Morphological Processing and Feature Extraction

Dr. Tran Anh Tuan,

Faculty of Mathematics and Computer Science,

University of Science, HCMC

Contents

Preview

- **"Morphology"** - a branch of biology that deals with the shape, form and structure of animals and plants.

- **Mathematical morphology** is a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries, skeletons, and convex hull

- **Morphological Techniques for pre- or post processing** - morphological filtering, thinning and pruning.

- **Morphology** is a cornerstone of the mathematical set of tools underlying the development of techniques that extract "meaning" from an image.

- Mathematical framework used for:
  - Pre-processing - noise filtering, shape simplification,
  - Enhancing object structure - skeletonization, convex hull
  - Segmentation - watershed
  - Quantitative description - area, perimeter, ...

Preliminaries

Basic concepts from Set Theory

- The language of mathematical morphology is Set Theory
- In binary images, the set elements are members of the **2-D integer space Z^2** Where each element (x,y) is a coordinate of black (or white) pixel in the image.

$$A = \{(x,y) \mid I_A(x,y) = 1\}$$

The vertical bar means "such that"

- "Union" of two sets A and B
  - The set of elements belonging to A,B, or both

$$A \cup B$$

- "Intersection" of two sets A and B
  - The set of elements belonging to both A and B

$$A \cap B$$

- w "is an element of" set A $w \in A$

Basic concepts from Set Theory (Cont.)

- "Complement"
  - The set of elements that are not in *A*

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

- "Difference" of two sets *A* and *B*
  - The set of elements belonging to *A*, but not to *B*

$$A - B = \{w \mid w \in A, w \notin B\}$$

Comma means "and"

- "Subset"
  - *A* is a subset of *B* if every element of *A* is also in *B*

$$A \subseteq B.$$

- The empty set { }

$$\varnothing$$

Basic operations of Set Theory- I

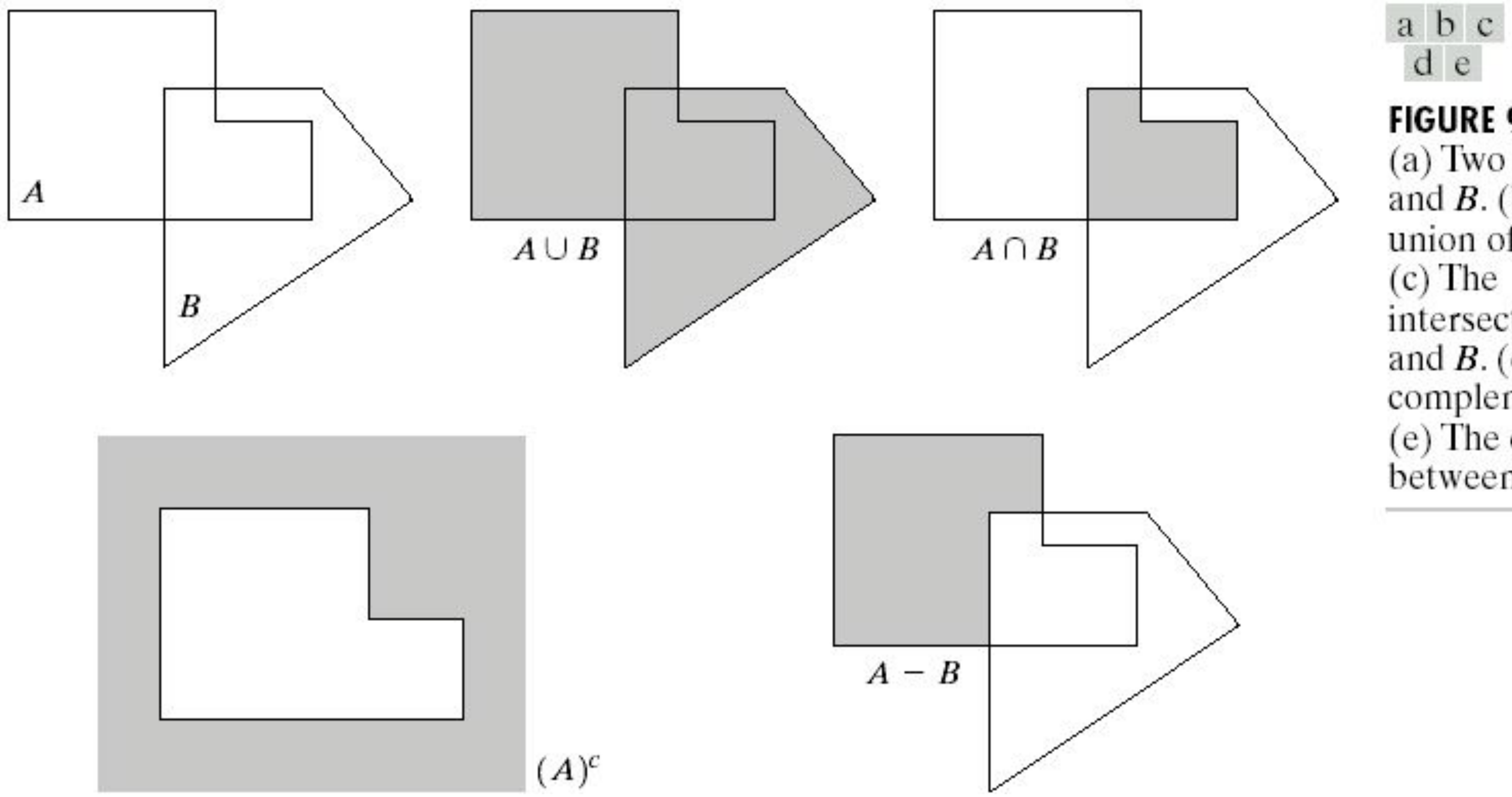


FIGURE 9.1
(a) Two sets A and B. (b) The union of A and B. (c) The intersection of A and B. (d) The complement of A. (e) The difference between A and B.

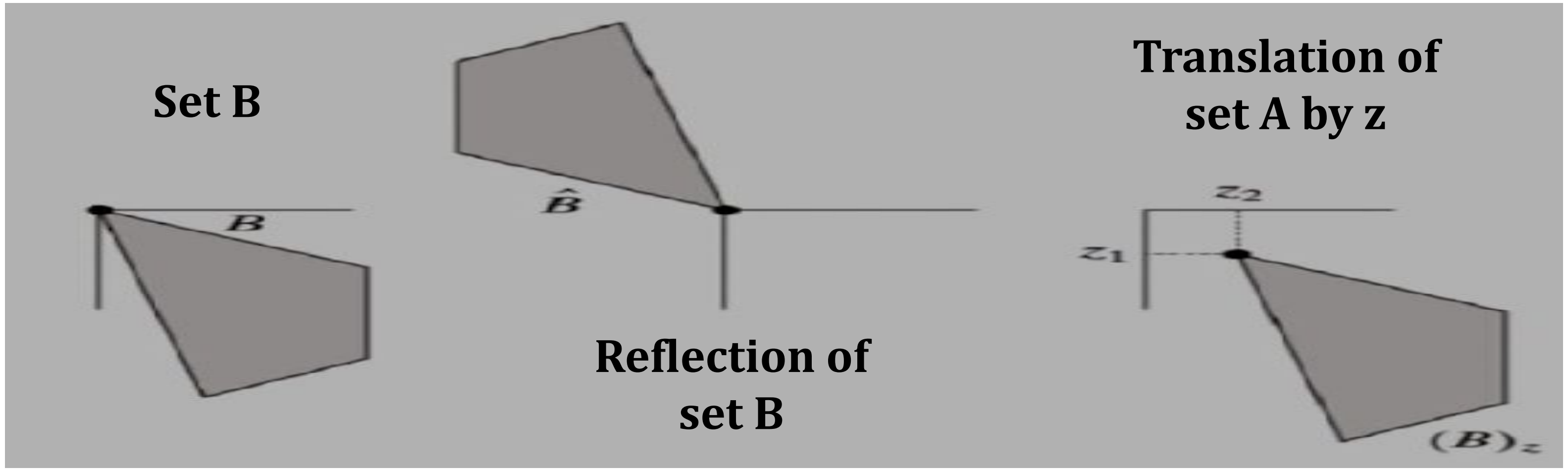
Basic operations of Set Theory- II

- The reflection of a set B

$$\hat{B} = \{ w \mid w = -b \text{ for } b \in B \}$$

The translation of set A by point z= (Z_1, Z_2)

$$(A)_z = \{ c \mid c = a + z \text{ for } a \in A \}$$



Logical operators and sets

TABLE 9.1
The three basic logical operations.

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

Set Operation	MATLAB Expression for Binary Images	Name
$A \cap B$	A & B	AND
$A \cup B$	A | B	OR
A^c	~A	NOT
$A - B$	A & ~B	DIFFERENCE

Using logical expressions in MATLAB to perform set operations on binary images.

Example - I

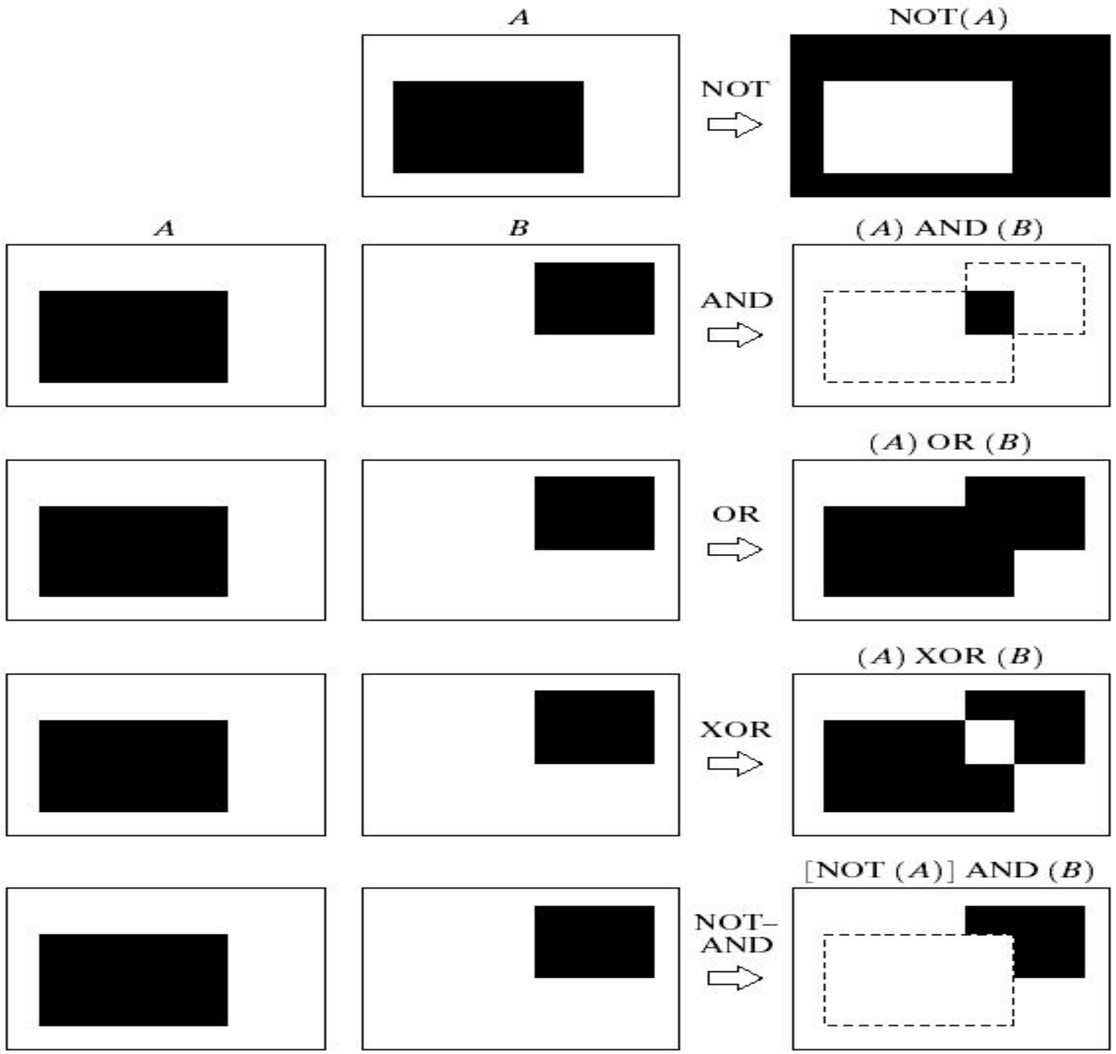
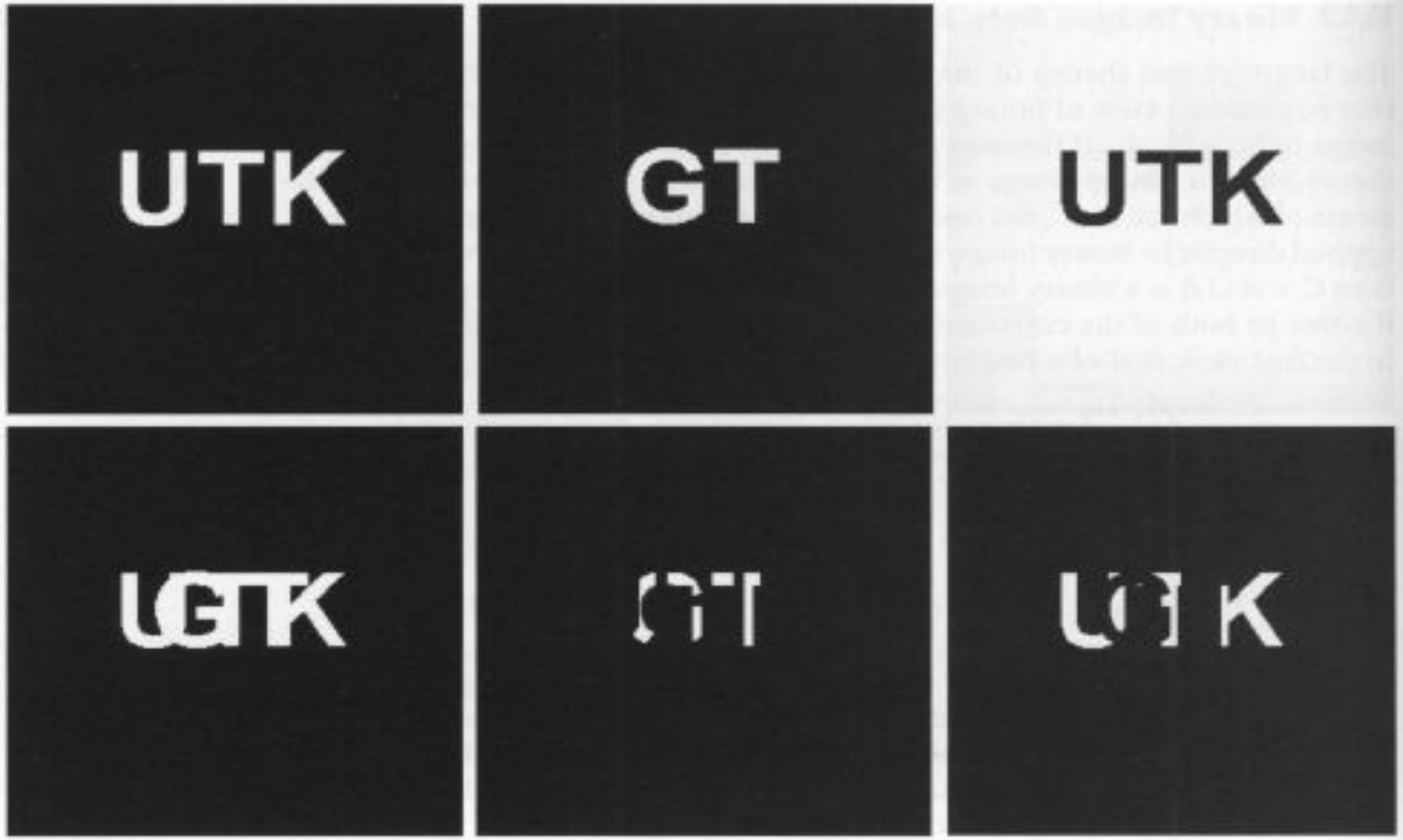


FIGURE 9.3 Some logic operations between binary images. Black represents binary 1s and white binary 0s in this example.

Example -II



a b c
d e f

FIGURE 10.3 (a) Binary image A. (b) Binary image B. (c) Complement ~A. (d) Union A | ~B. (e) Intersection A & B. (f) Set difference A & ~B.

Structure Elements (SE)

- Small sets or sub-images used to probe an image under study for properties of interest.

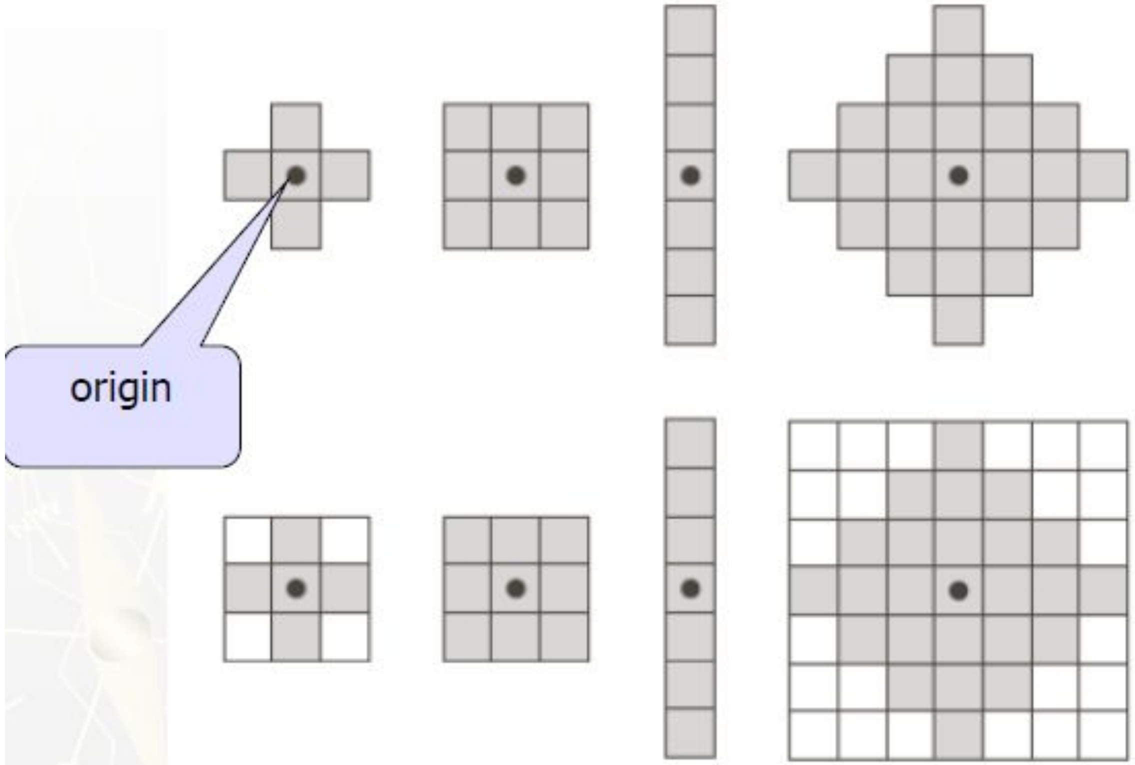


FIGURE 9.2 First row: Examples of structuring elements. Second row: Structuring elements converted to rectangular arrays. The dots denote the centers of the SEs.

Dilation and Erosion

Dilation

- Dilation is an operation that “grows” and “thickens” objects in an image.
- The specific manner and extent of this thickening is controlled by a shape referred to as a structuring element.
- The dilation of A by B, is defined as the set operation

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

where, $\varnothing$ is the empty set and B is the structuring element

In words, the dilation of A by B is the set consisting of <u>all the structuring element origin locations where the reflected and translated B overlaps at least one element of A</u>

Working of Dilation

The output image is <u>1 at each location of the origin of the structuring element</u> such that structuring element <u>overlaps</u> <u>at least one</u> 1- valued pixel in the input image.

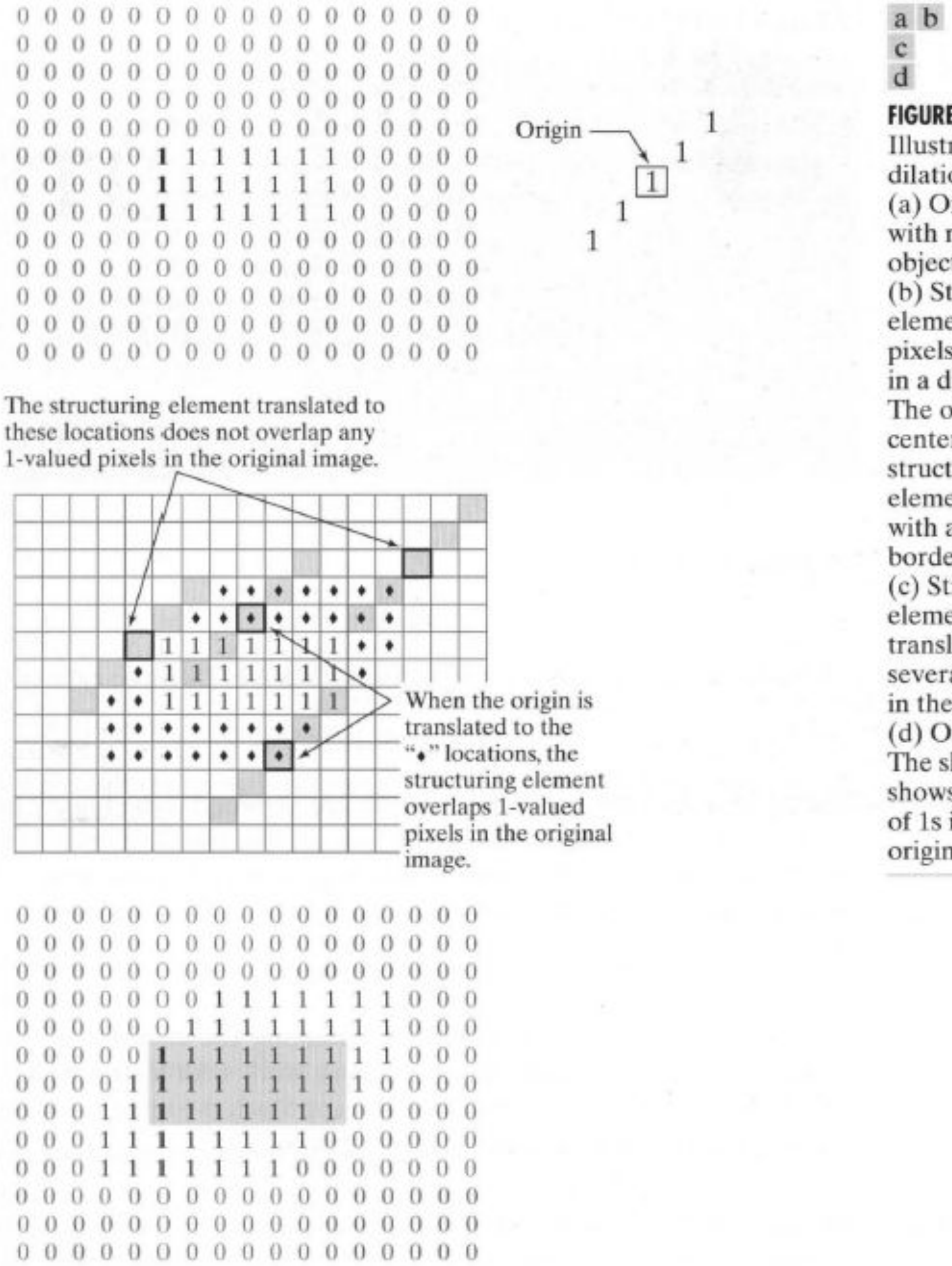
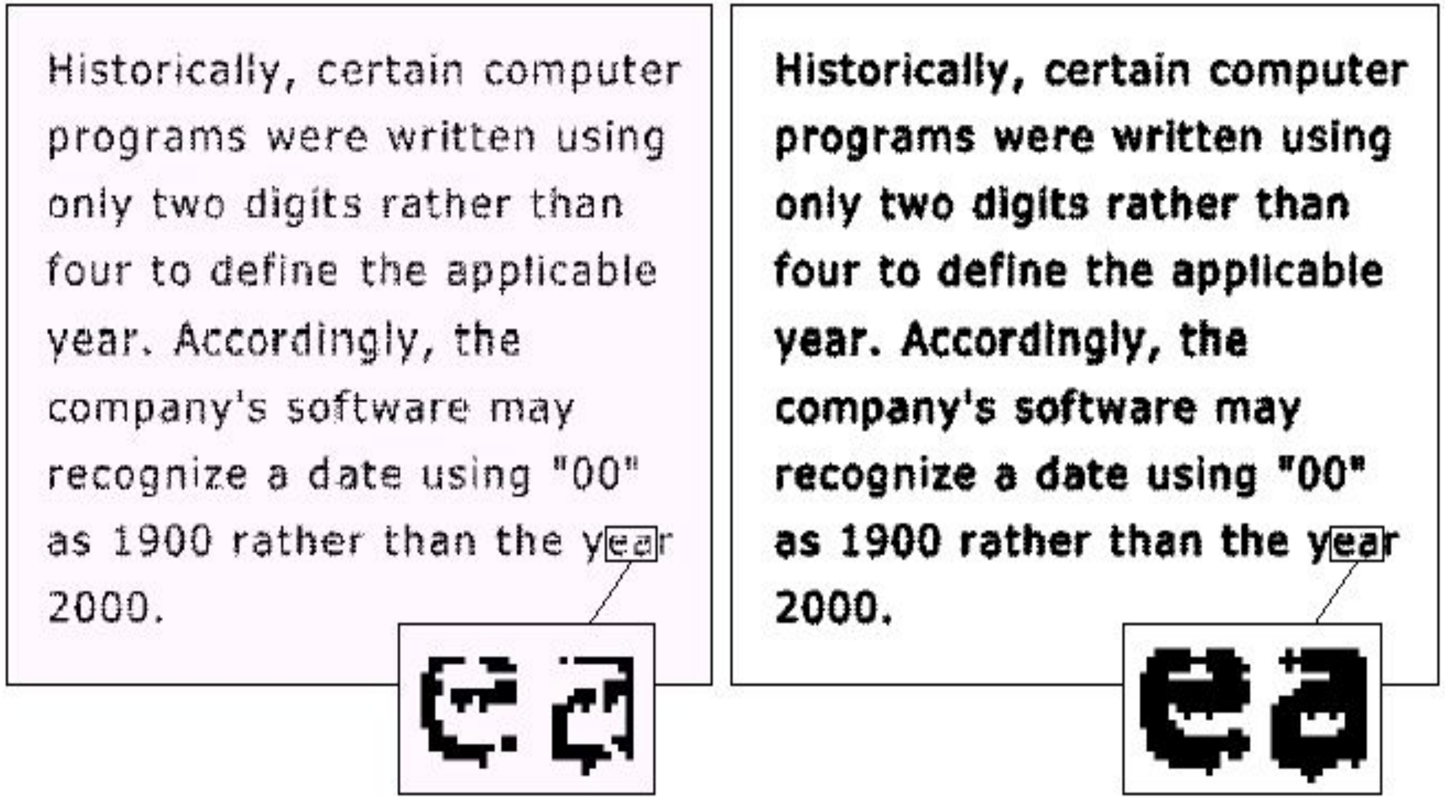


FIGURE 10.4 Illustration of dilation. (a) Original image with rectangular object. (b) Structuring element with five pixels arranged in a diagonal line. The origin, or center, of the structuring element is shown with a dark border. (c) Structuring element translated to several locations in the image. (d) Output image. The shaded region shows the location of 1s in the original image.

Example



0	1	0
1	1	1
0	1	0

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.

Erosion

- Erosion "shrinks" or "thins" objects in a binary image.
- As a dilation , the manner and extent of shrinking is controlled by a structuring element.

- The erosion of A by B , is defined as

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

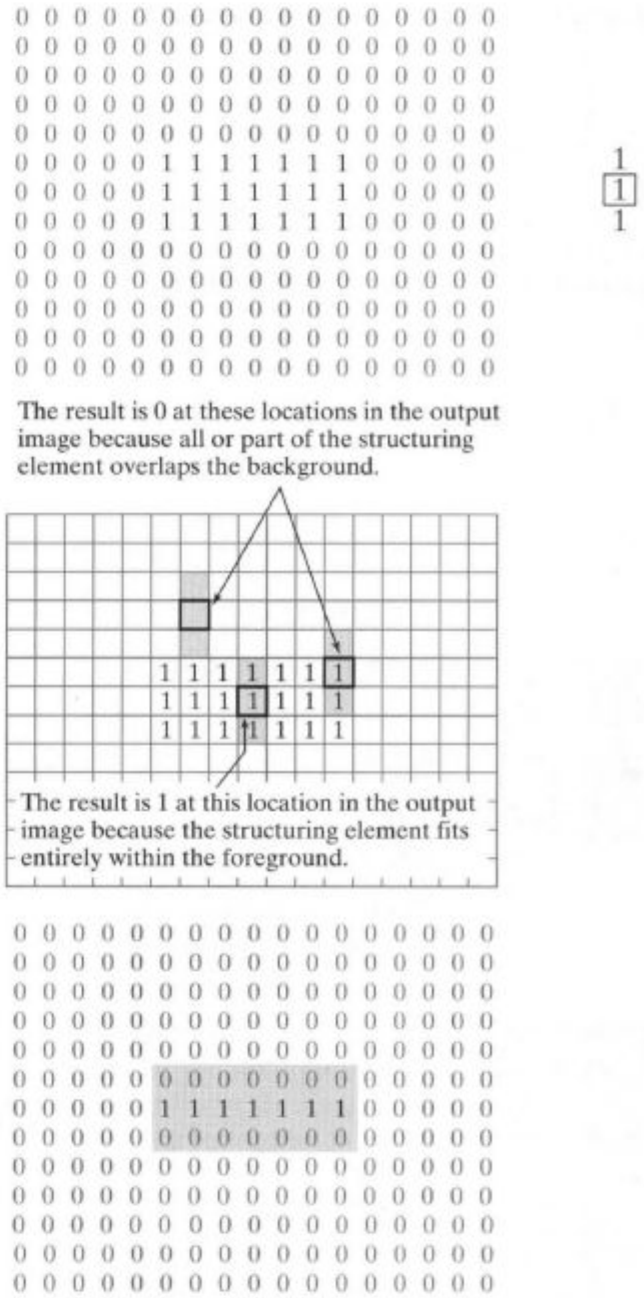
where, the notation $(B)_z \subseteq A$ means that $(B)_z$ is subset of A.

In words, erosion of A by B is the set of all points z such that B, translated by z , is contained in A.

Working of Erosion

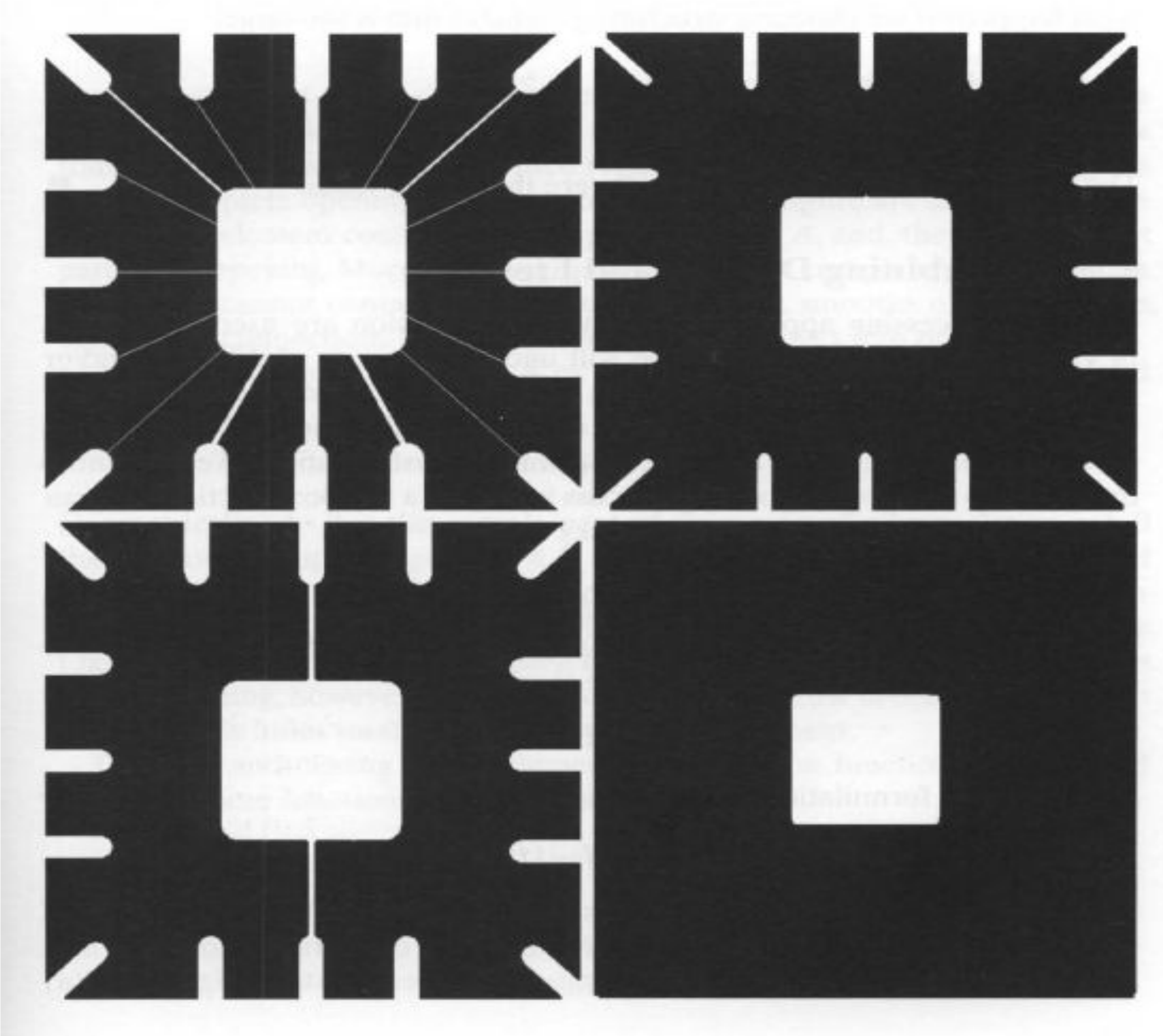
a b
c
d

FIGURE 10.7 Illustration of erosion. (a) Original image with rectangular object. (b) Structuring element with three pixels arranged in a vertical line. The origin of the structuring element is shown with a dark border. (c) Structuring element translated to several locations in the image. (d) Output image. The shaded region shows the location of 1s in the original image.



The output image has a value of 1 at each location of the origin of the structuring element such that the element overlap only 1 (not 0)-valued pixels of the input image.

Example



a b
c d

FIGURE 10.8
An illustration of erosion.
(a) Original image of size 486 × 486 pixels.
(b) Erosion with a disk of radius 10.
(c) Erosion with a disk of radius 5.
(d) Erosion with a disk of radius 20.

Combining Erosion and Dilation

- Dilation and erosion are duals of each other with respect to set complementation and reflection.

$$(A \ominus B)^c = A^c \oplus \hat{B}.$$

- Wanted
  remove structures / fill holes
  without affecting remaining parts

- Solution
  combine erosion and dilation
  (using same SE)

Example of combining Erosion and Dilation

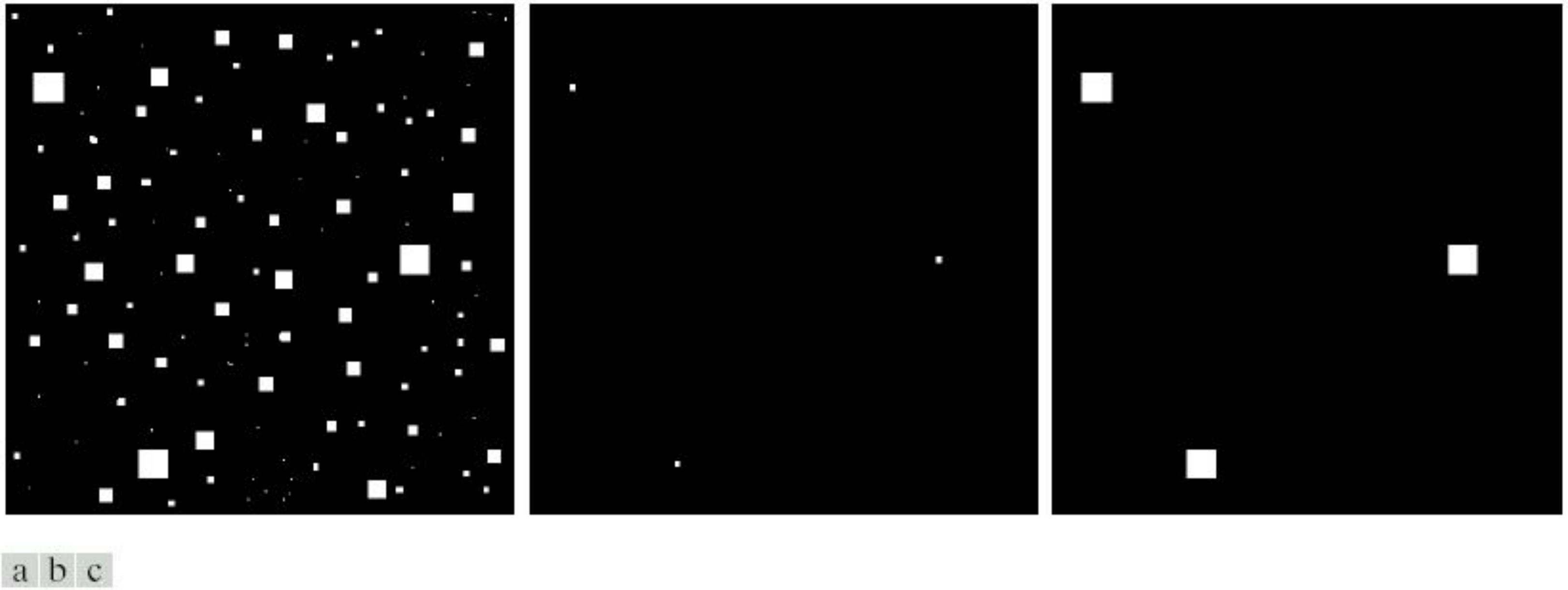


FIGURE 9.7 (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

Structuring element B=13x13 pixels

Opening and Closing

Opening and Closing

The *morphological opening* of A by B, denoted $A \circ B$, is defined as the erosion of A by B, followed by a dilation of the result by B:

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

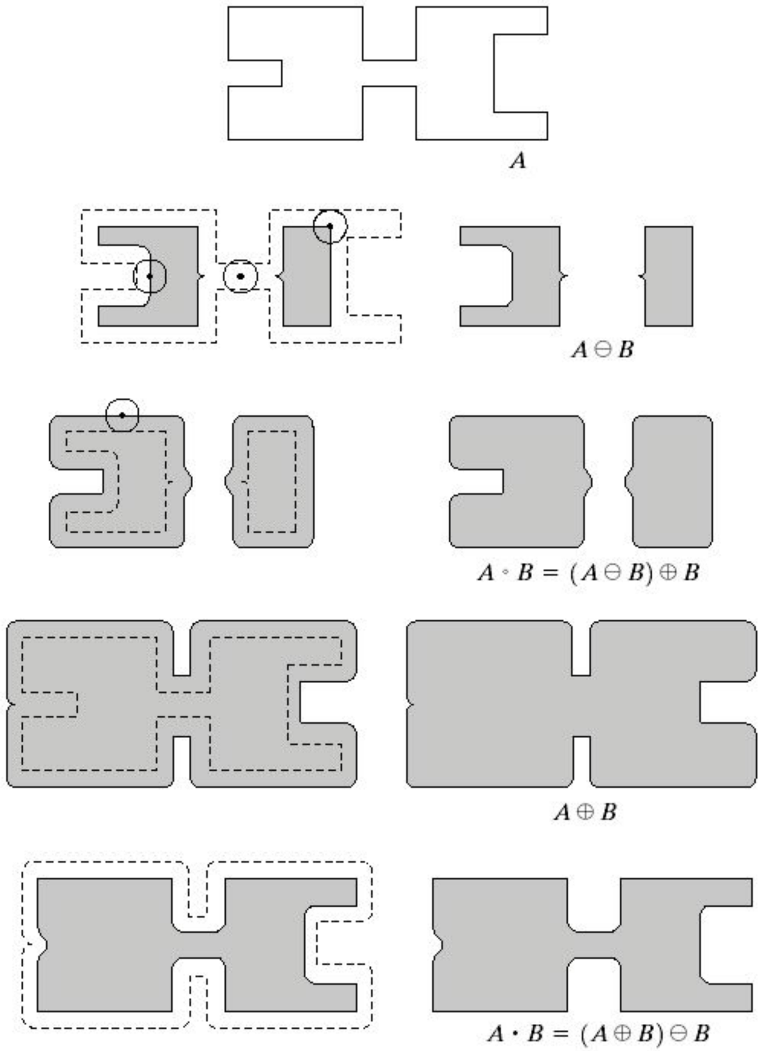
- Opening smoothes the contour of an object, breaks thin connections and removes thin protrusions.

The *morphological closing* of A by B, denoted $A \bullet B$, is a dilation followed by an erosion:

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

- Closing generally joins narrow breaks, fills long thin gulfs, and fills holes smaller than the structuring element.

Example



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.

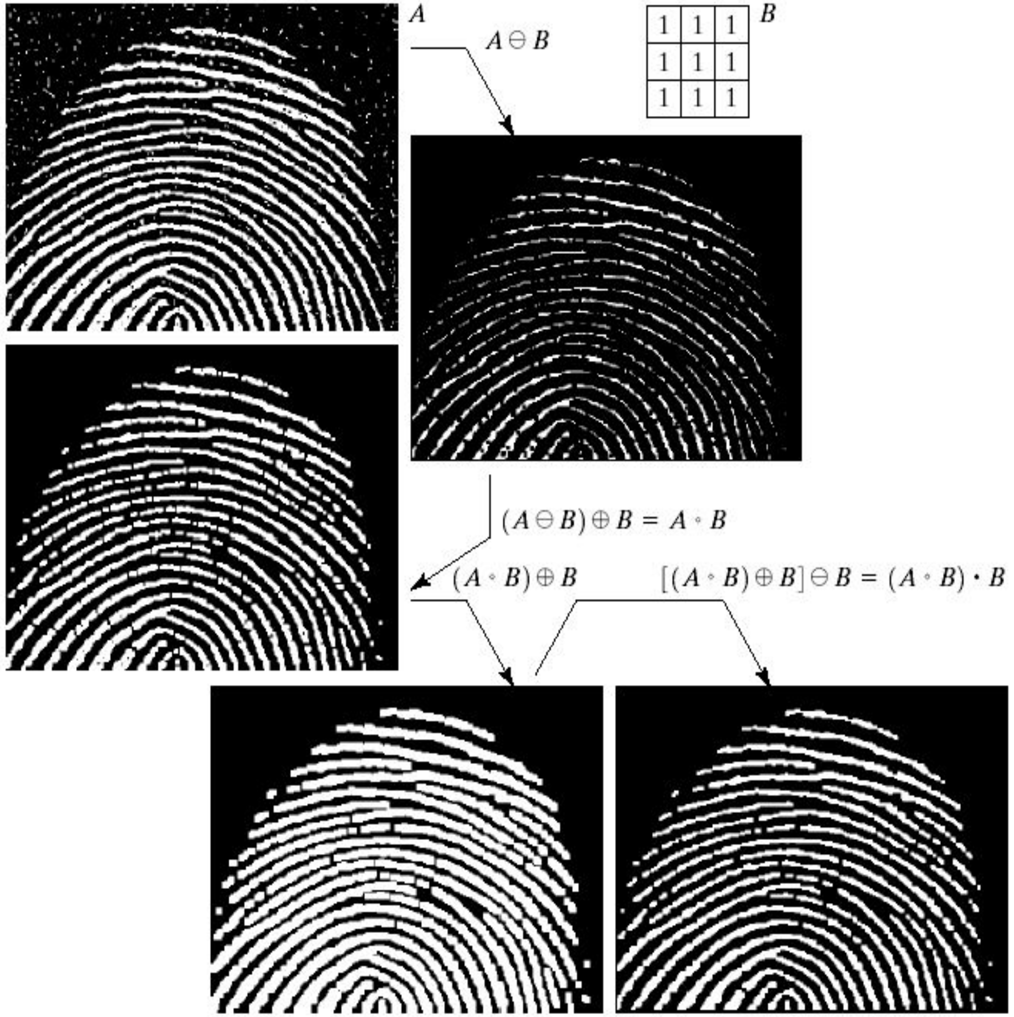
Opening and closing are duals of each other with respect to set complementation and reflection.

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

Morphological Filter

- Morphological operations can be used to construct filters which are similar in concept to spatial filters.

- A Morphological filter consisting of opening followed by closing can be used to eliminate the noise and its effect on the print.



a b
d c
e f

FIGURE 9.11
(a) Noisy image.
(c) Eroded image.
(d) Opening of A.
(d) Dilation of the opening.
(e) Closing of the opening. (Original image for this example courtesy of the National Institute of Standards and Technology.)

The Hit-or-Miss Transformation

The Hit-or-Miss Transformation

- A basic morphological tool for **shape detection** or **template matching**
- Let the origin of each shape be located at its center of gravity.
- If we want to find the location of a shape , say – **X** , at (larger) image, say – **A** :
  - Let X be enclosed by a small window, say – **W**.
  - The local background of X with respect to W is defined as the
    set difference (**W - X**).
  - Apply *erosion* operator of A by X, will get us the set of locations of the origin of X, such that X is completely contained in A.
  - It may be also view geometrically as the set of all locations of the origin of X at which X found a match (**hit**) in A.
  - Apply *erosion* operator on the complement of A by the local background set (W – X).
  - Notice, that the set of locations for which X **exactly** fits inside A is the intersection of these two last operators above. This intersection is precisely the location sought.

The Hit-or-Miss Transformation

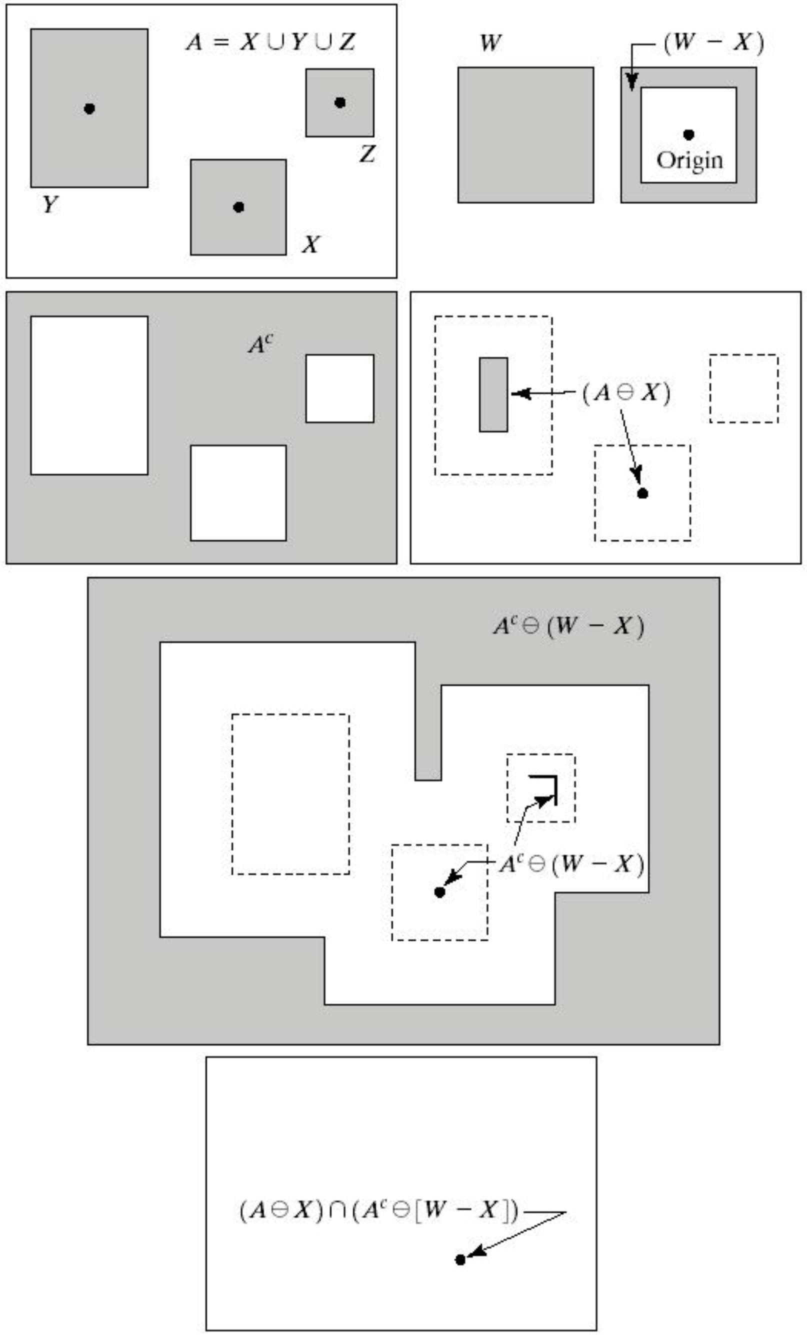


FIGURE 9.12 (a) Set A. (b) A window, W, and the local background of X with respect to W, $(W - X)$. (c) Complement of A. (d) Erosion of A by X. (e) Erosion of A^c by $(W - X)$. (f) Intersection of (d) and (e), showing the location of the origin of X, as desired.

a b
c
d e
f
g

FIGURE 10.12 (a) Original image A. (b) Structuring element B_1. (c) Erosion of A by B_1. (d) Complement of the original image, A^c. (e) Structuring element B_2. (f) Erosion of A^c by B_2. (g) Output image.

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

B_1

$$\begin{array}{ccc} & 1 & \\ 1 & \boxed{1} & 1 \\ & 1 & \end{array}$$

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

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

B_2

$$\begin{array}{ccc} 1 & & 1 \\ & \boxed{} & \\ 1 & & 1 \end{array}$$

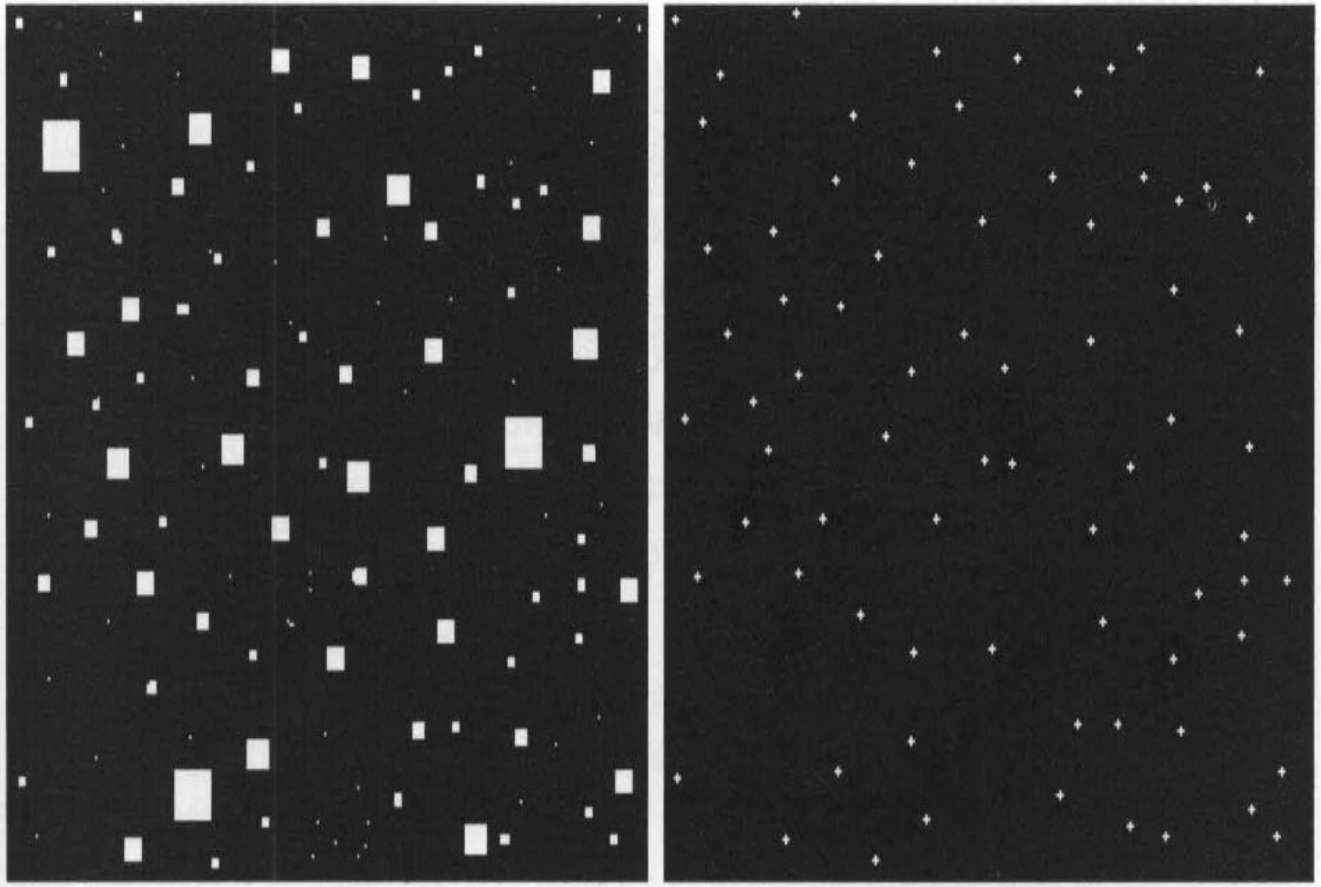
1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	1	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	1
1	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0
1	1	1	1	0	1	0	1	1	1	1	1	0	1	0	1
1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1

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

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

Working of Hit-or-Miss Transformation

The Hit-or-Miss Transformation - Example



a b

FIGURE 10.13 (a) Original image. (b) Result of applying the hit-or-miss transformation (the dots shown were enlarged to facilitate viewing).

Basic Morphological Algorithms

Basic Morphological Algorithms

1. Boundary Extraction
2. Region filling
3. Extraction of connected components
4. Convex hull
5. Thinning
6. Thickening
7. Skeletons
8. Pruning

1. Boundary Extraction

- First, erode A by B, then make set difference between A and the erosion
- The thickness of the contour depends on the size of structuring element – B

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

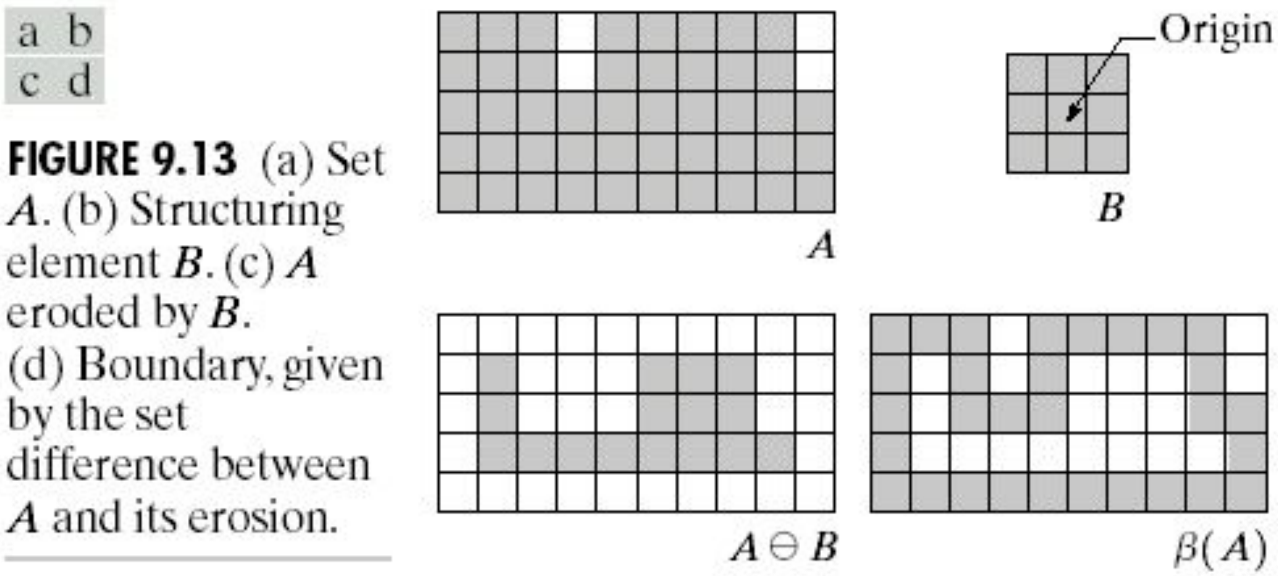
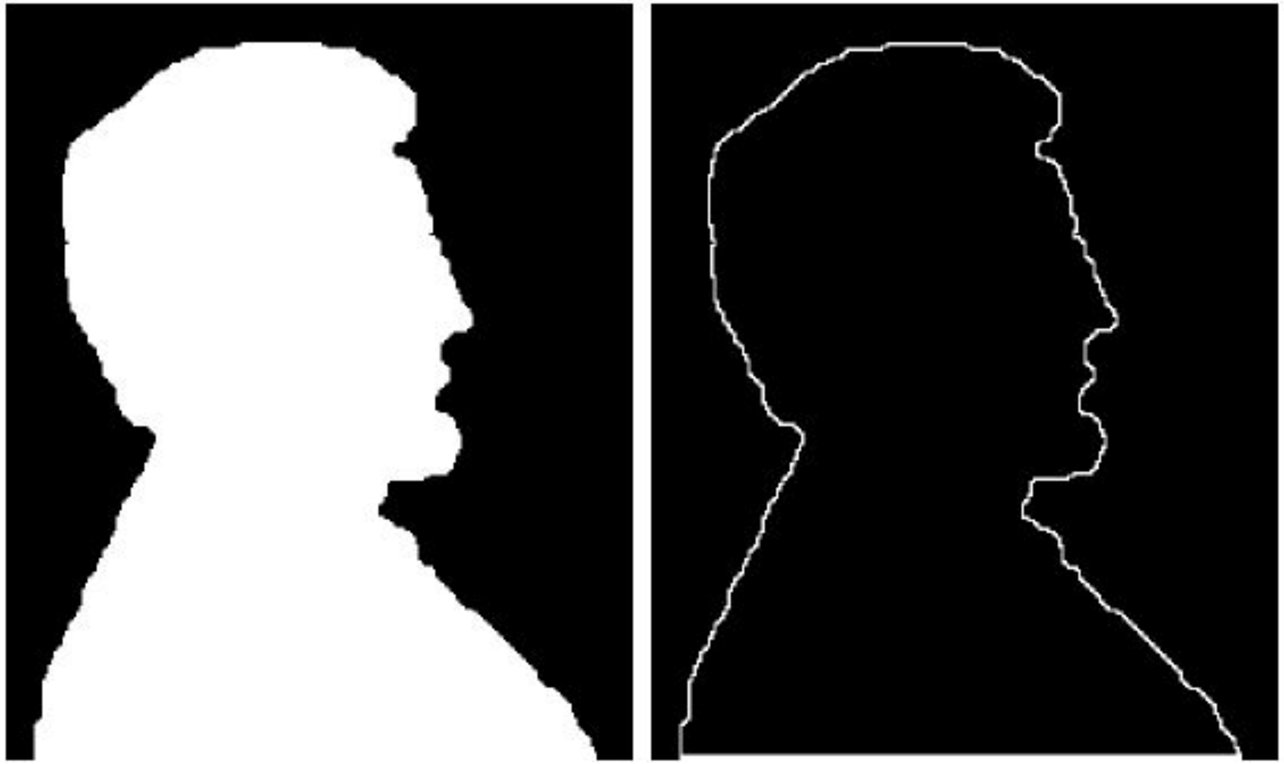


FIGURE 9.13 (a) Set A. (b) Structuring element B. (c) A eroded by B. (d) Boundary, given by the set difference between A and its erosion.

1. Boundary Extraction- Example



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).

2. Region Filling

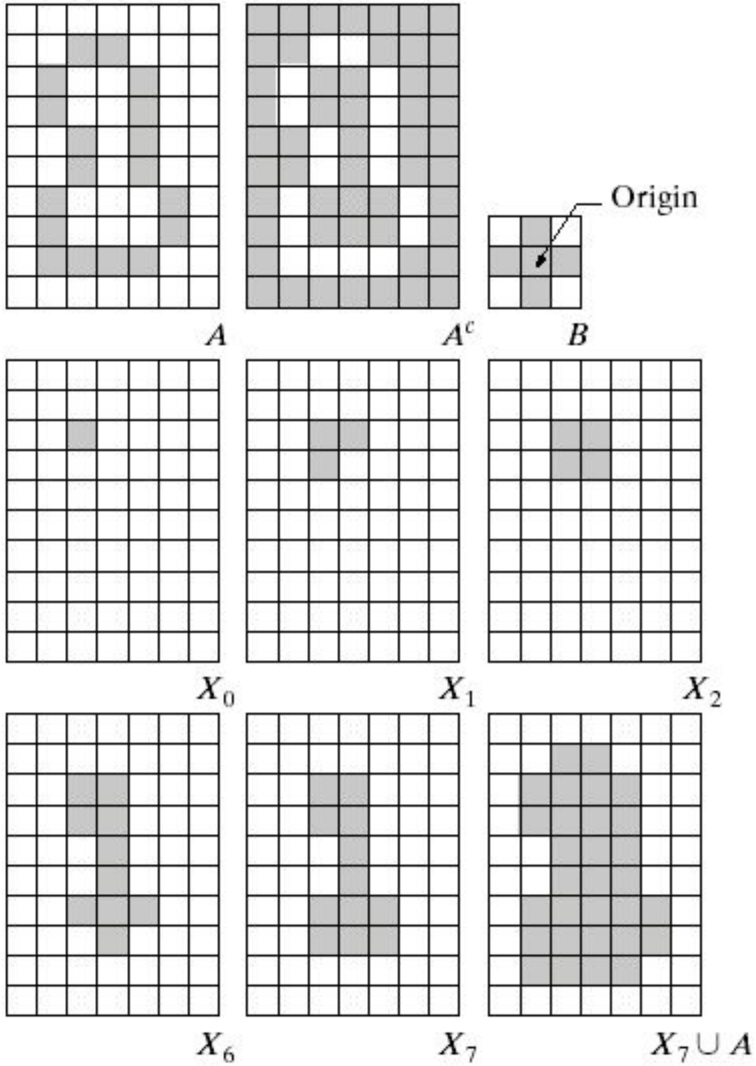
- This algorithm is based on a set of dilations, complementation and intersections
- Lets consider a set A containing a subset whose elements are 8-connected boundary points of a region
- p is the point inside the boundary, with the value of 1

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

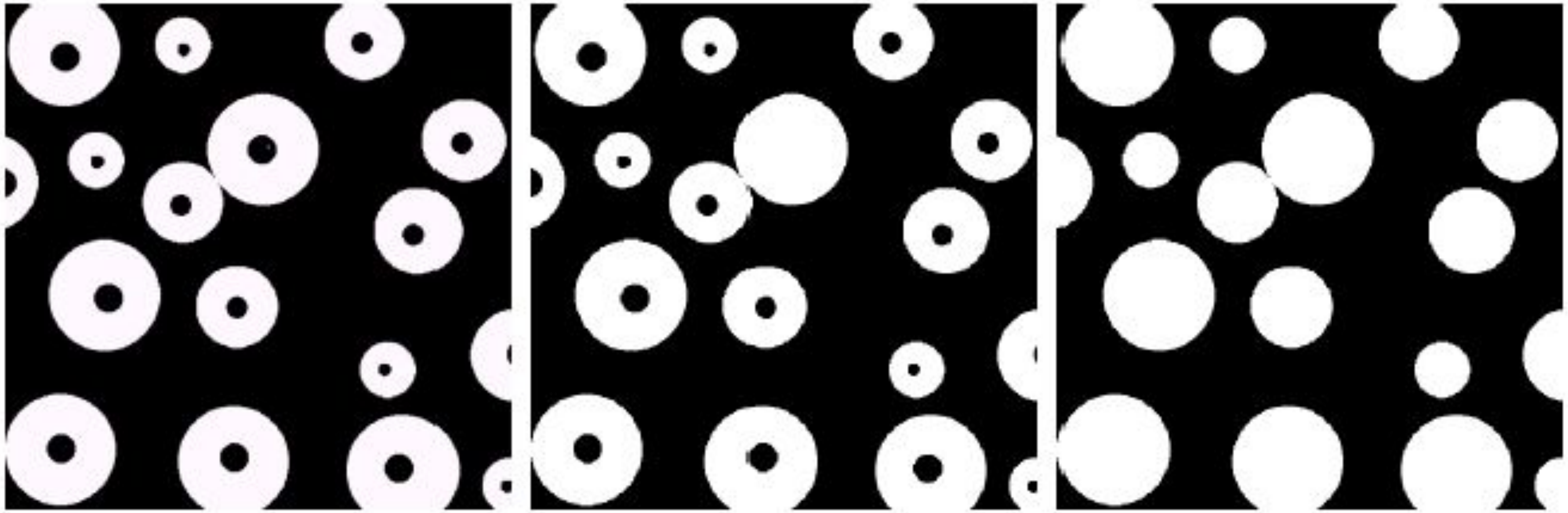
- The process stops when X(k) = X(k-1)
- The result that given by union of X(k) and A , is a set contains the filled set and the boundary

2. Region Filling-Working

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)].



2. Region Filling-Example II



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.

3. Extraction of Connected Components

- This algorithm extracts a component by selecting a point on a binary object A
- Works similar to region filling, but this time we use in the conjunction the object A, instead of it's complement
- Let Y represent a connected component contained in a set A
- Assume a point "p" of Y is known
- Below equation yields all the elements of Y :

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

Where X_0=p and B is suitable structuring element

3. Extraction of Connected Components - Working

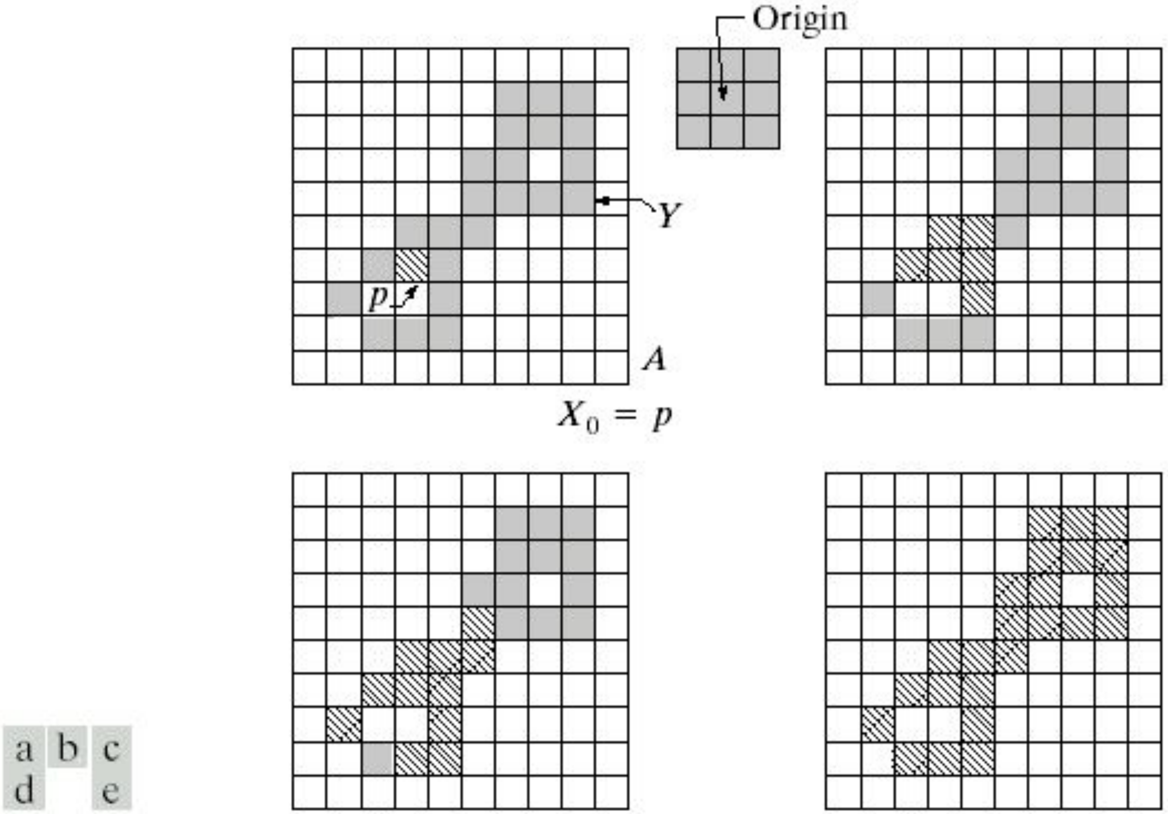


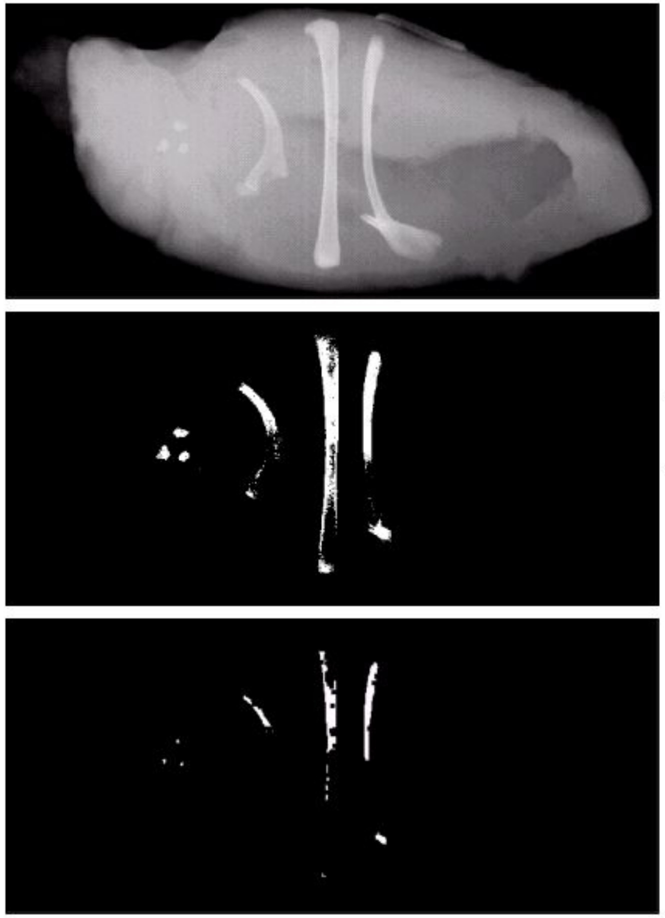
FIGURE 9.17 (a) Set A showing initial point p (all shaded points are valued 1, but are shown different from p to indicate that they have not yet been found by the algorithm). (b) Structuring element. (c) Result of first iterative step. (d) Result of second step. (e) Final result.

3. Extraction of Connected Components- Example

This shows automated inspection of chicken-breast, that contains bone fragment
The original image is threshold

We can get by using this algorithm the number of pixels in each of the connected components

Now we could know if this food contains big enough bones and prevent hazards



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

4. Convex Hull

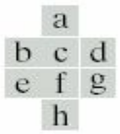
- Set A is said to be convex if a straight line segment joining any two points in A lies entirely within A

- The convex hull H of set S is the smallest convex set containing S

- Useful for object description
- This algorithm iteratively applying the hit-or-miss transform to A with the first of B element, unions it with A, and repeated with second element of B

$$X_k^i = (X_{k-1} \circledast B^i) \cup A \quad i = 1, 2, 3, 4 \quad \text{and} \quad k = 1, 2, 3, \ldots$$

4. Convex Hull - Working



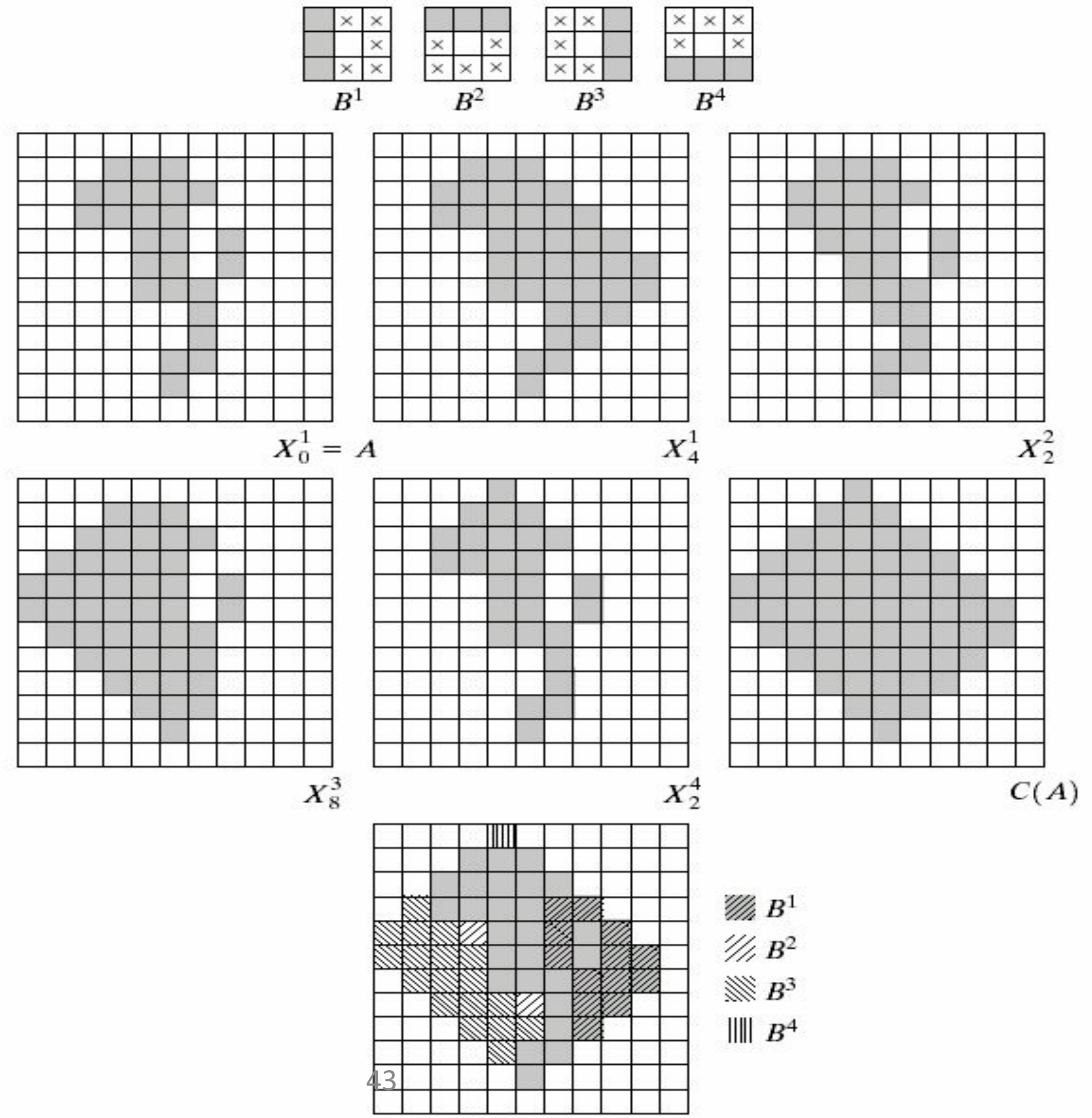


FIGURE 9.19 (a) Structuring elements. (b) Set A. (c)–(f) Results of convergence with the structuring elements shown in (a). (g) Convex hull. (h) Convex hull showing the contribution of each structuring element.

4. Convex Hull – Working (Cont.)

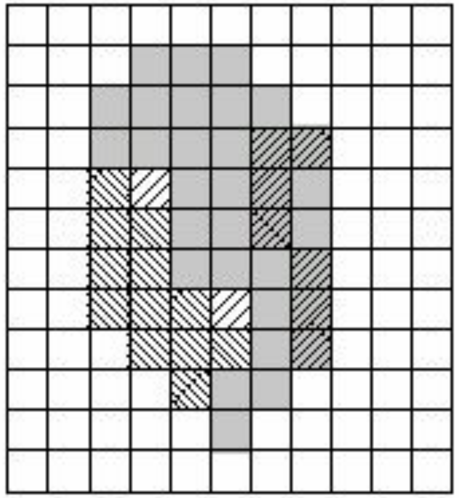
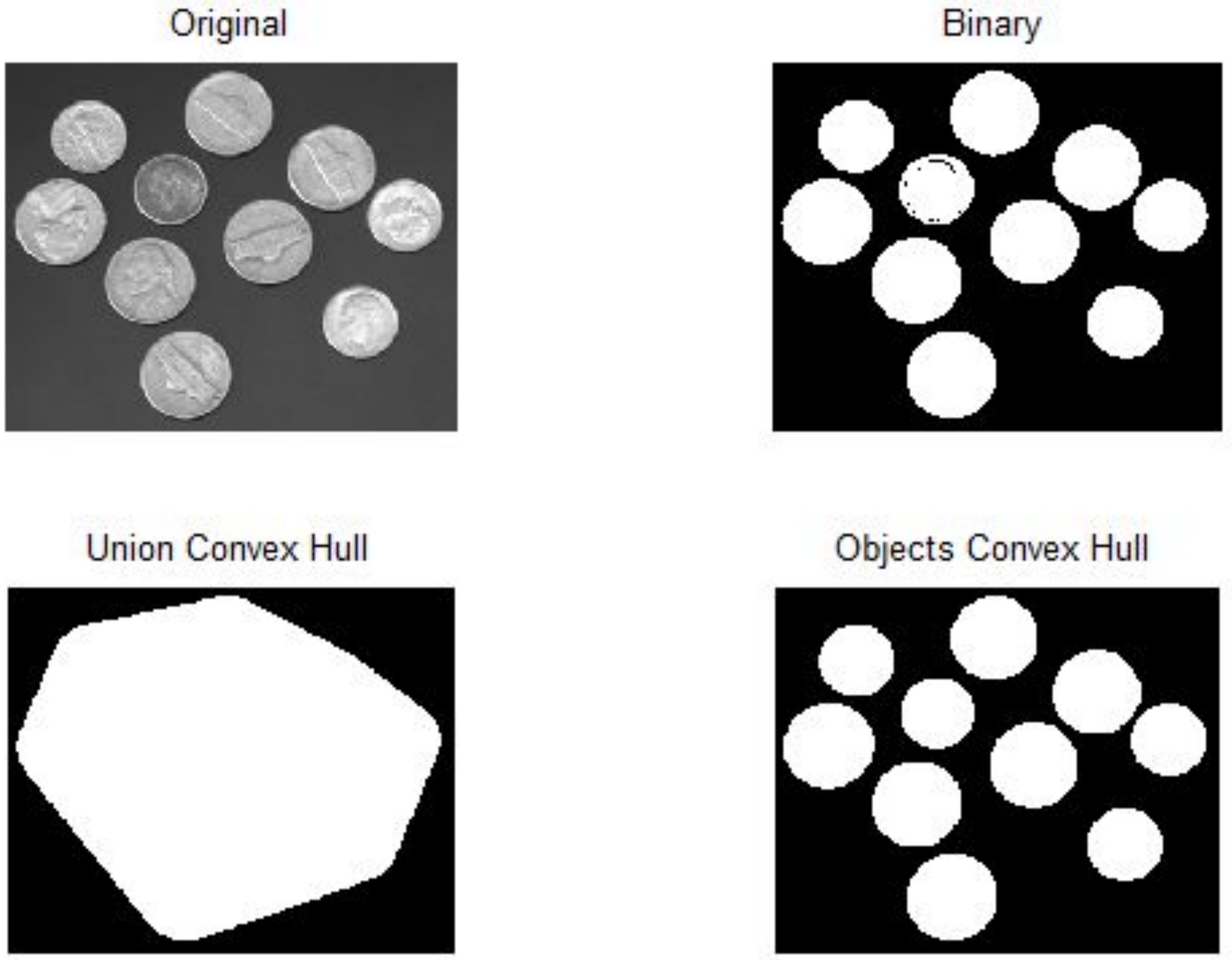


FIGURE 9.20 Result of limiting growth of convex hull algorithm to the maximum dimensions of the original set of points along the vertical and horizontal directions.

4. Convex Hull - Example



5. Thinning

- The thinning of a set A by a structuring element B, can be defined by terms of the hit-and-miss transform:

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

- A more useful expression for thinning A symmetrically is based on a sequence of structuring elements:

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

- Where B^i is a rotated version of B^{i-1}. Using this concept we define thinning by a sequence of structuring elements:

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

- The process is to thin by one pass with B^1, then thin the result with one pass with B^2, and so on until A is thinned with one pass with B^n.
- The entire process is repeated until no further changes occur.
- Each pass is preformed using the equation

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

5. Thinning- Working

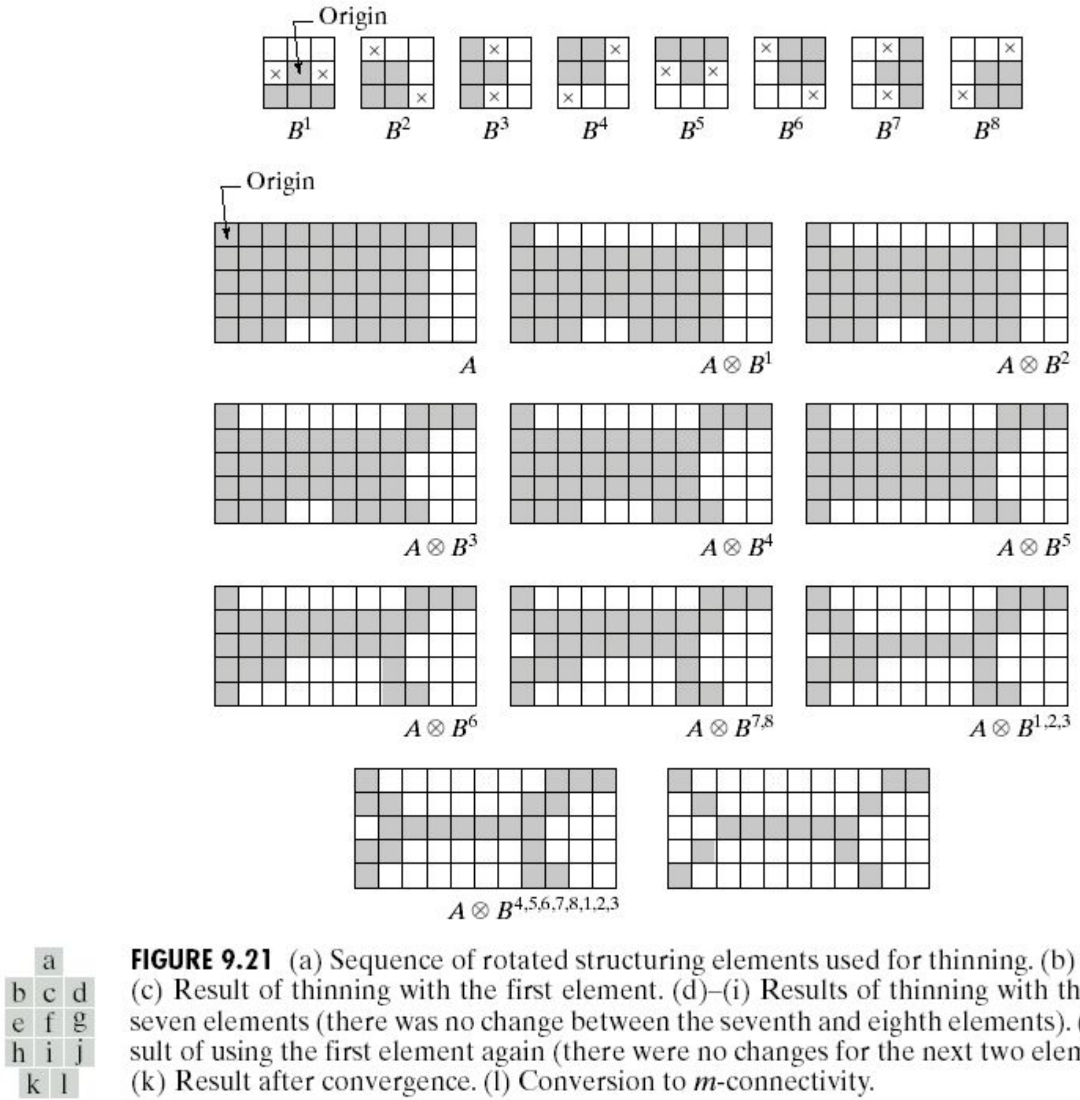
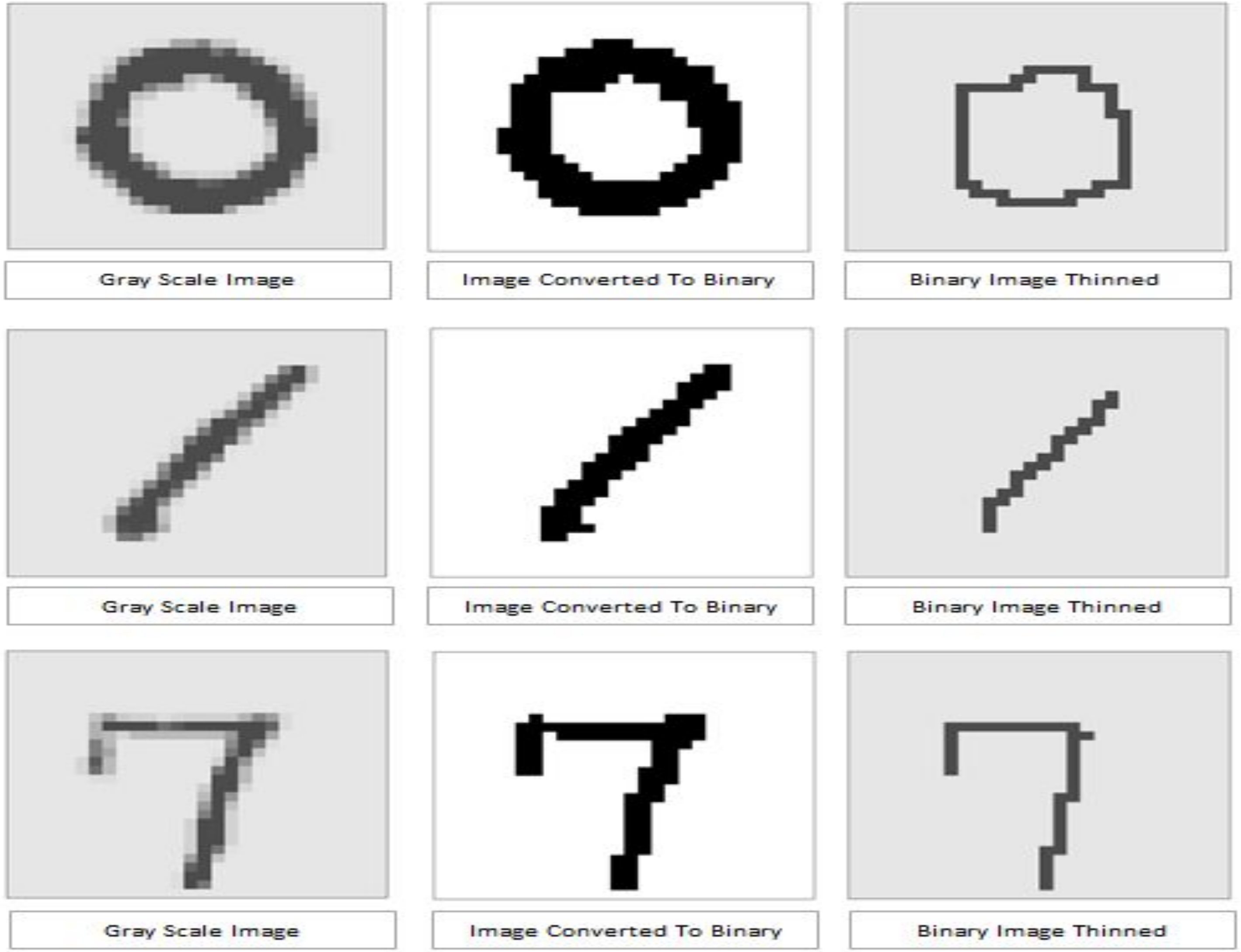


FIGURE 9.21 (a) Sequence of rotated structuring elements used for thinning. (b) Set A. (c) Result of thinning with the first element. (d)–(i) Results of thinning with the next seven elements (there was no change between the seventh and eighth elements). (j) Result of using the first element again (there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

5. Thinning- Example



6. Thickening

- Thickening is a morphological dual of thinning.
- Definition of thickening

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

- As in thinning, thickening can be defined as a sequential operation:

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

- the structuring elements used for thickening have the same form as in thinning, but with all 1's and 0's interchanged.

- A separate algorithm for thickening is often used in practice, Instead the usual procedure is to thin the background of the set in question and then complement the result.

- In other words, to thicken a set A, we form $C=A^c$, thin C and than form C^c.

6. Thickening - Working

- We will notice in the next example 9.22(c) that the thinned background forms a boundary for the thickening process, this feature does not occur in the direct implementation of thickening
- This is one of the reasons for using background thinning to accomplish thickening.

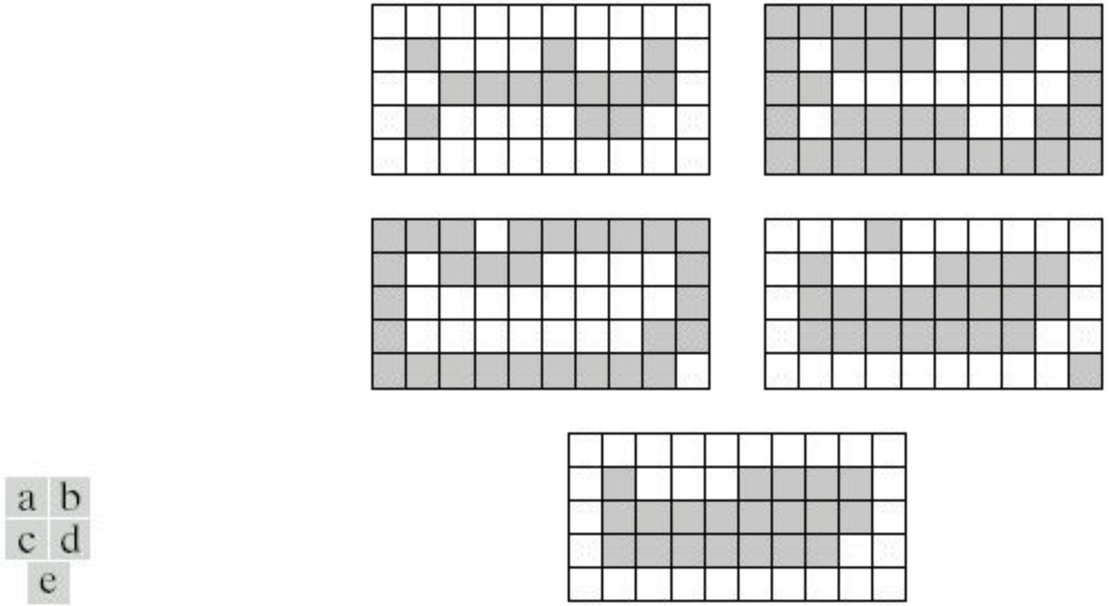
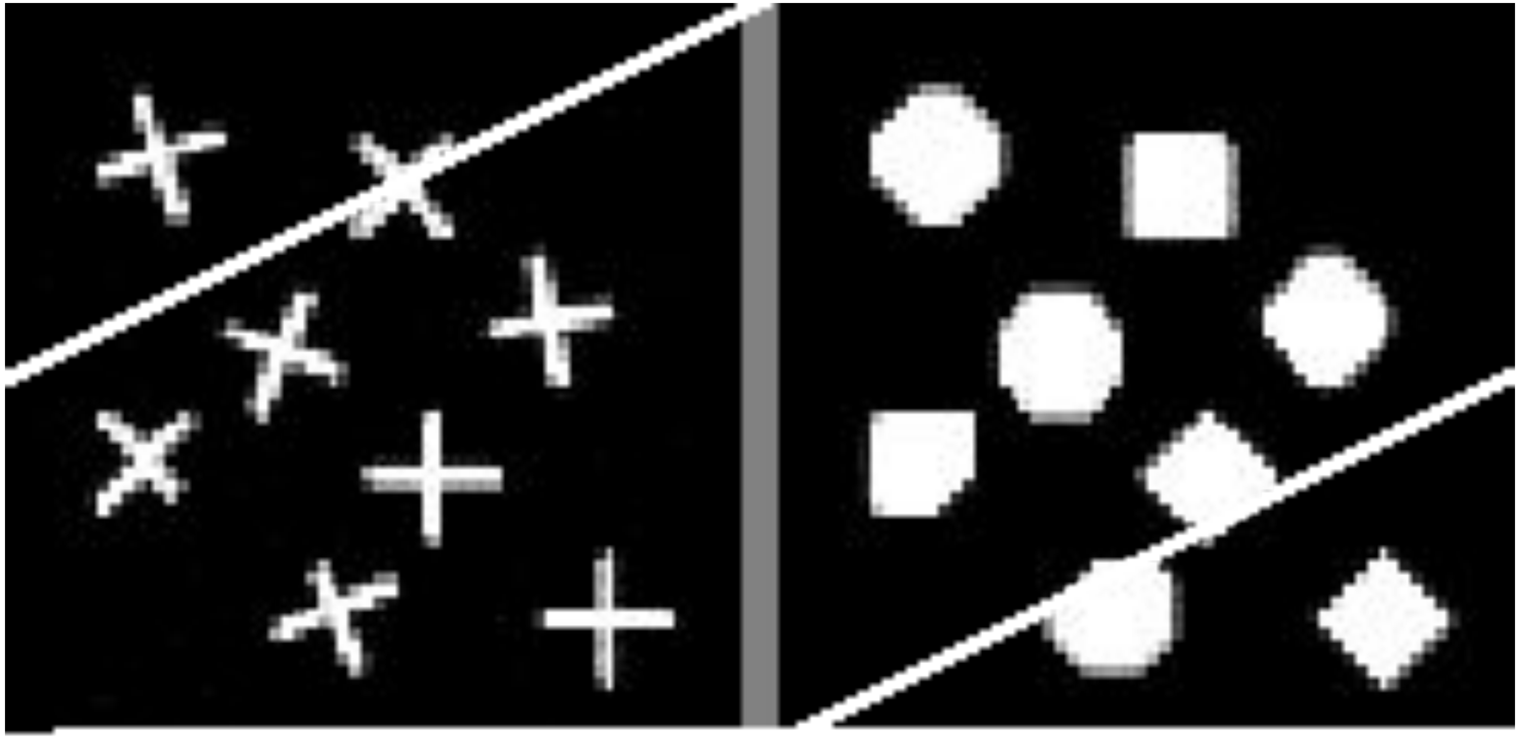


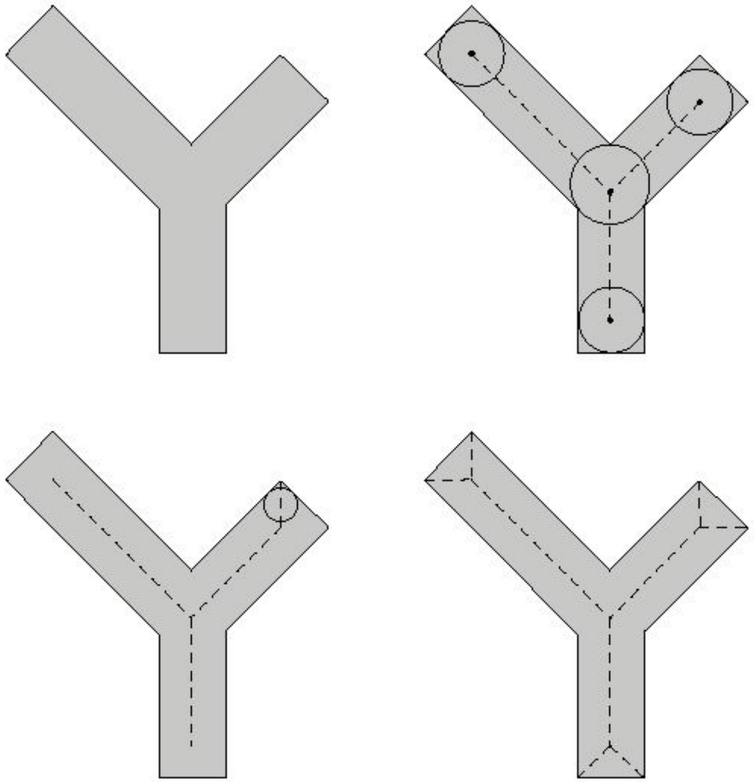
FIGURE 9.22 (a) Set A. (b) Complement of A. (c) Result of thinning the complement of A. (d) Thickened set obtained by complementing (c). (e) Final result, with no disconnected points.

6. Thickening - Example



Original   Thick

7. Skeleton



a b
c d

FIGURE 9.23
(a) Set A.
(b) Various positions of maximum disks with centers on the skeleton of A.
(c) Another maximum disk on a different segment of the skeleton of A.
(d) Complete skeleton.

- The notion of a skeleton S(A) of a set A is intuitively defined, we deduce from this figure that:

  If z is a point of S(A) and (D)z is the largest disk centered in z and contained in A (one cannot find a larger disk that fulfils this terms) – this disk is called “maximum disk”

  The disk (D)z touches the boundary of A at two or more different places.

7. Skeleton (Cont.)

- The skeleton of A is defined by terms of erosions and openings:

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

- With $S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B$
- Where B is the structuring element and $(A \ominus kB)$ indicates k successive erosions
- of A:

$$(A \ominus kB) = (\ldots((A \ominus B) \ominus B) \ominus \ldots) \ominus B$$

  - k times, and K is the last iterative step before A erodes to an empty set, in other words:

$$K = \max\{k \mid (A \ominus kB) \neq \emptyset\}$$

- in conclusion S(A) can be obtained as the union of skeleton subsets Sk(A).

7. Skeleton - Working

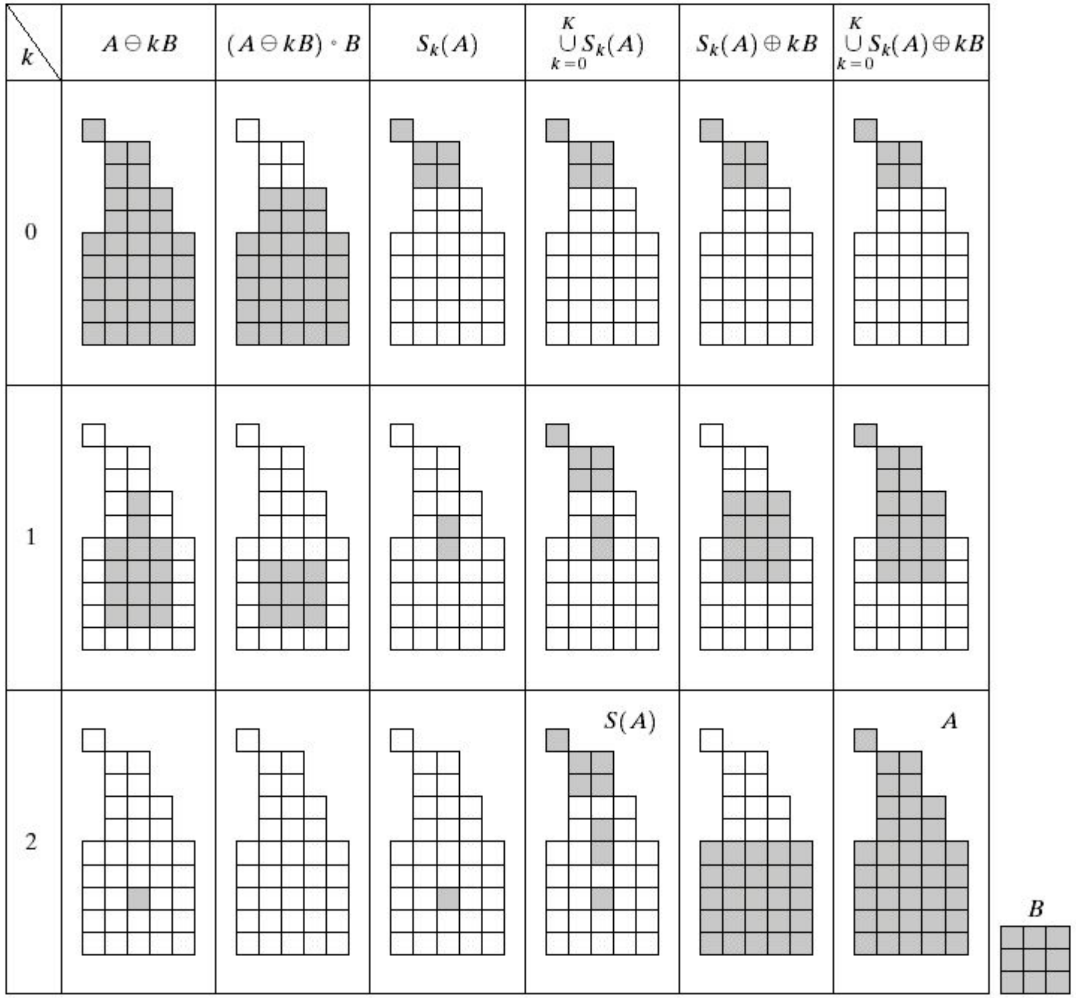
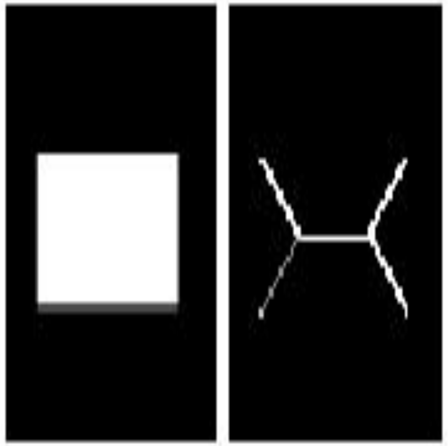
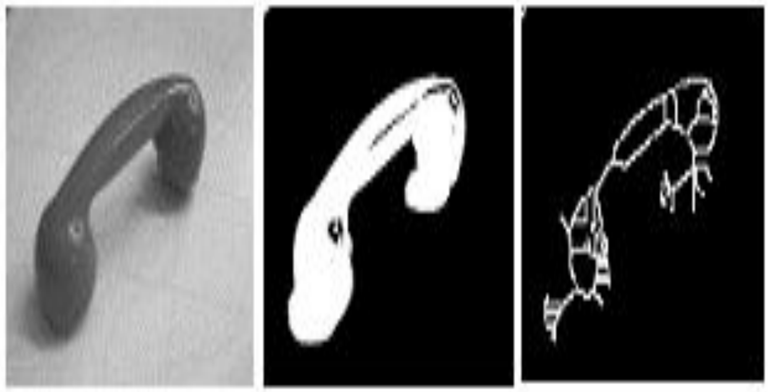
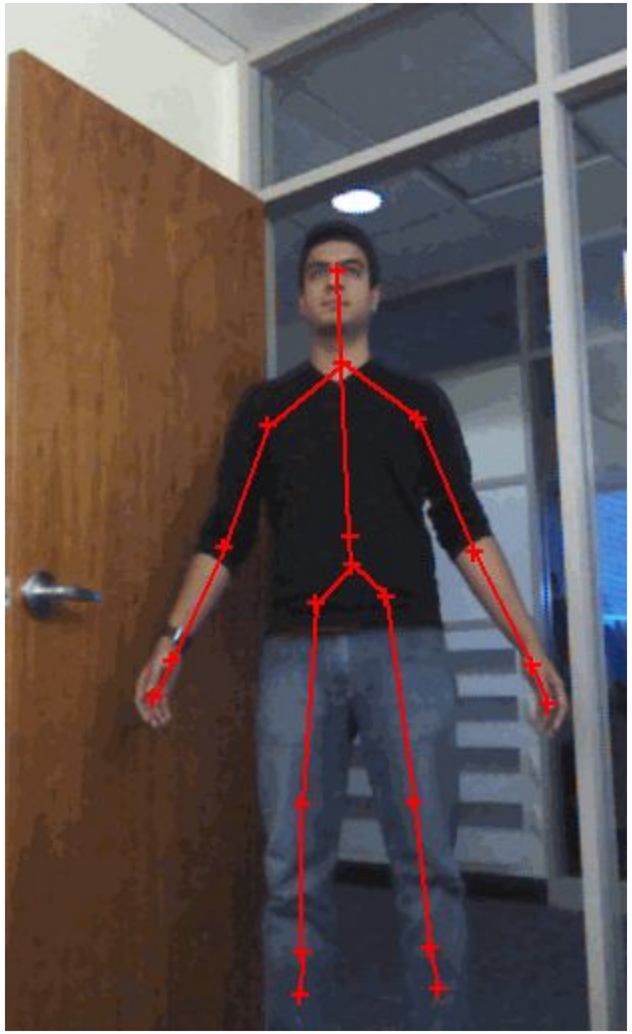


FIGURE 9.24 Implementation of Eqs. (9.5-11) through (9.5-15). The original set is at the top left, and its morphological skeleton is at the bottom of the fourth column. The reconstructed set is at the bottom of the sixth column.

7. Skeleton - Example







8. Pruning

- Pruning methods are essential complement to thinning and skeletonizing algorithms because tend to leave parasitic components that need to be "cleaned up" by post processing

$$X_1 = A \otimes \{B\}$$

$$X_2 = \bigcup_{k=1}^{8} \left(X_1 \circledast B^k \right)$$

$$X_3 = \left(X_2 \oplus H \right) \cap A$$

H : 3×3 structuring element

$$X_4 = X_1 \cup X_3$$

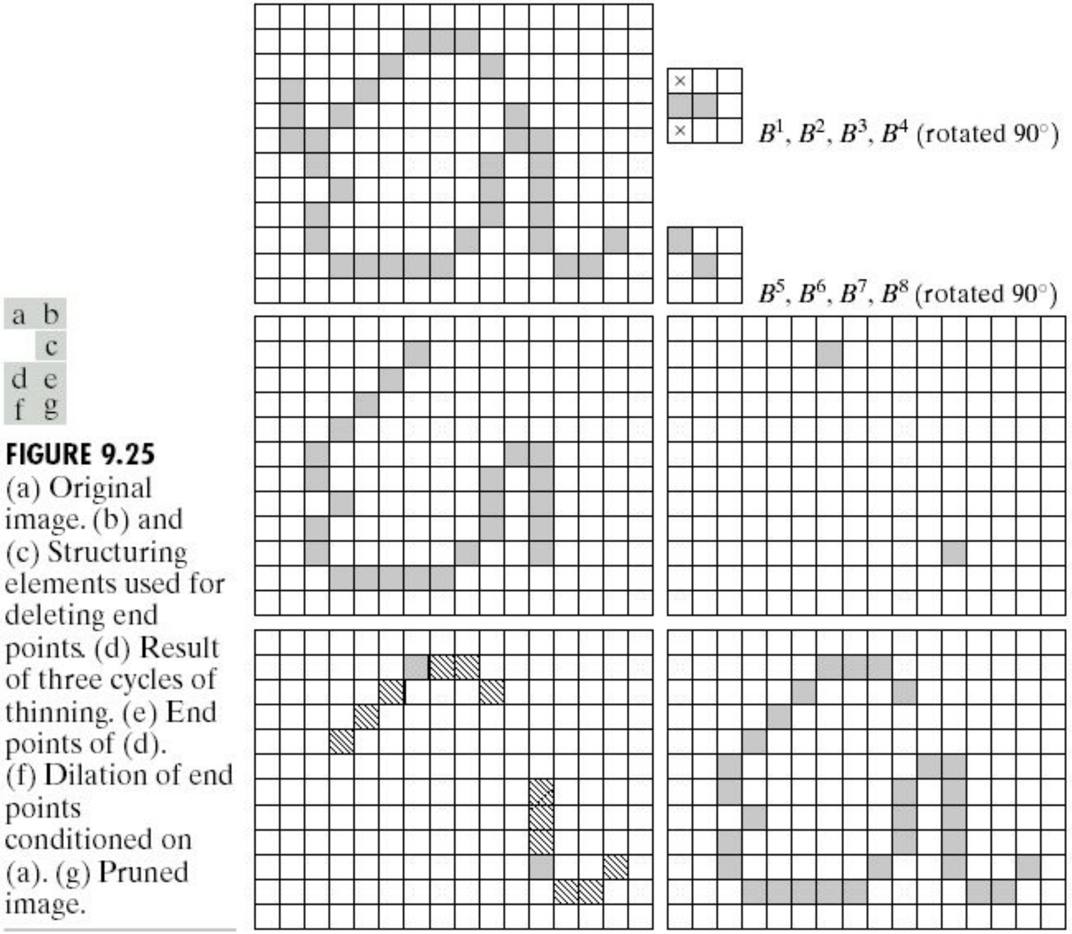
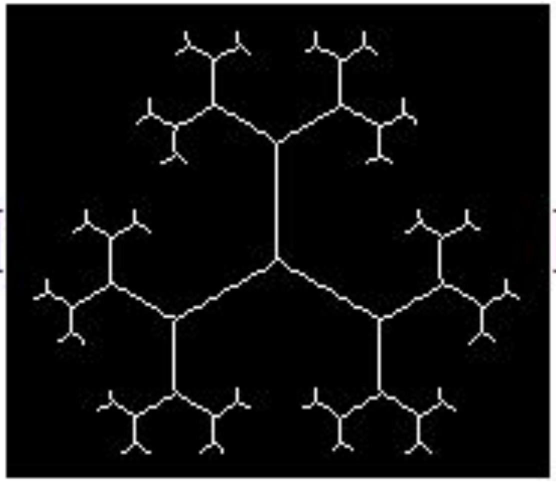
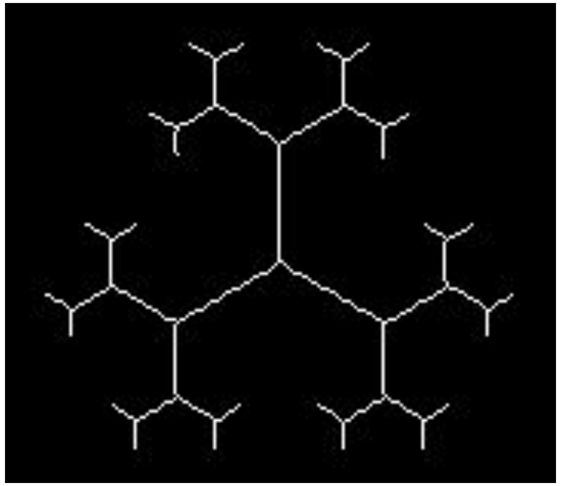
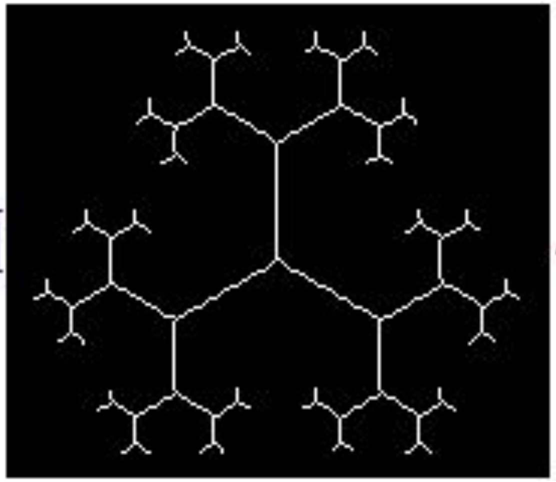


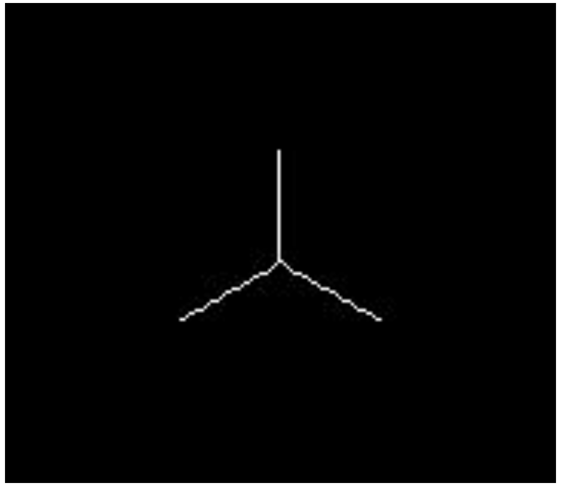
FIGURE 9.25 (a) Original image. (b) and (c) Structuring elements used for deleting end points. (d) Result of three cycles of thinning. (e) End points of (d). (f) Dilation of end points conditioned on (a). (g) Pruned image.

8. Pruning - Example

Pruning[]



Pruning[ , 60]



Summary of Morphological Operations on Binary Image

TABLE 9.2 Summary of morphological operations and their properties.

Operation	Equation	Comments (The Roman numerals refer to the structuring elements shown in Fig. 9.26).
Translation	$(A)_z = \{w \mid w = a + z, \quad \text{for } a \in A\}$	Translates the origin of A to point z.
Reflection	$\hat{B} = \{w \mid w = -b, \quad \text{for } b \in B\}$	Reflects all elements of B about the origin of this set.
Complement	$A^c = \{w \mid w \notin A\}$	Set of points not in A.
Difference	$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c$	Set of points that belong to A but not to B.
Dilation	$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \varnothing\}$	"Expands" the boundary of A. (I)
Erosion	$A \ominus B = \{z \mid (B)_z \subseteq A\}$	"Contracts" the boundary of A. (I)
Opening	$A \circ B = (A \ominus B) \oplus B$	Smoothes contours, breaks narrow isthmuses, and eliminates small islands and sharp peaks. (I)
Closing	$A \bullet B = (A \oplus B) \ominus B$	Smoothes contours, fuses narrow breaks and long thin gulfs, and eliminates small holes. (I)

Summary of Morphological Operations on Binary Image (Cont.)

Hit-or-miss transform	$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$ $= (A \ominus B_1) - (A \oplus \hat{B}_2)$	The set of points (coordinates) at which, simultaneously, B_1 found a match ("hit") in A and B_2 found a match in A^c.
Boundary extraction	$\beta(A) = A - (A \ominus B)$	Set of points on the boundary of set A. (I)
Region filling	$X_k = (X_{k-1} \oplus B) \cap A^c; X_0 = p$ and $k = 1, 2, 3, \ldots$	Fills a region in A, given a point p in the region. (II)
Connected components	$X_k = (X_{k-1} \oplus B) \cap A; X_0 = p$ and $k = 1, 2, 3, \ldots$	Finds a connected component Y in A, given a point p in Y. (I)
Convex hull	$X_k^i = (X_{k-1}^i \circledast B^i) \cup A; i = 1, 2, 3, 4;$ $k = 1, 2, 3, \ldots; X_0^i = A$; and $D^i = X_{\text{conv}}^i$.	Finds the convex hull $C(A)$ of set A, where "conv" indicates convergence in the sense that $X_k^i = X_{k-1}^i$. (III)

Summary of morphological results and their properties

Operation	Equation	Comments (The Roman numerals refer to the structuring elements shown in Fig. 9.26).
Thinning	$A \otimes B = A - (A \circledast B)$ $= A \cap (A \circledast B)^c$ $A \otimes \{B\} =$ $((\ldots((A \otimes B^1) \otimes B^2) \ldots) \otimes B^n)$ $\{B\} = \{B^1, B^2, B^3, \ldots, B^n\}$	Thins set A. The first two equations give the basic definition of thinning. The last two equations denote thinning by a sequence of structuring elements. This method is normally used in practice. (IV)
Thickening	$A \odot B = A \cup (A \circledast B)$ $A \odot \{B\} =$ $((\ldots(A \odot B^1) \odot B^2 \ldots) \odot B^n)$	Thickens set A. (See preceding comments on sequences of structuring elements.) Uses IV with 0's and 1's reversed.

TABLE 9.2 Summary of morphological results and their properties. *(continued)*

Summary of morphological results and their properties

Skeletons	$S(A) = \bigcup_{k=0} S_k(A)$ $S_k(A) = \bigcup_{k=0}^{K} \{(A \ominus kB) - [(A \ominus kB) \circ B]\}$ Reconstruction of A: $A = \bigcup_{k=0}^{K} (S_k(A) \oplus kB)$	Finds the skeleton $S(A)$ of set A. The last equation indicates that A can be reconstructed from its skeleton subsets $S_k(A)$. In all three equations, K is the value of the iterative step after which the set A erodes to the empty set. The notation $(A \ominus kB)$ denotes the kth iteration of successive erosion of A by B. (I)
Pruning	$X_1 = A \otimes \{B\}$ $X_2 = \bigcup_{k=1}^{8} (X_1 \circledast B^k)$ $X_3 = (X_2 \oplus H) \cap A$ $X_4 = X_1 \cup X_3$	X_4 is the result of pruning set A. The number of times that the first equation is applied to obtain X_1 must be specified. Structuring elements V are used for the first two equations. In the third equation H denotes structuring element I.

Basic Types of Structuring Elements

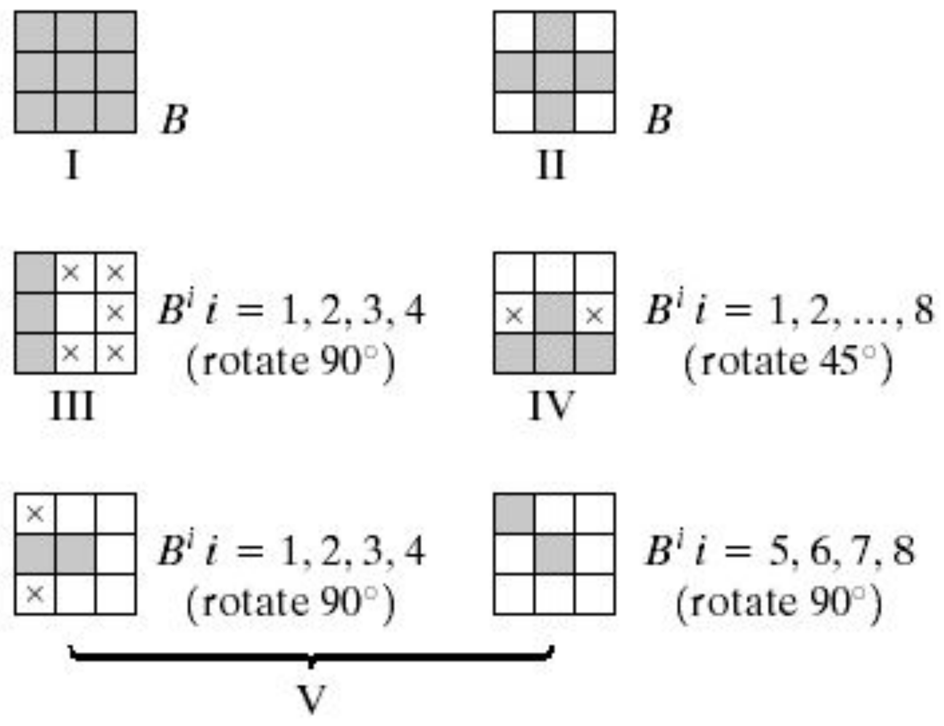
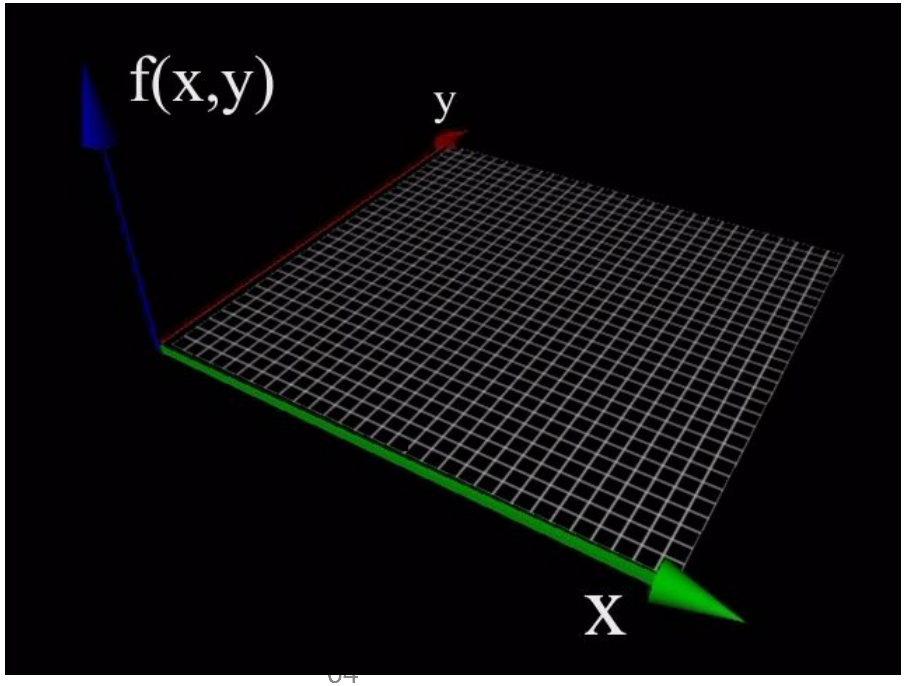


FIGURE 9.26 Five basic types of structuring elements used for binary morphology. The origin of each element is at its center and the ×'s indicate "don't care" values.

Extensions to Gray-Scale Images

Gray-Scale Images

- In gray scale images on the contrary to binary images we deal with digital image functions of the form f(x,y) as an input image and b(x,y) as a structuring element.
- (x,y) are integers from Z*Z that represent a coordinates in the image.
- f(x,y) and b(x,y) are functions that assign gray level value to each distinct pair of coordinates.
- For example the domain of gray values can be 0-255, whereas 0 – is black, 255- is white.



1. Dilation – Gray-Scale

- Equation for gray-scale dilation is:

$$(f \oplus b)(s,t) =$$

$$\max\{f(s-x,t-y)+b(x,y) \mid (s-x),(t-y) \in D_f,(x,y) \in D_b\}$$

- D_f and D_b are domains of f and b.
- The condition that (s-x),(t-y) need to be in the domain of f and x,y in the domain of b, is analogous to the condition in the binary definition of dilation, where the two sets need to overlap by at least one element.
- We will illustrate the previous equation in terms of 1-D.
  and we will receive an equation for 1 variable:

$$(f \oplus b)(s) = \max\{f(s-x)+b(x) \mid (s-x) \in D_f \; and \; x \in D_b\}$$

- The requirements the (s-x) is in the domain of f and x is in the domain of b imply that f and b overlap by at least one element.
- Unlike the binary case, f, rather than the structuring element b is shifted.
- Conceptually f sliding by b is really not different than b sliding by f.

1. Dilation – Gray-Scale (Cont.)

- The general effects of performing dilation on a gray scale image is twofold:

a. If all the values of the structuring elements are positive, then <u>the output image tends to be brighter than the input.</u>

b. Dark details either are reduced or eliminated, depending on how their values and shape relate to the structuring element used for dilation

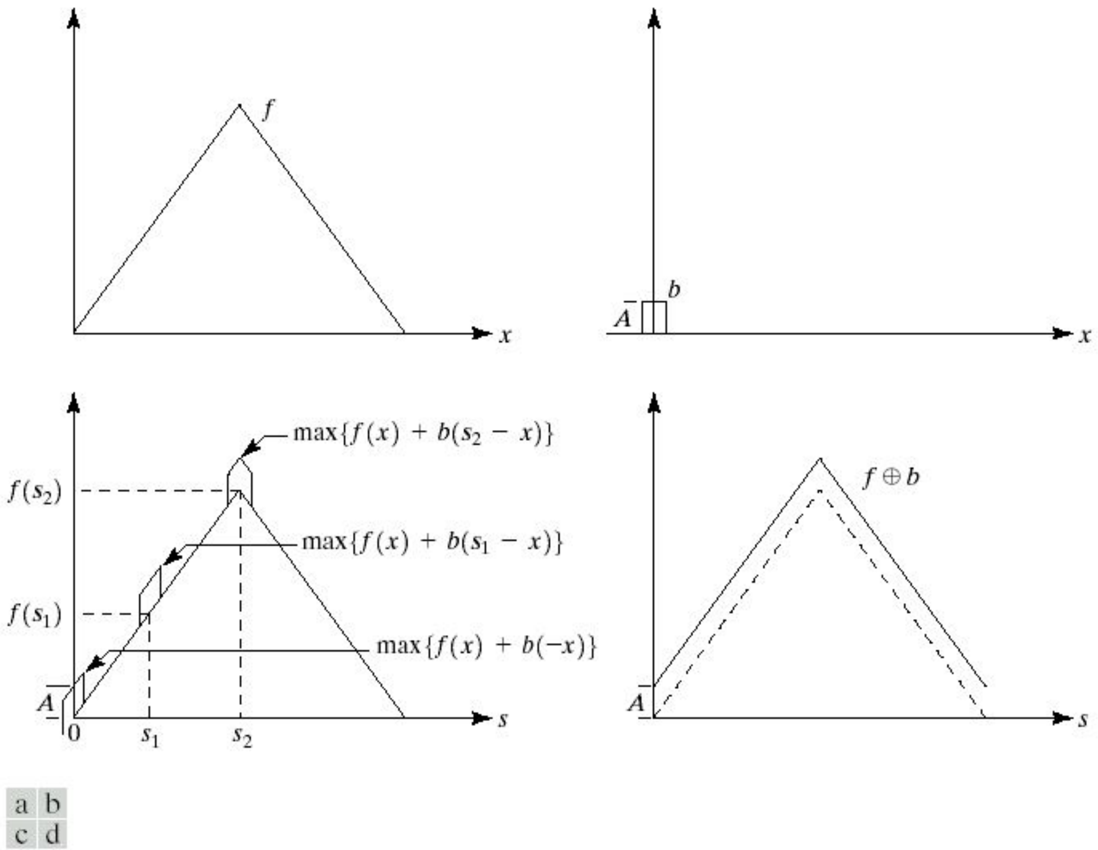


FIGURE 9.27 (a) A simple function. (b) Structuring element of height A. (c) Result of dilation for various positions of sliding b past f. (d) Complete result of dilation (shown solid).

2. Erosion – Gray-Scale

- Gray-scale erosion is defined as:

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

- The condition that (s+x),(t+y) have to be in the domain of f, and x,y have to be in the domain of b, is completely analogous to the condition in the binary definition of erosion, where the structuring element has to be completely combined by the set being eroded.
- The same as in erosion we illustrate with 1-D function

$$(f \ominus b)(s) = \min\{f(s+x) - b(x) | (s+x) \in D_f \text{ and } x \in D_b\}$$

- General effect of performing an erosion in grayscale images:
  - ❖ If all elements of the structuring element are positive, the output image tends to be darker than the input image.
  - ❖ The effect of bright details in the input image that are smaller in area than the structuring element is reduced, with the degree of reduction being determined by the grayscale values surrounding by the bright detail and by shape and amplitude values of the structuring element itself.
- Similar to binary image grayscale erosion and dilation are duals with respect to function complementation and reflection

2. Erosion – Gray-Scale

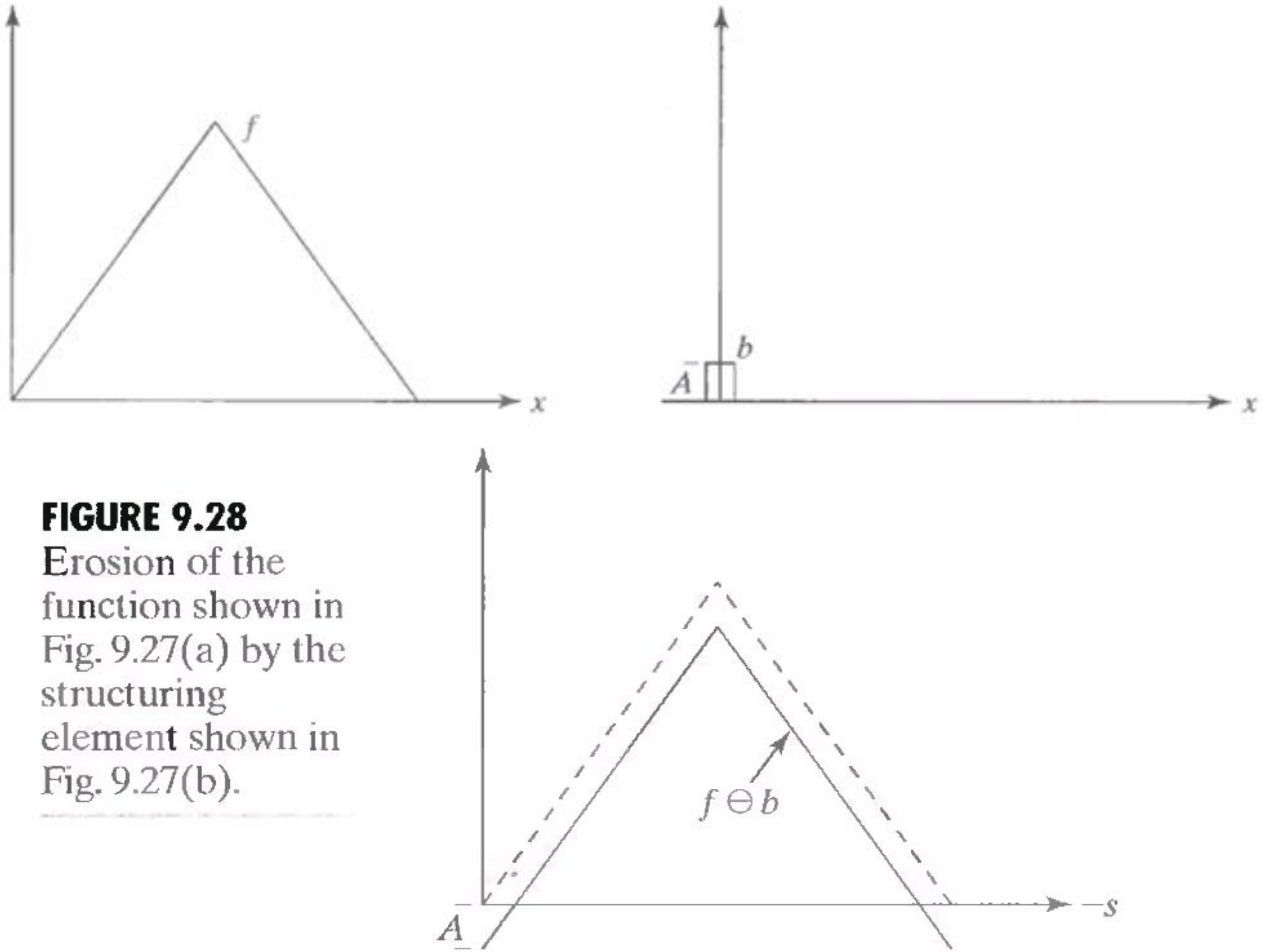
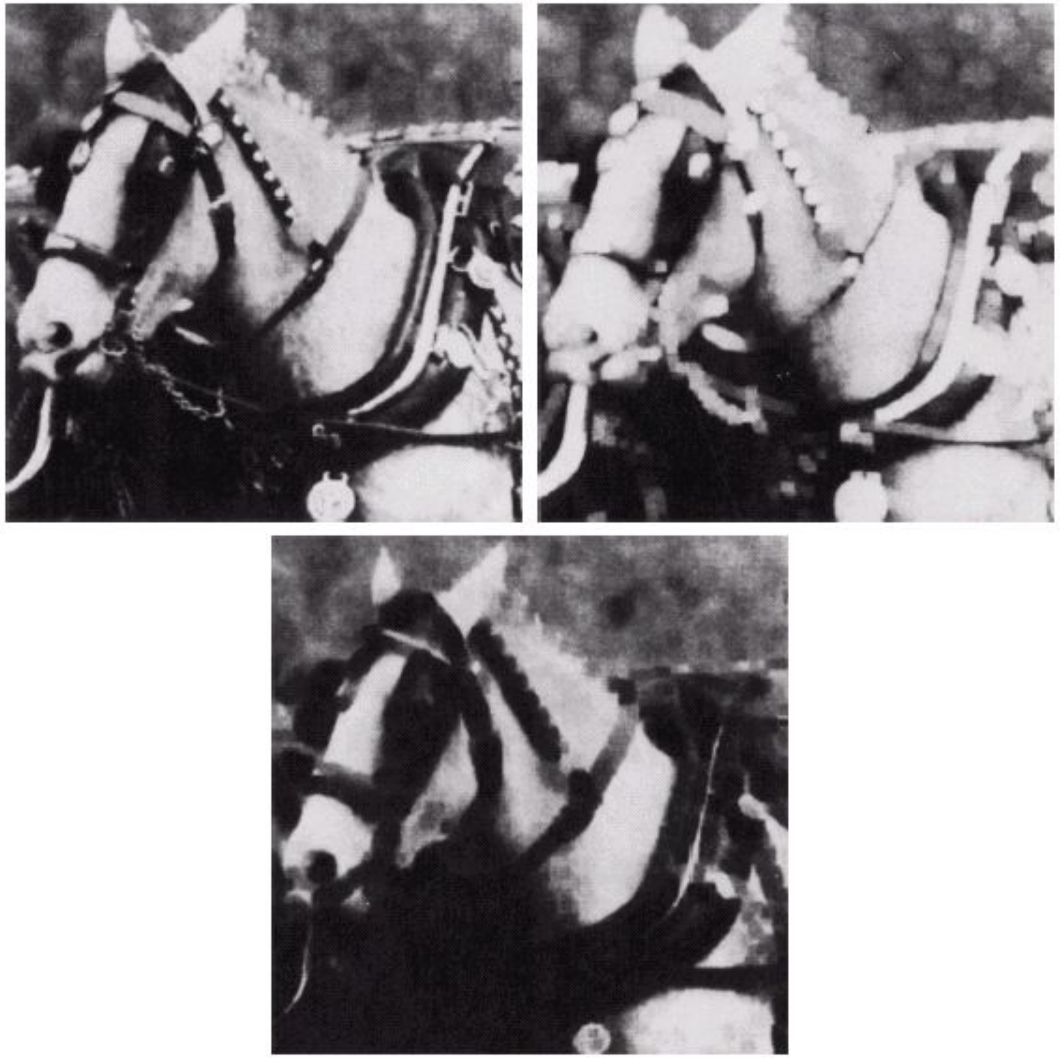


FIGURE 9.28
Erosion of the function shown in Fig. 9.27(a) by the structuring element shown in Fig. 9.27(b).

Example of Dilation & Erosion – Gray-Scale



a b
c

FIGURE 9.29 (a) Original image. (b) Result of dilation. (c) Result of erosion. (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

Example of Dilation & Erosion – Gray-Scale

a b
c d

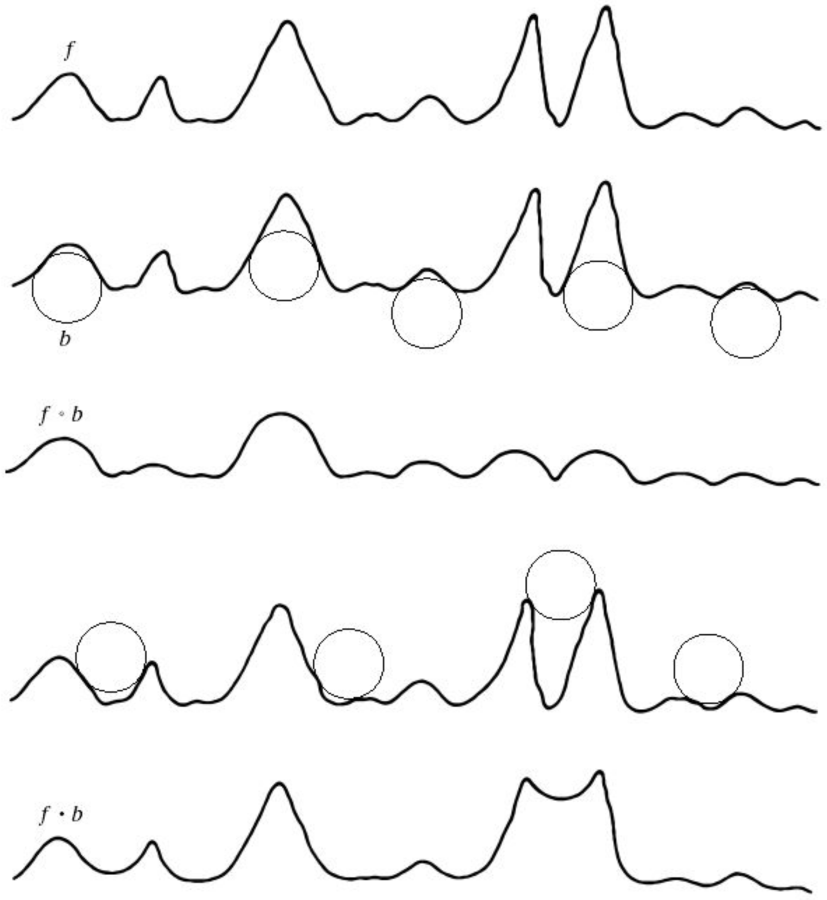
FIGURE 10.23 Dilation and erosion. (a) Original image. (b) Dilated image. (c) Eroded image. (d) Morphological gradient. (Original image courtesy of NASA.)

3. Opening & Closing

- Similar to the binary algorithms
- Opening – $f \circ b = (f \ominus b) \oplus b.$
- Closing – $f \bullet b = (f \oplus b) \ominus b.$
- In the opening of a gray-scale image, we remove small light details, while relatively undisturbed overall gray levels and larger bright features
- In the closing of a gray-scale image, we remove small dark details, while relatively undisturbed overall gray levels and larger dark features

3. Opening & Closing (cont.)

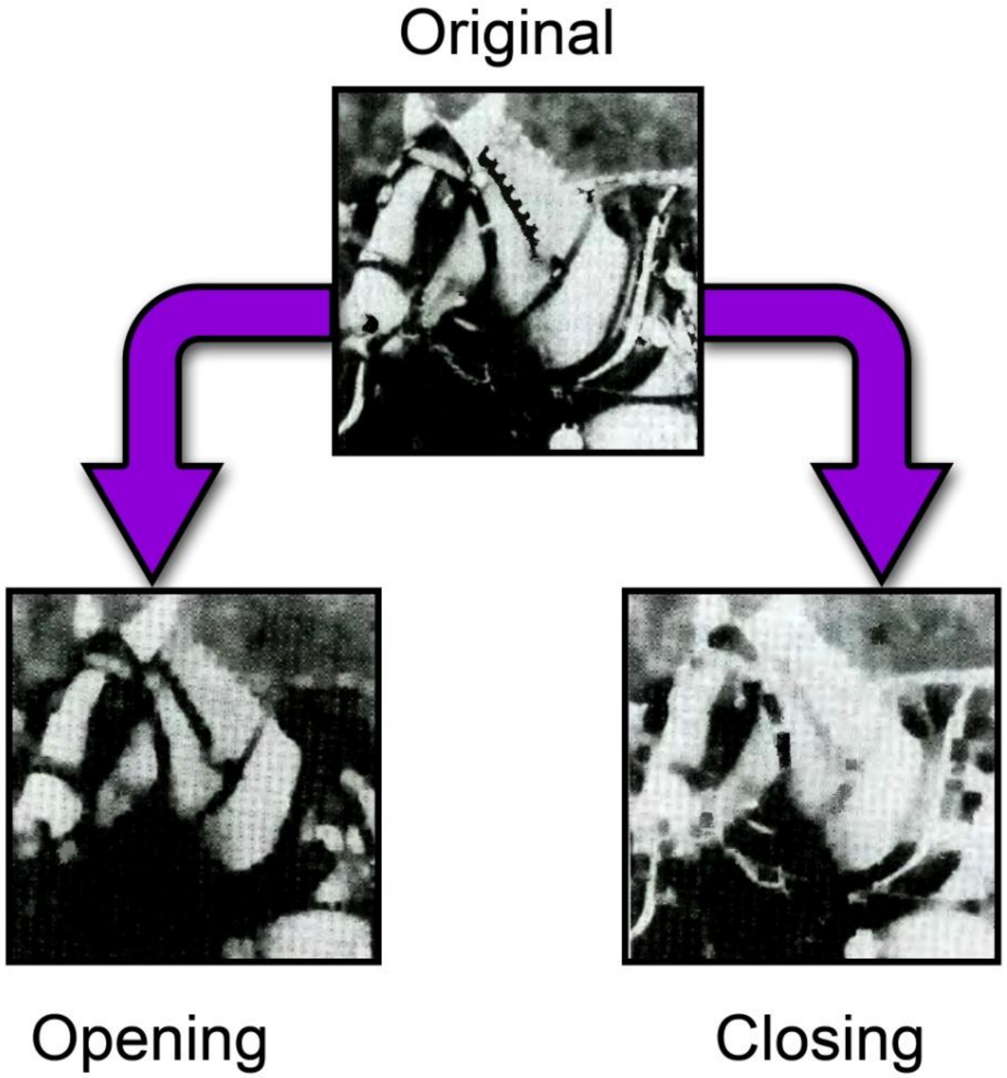
- Opening a G-S picture is describable as pushing object B under the scan-line graph, while traversing the graph according the curvature of B
- Closing a G-S picture is describable as pushing object B on top of the scan-line graph, while traversing the graph according the curvature of B
- The peaks are usually remains in their original form



a
b
c
d
e

FIGURE 9.30 (a) A gray-scale scan line. (b) Positions of rolling ball for opening. (c) Result of opening. (d) Positions of rolling ball for closing. (e) Result of closing.

3. Opening & Closing - Example



Summary

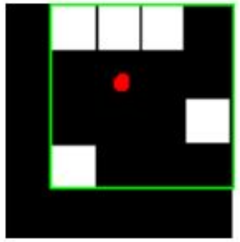
- The morphological concepts and techniques constitutes a powerful set of tools for extracting features of interest in an image.

- Morphological techniques have evolved from set-theory.

- Dilation and erosion are primitive operations that are the basis for morphological algorithms.

- Morphology cab be used as the basis for developing image segmentations procedures with a wide range of applications.

Shape Measurement

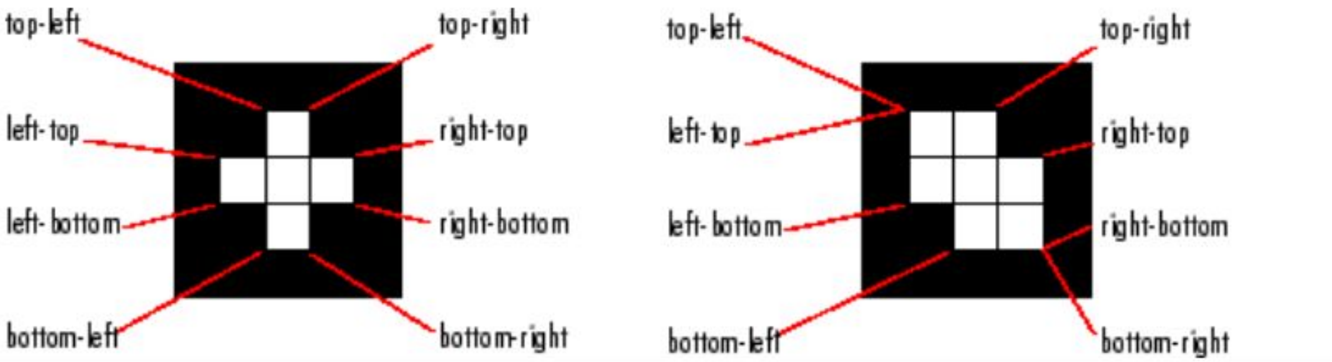
Property Name	Description
'Area'	Actual number of pixels in the region, returned as a scalar. (This value might differ slightly from the value returned by bwarea, which weights different patterns of pixels differently.) To find the equivalent to the area of a 3-D volume, use the 'Volume' property of regionprops3.
'BoundingBox'	Smallest rectangle containing the region, returned as a 1-by-Q*2 vector, where Q is the number of image dimensions. For example, in the vector [ul_corner width], ul_corner specifies the upper-left corner of the bounding box in the form [x y z ...]. width specifies the width of the bounding box along each dimension in the form [x_width y_width ...]. regionprops uses ndims to get the dimensions of label matrix or binary image, ndims(L), and numel to get the dimensions of connected components, numel(CC.ImageSize).
'Centroid'	Center of mass of the region, returned as a 1-by-Q vector. The first element of Centroid is the horizontal coordinate (or *x*-coordinate) of the center of mass. The second element is the vertical coordinate (or *y*-coordinate). All other elements of Centroid are in order of dimension. This figure illustrates the centroid and bounding box for a discontiguous region. The region consists of the white pixels; the green box is the bounding box, and the red dot is the centroid. 

Shape Measurement

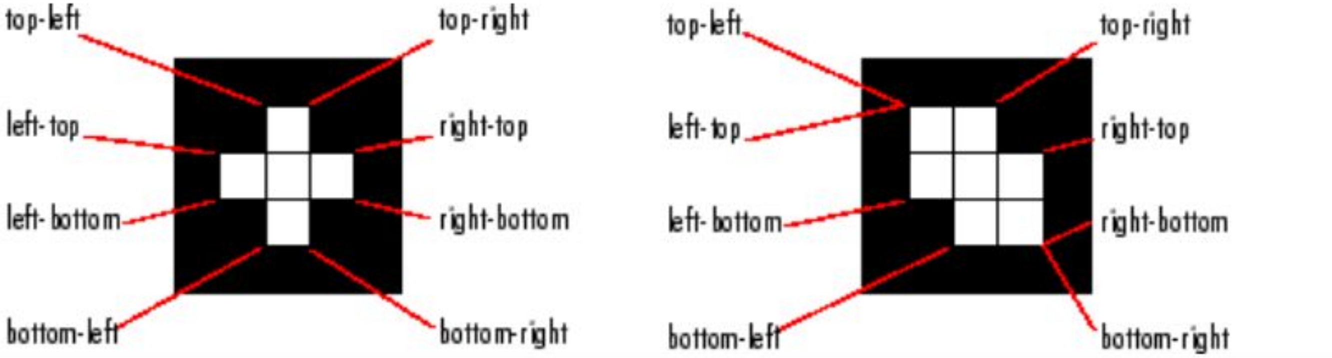
`'ConvexArea'`	Number of pixels in `'ConvexImage'`, returned as a scalar.
`'ConvexHull'`	Smallest convex polygon that can contain the region, returned as a p-by-2 matrix. Each row of the matrix contains the *x*- and *y*-coordinates of one vertex of the polygon.
`'ConvexImage'`	Image that specifies the convex hull, with all pixels within the hull filled in (set to `on`), returned as a binary image (`logical`). The image is the size of the bounding box of the region. (For pixels that the boundary of the hull passes through, `regionprops` uses the same logic as `roipoly` to determine whether the pixel is inside or outside the hull.)
`'Circularity'`	Circularity that specifies the roundness of objects, returned as a struct with field `Circularity`. The struct contains the circularity value for each object in the input image. The circularity value is computed as `(4*Area*pi)/(Perimeter`^{2}`)`. For a perfect circle, the circularity value is 1. The input must be a label matrix or binary image with contiguous regions. If the image contains discontiguous regions, `regionprops` returns unexpected results. **Circularity is not recommended for very small objects such as a 3*3 square. For such cases the results might exceed the circularity value for a perfect circle which is 1.**
`'Eccentricity'`	Eccentricity of the ellipse that has the same second-moments as the region, returned as a scalar. The eccentricity is the ratio of the distance between the foci of the ellipse and its major axis length. The value is between 0 and 1. (0 and 1 are degenerate cases. An ellipse whose eccentricity is 0 is actually a circle, while an ellipse whose eccentricity is 1 is a line segment.)

Shape Measurement

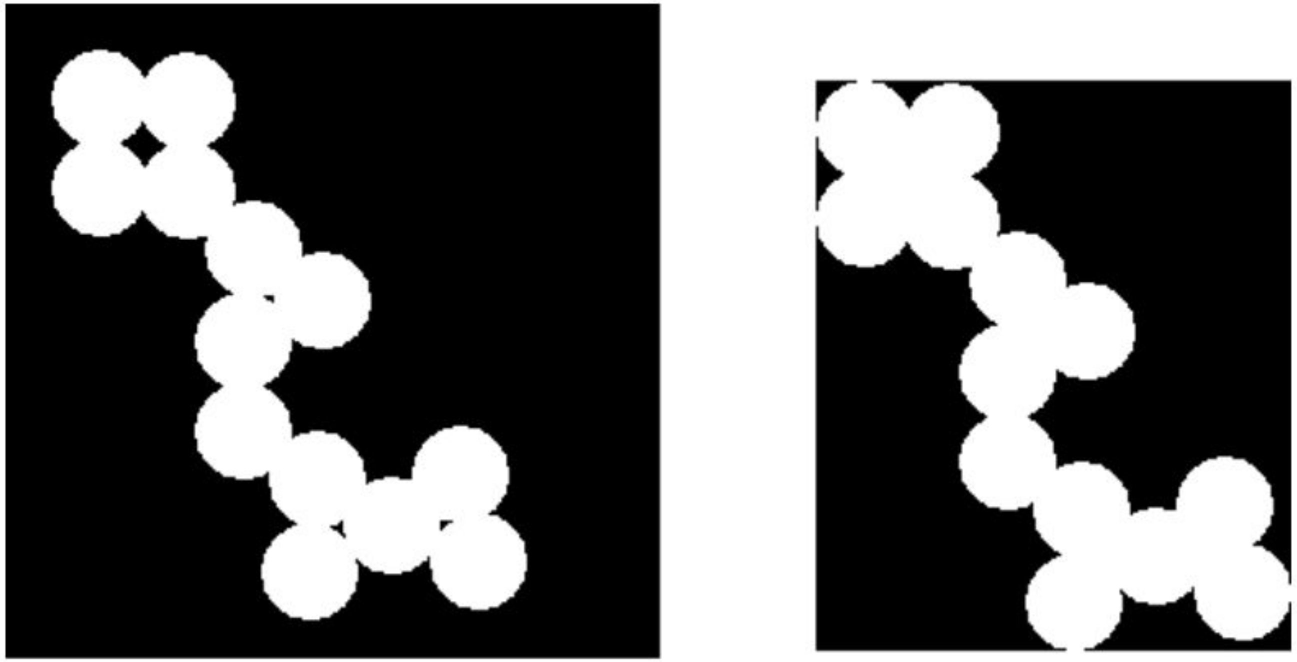
`'EquivDiameter'`	Diameter of a circle with the same area as the region, returned as a scalar. Computed as `sqrt(4*Area/pi)`.
`'EulerNumber'`	Number of objects in the region minus the number of holes in those objects, returned as a scalar. This property is supported only for 2-D label matrices. `regionprops` uses 8-connectivity to compute the Euler number measurement. To learn more about connectivity, see Pixel Connectivity.
`'Extent'`	Ratio of pixels in the region to pixels in the total bounding box, returned as a scalar. Computed as the `Area` divided by the area of the bounding box.
`'Extrema'`	Extrema points in the region, returned as an 8-by-2 matrix. Each row of the matrix contains the *x*- and *y*-coordinates of one of the points. The format of the vector is `[top-left top-right right-top right-bottom bottom-right bottom-left left-bottom left-top]`. This figure illustrates the extrema of two different regions. In the region on the left, each extrema point is distinct. In the region on the right, certain extrema points (e.g., `top-left` and `left-top`) are identical.



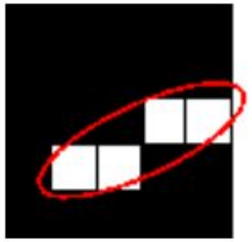
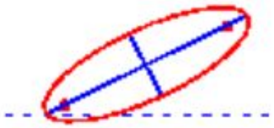
Shape Measurement

`'EquivDiameter'`	Diameter of a circle with the same area as the region, returned as a scalar. Computed as `sqrt(4*Area/pi)`.
`'EulerNumber'`	Number of objects in the region minus the number of holes in those objects, returned as a scalar. This property is supported only for 2-D label matrices. `regionprops` uses 8-connectivity to compute the Euler number measurement. To learn more about connectivity, see Pixel Connectivity.
`'Extent'`	Ratio of pixels in the region to pixels in the total bounding box, returned as a scalar. Computed as the `Area` divided by the area of the bounding box.
`'Extrema'`	Extrema points in the region, returned as an 8-by-2 matrix. Each row of the matrix contains the *x*- and *y*-coordinates of one of the points. The format of the vector is `[top-left top-right right-top right-bottom bottom-right bottom-left left-bottom left-top]`. This figure illustrates the extrema of two different regions. In the region on the left, each extrema point is distinct. In the region on the right, certain extrema points (e.g., `top-left` and `left-top`) are identical. 

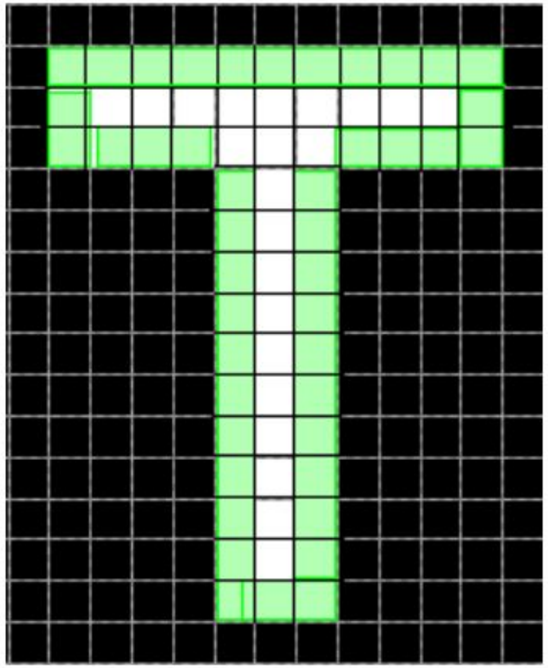
Shape Measurement

'FilledArea'	Number of on pixels in FilledImage, returned as a scalar.
'FilledImage'	Image the same size as the bounding box of the region, returned as a binary (logical) array. The on pixels correspond to the region, with all holes filled in, as shown in this figure.  Original Image, Containing a Single Region Image Returned
'Image'	Image the same size as the bounding box of the region, returned as a binary (logical) array. The on pixels correspond to the region, and all other pixels are off.

Shape Measurement

'MajorAxisLength'	Length (in pixels) of the major axis of the ellipse that has the same normalized second central moments as the region, returned as a scalar.
'MinorAxisLength'	Length (in pixels) of the minor axis of the ellipse that has the same normalized second central moments as the region, returned as a scalar.
'Orientation'	Angle between the *x*-axis and the major axis of the ellipse that has the same second-moments as the region, returned as a scalar. The value is in degrees, ranging from -90 degrees to 90 degrees. This figure illustrates the axes and orientation of the ellipse. The left side of the figure shows an image region and its corresponding ellipse. The right side shows the same ellipse with the solid blue lines representing the axes. The red dots are the foci. The orientation is the angle between the horizontal dotted line and the major axis.  

Shape Measurement

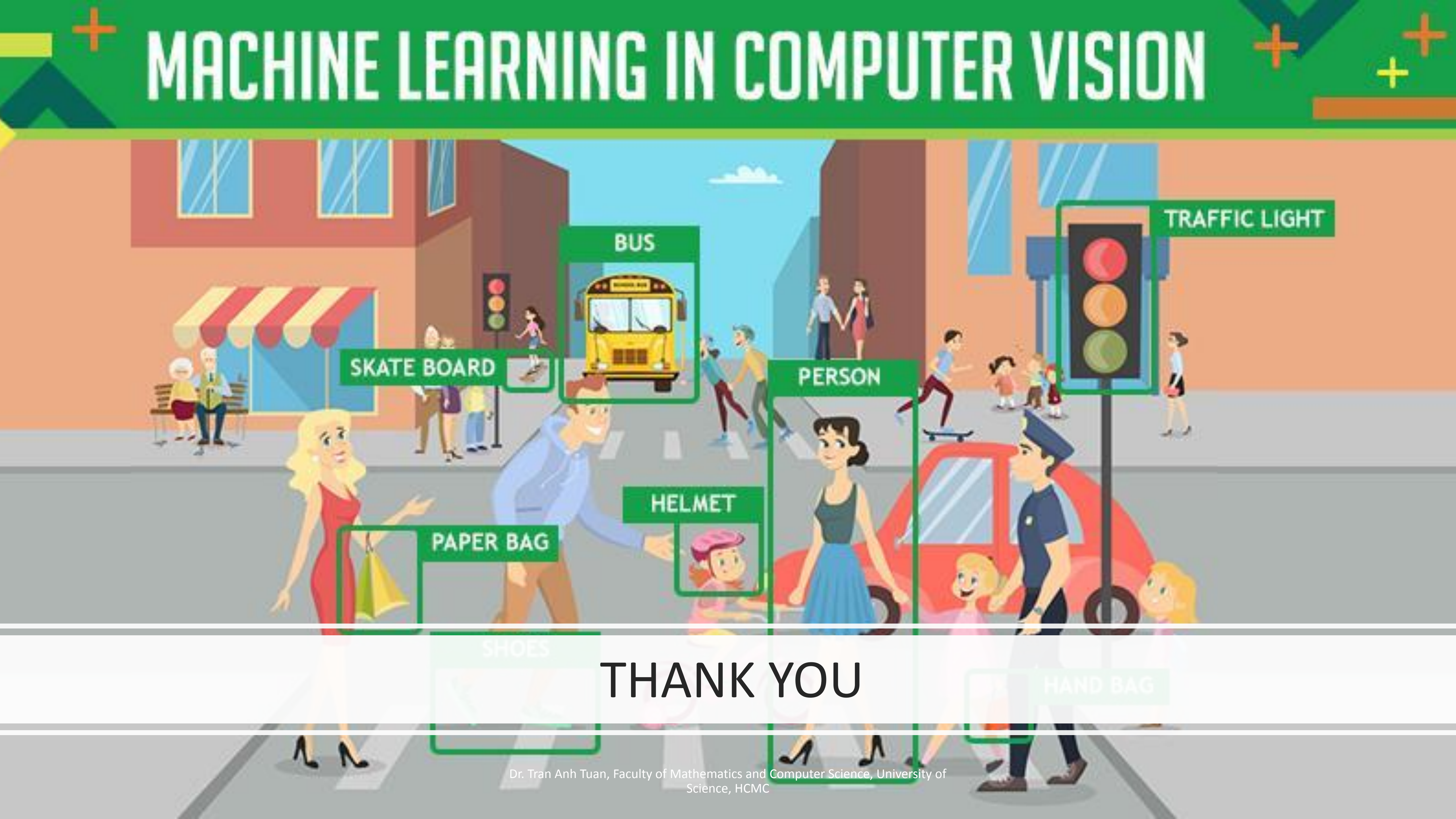
'Perimeter'	Distance around the boundary of the region returned as a scalar. `regionprops` computes the perimeter by calculating the distance between each adjoining pair of pixels around the border of the region. If the image contains discontiguous regions, `regionprops` returns unexpected results. This figure illustrates the pixels included in the perimeter calculation for this object. 
'PixelIdxList'	Linear indices of the pixels in the region, returned as a p-element vector.
'PixelList'	Locations of pixels in the region, returned as a p-by-`Q` matrix. Each row of the matrix has the form `[x y z ...]` and specifies the coordinates of one pixel in the region.

Shape Measurement

'PixelIdxList'	Linear indices of the pixels in the region, returned as a *p*-element vector.
'PixelList'	Locations of pixels in the region, returned as a *p*-by-Q matrix. Each row of the matrix has the form [x y z ...] and specifies the coordinates of one pixel in the region.
'Solidity'	Proportion of the pixels in the convex hull that are also in the region, returned as a scalar. Computed as Area/ConvexArea.
'SubarrayIdx'	Elements of L inside the object bounding box, returned as a cell array that contains indices such that L(idx{:}) extracts the elements.

Pixel Value Measurements

Property Name	Description
'MaxIntensity'	Value of the pixel with the greatest intensity in the region, returned as a scalar.
'MeanIntensity'	Mean of all the intensity values in the region, returned as a scalar.
'MinIntensity'	Value of the pixel with the lowest intensity in the region, returned as a scalar.
'PixelValues'	Number of pixels in the region, returned as a *p*-by-1 vector, where *p* is the number of pixels in the region. Each element in the vector contains the value of a pixel in the region.
'WeightedCentroid'	Center of the region based on location and intensity value, returned as a *p*-by-Q vector of coordinates. The first element of WeightedCentroid is the horizontal coordinate (or *x*-coordinate) of the weighted centroid. The second element is the vertical coordinate (or *y*-coordinate). All other elements of WeightedCentroid are in order of dimension.


MACHINE LEARNING IN COMPUTER VISION
TRAFFIC LIGHT
BUS
SKATE BOARD
PERSON
HELMET
PAPER BAG
SHOES
HAND BAG

THANK YOU