Contents

- **Chap 9. Image Segmentation**
  - Single & double thresholding
  - How to determine the threshold value
  - Adaptive thresholding
  - Edge detection: 1st and 2nd derivatives
  - Canny edge detector
  - Hough Transform
- **Chap 10. Mathematical Morphology**
  - Erosion & dilation -> boundary detection
  - Opening & closing -> noise removal
  - Hit-or-miss transform -> shape detection
  - Binary applications: region filling, connected components, skeletons
  - Grayscale morphology -> edge detection, noise removal

Chapter 10:

Mathematical Morphology

10.1 Introduction

- *Morphology* is a branch of image processing that is particularly useful for analyzing **shapes** in images.
- We will develop basic morphological tools for investigation of **binary images** and then show how to extend these tools to **grayscale images.**
- In this chapter, we will learn,
  - What is morphology?
  - Simple morphological operations: erosion, dilation
  - Compound operations: opening, closing
  - Morphological algorithms: boundary detection, region filling, connected components, skeleton

Translation & Reflection

- Translation

$$A_w = \{(a,b) + (x,y) : (a,b) \in A\}$$

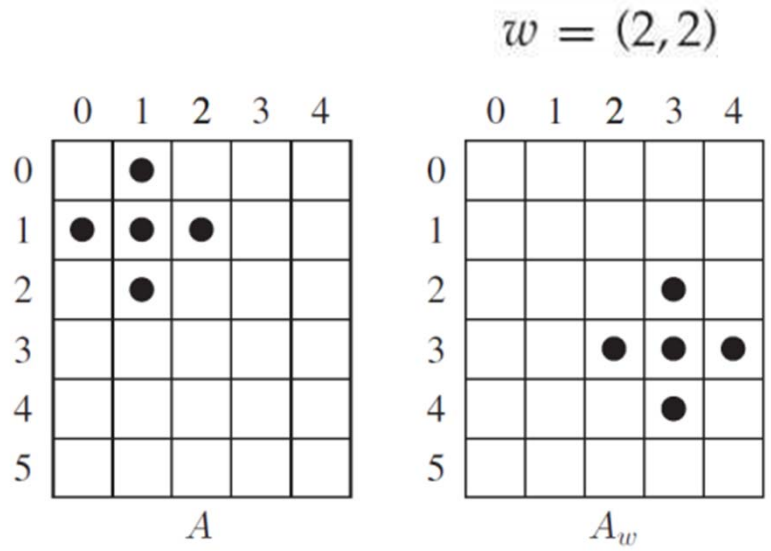


FIGURE 10.1 *Translation.*

- Reflection

$$\hat{A} = \{(-x,-y) : (x,y) \in A\}$$

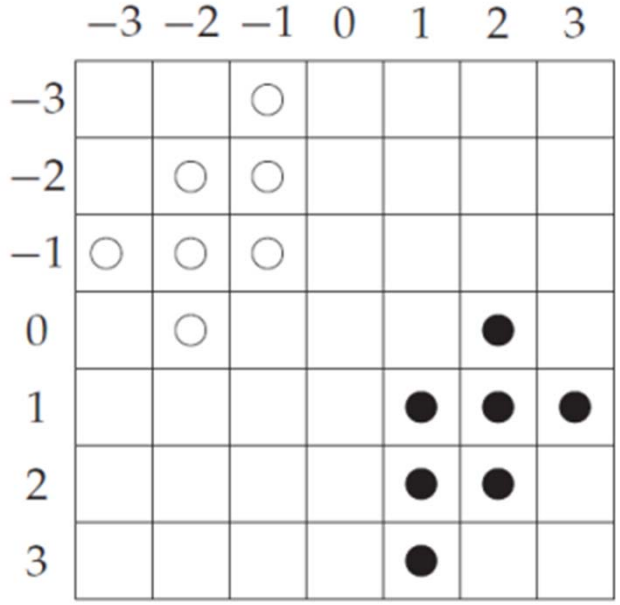


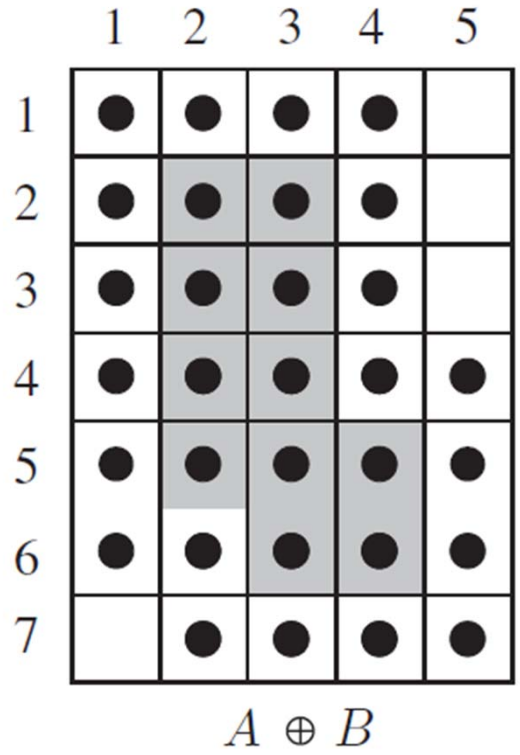
FIGURE 10.2 *Reflection.*

Dilation

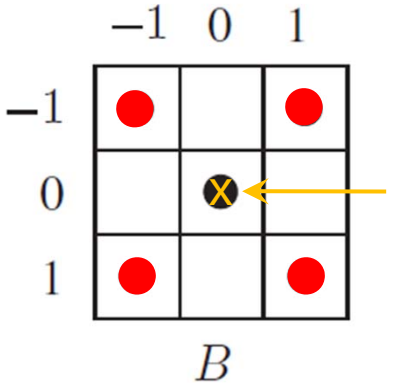
- A and B are sets of pixels
- Also known as **Minkowski addition**

$$A \oplus B = \{(x,y) + (u,v) : (x,y) \in A, (u,v) \in B\}$$

$$A \oplus B = \bigcup_{x \in B} A_x$$



structuring element (SE)



Center of SE should be specified.

FIGURE 10.3 *Dilation.*

Dilation in Matlab

```
>> t=imread('text.tif');
>> sq=ones(3,3);
>> td=imdilate(t,sq);
>> subplot(1,2,1),imshow(t)
>> subplot(1,2,2),imshow(td)
```

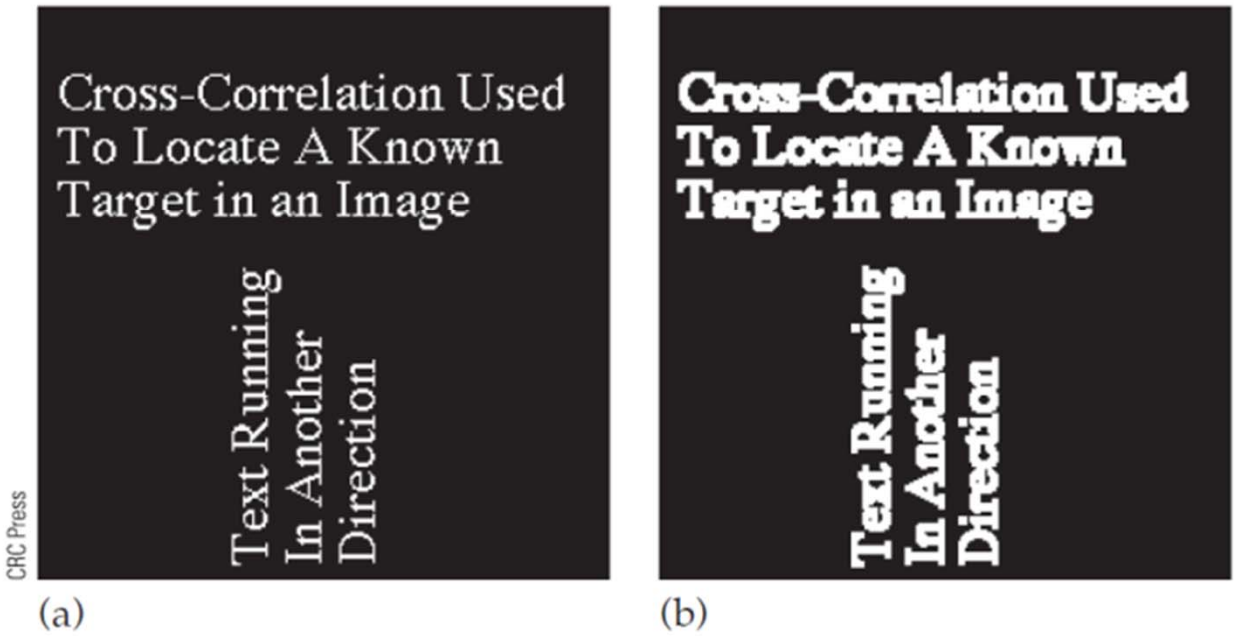


FIGURE 10.5 *Dilation of a binary image. (a) Text image. (b) Result of dilation.*

Erosion

- Also known as Minkowski substraction $A \ominus B = \{w : B_w \subseteq A\}$

$$A - B = \bigcap_{b \in B} A_b$$

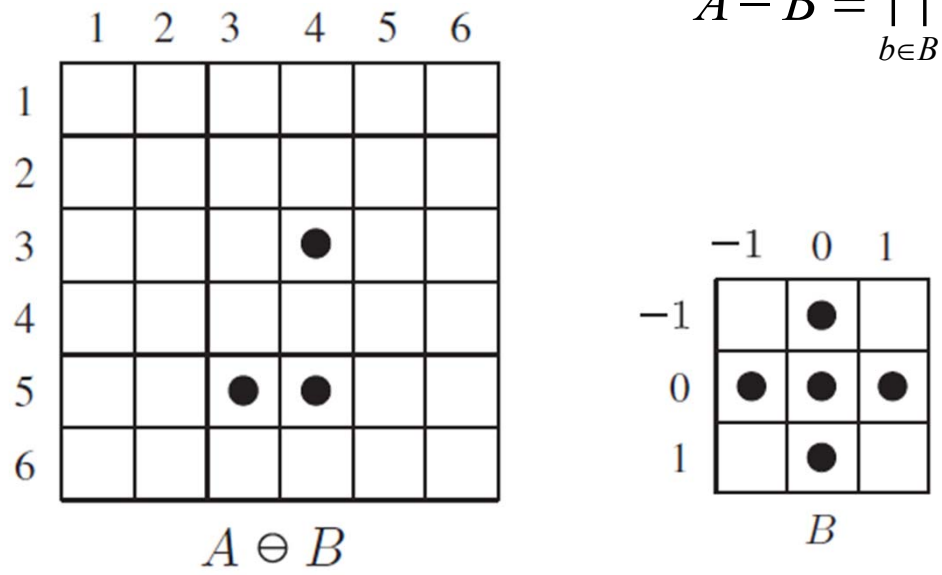


FIGURE 10.6 *Erosion with a cross-shaped structuring element.*

Erosion in Matlab

```
>> c=imread('circbw.tif');
>> ce=imerode(c,sq);
>> subplot(1,2,1),imshow(c)
>> subplot(1,2,2),imshow(ce)
```

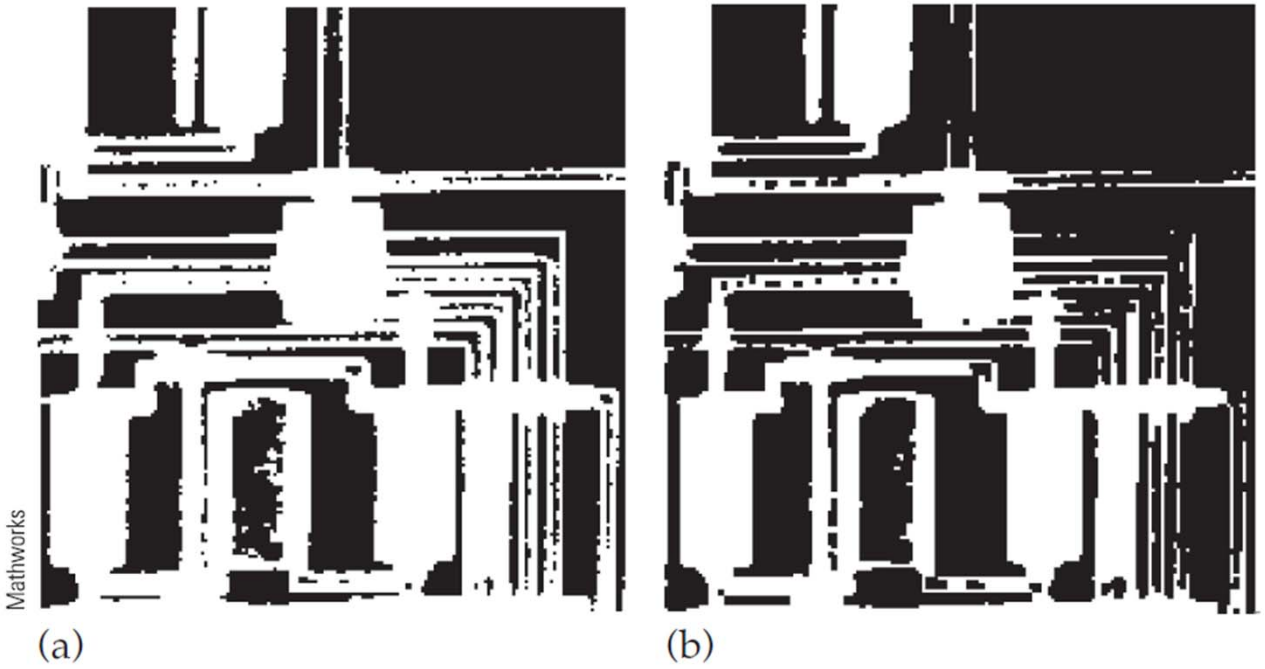


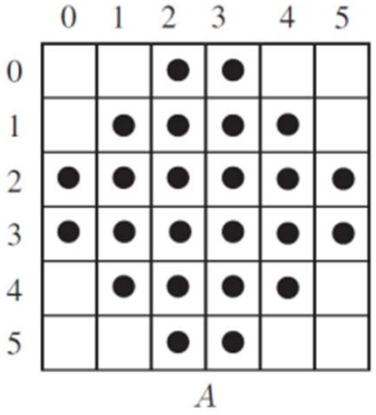
FIGURE 10.8 *Erosion of a binary image. (a) Original image. (b) Result of erosion.*

dilation과 erosion을 순차적으로 적용하면 boundary detection 이 가능

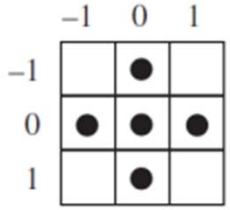
Application of Dilation and Erosion: Boundary Detection

- If A is an image and B a small structuring element,

(i) $A - (A \ominus B)$ internal boundary
(ii) $(A \oplus B) - A$ external boundary
(iii) $(A \oplus B) - (A \ominus B)$ morphological gradient

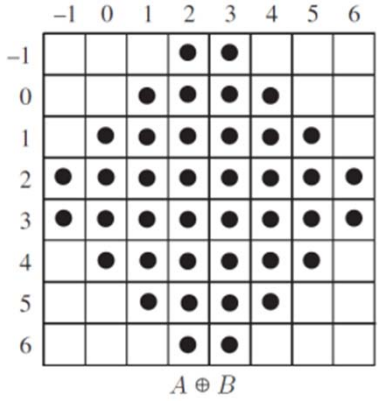


A

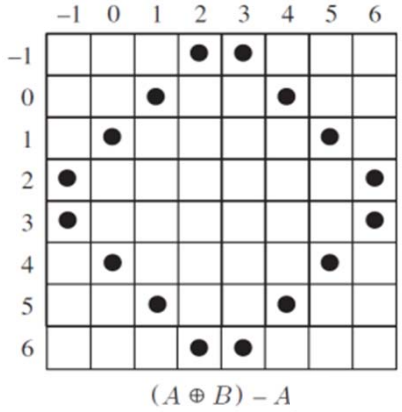


B

external boundary detection: outside of the object

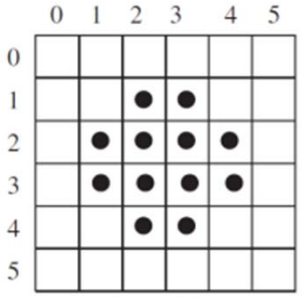


$A \oplus B$

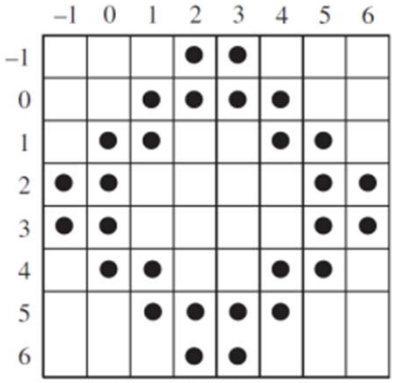


$(A \oplus B) - A$

morphological gradient: both int. and ext. boundaries



$(A \ominus B)$



$(A \oplus B) - (A \ominus B)$

Internal Boundary Detection in Matlab

```
>> re=imerode(r,sq);
>> r_int=r&~re;
>> subplot(1,2,1),imshow(r)
>> subplot(1,2,2),imshow(r_int)
```

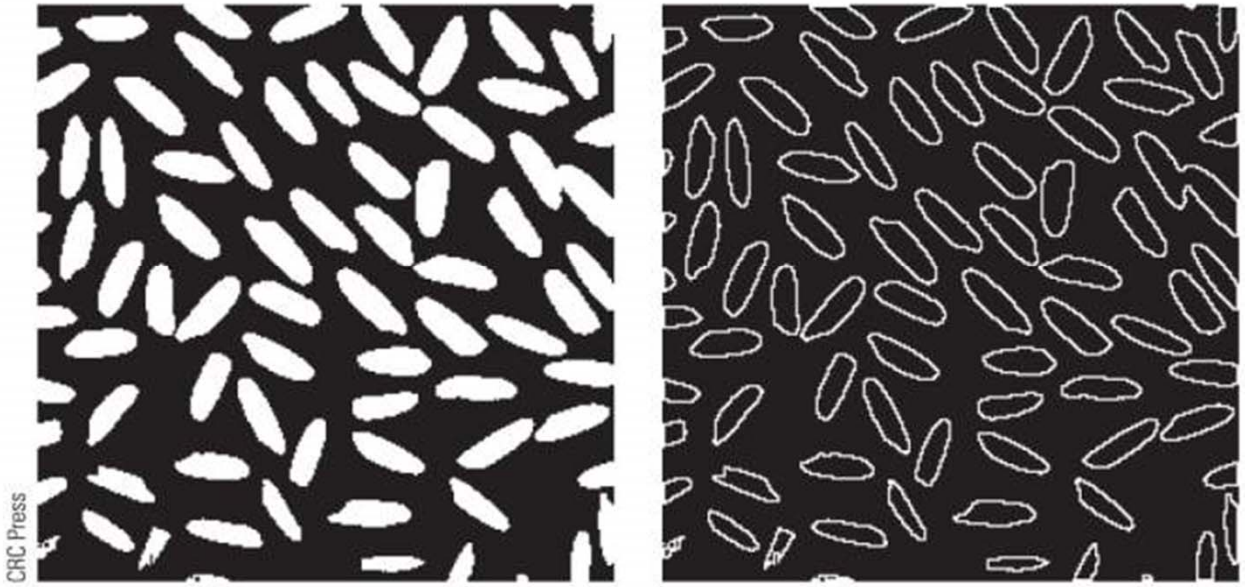


FIGURE 10.10 *Morphological edge detection. (a) The rice grains image. (b) The internal boundary.*

External and Gradient Boundary Detection

```
>> rd=imdilate(r,sq);
>> r_ext=rd&~r;
>> r_grad=rd&~re;
>> subplot(1,2,1),imshow(r_ext)
>> subplot(1,2,2),imshow(r_grad)
```

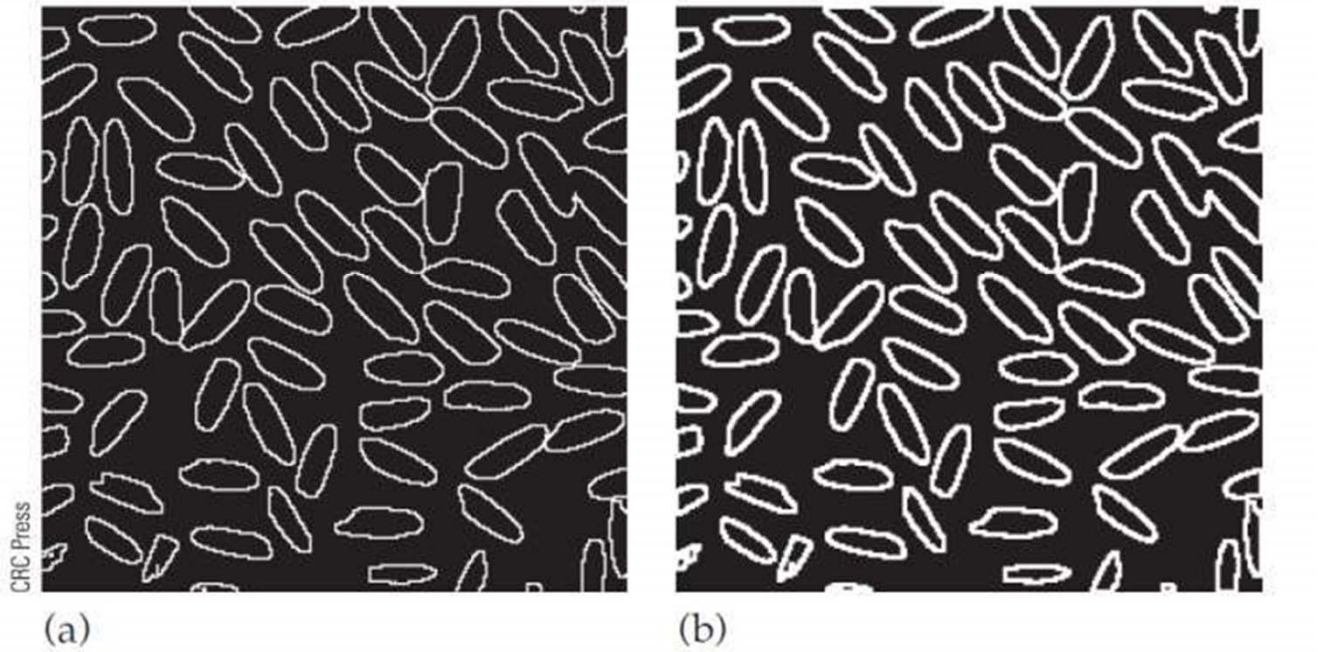


FIGURE 10.11 *More morphological edge detection. (a) External boundary. (b) Morphological gradient.*

Opening

- **Opening** (function: `imopen( )`)

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

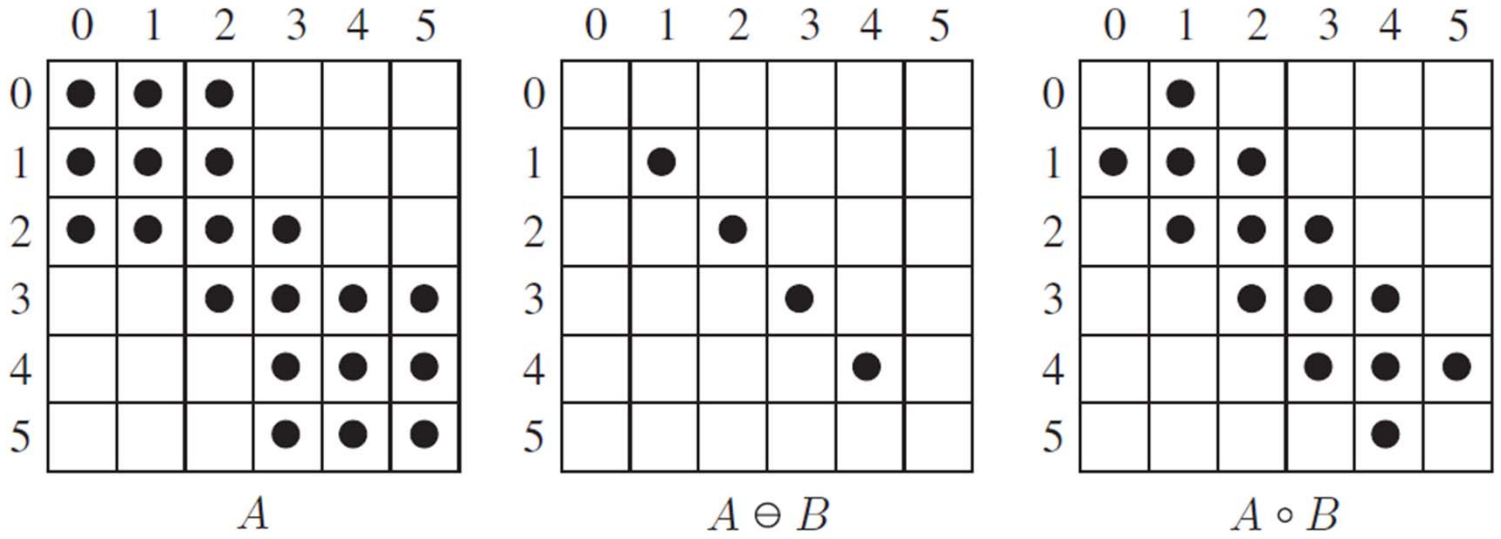
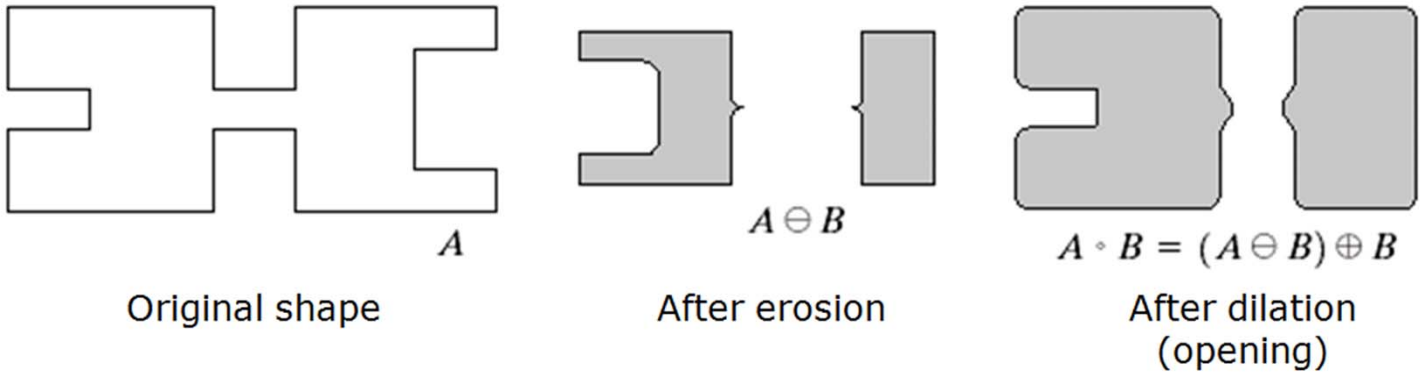


FIGURE 10.12 *Opening.*

Properties of Opening

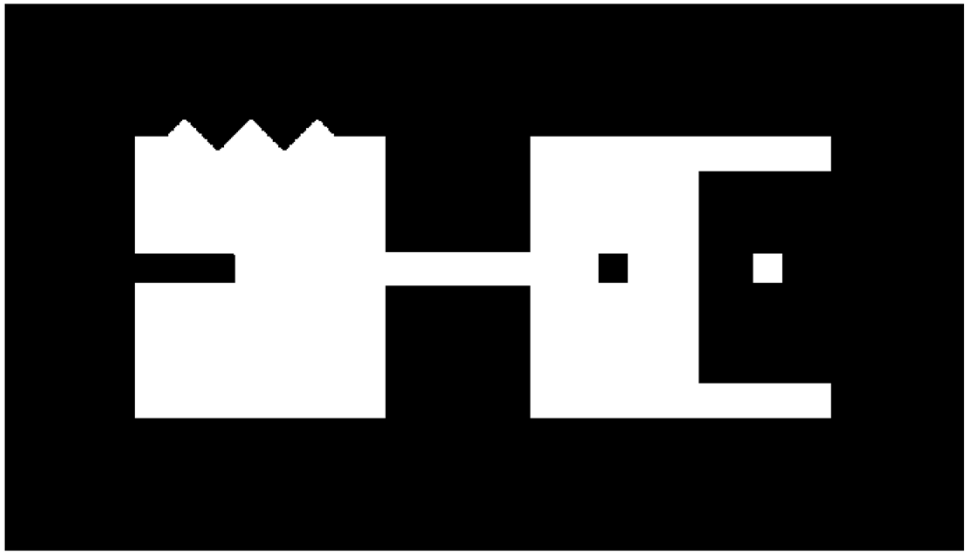
- ✓ $(A \circ B) \subseteq A$
  - ✓ Note that this is not the case with erosion. As we have seen, an erosion may not necessarily be a subset.
- ✓ $(A \circ B) \circ B = A \circ B$
  - ✓ That is, an opening can never be done more than once (**idempotent**).
- ✓ If $A \subseteq C$, then $(A \circ B) \subseteq (C \circ B)$
- ✓ Opening tends to smooth an image, to break narrow joins, and to remove thin protrusions.



약하게 연결된 것들을 끊어줌

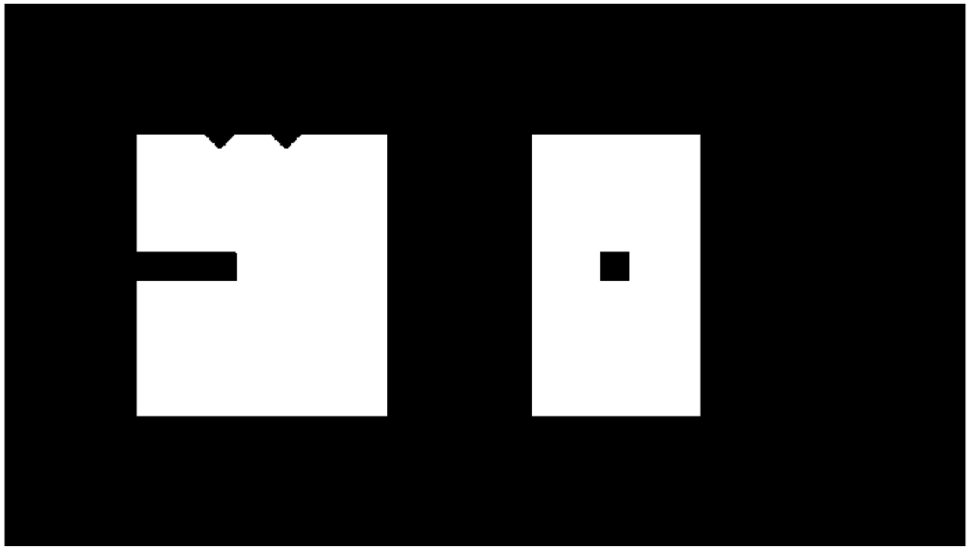
Opening Example

Original Image



se

Image After Opening



Closing

- **Closing** (function: `imclose()`)

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

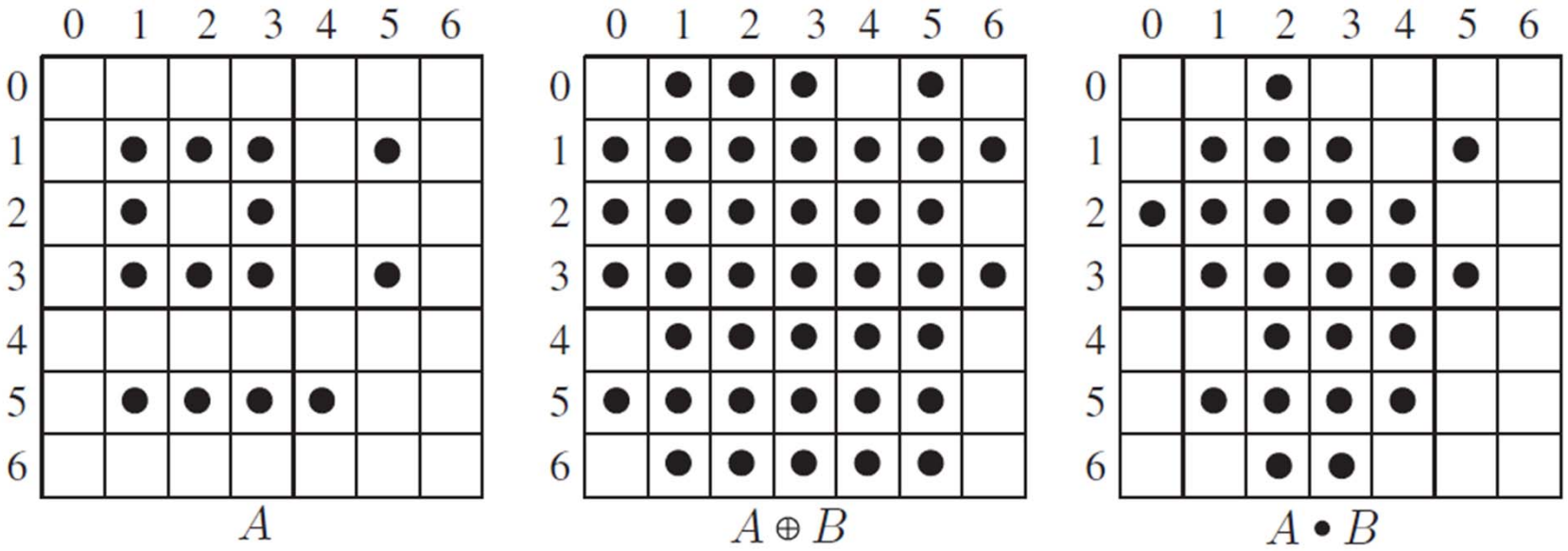
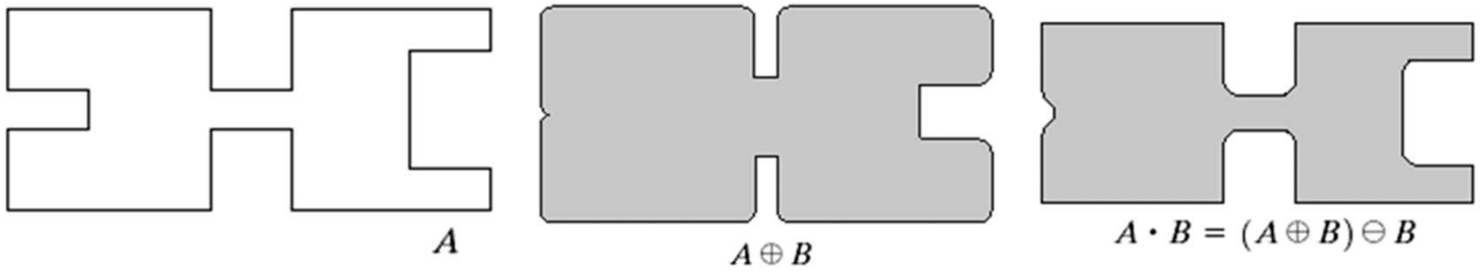


FIGURE 10.13 *Closing.*

Properties of Closing

- ✓ $A \subseteq (A \bullet B)$
- ✓ $(A \bullet B) \bullet B = A \bullet B$
  - ✓ That is, closing, like opening, is **idempotent.**
- ✓ If $A \subseteq C$, then $(A \bullet B) \subseteq (C \bullet B)$
- ✓ Closing also tends to smooth an image, but it fuses narrow breaks and thin gulfs and eliminates small holes.

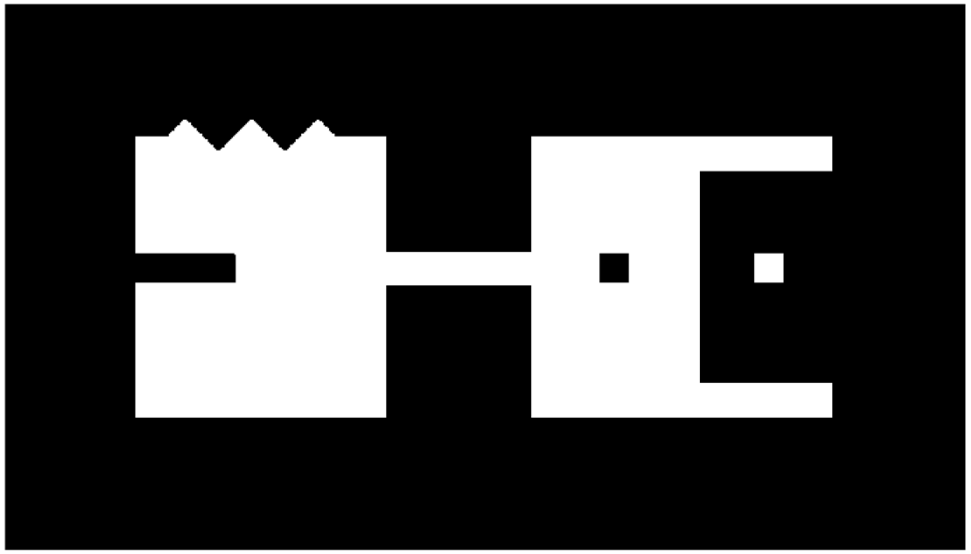


Original shape | After dilation | After erosion (closing)

약하게 연결되어 있던 것들을 더 강하게 이어줌

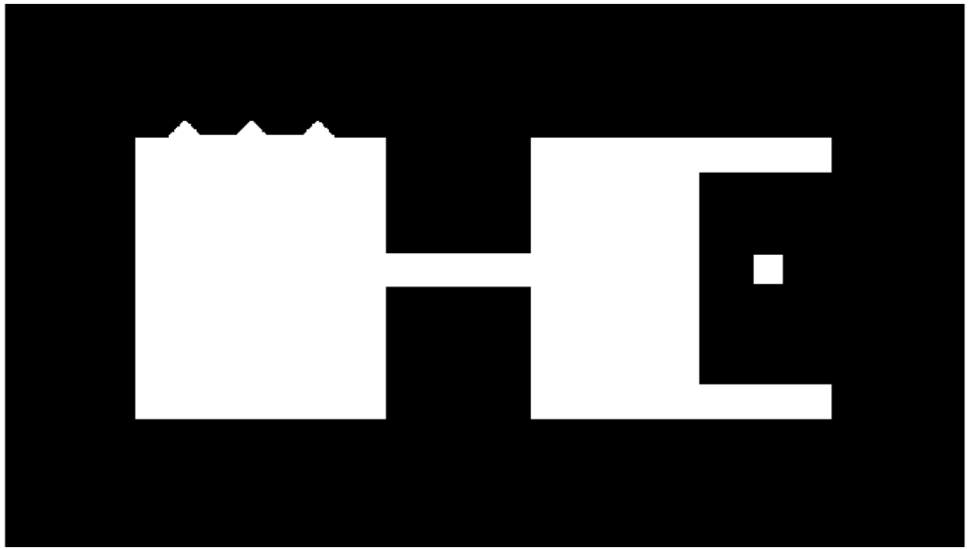
Closing Example

Original Image



se

Image After Closing

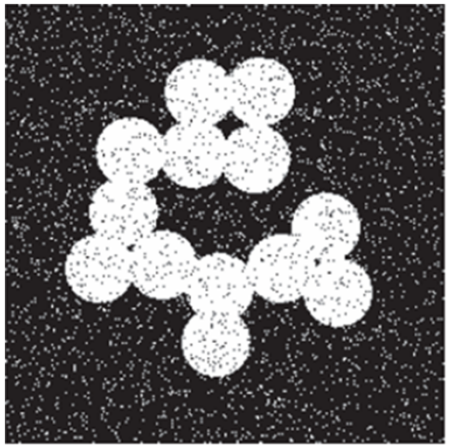


Application of Opening and Closing: Noise Removal

- Morphological filtering

```
>> c=imread('circles.tif');
>> x=rand(size(c));
>> d1=find(x<=0.05);
>> d2=find(x>=0.95);
>> c(d1)=0;
>> c(d2)=1;
>> imshow(c)
```

```
>> cf1=imclose(imopen(c,sq),sq);
>> figure,imshow(cf1)
>> cf2=imclose(imopen(c,cr),cr);
>> figure,imshow(cf2)
```



square SE

cross SE

Hit-or-Miss Transform

- Very useful for finding same shapes in the image
- defined with dilation and erosion
- If *B* is the 3x3 square structuring element,

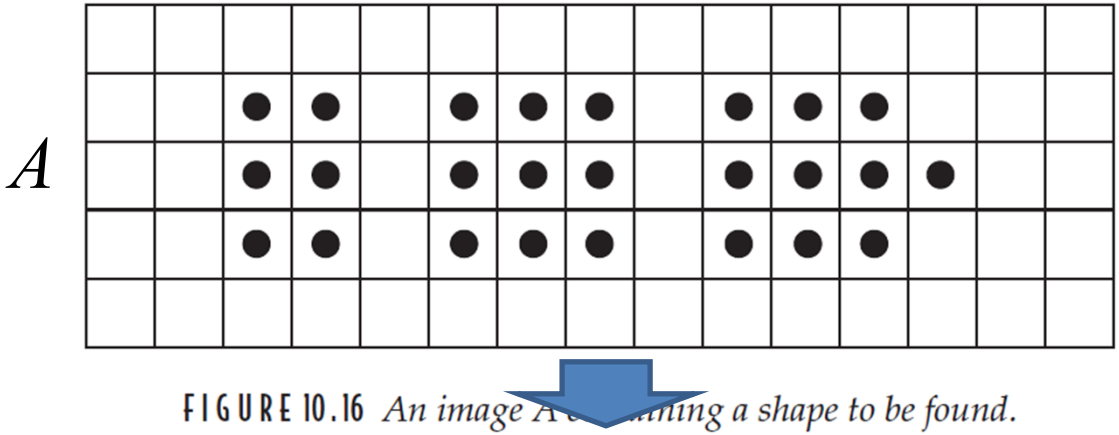
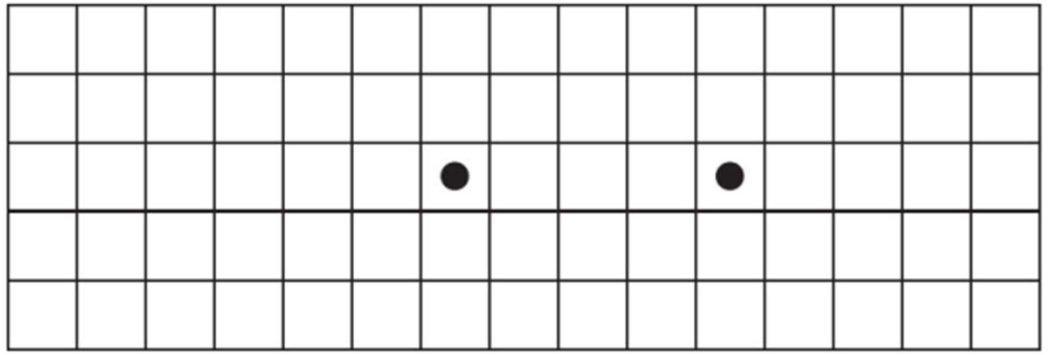
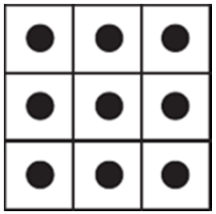


FIGURE 10.16 *An image A containing a shape to be found.*





B
square SE

FIGURE 10.17 *The erosion* $A \ominus B$.

Hit-or-Miss Transform

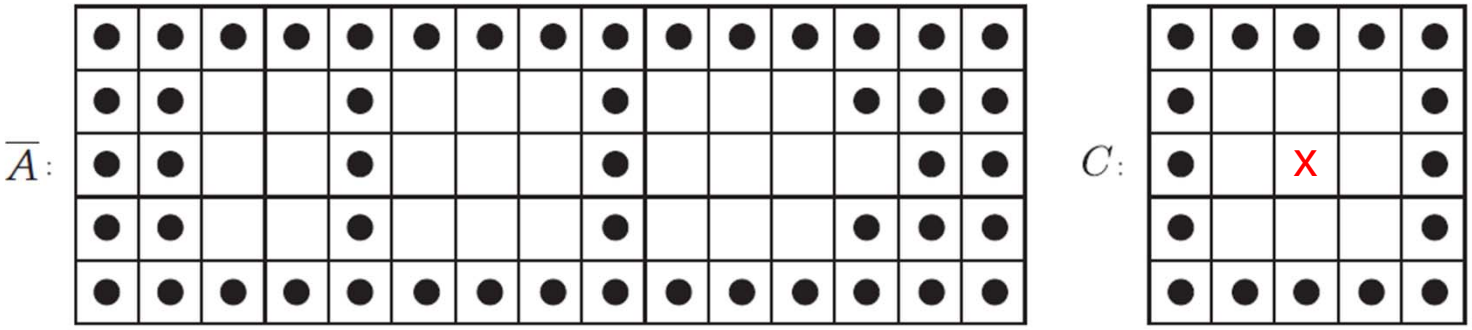
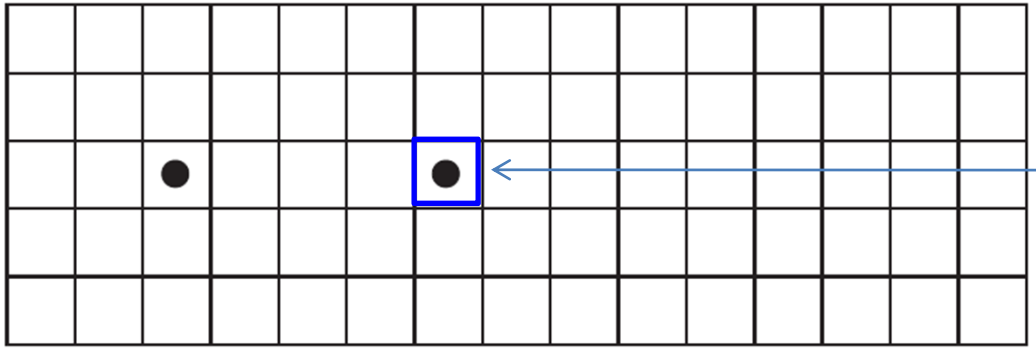


FIGURE 10.18 *The complement and the second structuring element.*



a point of detected object which has the same shape as B

FIGURE 10.19 *The erosion* $\overline{A} \ominus C$.

$$A \circledast B = (A \ominus B_1) \cap (\overline{A} \ominus B_2)$$

In this example, $B_1 = B$, $B_2 = C$.

Hit-or-Miss Transform

- In general, if we are looking for a particular shape in an image, we can design two structuring elements, B1 and B2.
- B1 should be the same shape that we want to find.
- B2 should fit around the shape.

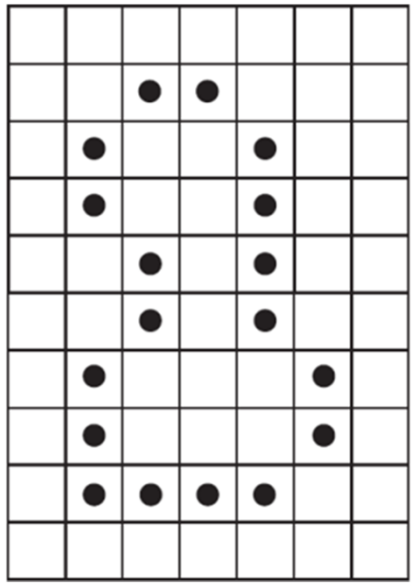
$$A \circledast B = (A \ominus B_1) \cap (\overline{A} \ominus B_2)$$

```
>> b1=ones(1,6);
>> b2=[1 1 1 1 1 1 1 1;1 0 0 0 0 0 0 1; 1 1 1 1 1 1 1 1];
>> tb1=imerode(t,b1);
>> tb2=imerode(~t,b2);
>> hit_or_miss=tb1&tb2;
>> [x,y]=find(hit_or_miss==1)
```

```
>> tb1=imerode(t,b1);
```

Region Filling

- How to fill inside of the boundary ?



- Given a pixel p within the region, we wish to fill up the entire region.

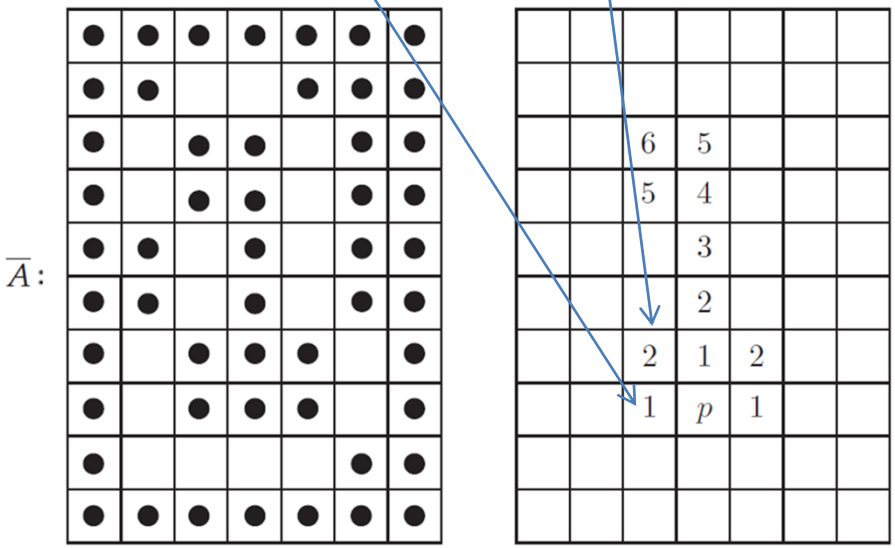
1. Dilate p with the cross-shaped structuring element B.
2. Take an intersection with complement of A.

$$\{p\} = X_0, X_1, X_2, \ldots, X_k = X_{k+1},$$

$$X_n = (X_{n-1} \oplus B) \cap \overline{A}$$

3. Repeat 1 and 2 until the result image is converged. $X_n = X_{n-1}$

$X_0 = \{p\}$, $X_1 = \{p,1\}$, $X_2 = \{p,1,2\}$, ...



Region Filling in Matlab: **`regfill()`**

```
function out=regfill(im,pos,kernel)
% REGFILL(IM,POS,KERNEL) performs region filling of binary
% image IMAGE,with kernel KERNEL, starting at point with
% coordinates given by POS.
% Example:
%           n=imread('nicework.tif');
%           nb=n&~imerode(n,ones(3,3));
%           nr=regfill(nb,[74,52],ones(3,3));
%
current=zeros(size(im));
last=zeros(size(im));
last(pos(1),pos(2))=1;
current=imdilate(last,kernel)&~im;
while any(current(:)~=last(:)),
  last=current;
  current=imdilate(last,kernel)&~im;
end;
out=current;
```

FIGURE 10.24 *A simple program for filling regions.*

Region Filling in Matlab

```
>> n=imread('nicework.tif');
>> imshow(n),pixval on
>> nb=n&~imerode(n,sq);
>> figure,imshow(nb)
>> nf=regfill(nb,[74,52],sq);
>> figure,imshow(nf)
```

FIGURE 10.25 *Region filling.*

Connected Components

- Cross shape SE for 4-connected components
- Square shape SE for 8-connected components

$$X_0 = p$$

$$X_n = (X_{n-1} \oplus B) \cap A, \quad n = 1,2,3,\ldots \quad \text{Until} \quad X_n = X_{n-1}$$

It is not A^c !

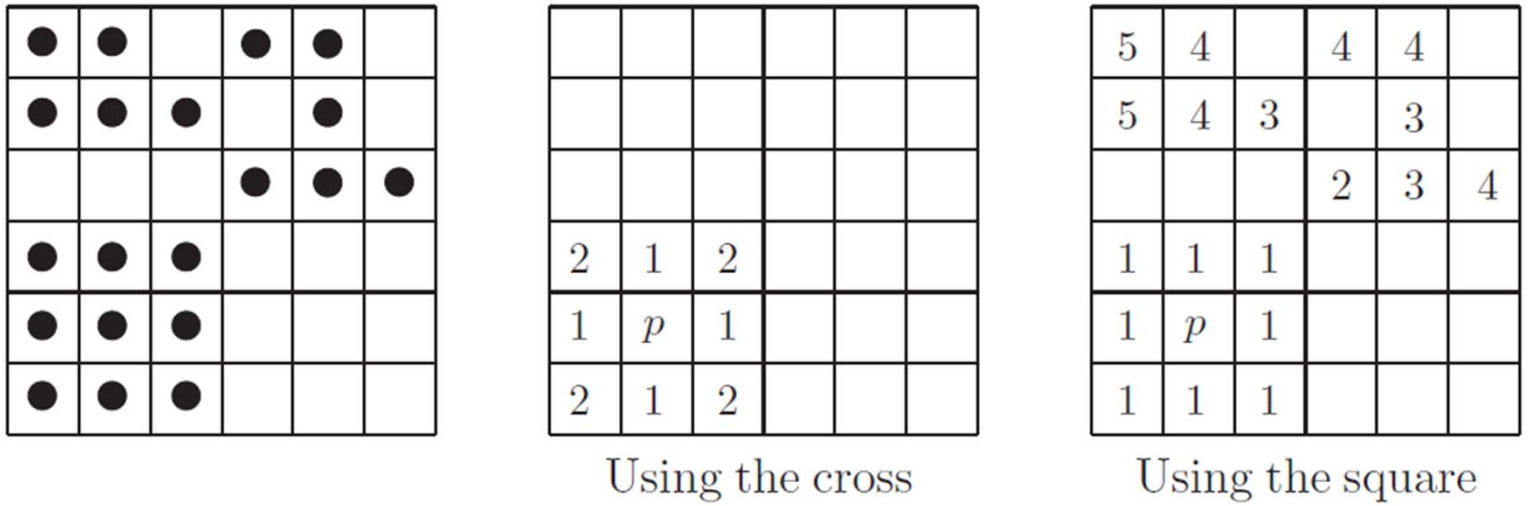


FIGURE 10.23 *Filling connected components.*

Connected Components in Matlab: **components()**

```
function out=components(im,pos,kernel)
% COMPONENTS(IM,POS,KERNEL) produces the connected component
% of binary image IMAGE which nicludes the point with coordinates given
% by POS, using kernel KERNEL.
%
% Example:
%           n=imread('nicework.tif');
%           nc=components(nb,[74,52],ones(3,3));
%
current=zeros(size(im));
last=zeros(size(im));
last(pos(1),pos(2))=1;
current=imdilate(last,kernel)&im;
while any(current(:)~=last(:)),
  last=current;
  current=imdilate(last,kernel)&im;
end;
out=current;
```

FIGURE 10.26 *A simple program for connected components.*

Connected Components in Matlab

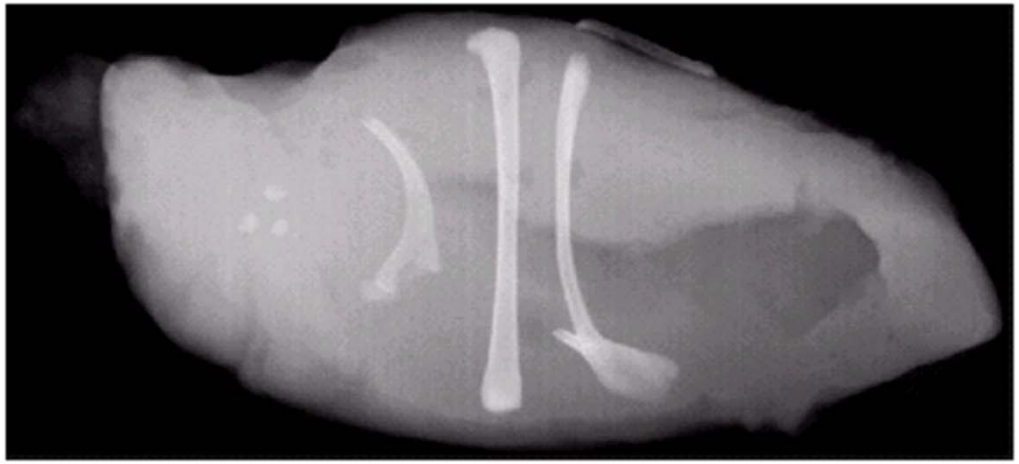
```
>> sq2=ones(11,11);
>> nc=components(n,[57,97],sq);
>> imshow(nc)
>> nc2=components(n,[57,97],sq2);
>> figure,imshow(nc2)
```

(a) 3x3 square SE (b) 11x11 square SE

FIGURE 10.27 *Connected components.*

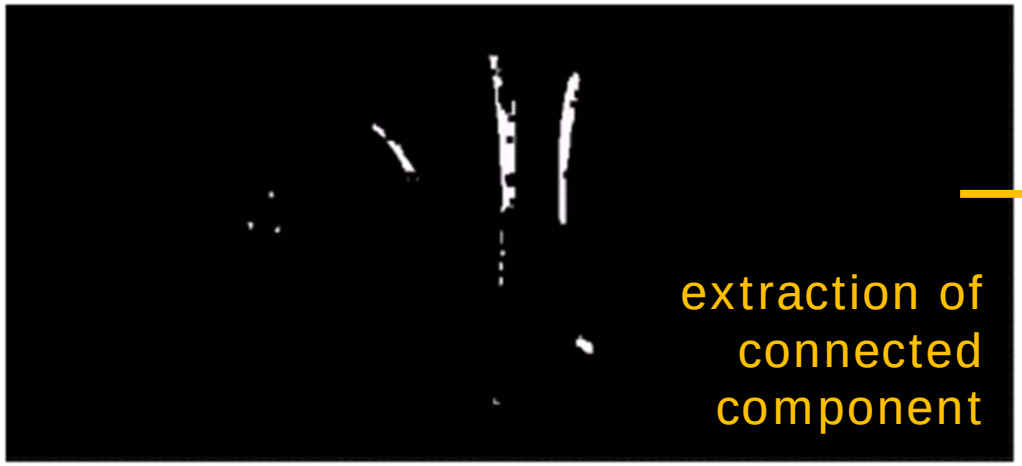
Connected Components Example

original image



thresholding

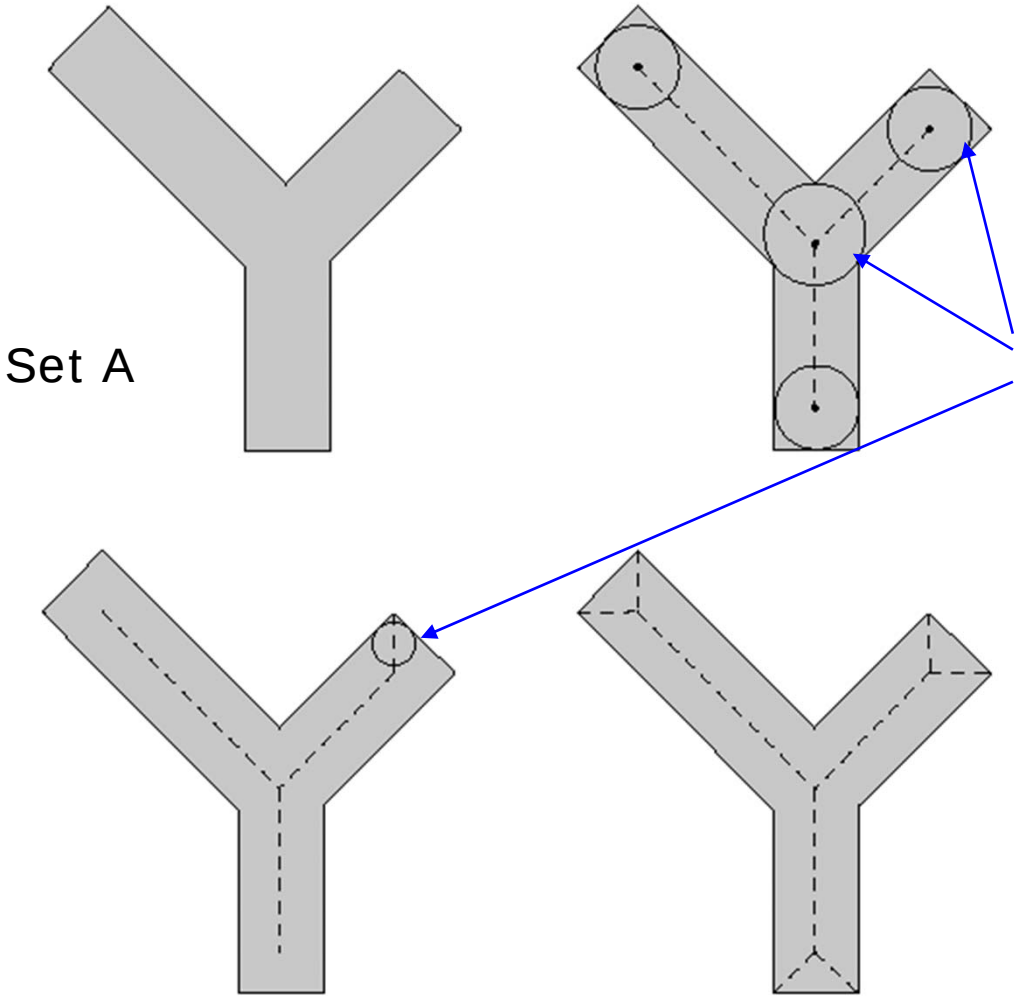
erosion with a 5x5 SE



extraction of connected component

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

Skeletons



How to define a Skeletons?

Maximum disk

1. The largest disk centered at a pixel

2. Touch the boundary of A at two or more places

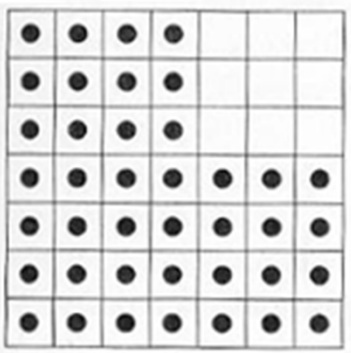
Skeleton

1. iterative erosions followed by an opening
2. eroded image - opened image

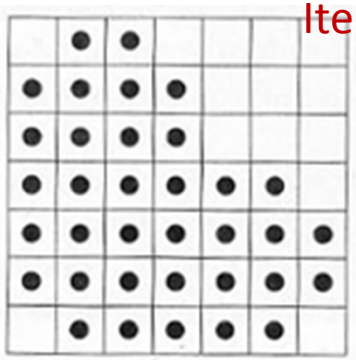
Erosions	Openings	Set differences
A	$A \circ B$	$A - (A \circ B)$
$A \ominus B$	$(A \ominus B) \circ B$	$(A \ominus B) - ((A \ominus B) \circ B)$
$A \ominus 2B$	$(A \ominus 2B) \circ B$	$(A \ominus 2B) - ((A \ominus 2B) \circ B)$
$A \ominus 3B$	$(A \ominus 3B) \circ B$	$(A \ominus 3B) - ((A \ominus 3B) \circ B)$
$\vdots$	$\vdots$	$\vdots$
$A \ominus kB$	$(A \ominus kB) \circ B$	$(A \ominus kB) - ((A \ominus kB) \circ B)$

귀퉁이 값만 남음

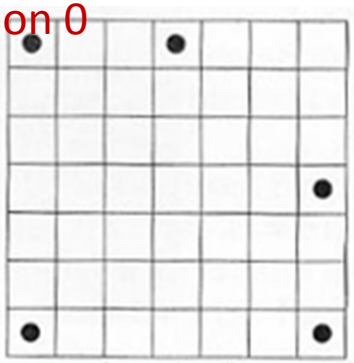
Iteration 0



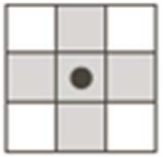
A



$A \circ B$



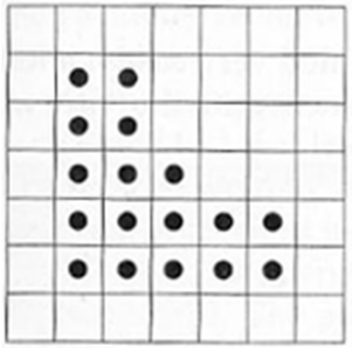
$A-(A \circ B)$



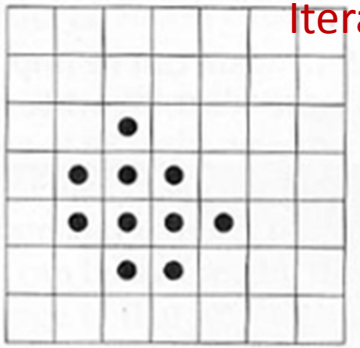
B

$$S_k(A) = (A \Theta kB) - (A \Theta kB) \circ B$$

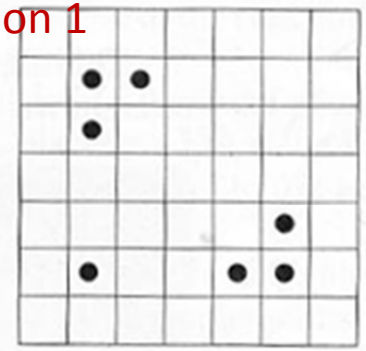
Iteration 1



$A \ominus B$



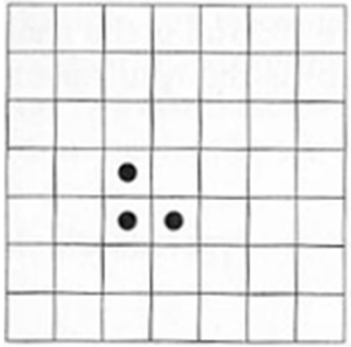
$(A \ominus B) \circ B$



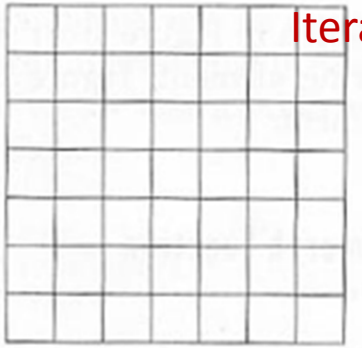
$(A \ominus B) - ((A \ominus B) \circ B)$

$$S(A) = \bigcup_{k=0}^{K} S_k(A)$$

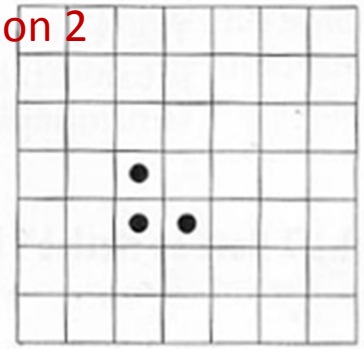
Iteration 2



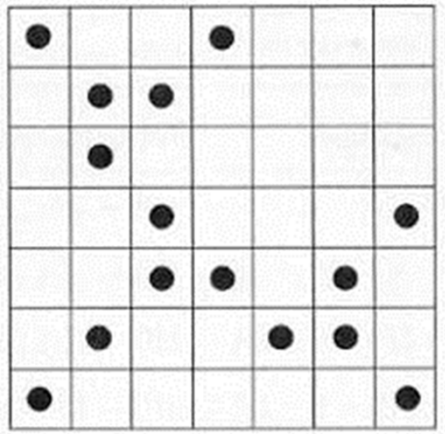
$A \ominus 2B$



$(A \ominus 2B) \circ B$



$(A \ominus 2B) - ((A \ominus 2B) \circ B)$



Final skeleton,
S(A)

Skeleton in Matlab

```
function skel = imskel(image,str)
% IMSKEL(IMAGE,STR) - Calculates the skeleton of binary image IMAGE using
% structuring element STR.  This function uses Lantejoul's algorithm.
%
skel=zeros(size(image));
e=image;
while (any(e(:))),
    o=imopen(e,str);
    skel=skel | (e&~o);
    e=imerode(e,str);
end
```

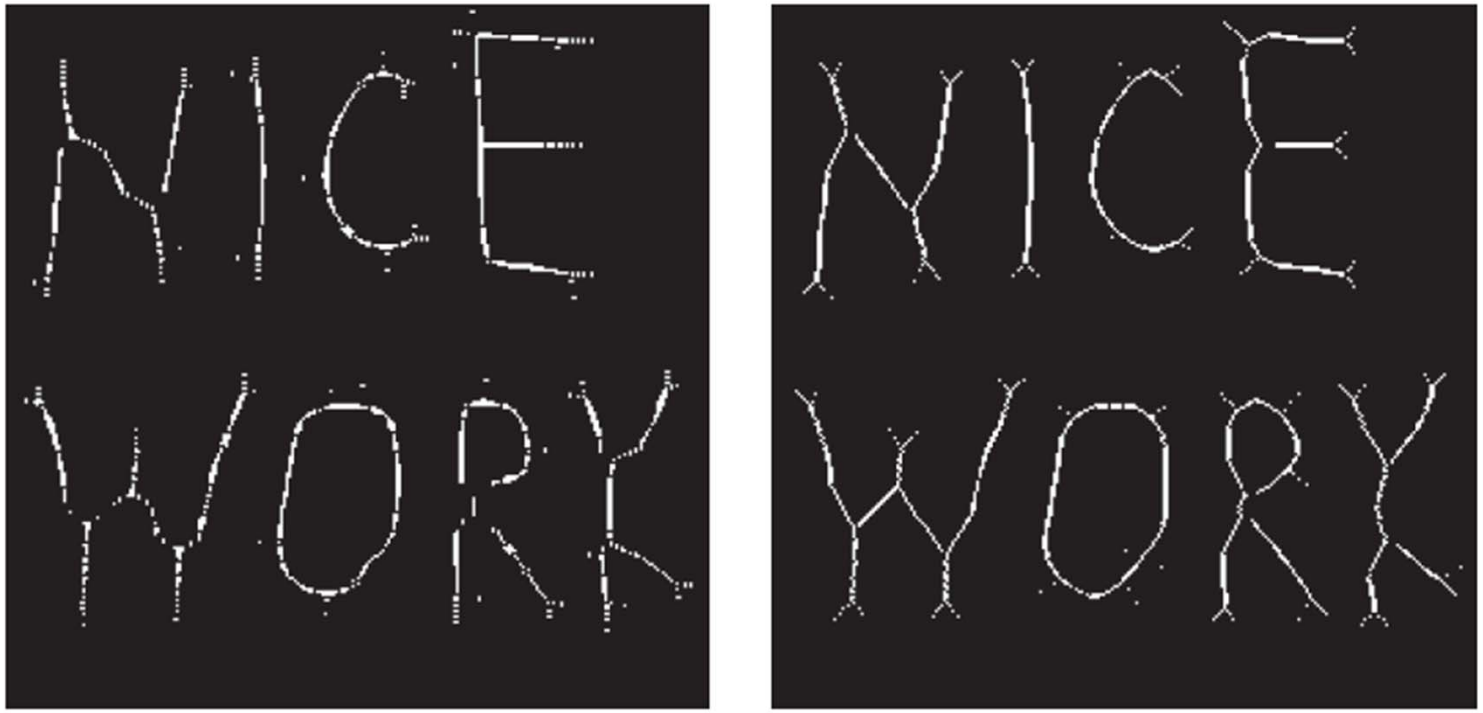
FIGURE 10.30 *A simple program for computing skeletons.*

```
>> nk=imskel(n,sq);
>> imshow(nk)
>> nk2=imskel(n,cr);
>> figure,imshow(nk2)
```

FIGURE 10.31

Square-structuring element

Cross-structuring element



(a) (b)

FIGURE 10.31 *Skeletonization of a binary image.*

A Note on MATLAB's bwmorph Function

- Based on **lookup tables** (ch11)
  - ✓ Consider the 3×3 neighborhood of a pixel.
  - ✓ Since each pixel in the neighborhood can have only two values, there are $2^9 = 512$ different possible neighborhoods.
  - ✓ Define a morphological operation to be a function that maps these neighborhoods to the values 0 and 1.
  - ✓ Each possible neighborhood state can be associated with a numeric value from 0 (all pixels have value 0) to 511 (all pixels have value 1).
  - ✓ The lookup table is then a binary vector of length 512. Its kth element is the value of the function for state k.

- Many other operations can be defined by this method (see the help file for `bwmorph`)

많이 쓰이지 않음

Grayscale Morphology

- **Another way for Erosion**
  - ✓ 1. Find the 3×3 neighborhood N_p of p
  - ✓ 2. Compute the matrix $\{N_p - B\}$
  - ✓ 3. Find the minimum value in the matrix $\{N_p - B\}$

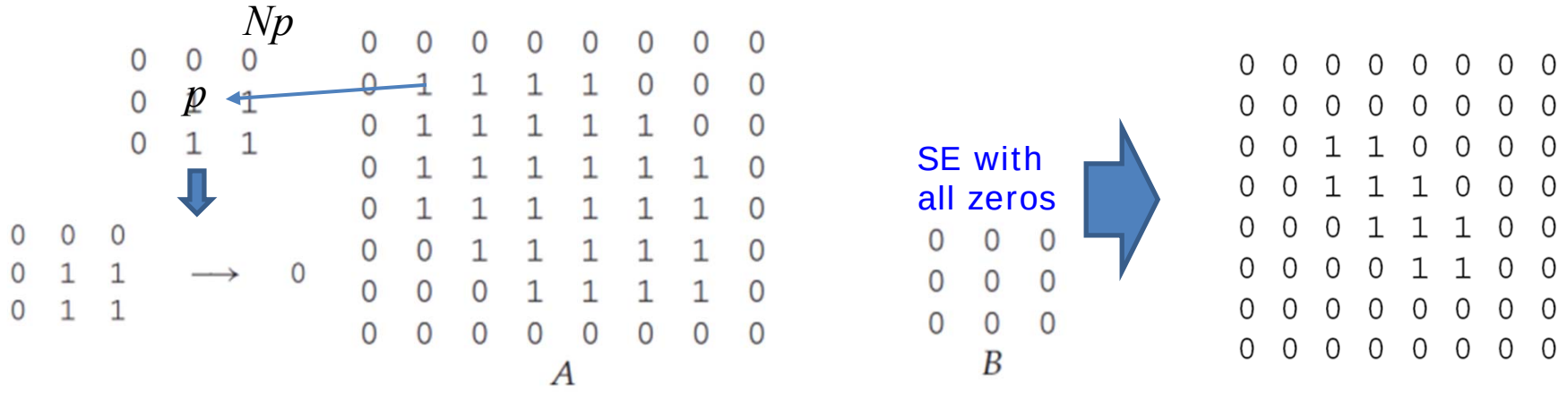


FIGURE 10.32 *An example for erosion.*

Grayscale Morphology

- **Another way for dilation**
  - ✓ 1. Find the 3×3 neighborhood N_p of p
  - ✓ 2. Compute the matrix $N_p + B$
  - ✓ 3. Find the maximum of that result

- **Summary**

$$(A \ominus B)(x,y) = \min\{A(x+s, y+t) - B(s,t), (s,t) \in D_B\},$$
$$(A \oplus B)(x,y) = \max\{A(x+s, y+t) + B(s,t), (s,t) \in D_B\}.$$

Grayscale Morphology Example

x \ y	1	2	3	4	5
1	10	20	20	20	30
2	20	30	30	40	50
3	20	30	30	50	60
4	20	40	50	50	60
5	30	50	60	60	70

A

s \ t	−1	0	1
−1	1	2	3
0	4	5	6
1	7	8	9

B

FIGURE 10.33 *An example for grayscale erosion and dilation.*

$$A \ominus B = \begin{matrix} 5 & 6 & 14 & 15 & 16 \\ 8 & 9 & 17 & 18 & 19 \\ 12 & 13 & 25 & 26 & 39 \\ 15 & 16 & 28 & 29 & 46 \\ 18 & 19 & 39 & 48 & 49 \end{matrix}$$

$$A \oplus B = \begin{matrix} 39 & 39 & 49 & 59 & 58 \\ 39 & 39 & 59 & 69 & 68 \\ 49 & 59 & 59 & 69 & 68 \\ 59 & 69 & 69 & 79 & 78 \\ 56 & 66 & 66 & 76 & 75 \end{matrix}$$

Grayscale Morphology in Matlab

- The `arbitrary` parameter of `strel` allows us to create a structuring element containing any values we like.
- `ones(3,3)` provides the neighborhood.
- Matrix as third parameter provides the values of 3x3 SE.

```
>> str=strel('arbitrary',ones(3,3),[1 2 3;4 5 6;7 8 9])

str =

Nonflat STREL object containing 9 neighbors.

Neighborhood:
     1     1     1
     1     1     1
     1     1     1


Height:
     1     2     3
     4     5     6
     7     8     9
```

```
>> A=[10 20 20 20 30;20 30 30 40 50;20 30 30 50 60;20 40 50 50 60;30 50 60 60 70];
>> imerode(A,str)

ans =

     5     6    14    15    16
     8     9    17    18    19
    12    13    25    26    39
    15    16    28    29    46
    18    19    39    48    49
```

Grayscale Morphology in Matlab

```
>> str2=strel('arbitrary',ones(3,3),[9 8 7;6 5 4;3 2 1])
str2 =
Nonflat STREL object containing 9 neighbors.

Neighborhood:
     1     1     1
     1     1     1
     1     1     1

Height:
     9     8     7
     6     5     4
     3     2     1

>> imdilate(A,str2)

ans =

    39    39    49    59    58
    39    39    59    69    68
    49    59    59    69    68
    59    69    69    79    78
    56    66    66    76    75
```

Grayscale Morphology in Matlab: Erosion and Dilation

```
>> c=imread('caribou.tif');
>> str=strel('square',5)
```

```
>> cd=imdilate(c,str);
>> ce=imerode(c,str);
>> imshow(cd),figure,imshow(ce)
```

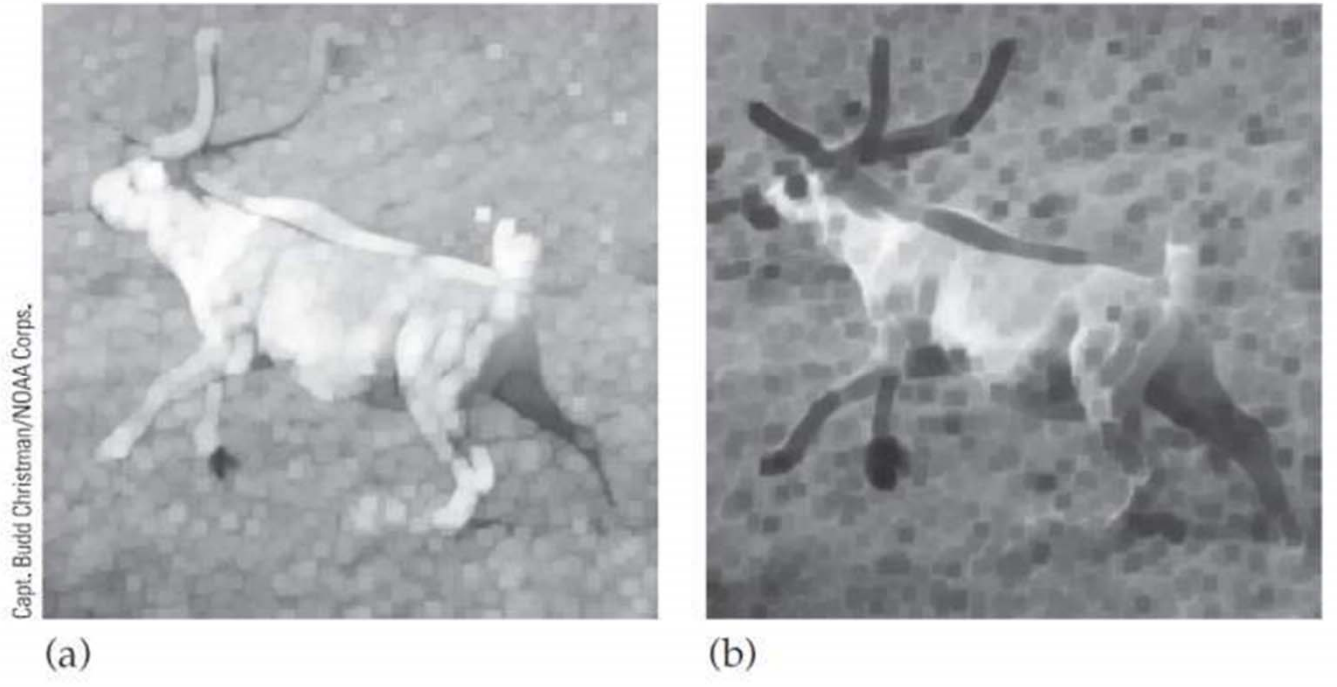


FIGURE 10.34 *Morphology. (a) Dilation. (b) Erosion.*

Grayscale Morphology: Opening and Closing

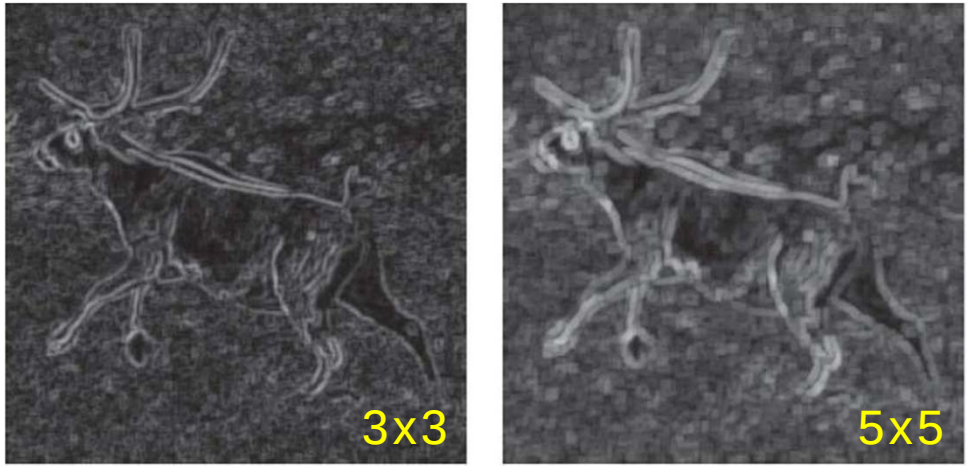
```
>> co=imopen(c,str);
>> cc=imclose(c,str);
>> imshow(co),figure,imshow(cc)
```

FIGURE 10.35 *Grayscale opening and closing. (a) Opening. (b) Closing.*

Applications of Grayscale Morphology: Edge Detection

- morphological gradient in grayscale $(A \oplus B) - (A \ominus B)$
- 3x3 and 5x5 square SE

```
>> str1=strel('square',3);
>> str2=strel('square',5);
>> ce1=imerode(c,str1);
>> ce2=imerode(c,str2);
>> cd1=imdilate(c,str1);
>> cd2=imdilate(c,str2);
>> cg1=imsubtract(cd1,ce1);
>> cg2=imsubtract(cd2,ce2);
>> imshow(cg1),figure,imshow(cg2)
```



Applications of Grayscale Morphology: Noise Removal

- opening followed by a closing in grayscale
- In general, it does not performed well on Gaussian noise

가우시안 노이즈에는 효과적이지 않다

```
>> cn=imnoise(c,'salt & pepper');
>> cf=imclose(imopen(cn,str),str);
>> imshow(cn),figure,imshow(cf)
```

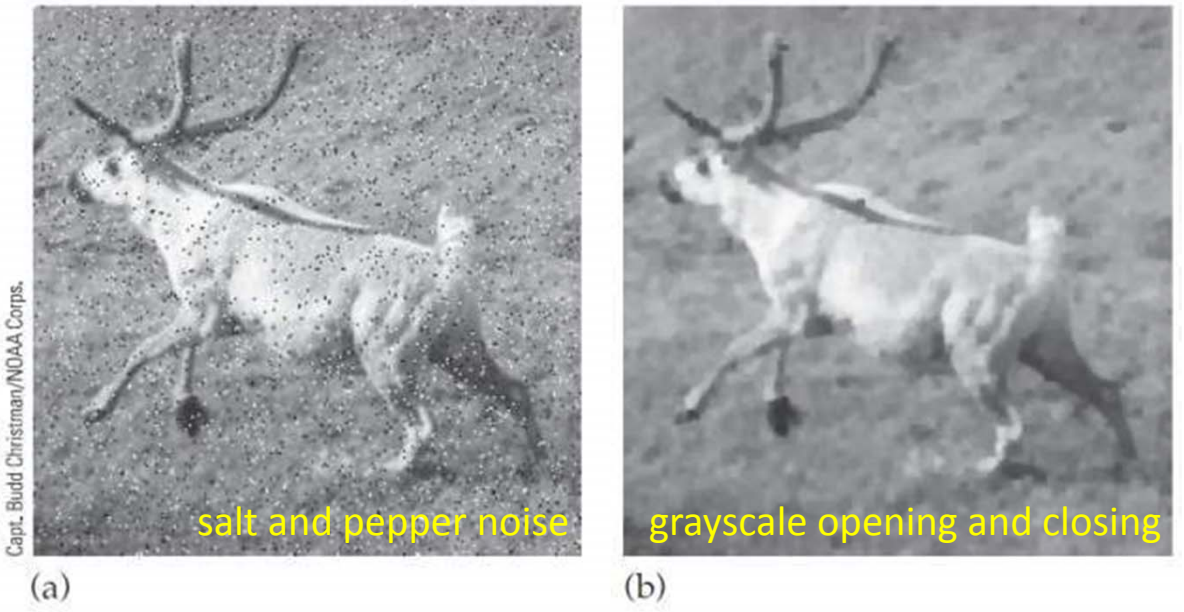


FIGURE 10.37 *Use of morphological filtering to remove noise.*

Summary

- **Chap 9. Image Segmentation**
  - Single & double thresholding
  - How to determine the threshold value
  - Adaptive thresholding
  - Edge detection: 1st and 2nd derivatives
  - Canny edge detector
  - Hough Transform
- **Chap 10. Mathematical Morphology**
  - Erosion & dilation -> boundary detection
  - Opening & closing -> noise removal
  - Hit-or-miss transform -> shape detection
  - Binary applications: region filling, connected components, skeletons
  - Grayscale morphology -> edge detection, noise removal