5),np.uint8)

opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
```

Cont..

```
plt.subplot(131),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(kernel,cmap = 'gray')
plt.title('Kernel'), plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(opening,cmap = 'gray')
plt.title('erosion followed by dilation in Image'), plt.xticks([]), plt.yticks([])
plt.show()
```

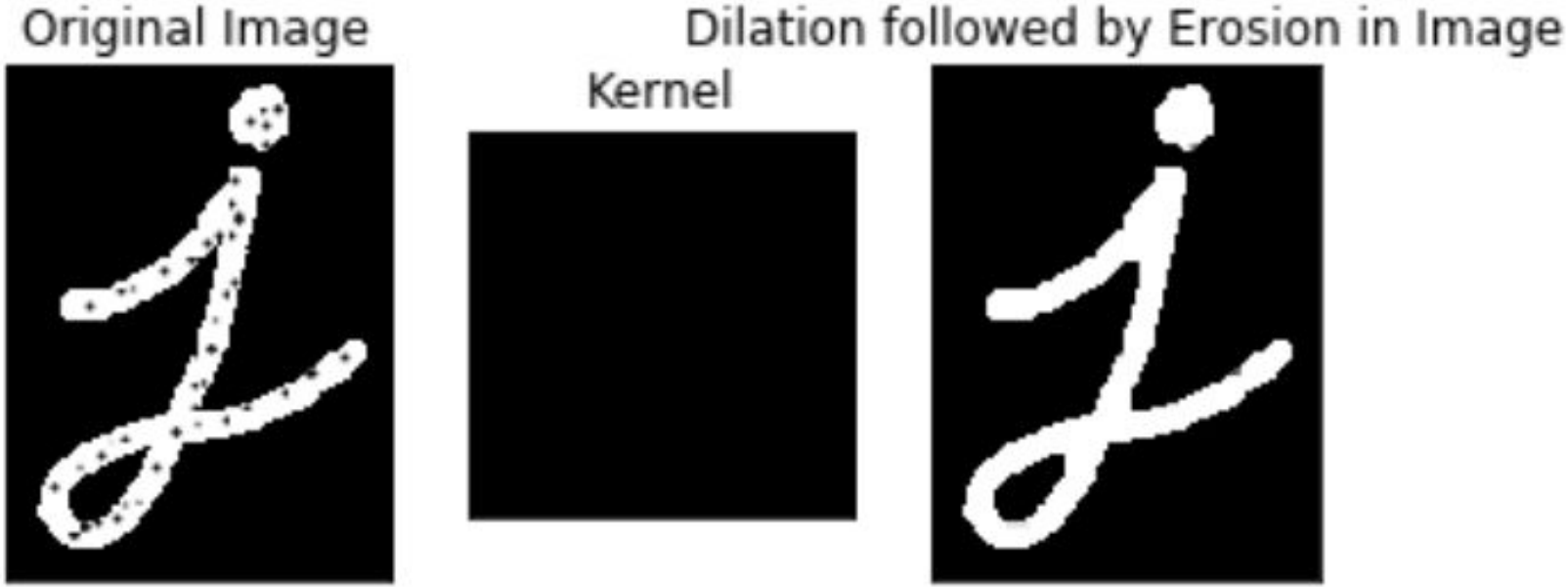
Closing

The closing of a set A by the kernel B, denoted $A \cdot B$, is defined as

$$A \cdot B = (A \oplus B) \ominus B$$

- Thus opening A by B is the dilation of A by B, followed by a erosion of the result by B

Closing


Original Image
Kernel
Dilation followed by Erosion in Image

Cont..

```
import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('jforclosing.png',0)

kernel = np.ones((5,5),np.uint8)

closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

Cont..

```
plt.subplot(131),plt.imshow(img,cmap = 'gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(132),plt.imshow(kernel,cmap = 'gray')

plt.title('Kernel'), plt.xticks([]), plt.yticks([])

plt.subplot(133),plt.imshow(closing,cmap = 'gray')

plt.title('Dilation followed by Erosion in Image'), plt.xticks([]), plt.yticks([])

plt.show()
```

Hit or Miss Transform

- Let B = (B1,B2) such that
  - B1 is the set formed from elements of B associated with an object
  - B2 is the set formed from elements of B associated with background

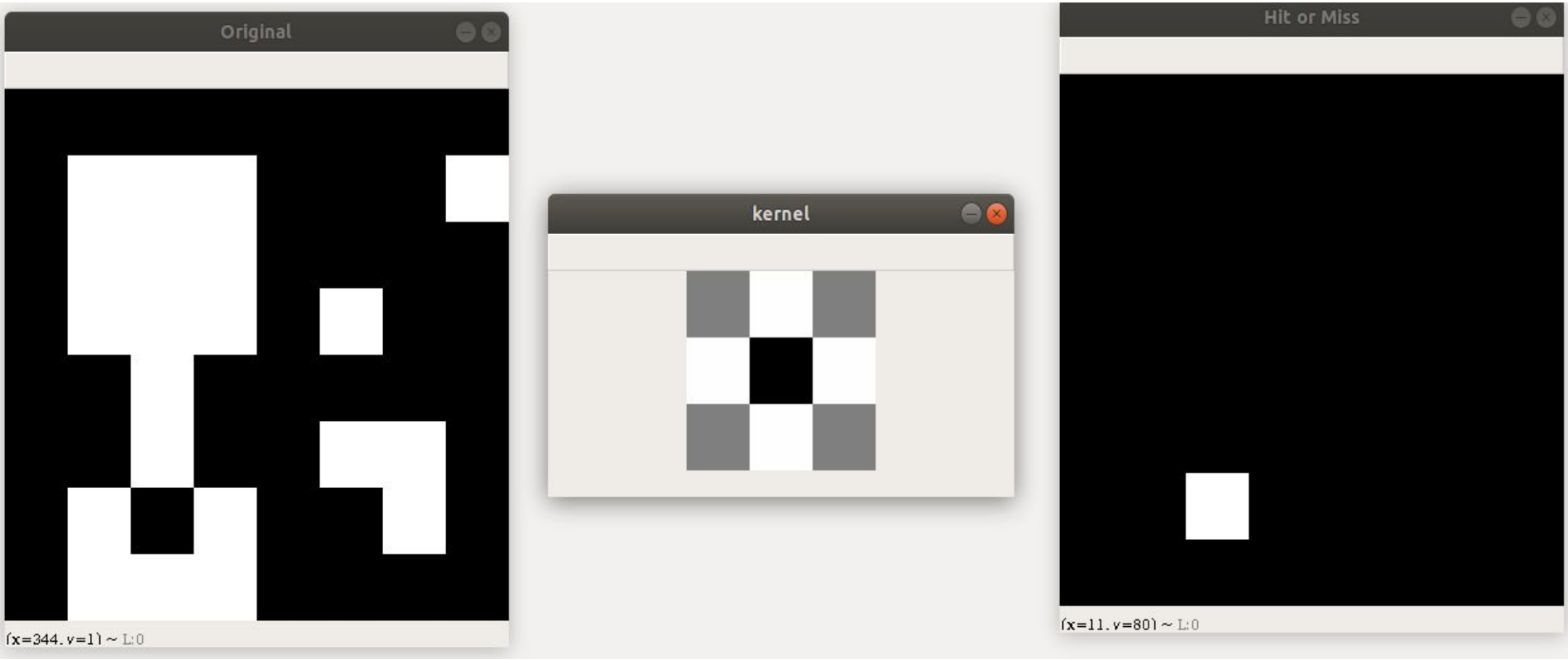
The hit-or-miss transform of a set A with B, denoted $A \circledast B$, is defined as

$A \circledast B = (A \ominus B_1) \cap (A^C \ominus B_2)$ --------- Eq.1

- The set $A \circledast B$ contains all the origin points at which, simultaneously B_1 found a match (“hit”) in A and B2 found a match in A^C.

Eq.1 can write as

$A \circledast B = (A \ominus B_1) - (A \oplus Ref(B_2))$


Original
kernel
Hit or Miss

Cont..

```
import cv2 as cv

import numpy as np

input_image = np.array((

    [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, 255, 0, 0, 0, 0]), dtype="uint8")
```

Cont..

```
kernel = np.array(( [0, 1, 0], [1, -1, 1], [0, 1, 0]), dtype="int")

output_image = cv.morphologyEx(input_image, cv.MORPH_HITMISS, kernel)

rate = 50

kernel = (kernel + 1) * 127

kernel = np.uint8(kernel)

kernel = cv.resize(kernel, None, fx = rate, fy = rate, interpolation =
cv.INTER_NEAREST)
```

Cont..

```
cv.imshow("kernel", kernel)

cv.moveWindow("kernel", 0, 0)

input_image = cv.resize(input_image, None, fx = rate, fy = rate, interpolation =
cv.INTER_NEAREST)

cv.imshow("Original", input_image)

cv.moveWindow("Original", 0, 200)
```

Cont..

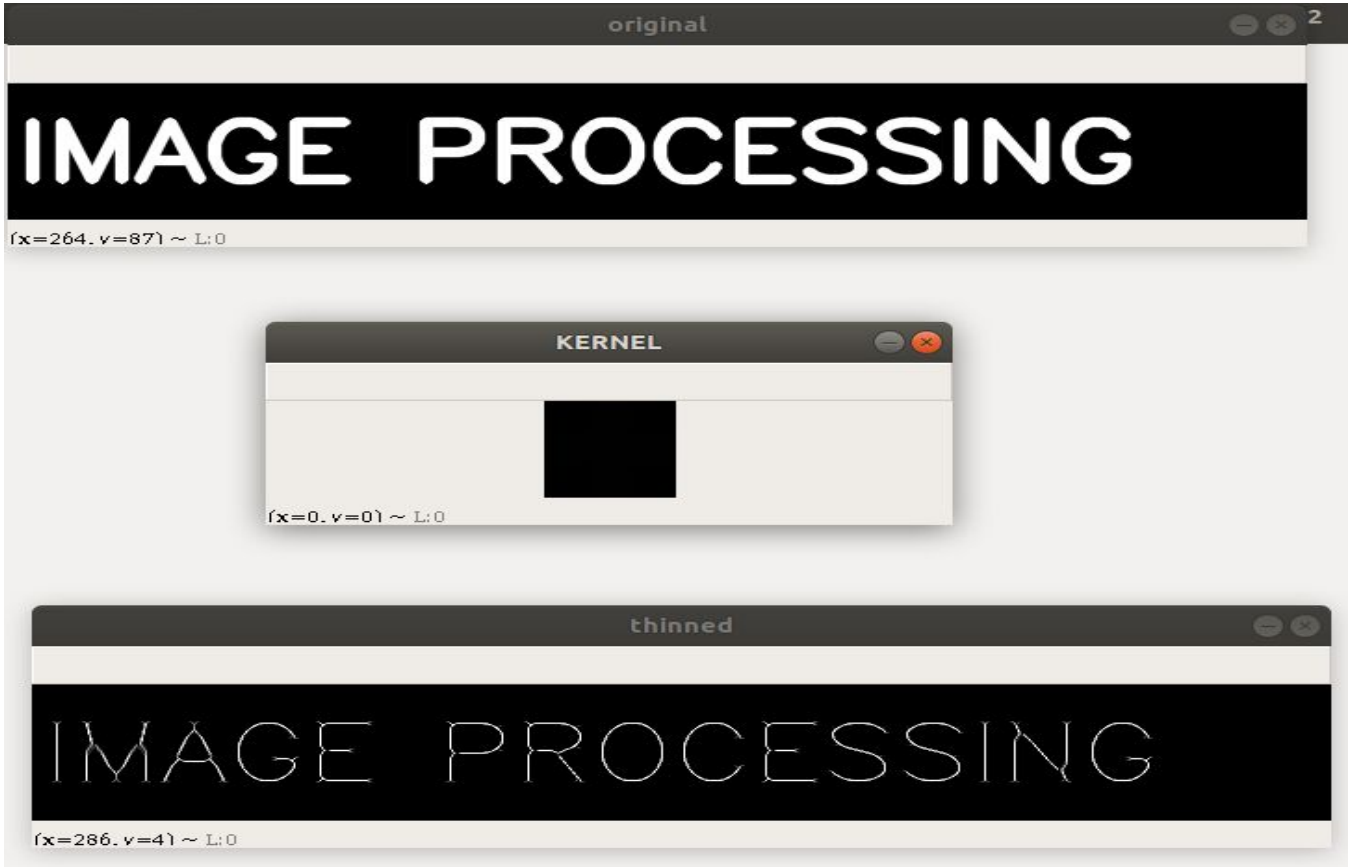
```
output_image = cv.resize(output_image, None , fx = rate, fy = rate,
interpolation = cv.INTER_NEAREST)

cv.imshow("Hit or Miss", output_image)

cv.moveWindow("Hit or Miss", 500, 200)

cv.waitKey(0)

cv.destroyAllWindows()
```

Thinning

The thinning of a set A by a kernel B, denoted as $A \otimes B$, can be defined in terms of the hit-or-miss transform

$$A \otimes B = A - (A \circledast B)$$

$$= A \cap (A \circledast B)^{C}$$


original
IMAGE PROCESSING
KERNEL
thinned
IMAGE PROCESSING

Cont..

```
import cv2

import numpy as np

# Create an image with text on it

img = np.zeros((100,700),dtype='uint8')

font = cv2.FONT_HERSHEY_SIMPLEX

cv2.putText(img,'IMAGE PROCESSING',(5,70), font, 2,(255),5,cv2.LINE_AA)

img1 = img.copy()
```

Cont..

```
# Structuring Element

kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))

# Create an empty output image to hold values

thin = np.zeros(img.shape,dtype='uint8')

# Loop until erosion leads to an empty set
```

Cont..

```
while (cv2.countNonZero(img1)!=0):
    erode = cv2.erode(img1,kernel)
    opening = cv2.morphologyEx(erode,cv2.MORPH_OPEN,kernel)
    subset = erode - opening
    thin = cv2.bitwise_or(subset,thin)
    img1 = erode.copy()
```

Cont..

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

cv2.imshow('thinned',thin)

cv2.imshow('KERNEL',kernel)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Thickening

The thickening of a set A by a kernel B, denoted as $A \odot B$, can be defined in terms of the hit-or-miss transform

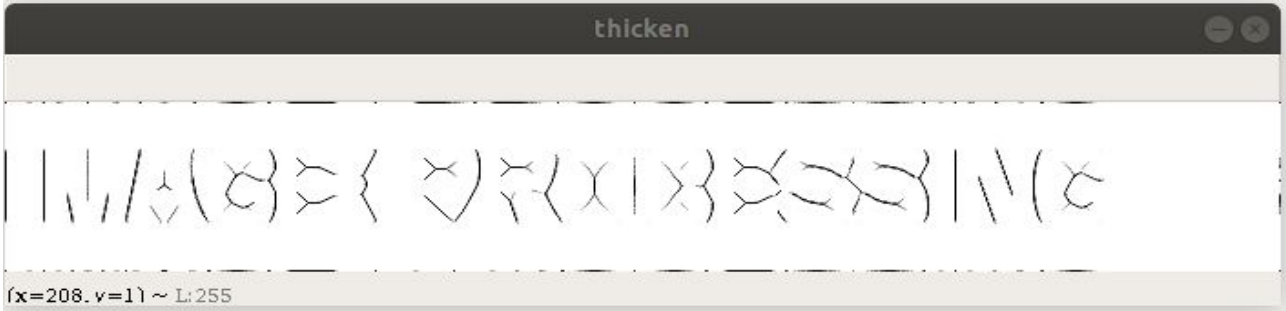
$$A \odot B = A \cup (A \circledast B)$$

Note: Thickening is the morphological dual of thinning.

To thicken a set A, we form $C = A^C$, thin C, and then form C^C.


original
IMAGE PROCESSING
(x=405, y=46) ~ L:255


kernel
(x=2, y=2) ~ L:0


thicken
(x=208, y=1) ~ L:255

Cont..

```
import cv2

import numpy as np

# Create an image with text on it

img = np.zeros((100,700),dtype='uint8')

font = cv2.FONT_HERSHEY_SIMPLEX

cv2.putText(img,'IMAGE PROCESSING',(5,70), font, 2,(255),5,cv2.LINE_AA)

img2 = img.copy()
```

Cont..

```
Img1 = ~img2

# Structuring Element

kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))

# Create an empty output image to hold values

thin = np.zeros(img.shape,dtype='uint8')

# Loop until erosion leads to an empty set
```

Cont..

```
while (cv2.countNonZero(img1)!=0):
    erode = cv2.erode(img1,kernel)
    opening = cv2.morphologyEx(erode,cv2.MORPH_OPEN,kernel)
    subset = erode - opening
    thin = cv2.bitwise_or(subset,thin)
    img1 = erode.copy()
```

Cont..

thick = ~thin

cv2.imshow('original',img)

cv2.imshow('thicken',thick)

cv2.imshow('KERNEL',kernel)

cv2.waitKey(0)

cv2.destroyAllWindows()

Thank You