Segment
- discontinuities
  - Laplacian / Canny edge
  - Line ~ hough trans
    - circle
    - parabola
- similarity
  - otsu's t
  - region growing

LECTURE 09
MORPHOLOGICAL IMAGE PROCESSING

Punnarai Siricharoen, Ph.D.

OBJECTIVES

morphology

- To understand basic morphological image processing techniques
- To be able to apply morphological techniques as development basis for segmentation, automated inspection, or describing an image with the simple to moderate complexity of the problems

mostly we apply on binary

CONTENT

- Introduction
- Logic operation with binary images
- Dilation / Erosion
- Closing / Opening
- Hit-or-miss transformation
- Other morphological algorithms
- Thinning/Thickening/Skeleton/Pruning
- Applications of grayscale morphology

INTRODUCTION

- **Morphology** post-process
  - A technique used to analyze and manipulate the **shape and structure** of objects within an image.
  - applying a set of mathematical operations to an input image to extract useful information or to modify the image in a way that **enhances certain features or removes unwanted details**.
- Morphological operations are primarily used in **binary or grayscale** images and are especially useful in tasks like image segmentation, noise reduction, object extraction, and feature enhancement by for extracting image components useful for representation and description of region shape, such as
  - **Boundaries**
  - **Skeletons**
  - **Convex hull**
- **Morphological techniques** – pre- or post-processing, such as morphological filtering, thinning and pruning.

INTRODUCTION

Preliminaries: Intro to set

- Let A be a set in Z^2 if a = (a_1, a_2) is an element of A, we write

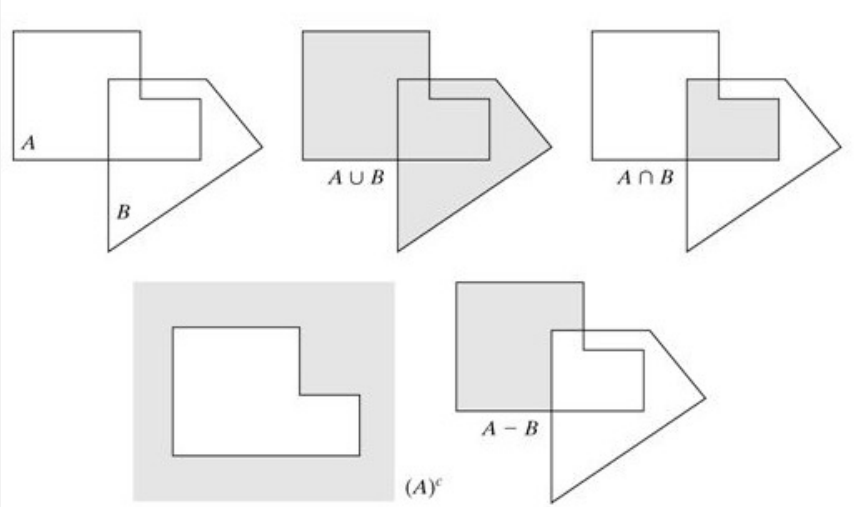
$$a \in A$$

- If a is not an element of A, we write

$$a \notin A$$

- Null or empty set: $\varnothing$
- Every element of set A is also an element of another set B, or A is a subset of B

$$A \subseteq B$$



a b c
d e

FIGURE 9.1
(a) Two sets A and B. (b) The union of A and B. (c) The intersection of A and B. (d) The complement of A. (e) The difference between A and B.

Image from Gonzalez & Woods, Digital Image Processing

INTRODUCTION

Preliminaries:

- The union of two sets, A and B:

$$C = A \cup B$$

- The intersection of two sets, A and B:

$$C = A \cap B$$

- Two sets are mutually exclusive.

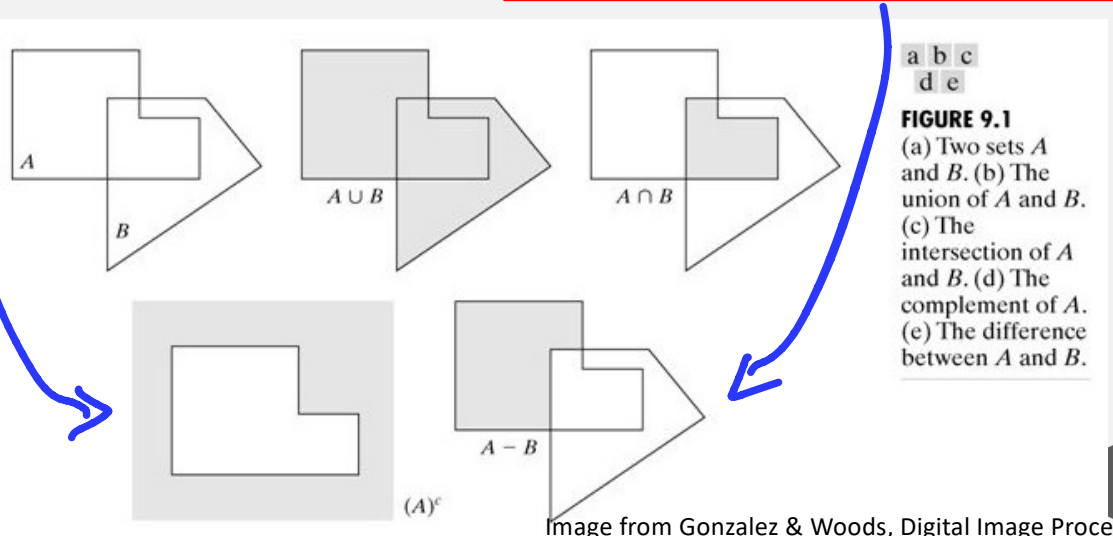
$$A \cap B = \varnothing$$

- complement:

$$A^c = \{w | w \notin A\}$$

- Difference of the two:

$$A - B = A \cap B^c$$



a b c
d e

FIGURE 9.1
(a) Two sets A and B. (b) The union of A and B. (c) The intersection of A and B. (d) The complement of A. (e) The difference between A and B.

Image from Gonzalez & Woods, Digital Image Processing

INTRODUCTION

Preliminaries:

- Reflection:

$$\hat{B} = \{w | w = -b \; for \; b \in B)$$

- Translation of A by point z = (z_1, z_2) denoted $(A)_z$

$$(A)_z = \{c | c = a + z \; for \; a \in A)$$

a b

FIGURE 9.2
(a) Translation of A by z.
(b) Reflection of B. The sets A and B are from Fig. 9.1.

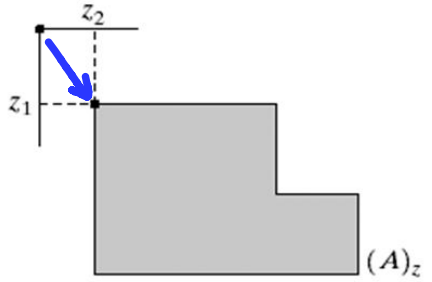


Image from Gonzalez & Woods, Digital Image Processing

LOGIC OPERATION WITH BINARY IMAGES

p	q	p AND q (also $p \cdot q$)	p OR q (also $p + q$)	NOT (p) (also $\bar{p}$)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

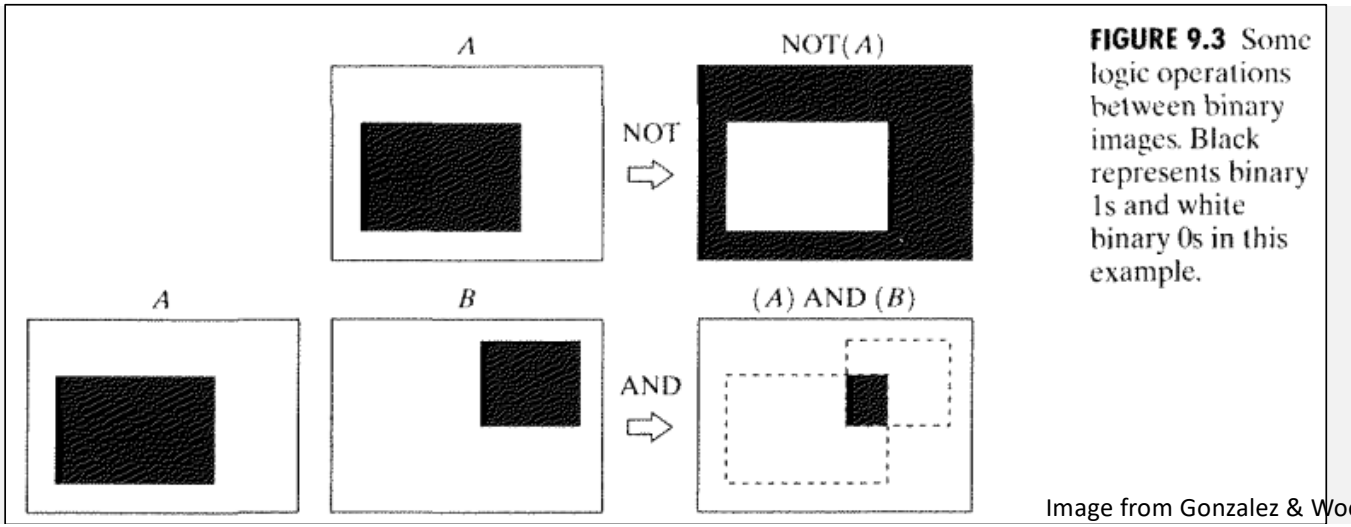
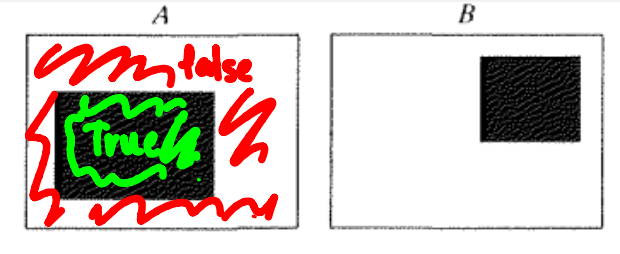
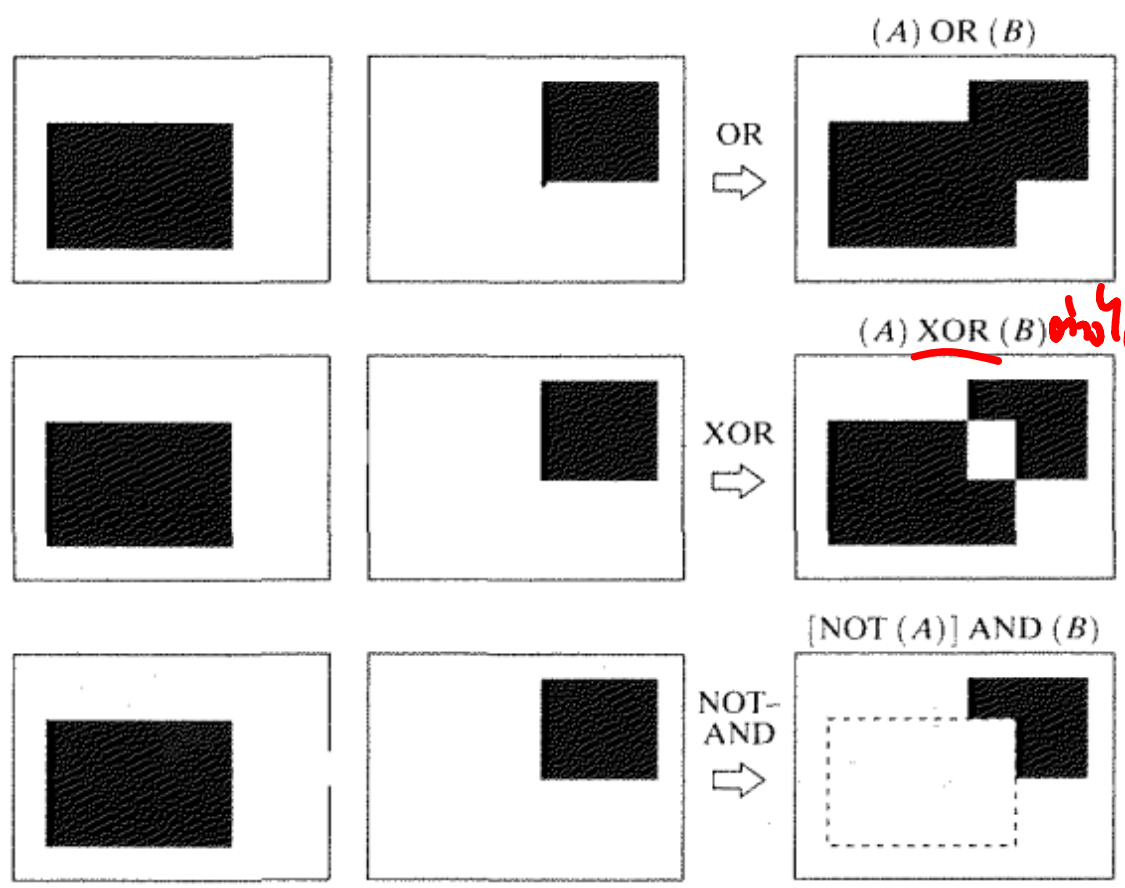


FIGURE 9.3 Some logic operations between binary images. Black represents binary 1s and white binary 0s in this example.

Image from Gonzalez & Woods, Digital Image Processing

LOGIC OPERATION WITH BINARY IMAGES





pixel-wise

Image from Gonzalez & Woods, Digital Image Processing

DILATION / EROSION

ขยาย / พองตัว กร่อน

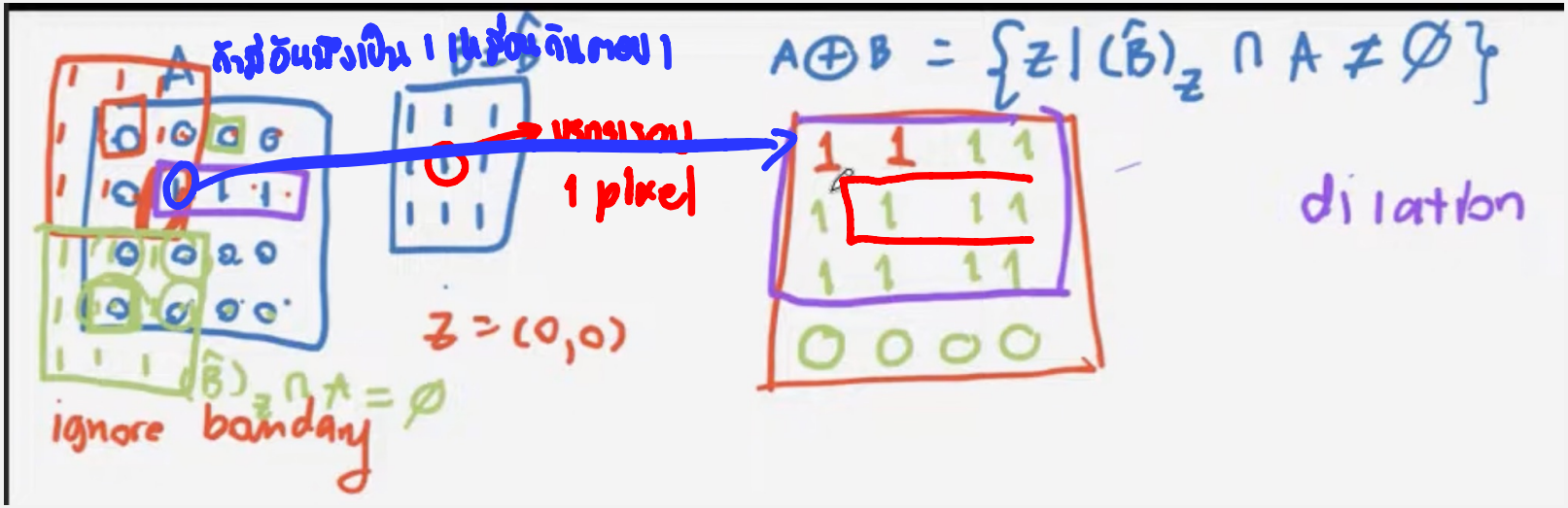
- Fundamental morphological operations.
- Many other morphological algorithms are based on these primitive operations.
- **Dilation**:
  - With A and B as sets in Z^2, the dilation of A by B, denoted $A \oplus B$

$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \varnothing\}$$

Flip & displace

Image kernel

  - Dilation of A by B is the set of all displacements that $(\hat{B})$ overlap A by at least elements.
  - B is called '**Structuring element**' (Convolution mask).


A
B
1 pixel
$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$
dilation
$z = (0,0)$
$(\hat{B})_z \cap A = \emptyset$
ignore boundary

DILATION / EROSION

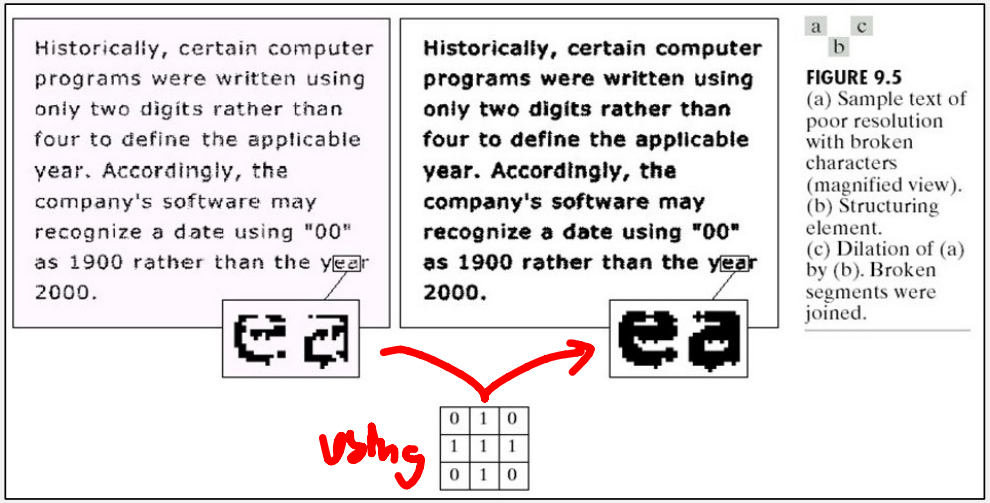
Dilation – set operations
Convolution - arithmetic operations

- **Dilation**: or can write,

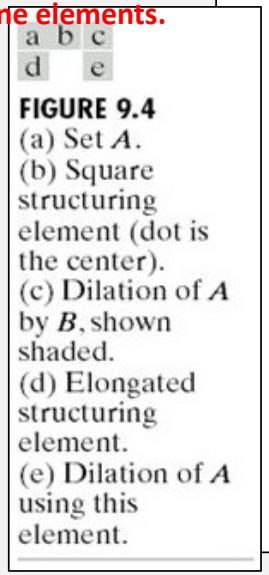
$$A \oplus B = \{z | (\hat{B})_z \cap A \subseteq A\}$$

- Application: use dilation for bridging gaps. make it thicker

The dilation of A by B is the set of all displacements z that B and A overlap at lease one elements.



a c
b
FIGURE 9.5
(a) Sample text of poor resolution with broken characters (magnified view). (b) Structuring element. (c) Dilation of (a) by (b). Broken segments were joined.



a b c
d e
FIGURE 9.4
(a) Set A. (b) Square structuring element (dot is the center). (c) Dilation of A by B, shown shaded. (d) Elongated structuring element. (e) Dilation of A using this element.

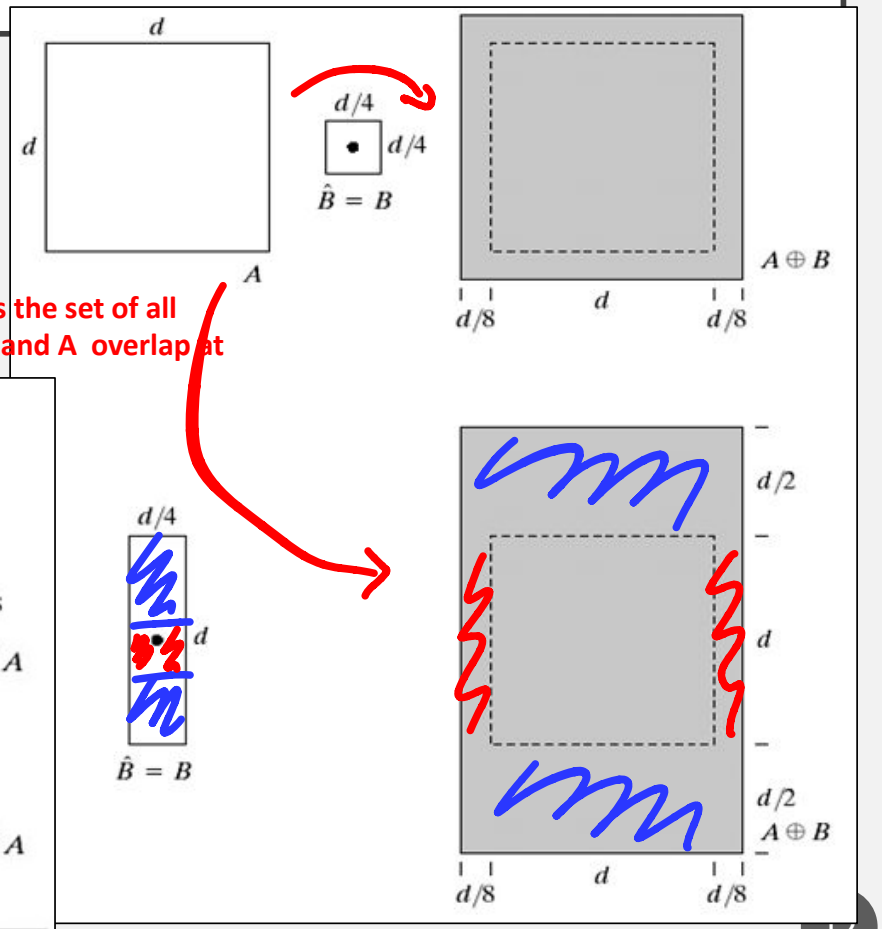


Image from Gonzalez & Woods, Digital Image Processing

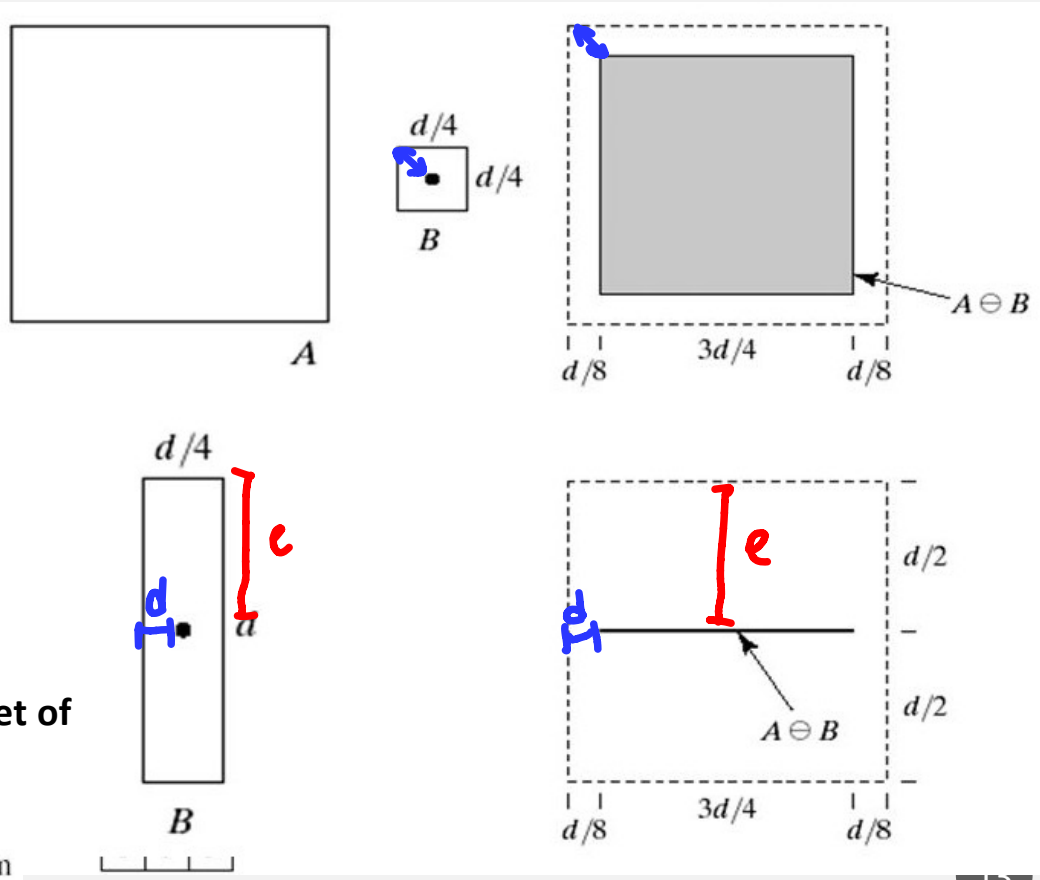
DILATION / EROSION

- **Erosion**: B translated by z is
- contained in A.

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

- Dilation and erosion are duals of each other with respect to set complementation and reflection: $(A \ominus B)^c = A^c \oplus \hat{B}$

Erosion – Erosion of A by B is the set of all points z that B translated by z is contained in A!



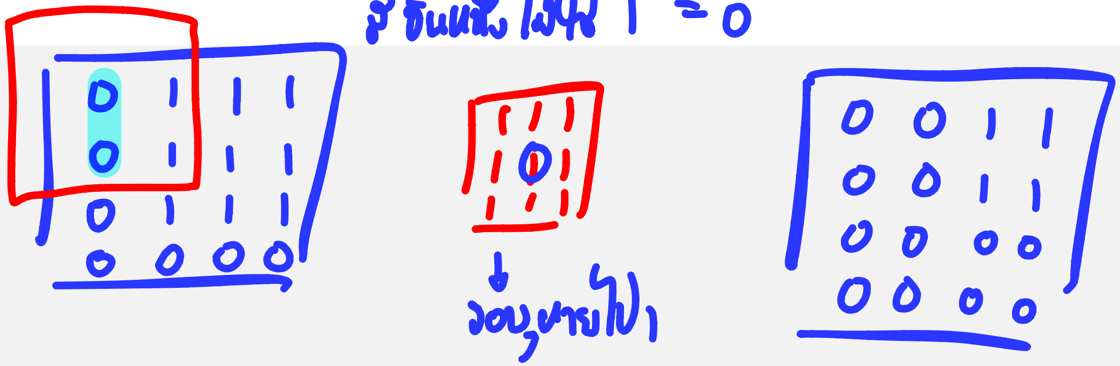
a b c
d e

FIGURE 9.6 (a) Set A. (b) Square structuring element. (c) Erosion of A by B, shown shaded. (d) Elongated structuring element. (e) Erosion of A using this element.

Image from Gonzalez & Woods, Digital Image Processing

Ignore boundarie

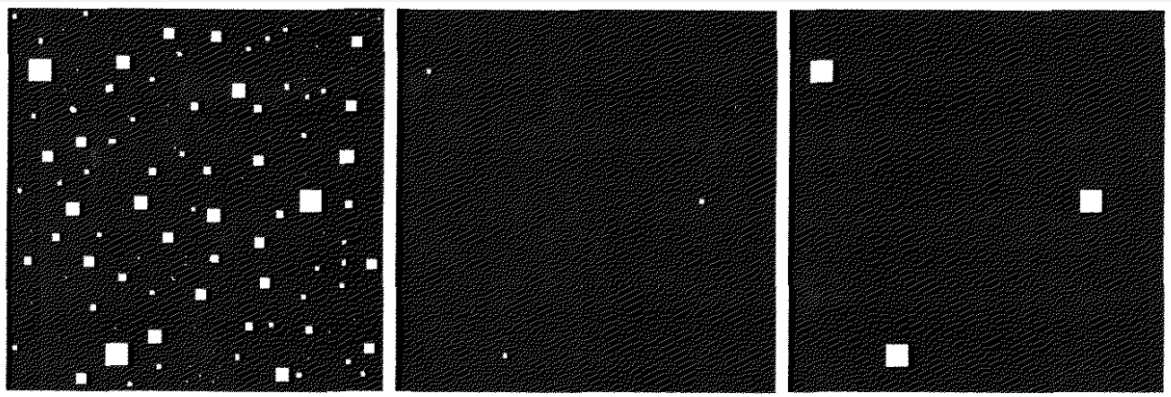
ถ้าขอบที่ไม่ใช่ 1 = 0



ขอบหายไป

DILATION / EROSION

- **Erosion**:
  - One application is to remove image components, particularly, irrelevant details from binary image.
  - Eliminating small squares shows below: remove noise ทำได้โดย Erosion → dilation (opening) ได้ผลเท่าเดิม แต่ noise หายไป



a b c

FIGURE 9.7 (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

Image from Gonzalez & Woods, Digital Image Processing

DILATION / EROSION

Python:

```
import cv2
import numpy as np

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

Python: `cv2.dilate(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]]) → dst`

Python: `cv2.erode(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]]) → dst`

Parameters:

- **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 image of the same size and type as `src`.
- **element** – structuring element used for erosion; if `element=Mat()`, a `3 x 3` rectangular structuring element is used.
- **anchor** – position of the anchor within the element; default value `(-1, -1)` means that the anchor is at the element center.
- **iterations** – number of times erosion is applied.
- **borderType** – pixel extrapolation method (see `borderInterpolate()` for details).
- **borderValue** – border value in case of a constant border (see `createMorphologyFilter()` for details).

Image from Gonzalez & Woods, Digital Image Processing

CLOSING / OPENING

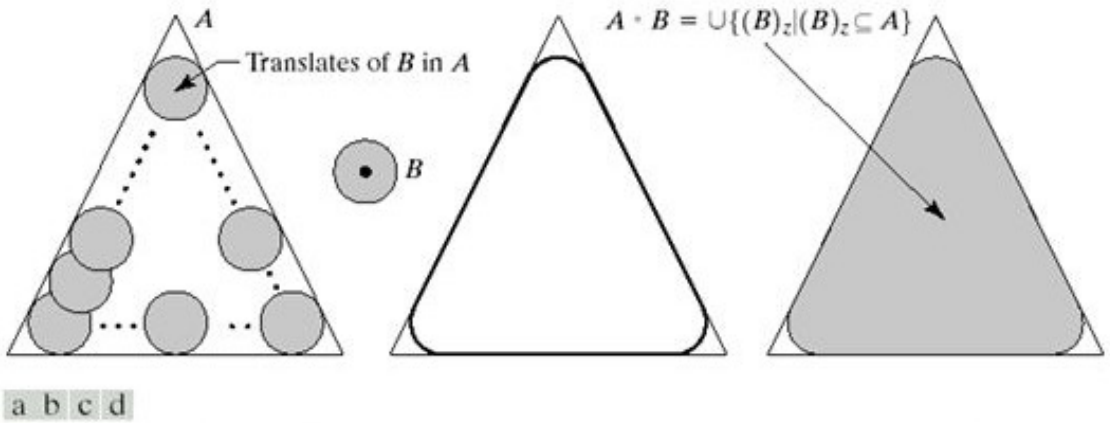
Dilate – expand
Erode – shrink

separate object

- **Opening** – smooth the contour of an object. Break narrow isthmus and eliminates thin protrusions.

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

Erode then dilate..



a b c d

FIGURE 9.8 (a) Structuring element B "rolling" along the inner boundary of A (the dot indicates the origin of B). (c) The heavy line is the outer boundary of the opening. (d) Complete opening (shaded).

Image from Gonzalez & Woods, Digital Image Processing

CLOSING / OPENING

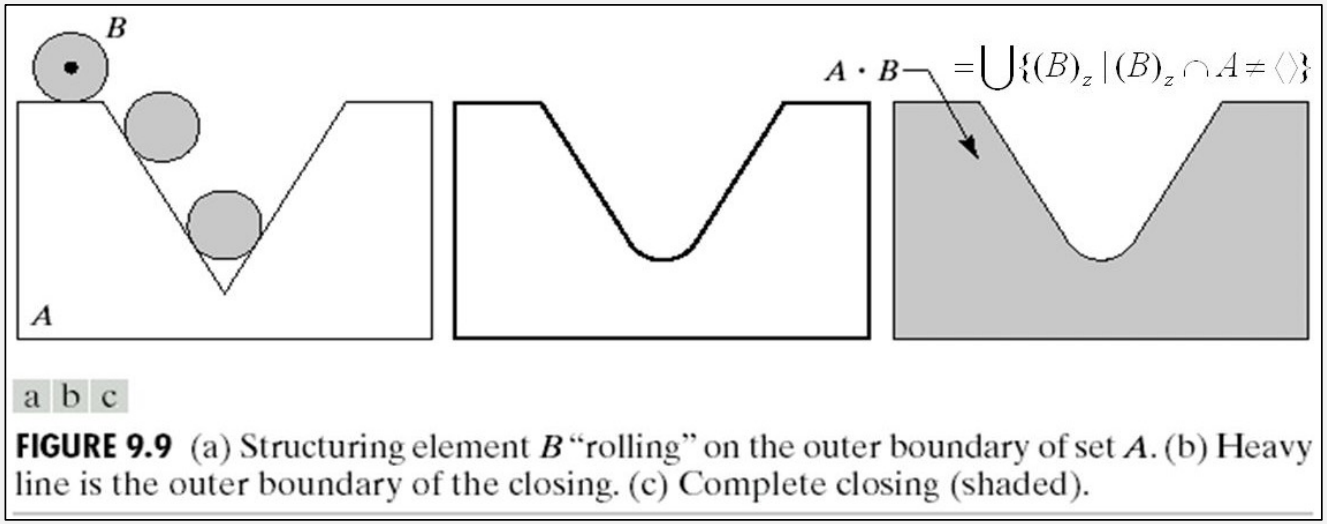
fill the hole, merge gap

Dilate – expand
Erode – shrink

- **Closing** – smooth sections of contours but fuse narrow breaks and long thin gulfs. Eliminate small holes and fill gaps in the contour.

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

Dilate then erode..

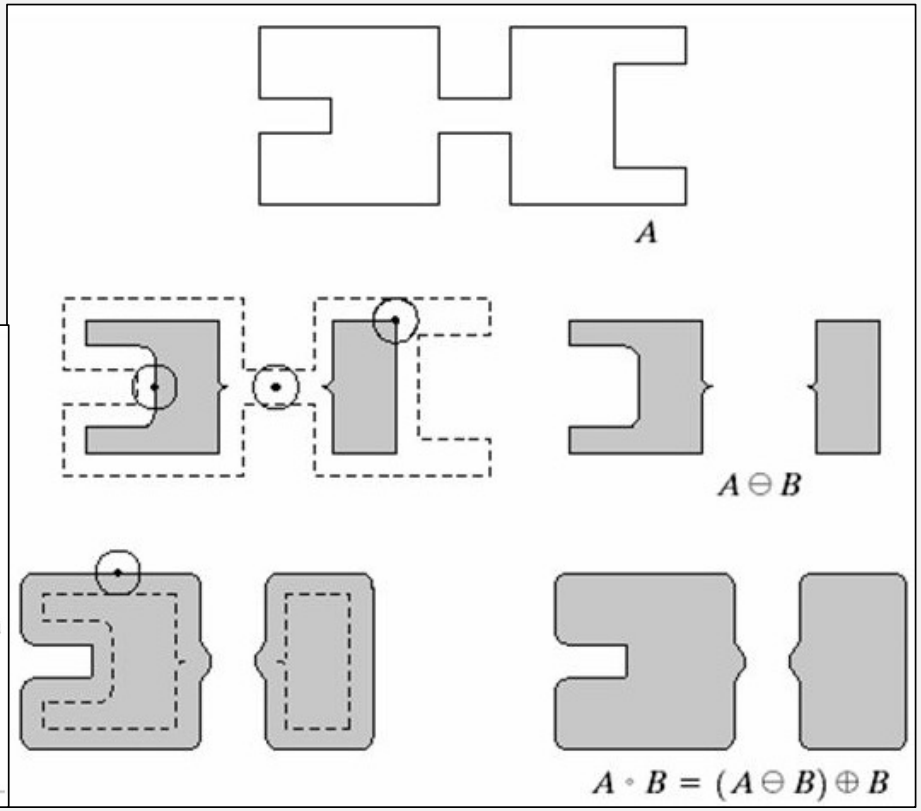


a b c

FIGURE 9.9 (a) Structuring element B "rolling" on the outer boundary of set A. (b) Heavy line is the outer boundary of the closing. (c) Complete closing (shaded).

Image from Gonzalez & Woods, Digital Image Processing

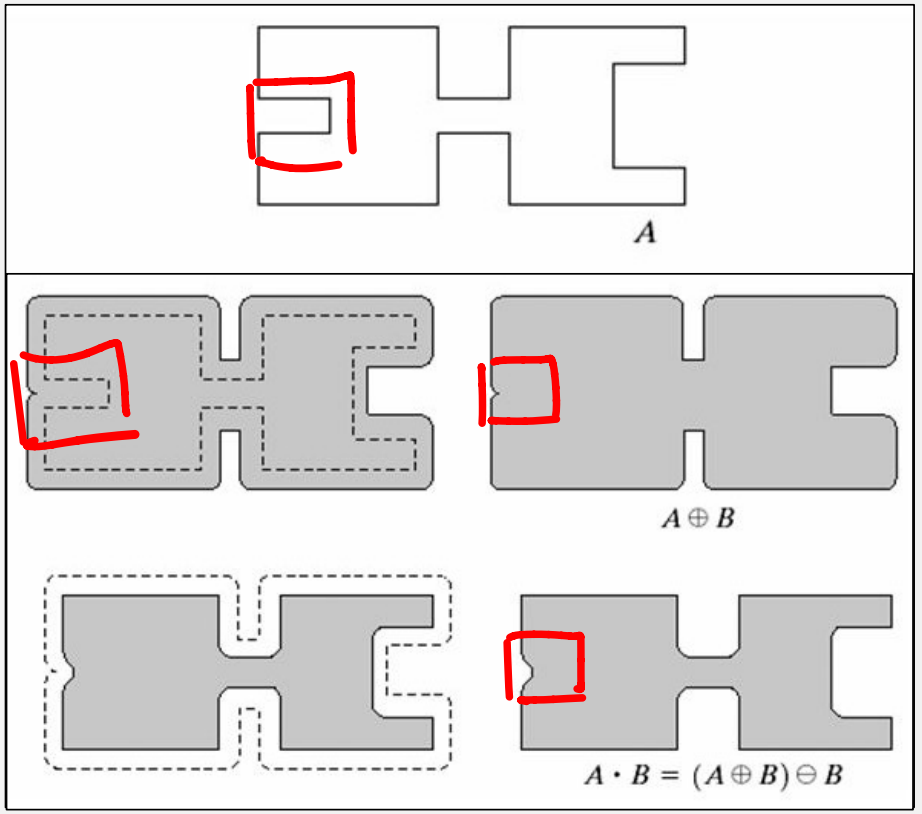
CLOSING / OPENING



a
b c
d e
f g
h i

FIGURE 9.10 Morphological opening and closing. The structuring element is the small circle shown in various positions in (b). The dark dot is the center of the structuring element.

OPENING



CLOSING

Image from Gonzalez & Woods, Digital Image Processing

CLOSING / OPENING

- Fingerprint problem:

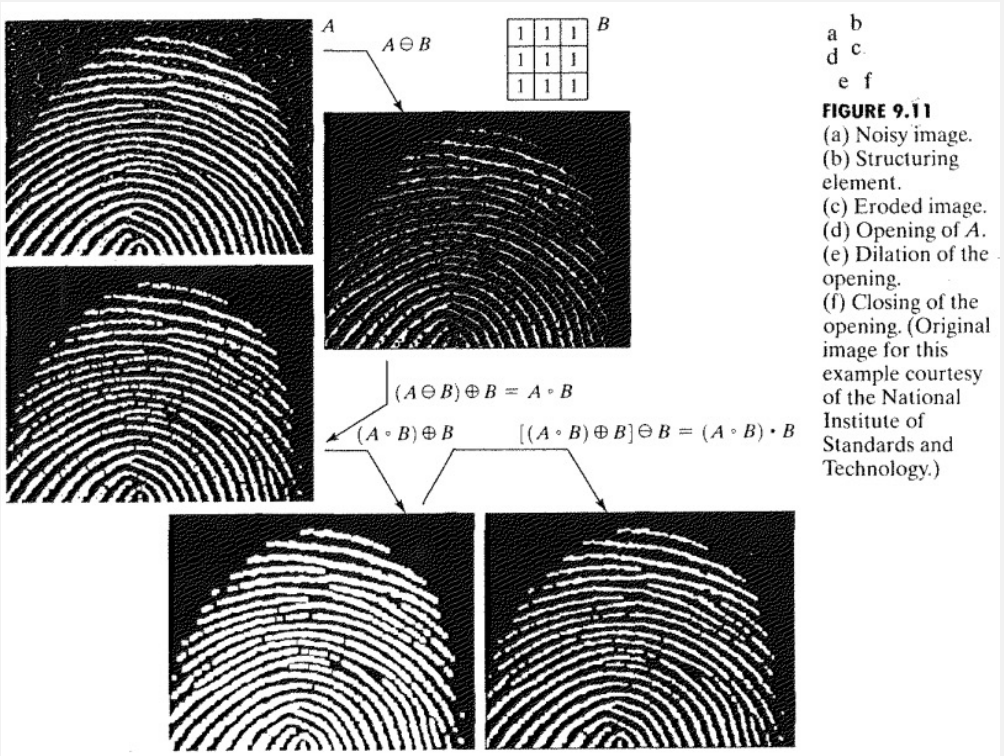
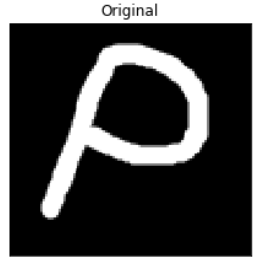


Image from Gonzalez & Woods, Digital Image Processing

- Opening and closing are also duals of each other:

$$(A \cdot B)^c = A^c \circ \hat{B}$$

OPENING / CLOSING




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

img = cv2.imread('P.png',0)
kernel = np.ones((5,5),np.uint8)

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

type of morphological operation

Enumerator
MORPH_ERODE Python: cv.MORPH_ERODE
MORPH_DILATE Python: cv.MORPH_DILATE
MORPH_OPEN Python: cv.MORPH_OPEN
MORPH_CLOSE Python: cv.MORPH_CLOSE
MORPH_GRADIENT Python: cv.MORPH_GRADIENT
MORPH_TOPHAT Python: cv.MORPH_TOPHAT
MORPH_BLACKHAT Python: cv.MORPH_BLACKHAT
MORPH_HITMISS Python: cv.MORPH_HITMISS

§ morphologyEx()

```
void cv::morphologyEx ( InputArray      src,
                        OutputArray     dst,
                        int             op,
                        InputArray      kernel,
                        Point           anchor = Point(-1,-1),
                        int             iterations = 1,
                        int             borderType = BORDER_CONSTANT,
                        const Scalar &  borderValue = morphologyDefaultBorderValue()
                      )
```

Python:

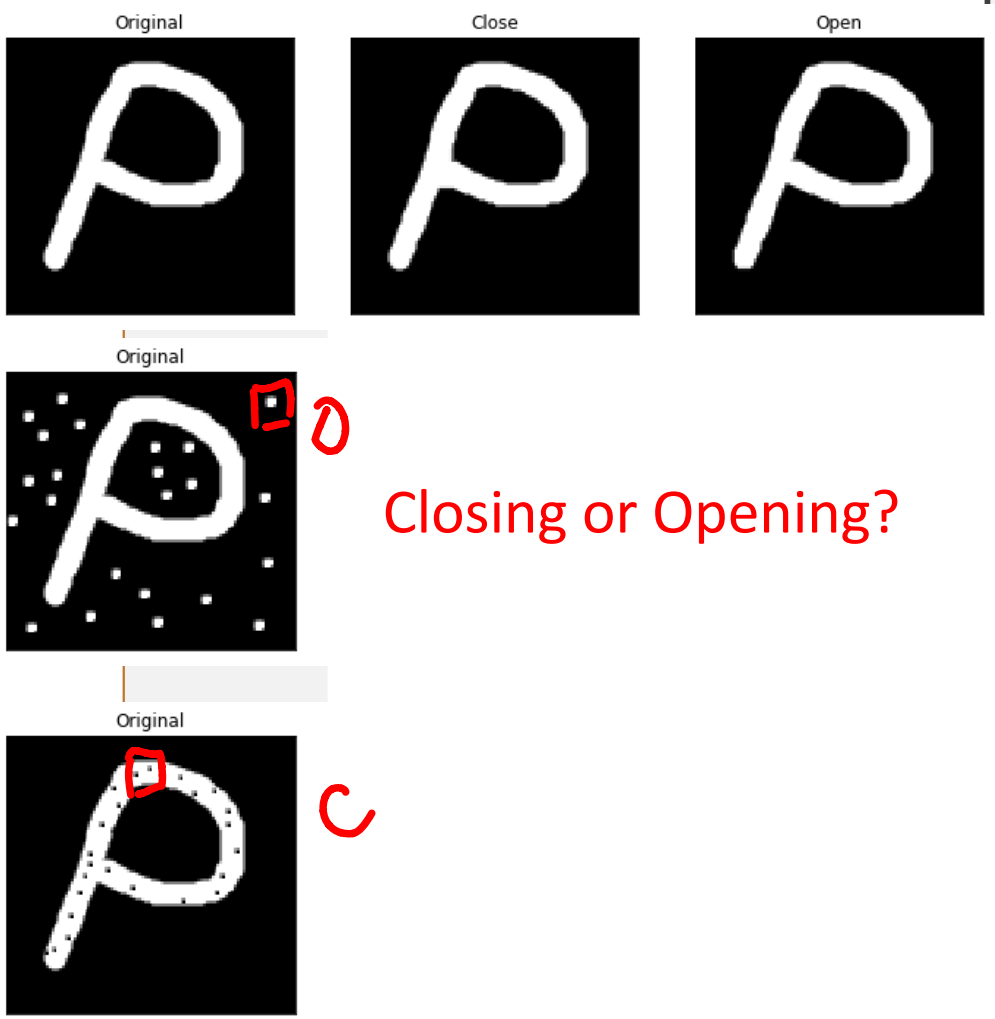
dst = cv.morphologyEx(src, op, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]])

Parameters

src	Source image. The number of channels can be arbitrary. The depth should be one of CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
dst	Destination image of the same size and type as source image.
op	Type of a morphological operation, see **MorphTypes**
kernel	Structuring element. It can be created using **getStructuringElement**.
anchor	Anchor position with the kernel. Negative values mean that the anchor is at the kernel center.
iterations	Number of times erosion and dilation are applied.
borderType	Pixel extrapolation method, see **BorderTypes**
borderValue	Border value in case of a constant border. The default value has a special meaning.

OPENING / CLOSING

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


img = cv2.imread('P.png',0)
kernel = np.ones((5,5),np.uint8)


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



HIT-OR-MISS TRANSFORMATION

If we try to find object X, B_1 = X, $B_2 = W - X$ where X is enclosed by a small window W.

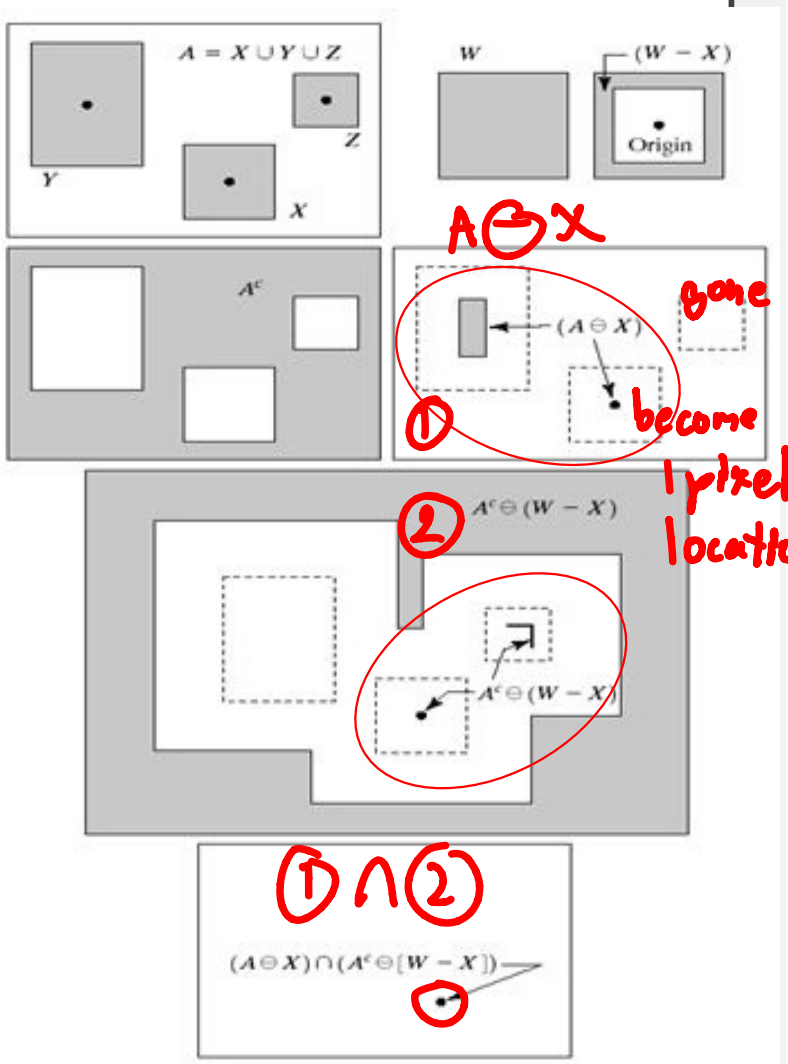
- The morphological hit-or-miss transform is a basic tool for shape detection.
- The objective is to find location of the shapes in the figure →

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

- Or (dual relationship)

$$A \circledast B = (A \ominus B_1) - (A^c \oplus \hat{B}_2)$$

- $A \circledast B$ contains points (origin) B_1 found match in A and B_2 found match in A^c



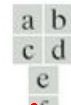


FIGURE 9.12 (a) Set A. (b) A window, W, and the local background of X with respect to W, $(W - X)$. (c) Complement of A. (d) Erosion of A by X. (e) Erosion of A^c by $(W - X)$. (f) Intersection of (d) and (e), showing the location of the origin of X, as desired.

Image from Gonzalez & Woods, Digital Image Processing

SOME BASIC MORPHOLOGICAL ALGORITHMS

- Boundary extraction
- Region filling
- Extraction of connected components
- Convex Hull
- Thinning
- Skeletons
- Pruning
- Summary of Morphological Operations on binary images

BOUNDARY EXTRACTION

- The boundary of set A, denoted by $\beta(A)$
  - Eroding A by B then performing set difference:

$$\beta(A) = A - (A \ominus B)$$

  - B is a suitable structuring element.

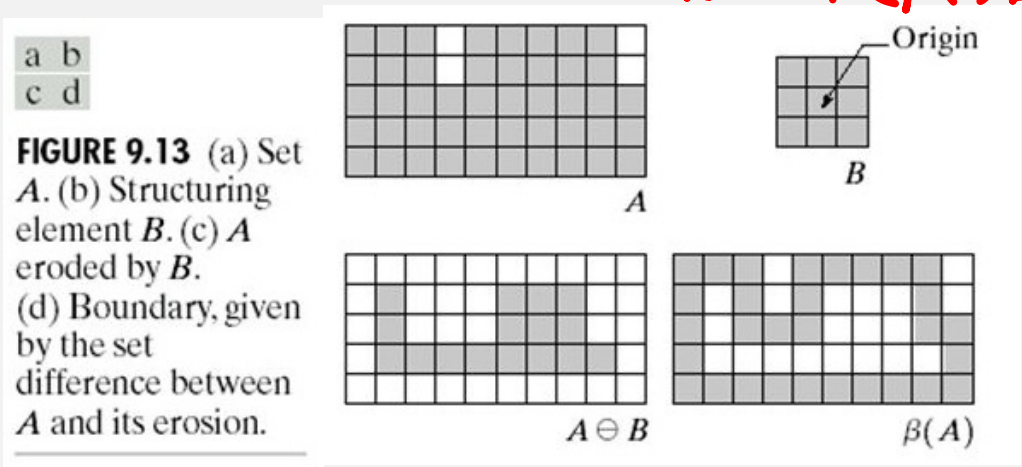


Image from Gonzalez & Woods, Digital Image Processing

BOUNDARY EXTRACTION

- Use different size of structuring element of 1, such as 5 x 5 result in a boundary between 2-3 pixels thick.

a b

FIGURE 9.14
(a) A simple binary image, with 1's represented in white. (b) Result of using Eq. (9.5-1) with the structuring element in Fig. 9.13(b).

Image from Gonzalez & Woods, Digital Image Processing

REGION FILLING เติมสี 4ทิศ

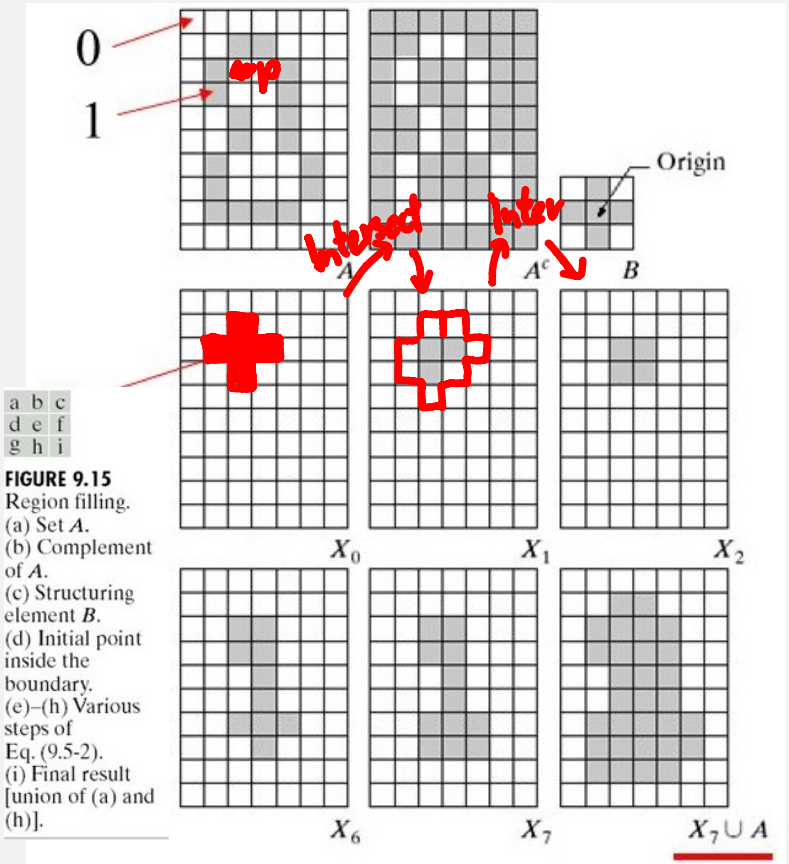
- The objective is to fill the entire region (starting at point p) with 1's.
- Non-boundary (background) points are labeled 0.
- The following procedure then fills the region with 1's:

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1,2,3,..$$

- X_0= p or the starting point
- B is the symmetric structuring element. The algorithm terminates at iteration step k.
- Assuming a point inside each boundary is given.
- The algorithm terminates at iteration step k if $X_k = X_{k-1}$.
- X_k contains the filled holes; the set union of X_k and the original image contains all holes and its boundaries.

REGION FILLING

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1,2,3,..$$



a b c
d e f
g h i

FIGURE 9.15 Region filling. (a) Set A. (b) Complement of A. (c) Structuring element B. (d) Initial point inside the boundary. (e)–(h) Various steps of Eq. (9.5-2). (i) Final result [union of (a) and (h)].

a b c

FIGURE 9.16 (a) Binary image (the white dot inside one of the regions is the starting point for the region-filling algorithm). (b) Result of filling that region (c) Result of filling all regions.

Image from Gonzalez & Woods, Digital Image Processing

EXTRACTION OF CONNECTED COMPONENTS

- Let Y represents a connected component contained in a set A and assume that a point p of Y is known. Then the following iterative expression yields all the elements of Y: X_0 เป็น pixel ในเซตที่ต้องการ

$$X_k = (X_{k-1} \oplus B) \cap A \quad k = 1,2,3,..$$

- The algorithm terminates at iteration step k if $X_k = X_{k-1}$.
- Similar to region filling except use A instead of its complement. A^c

EXTRACTION OF CONNECTED COMPONENTS

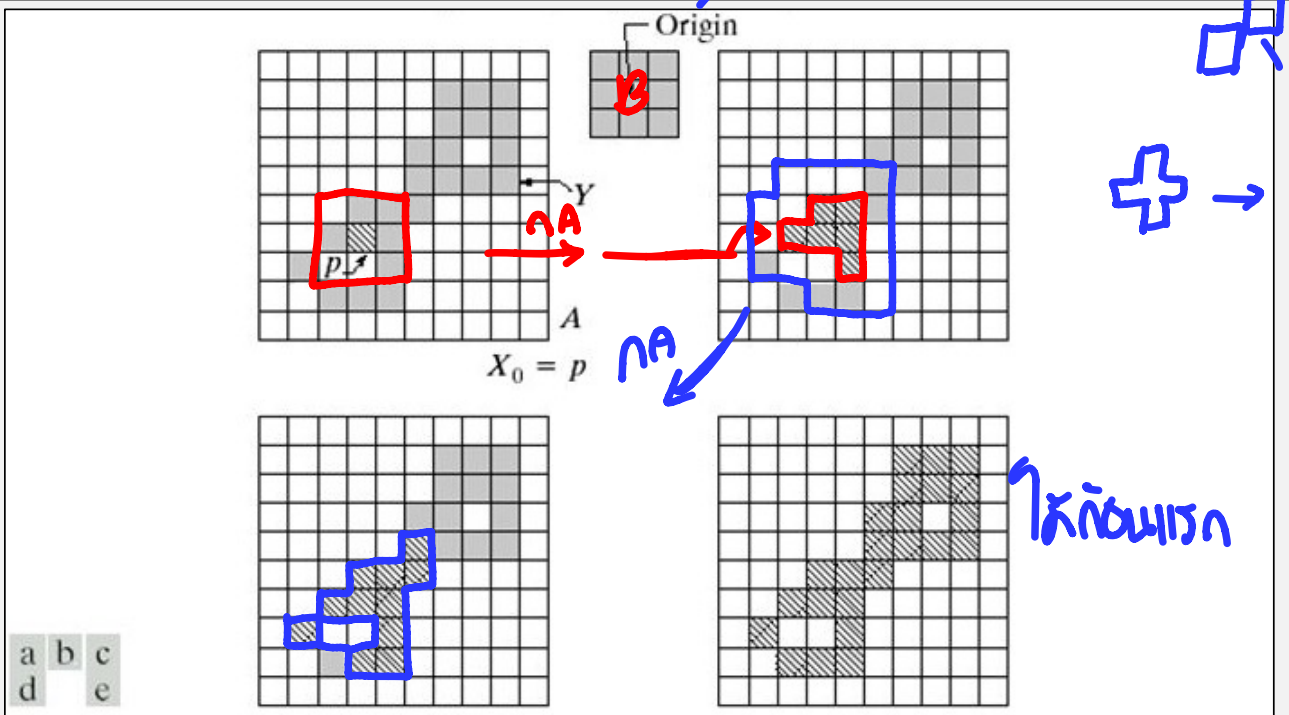


FIGURE 9.17 (a) Set A showing initial point p (all shaded points are valued 1, but are shown different from p to indicate that they have not yet been found by the algorithm). (b) Structuring element. (c) Result of first iterative step. (d) Result of second step. (e) Final result.

Image from Gonzalez & Woods, Digital Image Processing

EXTRACTION OF CONNECTED COMPONENTS

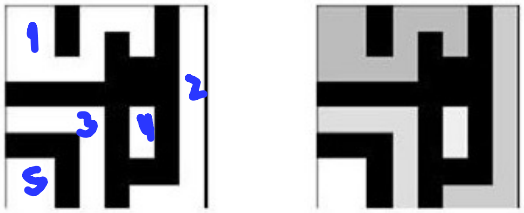
- Automated inspection

1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	0	1	0	0	0	1
1	1	0	1	0	1	1	1

a) binary image

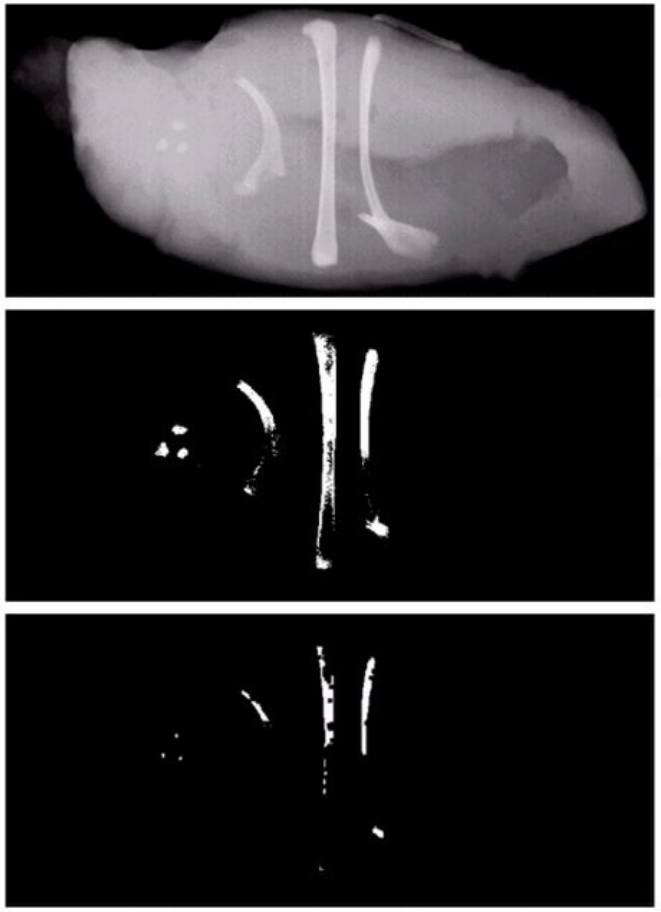
1	1	0	1	1	1	0	2
1	1	0	1	0	1	0	2
1	1	1	1	0	0	0	2
0	0	0	0	0	0	0	2
3	3	3	3	0	4	0	2
0	0	0	3	0	4	0	2
5	5	0	3	0	0	0	2
5	5	0	3	0	2	2	2

b) connected components labeling



c) binary image and labeling, expanded for viewing

Image from Gonzalez & Woods, Digital Image Processing



a
b
c d

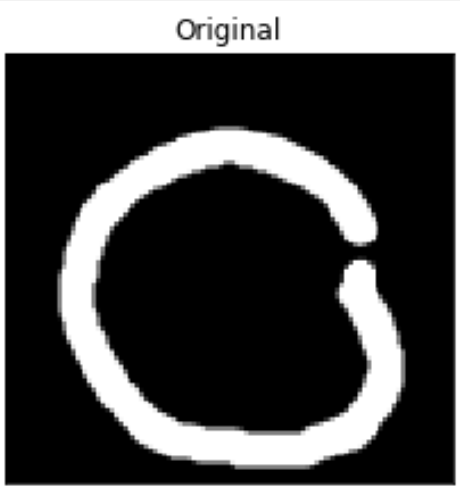
FIGURE 9.18 (a) X-ray image of chicken filet with bone fragments. (b) Thresholded image. (c) Image eroded with a 5×5 structuring element of 1's. (d) Number of pixels in the connected components of (c). (Image courtesy of NTB Elektronische Geraete GmbH, Diepholz, Germany, www.ntbxray.com.)

Connected component	No. of pixels in connected comp
01	11
02	9
03	9
04	39
05	133
06	1
07	1
08	743
09	7
10	11
11	11
12	9
13	9
14	674
15	85

REGION FILLING

Python: `cv2.floodFill(image, mask, seedPoint, newVal[, loDiff[, upDiff[, flags]]]) → retval, rect`

- **seedPoint** – Starting point.
- **newVal** – New value of the repainted domain pixels.
- **loDiff** – Maximal lower brightness/color difference between the currently observed pixel and one of its neighbors belonging to the component, or a seed pixel being added to the component.
- **upDiff** – Maximal upper brightness/color difference between the currently observed pixel and one of its neighbors belonging to the component, or a seed pixel being added to the component.
- **rect** – Optional output parameter set by the function to the minimum bounding rectangle of the repainted domain.



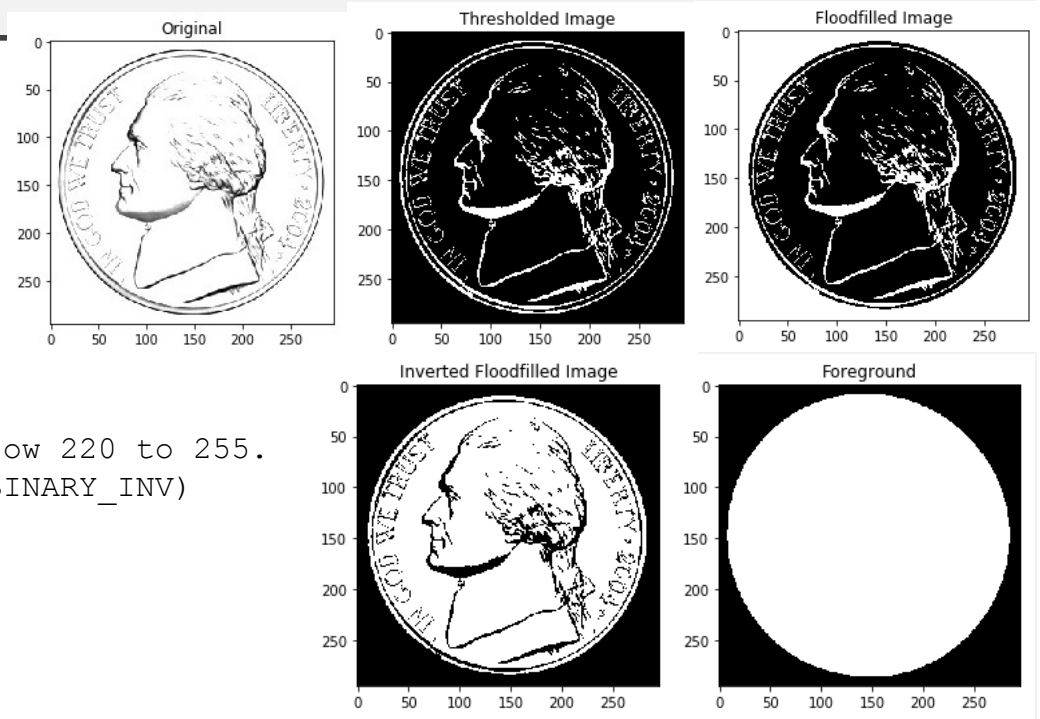

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Read image
img = cv2.imread("zen.png",0)

# Threshold.
# Set values equal to or above 127 to 0. Set values below 127 to 255.
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

# Closing to connect edges
kernel = np.ones((5,5),np.uint8)
closing = cv2.morphologyEx(thresh1, cv2.MORPH_CLOSE, kernel)

# Copy the thresholded image.
im_floodfill = closing.copy()

# Mask used to flood filling.# Notice the size needs to be 2 pixels than the image.
h, w = closing.shape[:2]
mask = np.zeros((h+2, w+2), np.uint8)

# Floodfill from point (0, 0)
 cv2.floodFill(im_floodfill, mask, (55,55), 255)
```

https://learnopencv.com/filling-holes-in-an-image-using-opencv-python-c/

https://docs.opencv.org/3.4.15/d7/d1b/group__imgproc__misc.html#ga366aae45a6c1289b341d140839f18717

REGION FILLING

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

# Read image
im_in = cv2.imread("nickel.png",0)

# Set values equal to or above 220 to 0. Set values below 220 to 255.
th, im_th = cv2.threshold(im_in, 220, 255, cv2.THRESH_BINARY_INV)

# Copy the thresholded image.
im_floodfill = im_th.copy()

# Mask used to flood filling.
# Notice the size needs to be 2 pixels than the image.
h, w = im_th.shape[:2]
mask = np.zeros((h+2, w+2), np.uint8)

# Floodfill from point (0, 0)
cv2.floodFill(im_floodfill, mask, (0,0), 255)

# Invert floodfilled image
im_floodfill_inv = cv2.bitwise_not(im_floodfill)

# Combine the two images to get the foreground.
im_out = im_th | im_floodfill_inv
```



https://learnopencv.com/filling-holes-in-an-image-using-opencv-python-c/

https://docs.opencv.org/4.5.3/d3/dc0/group__imgproc__shape.html#gaedef8c7340499ca391d459122e51bef5

◆ connectedComponents() [2/2]

binary image

```
int cv::connectedComponents ( InputArray  image,
                              OutputArray labels,
                              int         connectivity = 8 ,
                              int         ltype = CV_32S
                            )
```

Python:

```
cv.connectedComponents(                  image[, labels[, connectivity[, ltype]]]         ) -> retval, labels
cv.connectedComponentsWithAlgorithm( image, connectivity, ltype, ccltype[, labels] ) -> retval, labels
```

```
#include <opencv2/imgproc.hpp>
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

image	the 8-bit single-channel image to be labeled
labels	destination labeled image
connectivity	8 or 4 for 8-way or 4-way connectivity respectively
ltype	output image label type. Currently CV_32S and CV_16U are supported.

https://docs.opencv.org/4.5.3/d3/dc0/group__imgproc__shape.html#gaedef8c7340499ca391d459122e51bef5

◆ connectedComponentsWithStats() [2/2]

```
int cv::connectedComponentsWithStats ( InputArray  image,
                                       OutputArray labels,
                                       OutputArray stats,
                                       OutputArray centroids,
                                       int         connectivity = 8 ,
                                       int         ltype = CV_32S
                                     )
```

Python:

```
cv.connectedComponentsWithStats(                  image[, labels[, stats[, centroids[, connectivity[, ltype]]]]]          ) -> retval, labels, stats, centroids
cv.connectedComponentsWithStatsWithAlgorithm( image, connectivity, ltype, ccltype[, labels[, stats[, centroids]]] ) -> retval, labels, stats, centroids
```

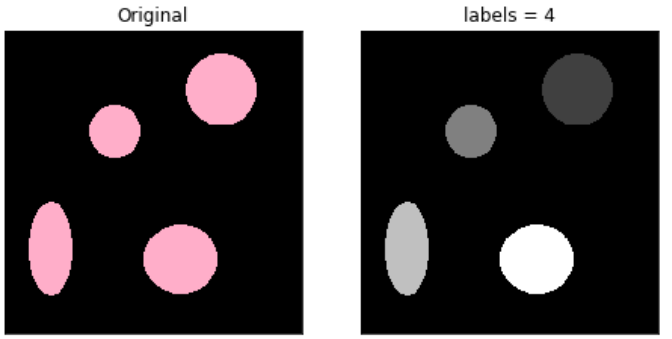
```
#include <opencv2/imgproc.hpp>
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

image	the 8-bit single-channel image to be labeled
labels	destination labeled image
stats	statistics output for each label, including the background label. Statistics are accessed via stats(label, COLUMN) where COLUMN is one of **ConnectedComponentsTypes**, selecting the statistic. The data type is CV_32S.
centroids	centroid output for each label, including the background label. Centroids are accessed via centroids(label, 0) for x and centroids(label, 1) for y. The data type CV_64F.
connectivity	8 or 4 for 8-way or 4-way connectivity respectively
ltype	output image label type. Currently CV_32S and CV_16U are supported.

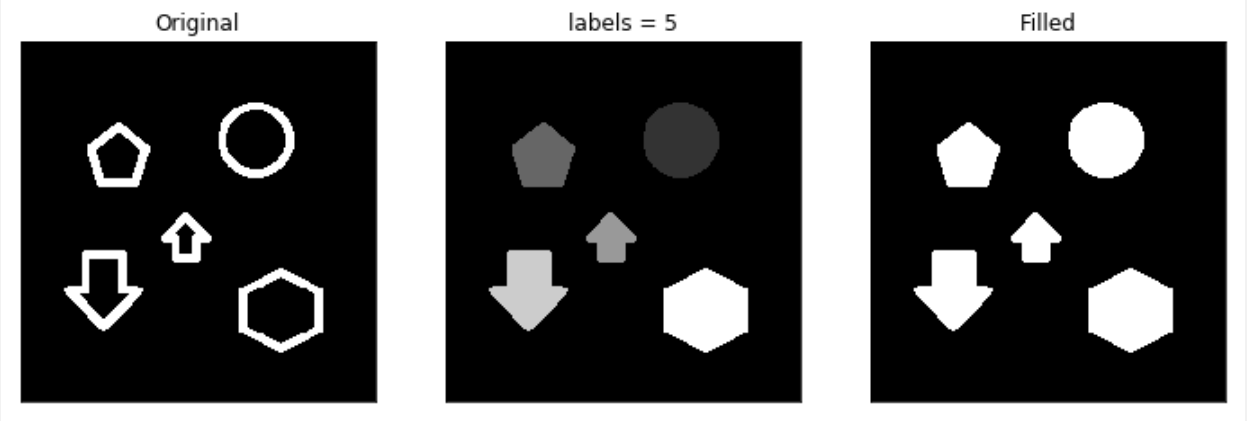
CONNECTED COMPONENT




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

img_rgb = cv2.imread("4objects.png")
img = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)

ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

# You need to choose 4 or 8 for connectivity type
connectivity = 4

output = cv2.connectedComponentsWithStats(thresh1, connectivity, cv2.CV_32S)

# Get the results
# The first cell is the number of labels
num_labels = output[0]-1
# The second cell is the label matrix
labels = output[1]
# The third cell is the stat matrix
stats = output[2]
# The fourth cell is the centroid matrix
centroids = output[3]
```

CONNECTED COMPONENT

- Fill The region and count the number of objects.
- How many Labels?
- Let’s calculate!

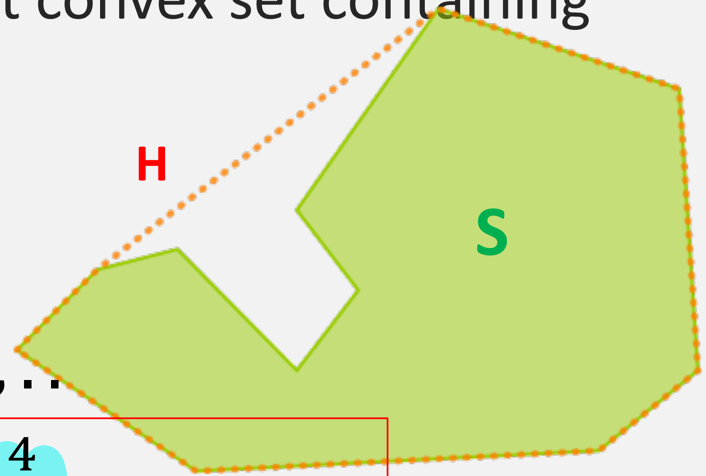


CONVEX HULL

หารูปหลายเหลี่ยมที่เล็กที่สุด และต้อง convex

not convex

- A set A is said to be ***convex*** if the straight line segment joining any two points in A lies entirely within A.
- The convex hull H of an arbitrary set S is the smallest convex set containing S.
- Let B^i, i = 1,2,3,4, represent 4 structuring elements, the procedure: $X_0^1 = A$



$$X_k^i = (X_k \circledast B^i) \cup A \; ; k = 1,2,3,...$$

Convergence in the sense that $X_k^i = X_{k-1}^i$

$$D^i = X_{conv}^i$$

Convex hull of A, C(A) is

$$C(A) = \bigcup_{i=1}^{4} D^i$$

CONVEX HULL

x – don't care

Simplified version of hit-or-miss transform

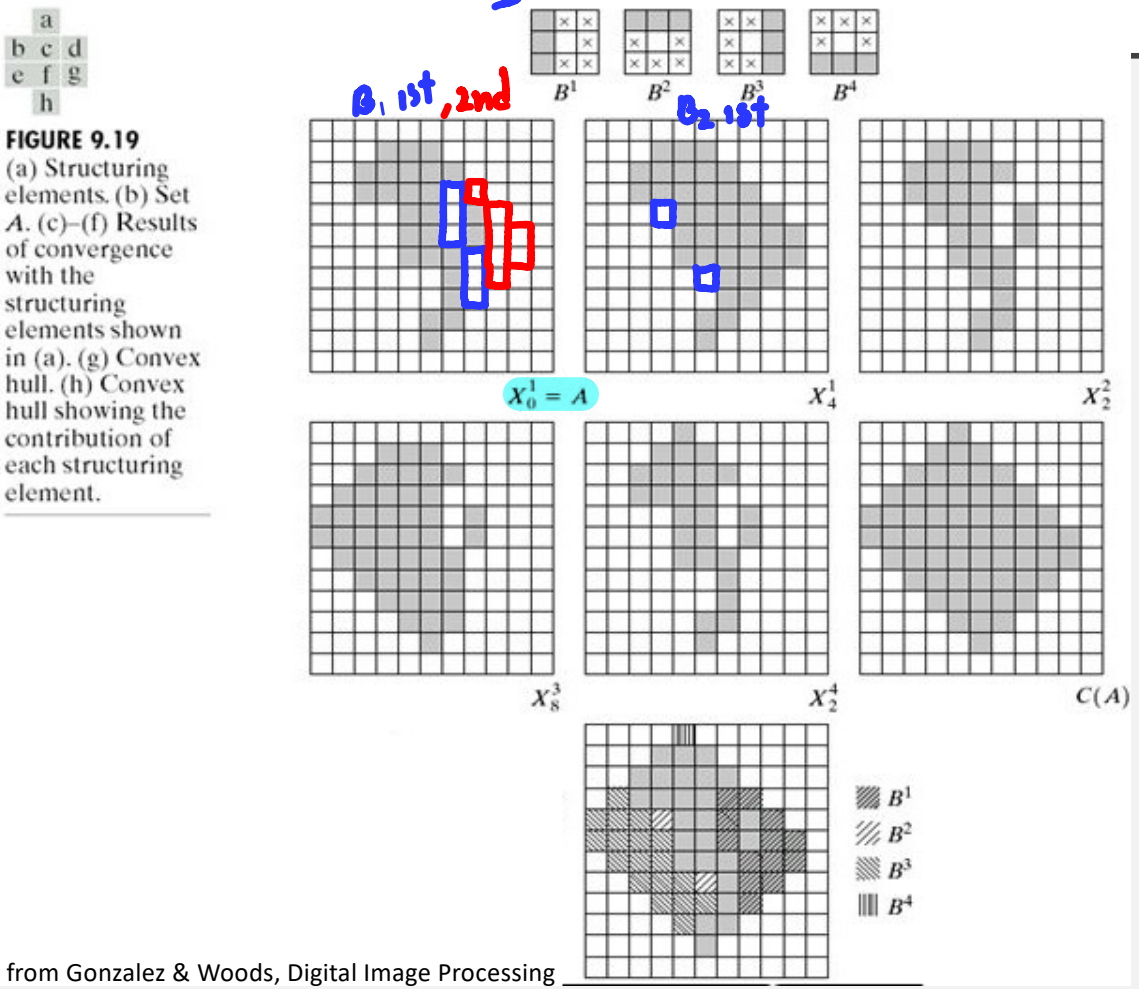


FIGURE 9.19 (a) Structuring elements. (b) Set A. (c)–(f) Results of convergence with the structuring elements shown in (a). (g) Convex hull. (h) Convex hull showing the contribution of each structuring element.

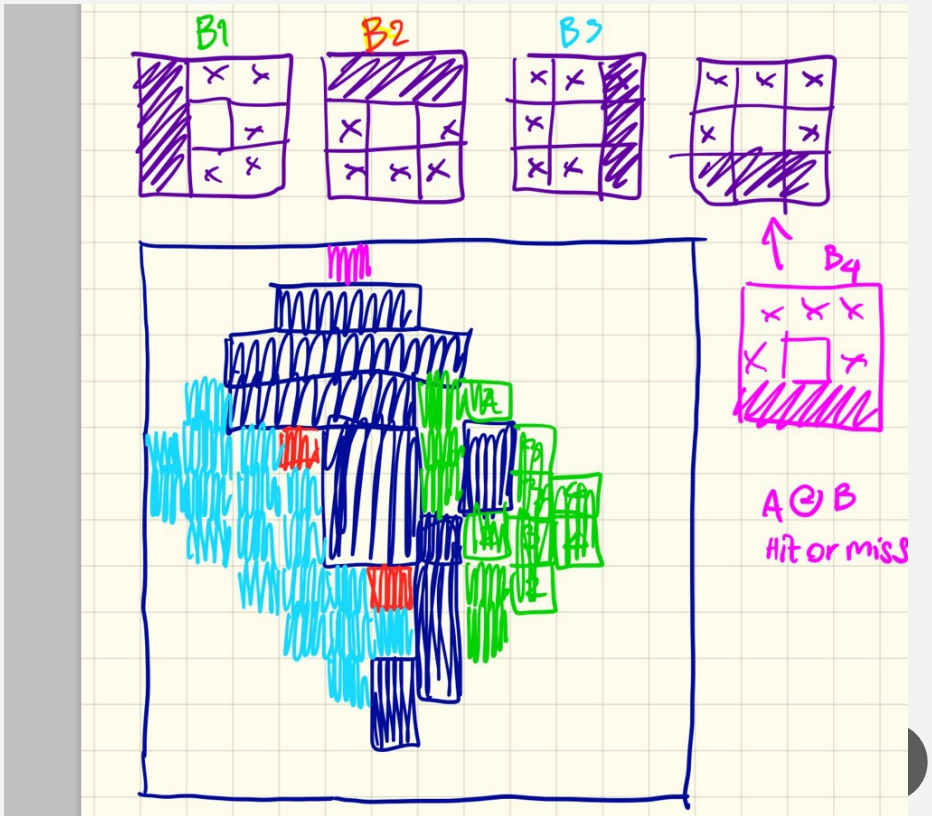


Image from Gonzalez & Woods, Digital Image Processing

THINNING หา แกนเส้น

- Thinning of a set A by a structural element B is denoted $A \otimes B$

$$A \otimes B = A - (A \circledast B)$$ Only pattern matching

$$A \otimes B = A \cap (A \circledast B)^c$$

- A sequence of structural elements:

$$\{B\} = \{B^1, B^2, B^3, \ldots, B^n\}$$ Different rotated versions

$$A \otimes \{B\} = ((\ldots((A \otimes B^1) \otimes B^2)\ldots) \otimes B^n)$$

- The process is to thin A by one pass with B^1 then thin the result by one pass with B^2…. Until B^n repeat until no further changes occur.

THINNING

- Thinning process:

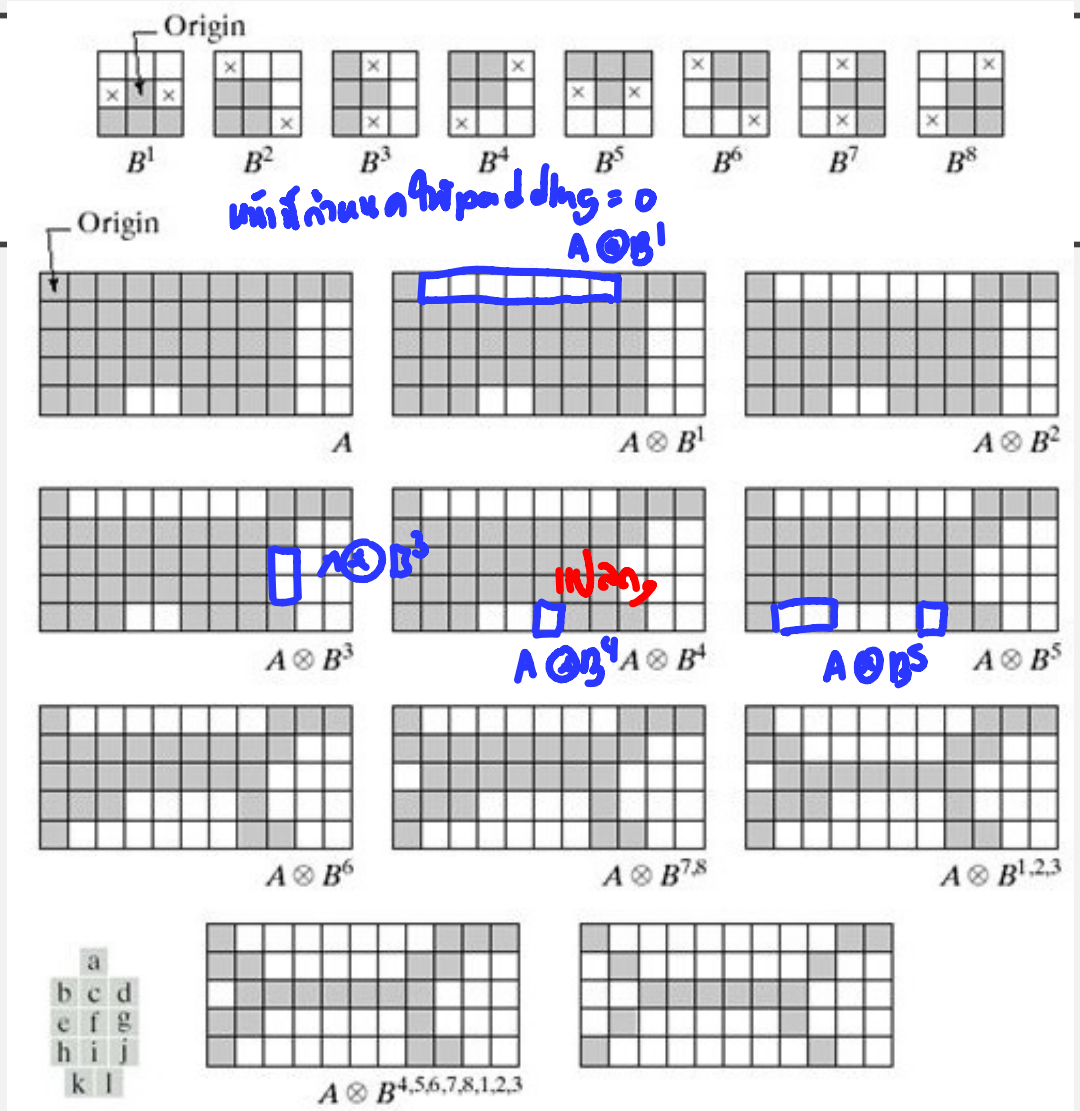


FIGURE 9.21 (a) Sequence of rotated structuring elements used for thinning. (b) Set A. (c) Result of thinning with the first element. (d)–(i) Results of thinning with the next seven elements (there was no change between the seventh and eighth elements). (j) Result of using the first element again (there were no changes for the next two elements). (k) Result after convergence. (l) Conversion to m-connectivity.

Image from Gonzalez & Woods, Digital Image Processing

THICKENING

B : same as thining

- Morphological dual of thinning:

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

$$A \odot \{B\} = ((\ldots((A \circledast B^1) \circledast B^2)\ldots) \circledast B^n)$$

A separate algorithm for thickening is seldom used in practice.
The usual procedure is to thin the background of the set in question and then complement the result.
Depending on the nature of A, may result in disconnected points.

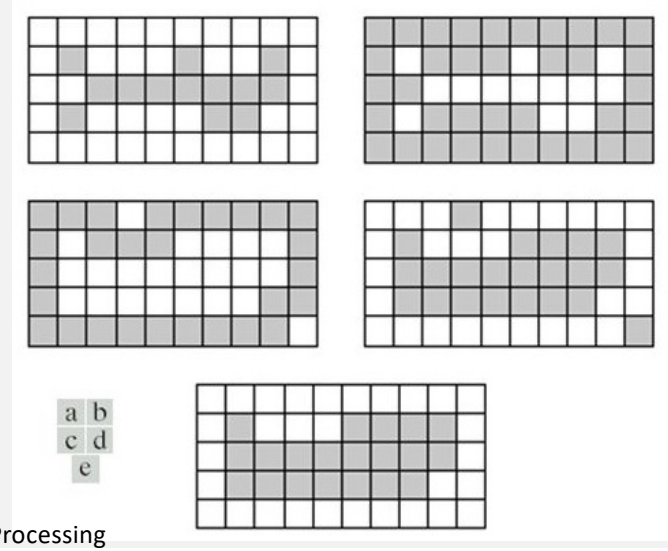


FIGURE 9.22 (a) Set A. (b) Complement of A. (c) Result of thinning the complement of A. (d) Thickened set obtained by complementing (c). (e) Final result, with no disconnected points.

Image from Gonzalez & Woods, Digital Image Processing

SKELETONS

- A skeleton of a set A consists of points z that is the center of a maximum disk.
- The disk touches the boundary of A at two or more different places.
- A maximum disk is a circle in A that cannot be enclosed by another circle that is also in A.
- The figure on the right → shows sets of possible maximum disks and dotted line is the skeleton.

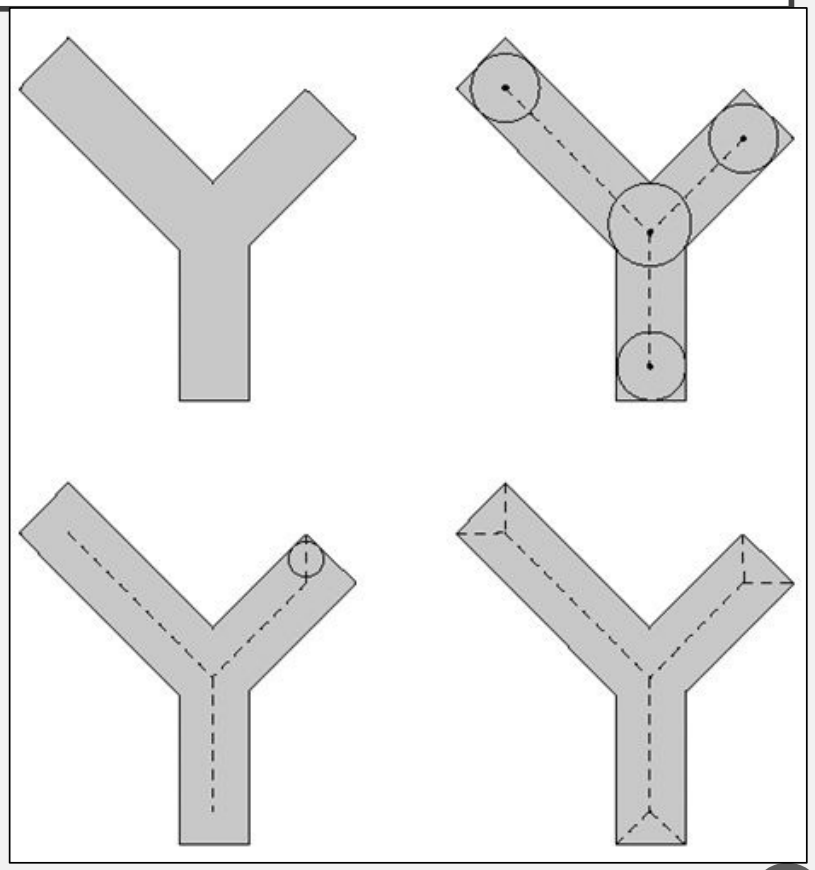
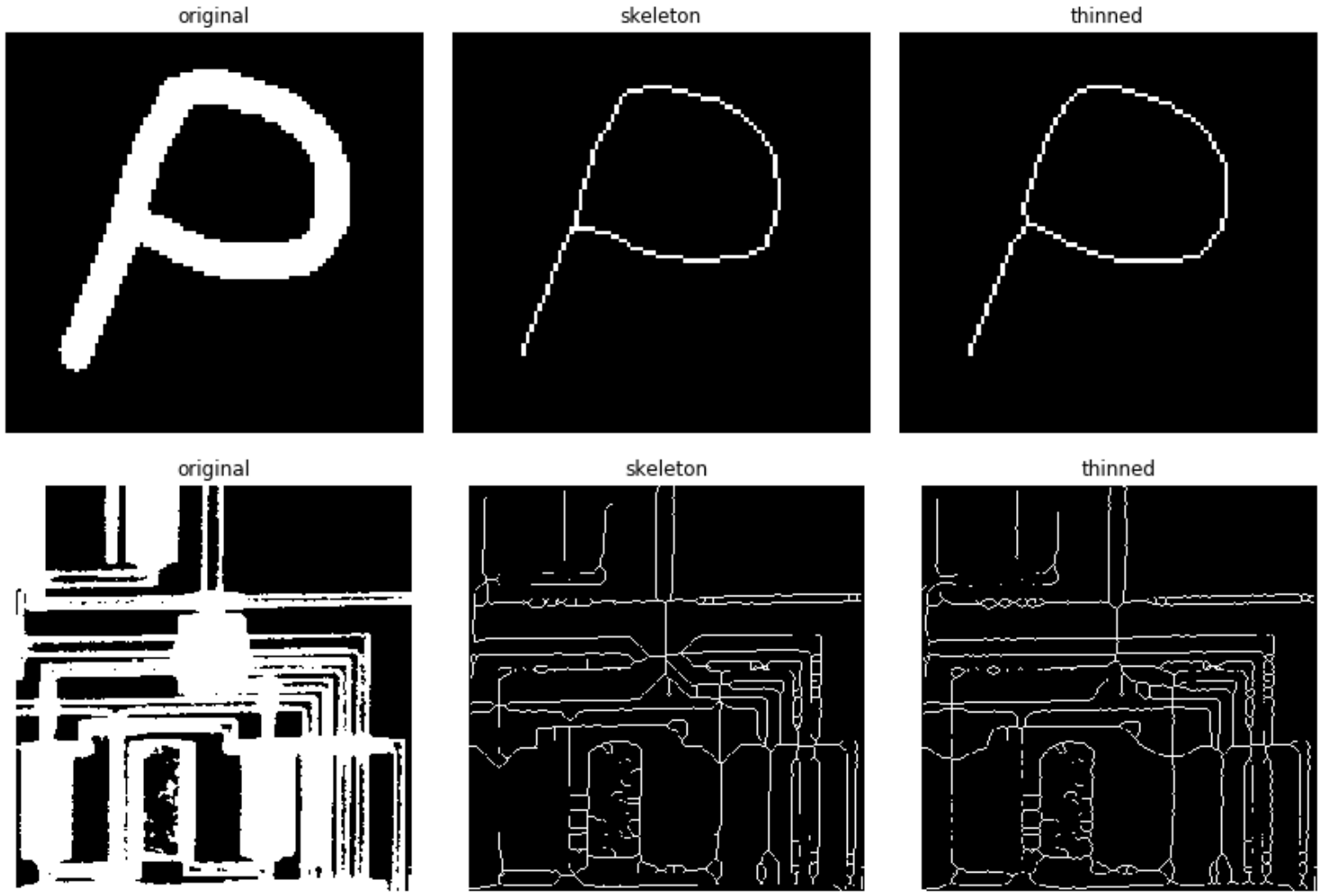


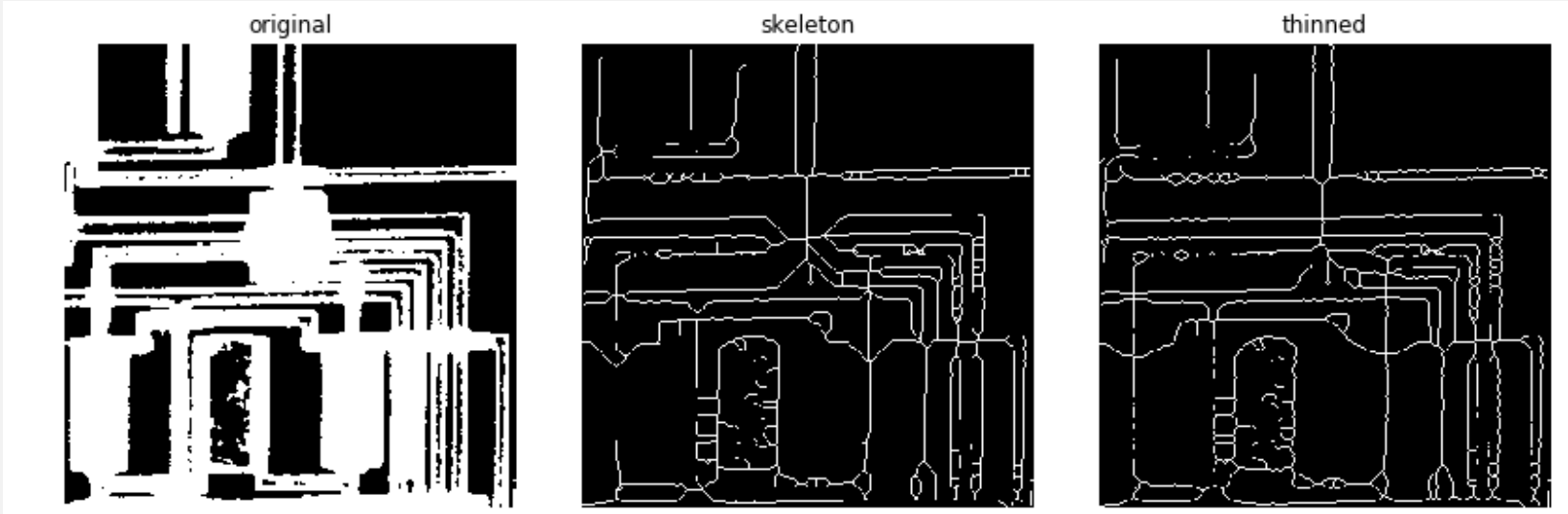
Image from Gonzalez & Woods, Digital Image Processing

SKELETON VS. THINNING



SKELETON VS. THINNING

```
from skimage.morphology import skeletonize, thin

image  = cv2.imread("circuitbw.png",0)
th, image = cv2.threshold(image,127,1,cv2.THRESH_BINARY)
skeleton = skeletonize(image)
thinned = thin(image)
```



PRUNING

- An essential complement to thinning and skeletonizing algorithms because these procedures tend to leave parasitic components (cleaned up by post-processing).
- To handle spurs caused during erosion e.g., “a” character →
- Suppressing the spurs by successively eliminating its end point.

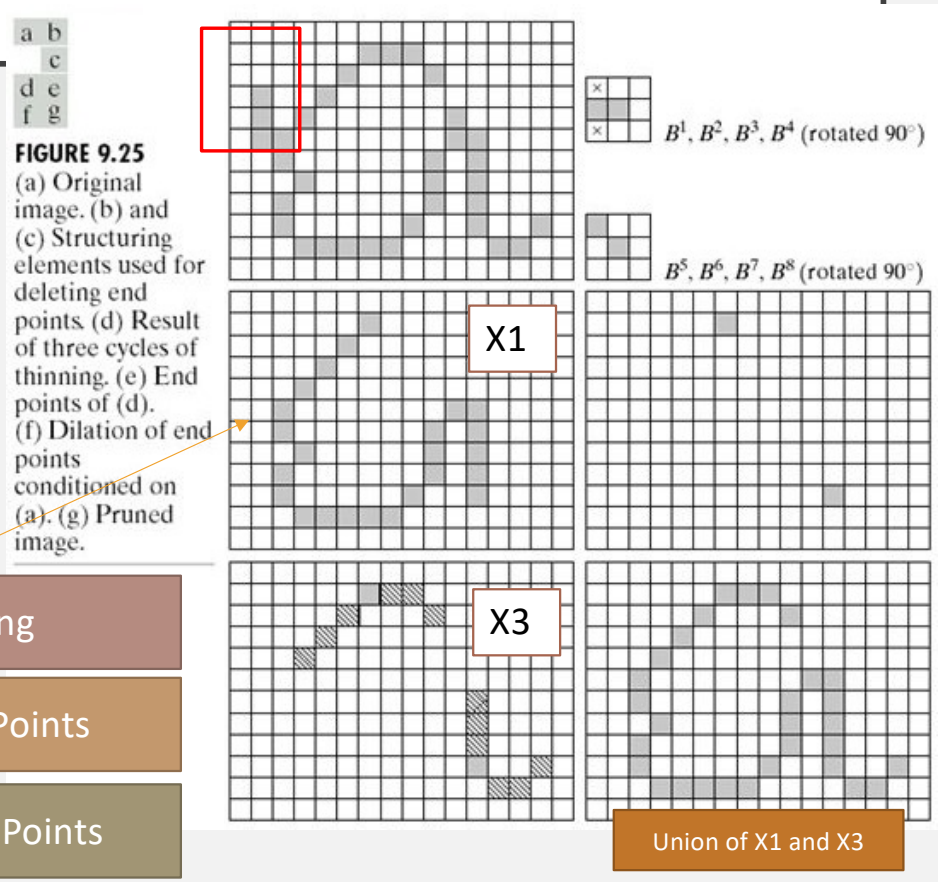
3 times all the spurs are gone

Thinning

Find End Points

Dilate End Points

Union of X1 and X3



Skeleton of a hand-printed “a”

Image from Gonzalez & Woods, Digital Image Processing

EXTENSION TO GRAYSCALE IMAGES

D_f and D_b are domains of f and b

- Dilation : ขยายขึ้น เลือกค่า max

$$(f \oplus b)(s,t) = \max\{f(s-x, t-y) + b(x,y) | (s-x), (t-y) \in D_f; (x,y) \in D_b\}$$

- Erosion : หดลง เลือกค่า min

Similar to 2D convolution with max operation replacing sum and sum replacing the products

$$(f \ominus b)(s,t) = \min\{f(s+x, t+y) - b(x,y) | (s+x), (t+y) \in D_f; (x,y) \in D_b\}$$

- Opening

Similar to 2D filtering with min operation replacing sum and subtraction replacing the products

$$f \circ b = (f \ominus b) \oplus b$$

- Closing

$$f \cdot b = (f \oplus b) \ominus b$$

Dilation	$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$
Erosion	$A \ominus B = \{z \mid (B)_z \subseteq A\}$
Opening	$A \circ B = (A \ominus B) \oplus B$
Closing	$A \bullet B = (A \oplus B) \ominus B$

EXTENSION TO GRAYSCALE IMAGES

- Erosion and Dilation in Grayscale

Original Image

Erosion

Dilation

APPLICATIONS OF GRAY-SCALE MORPHOLOGY

- Morphological smoothing
  - Opening followed by closing to remove or attenuate both bright and dark artifacts or noise.
- Morphological gradient
  - Dilation and erosion are used to compute morphological gradient
  - $g = (f \oplus g) - (f \ominus g)$

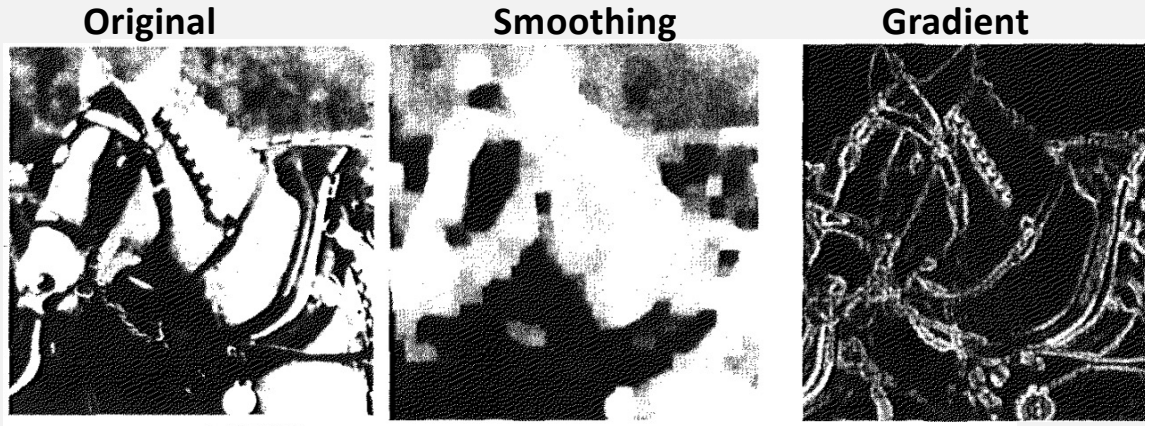


FIGURE 9.32 Morphological smoothing of the image in Fig. 9.29(a). (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

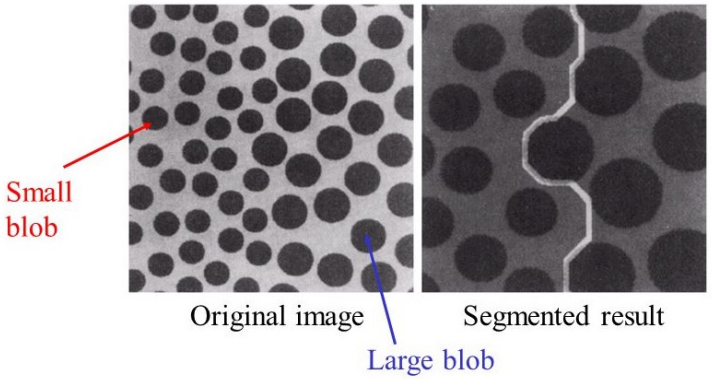
Image from Gonzalez & Woods, Digital Image Processing

APPLICATIONS OF GRAY-SCALE MORPHOLOGY

- Textural segmentation
  - Two texture regions in figure below (circular blobs with different diameters).
  - The objective is to find the boundary between two regions based on their textural content.

a b

FIGURE 9.35 (a) Original image. (b) Image showing boundary between regions of different texture. (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)



Algorithm:

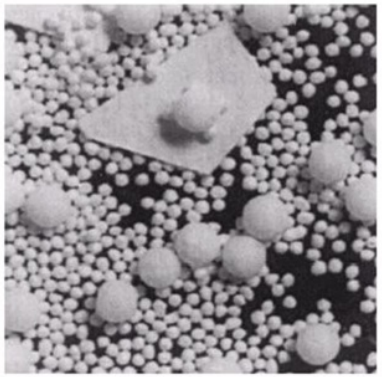
1. Perform opening on the image by using successively larger structuring elements until small blobs are vanished.
2. Perform closing to join large blobs together
3. Perform intensity thresholding

APPLICATIONS OF GRAY-SCALE MORPHOLOGY

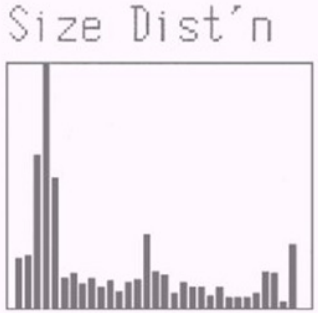
- Granulometry 입자크기분포
  - To determine size distribution of particles in an image.

Method:

1. Perform opening using structuring elements of increasing size
2. Compute the difference between the original image and the result after each opening operation
3. The differenced image obtained in Step 2 are normalized and used to construct the size-distribution graph.



Original image



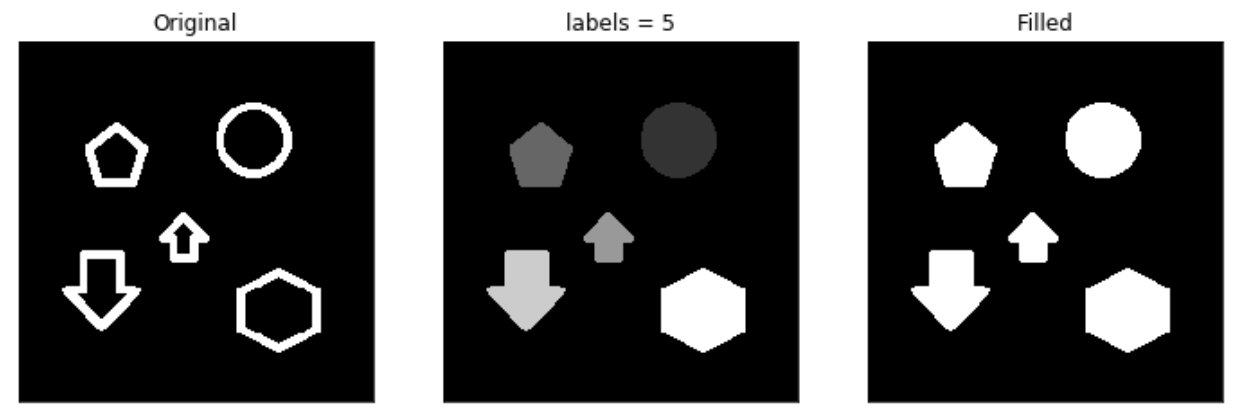
Size distribution graph

a b

FIGURE 9.36
(a) Original image consisting of overlapping particles; (b) size distribution. (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

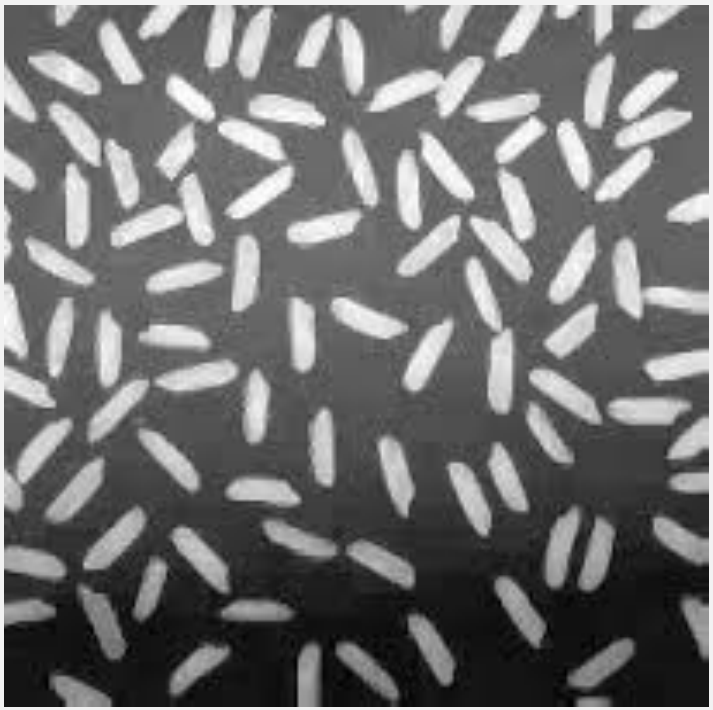
EXERCISE #1:

- Fill The region and count the number of objects.
- How many Labels?
- Let’s calculate!



EXERCISE #2:

- Using morphological operation to count the number of ‘bacteria.png’ below:
- How do you measure the performance?



REFERENCES

- Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison- Wesley
  - Chapter 9 Morphological Image Processing