Image and Video Processing

Morphological Image Processing

Course Outline

Image Enhancement

Image Acquisition

Problem Domain

Image Restoration

Segmentation

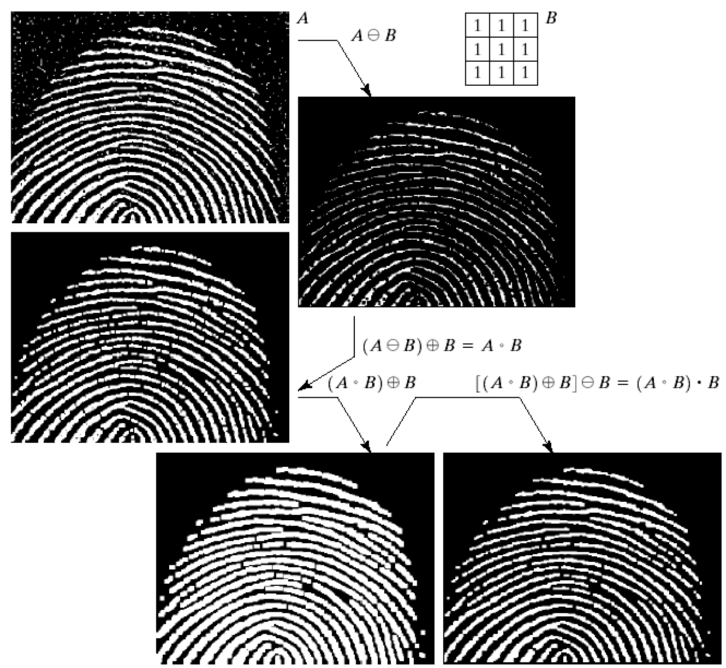
Morphological Processing

Representation & Description

Object Recognition

Colour Image Processing

Image Compression



Contents

Morphological operations can be used to remove imperfections in the segmented image and provide information on the form and structure of a region shape

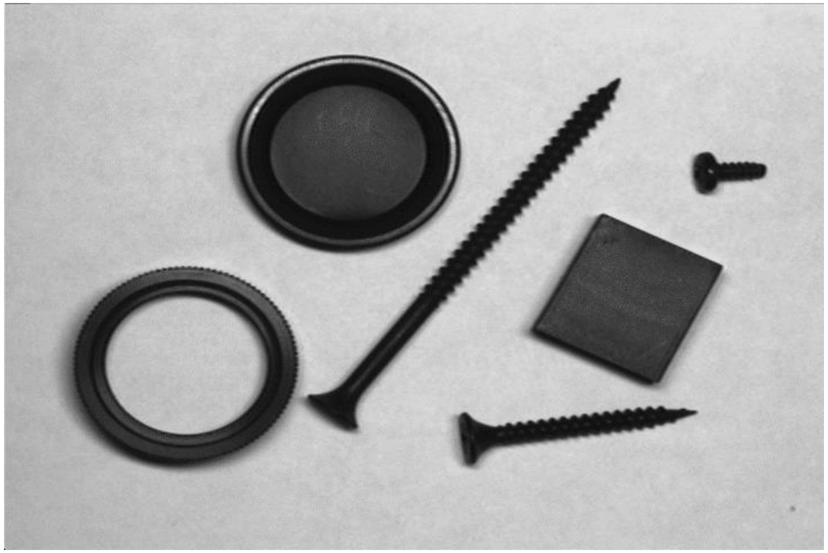
In this topic we will consider

- What is morphology?
- Simple morphological operations
- Compound operations
- Morphological algorithms

What Is Morphology?

- Morphological image processing (or *morphology*) describes a range techniques for extracting image components that are useful in the representation & description of region shape such as boundaries, skeletons etc.
- Some of the operations are frequently applied as post-processing to remove imperfections introduced during segmentation, and so typically operate on binary images.

Quick Example 1



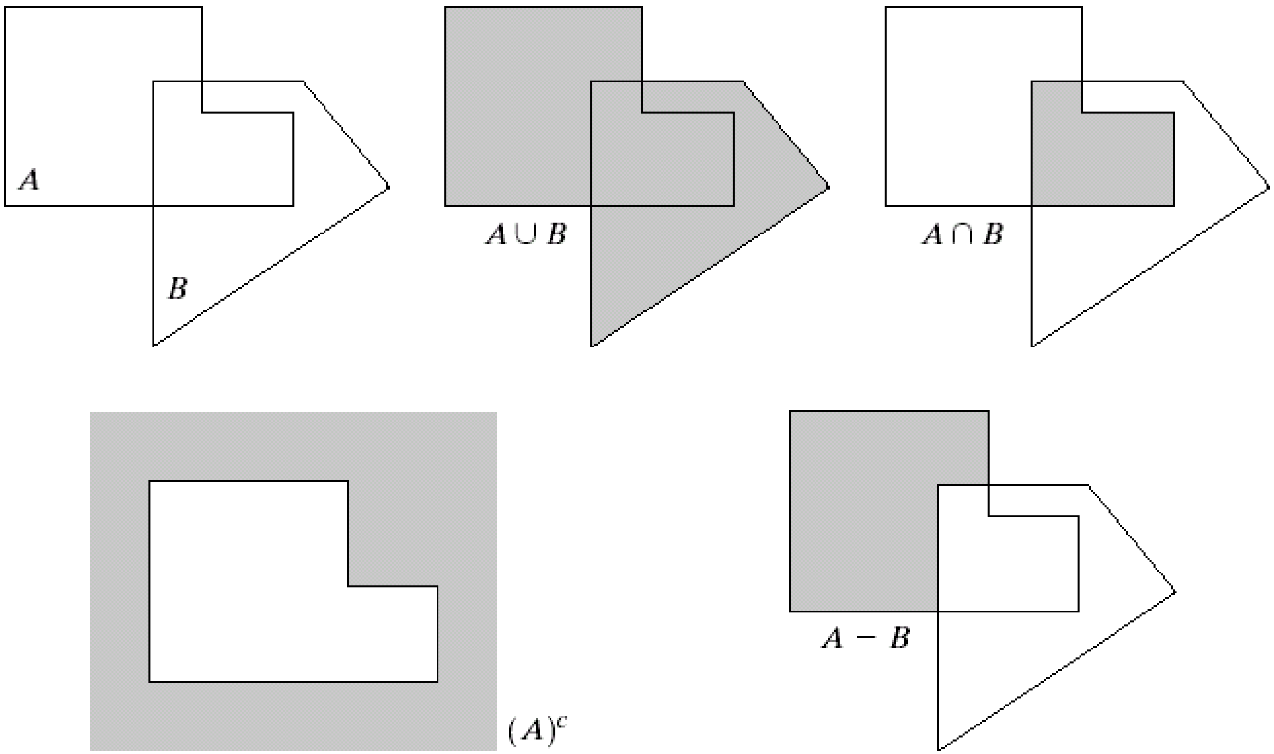
Quick Example 2

Image after segmentation

Image after segmentation and morphological processing

Preliminaries

- Language of mathematical morphology is set theory



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.

Preliminaries

- Logic operations involving binary images

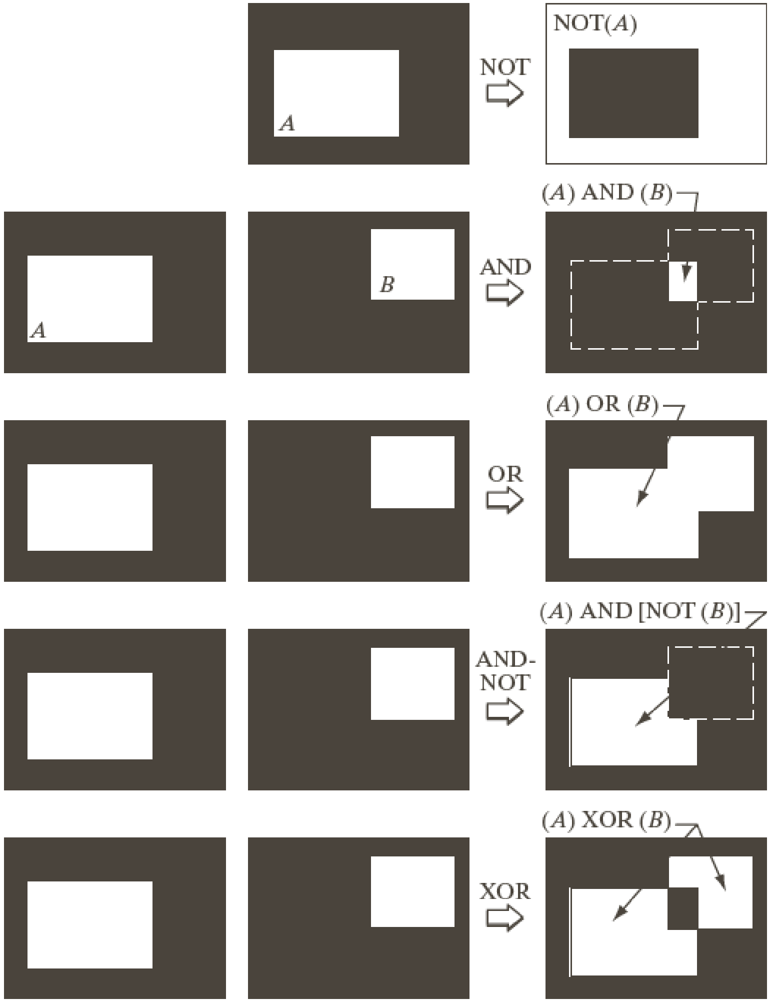


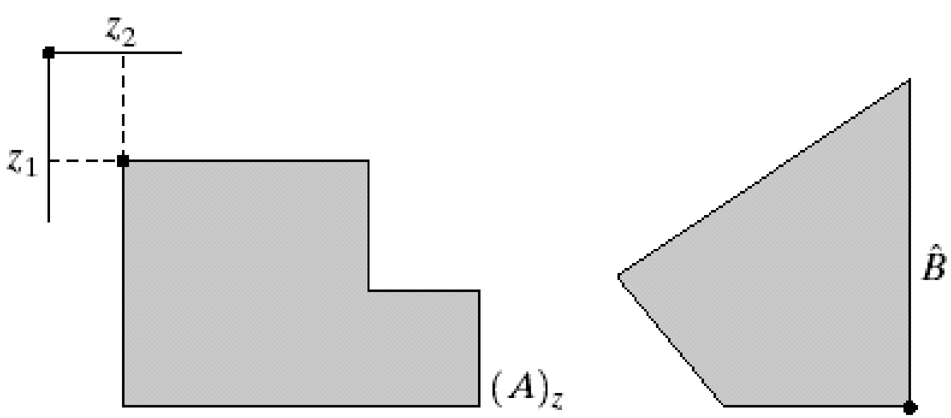
FIGURE 2.33 Illustration of logical operations involving foreground (white) pixels. Black represents binary 0s and white binary 1s. The dashed lines are shown for reference only. They are not part of the result.

Preliminaries

- Reflection and Translation:

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

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

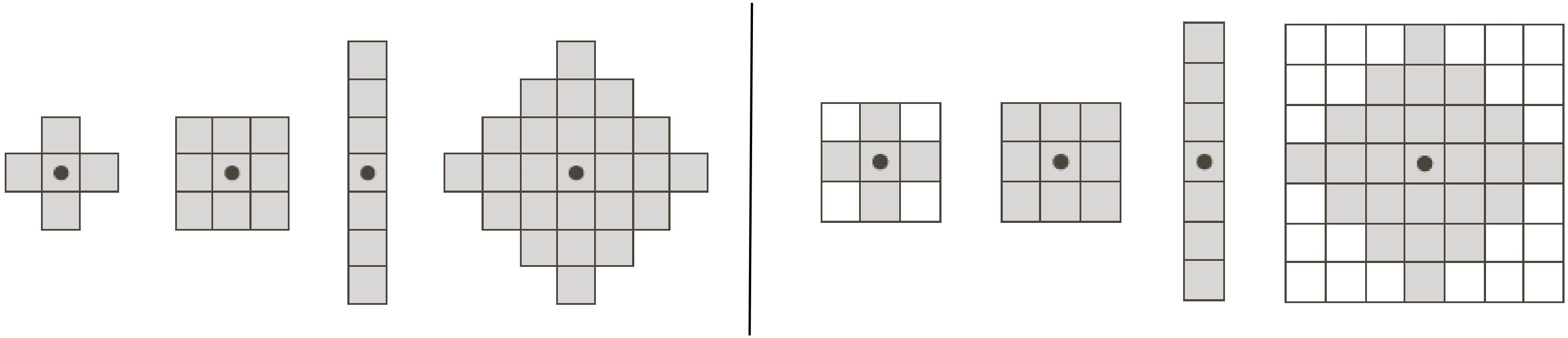


a b

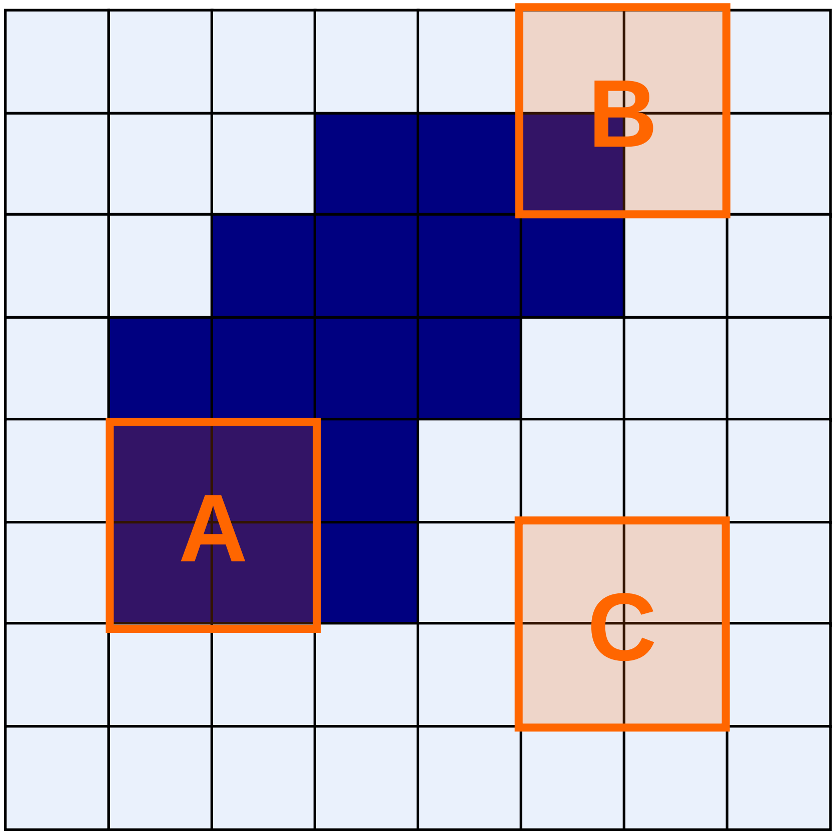
FIGURE 9.2
(a) Translation of A by z.
(b) Reflection of B. The sets A and B are from Fig. 9.1.

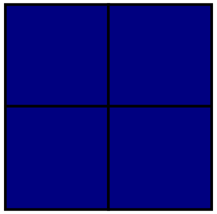
Structuring Elements

- Structuring elements are small sets/sub-images used to probe an image under study
- For each SE, define its origin
- shape and size must be adapted to geometric properties for the objects
  - For simplicity we will use rectangular structuring elements with their origin at the middle pixel



Structuring Elements, Hits & Fits





Structuring Element

Fit: All *on pixels* in the structuring element cover *on pixels* in the image

Hit: Any *on pixel* in the structuring element covers an *on pixel* in the image

Basic morphological processing operations are based on these simple ideas (informally)

Fitting & Hitting

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	**B** 1	1	1	1	0	**C** 0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	**A** 1	1	1	1	0
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0

1	1	1
1	1	1
1	1	1

Structuring Element 1

0	1	0
1	1	1
0	1	0

Structuring Element 2

Fundamental Operations

- Fundamentally morphological image processing is very much like spatial filtering
- The structuring element is moved across every pixel in the original image to give a pixel in a new processed image
- The value of this new pixel depends on the operation performed
- There are two basic morphological operations: **erosion** and **dilation**

Erosion

Erosion of image A by structuring element B represented as sets in Z^2, is defined as:

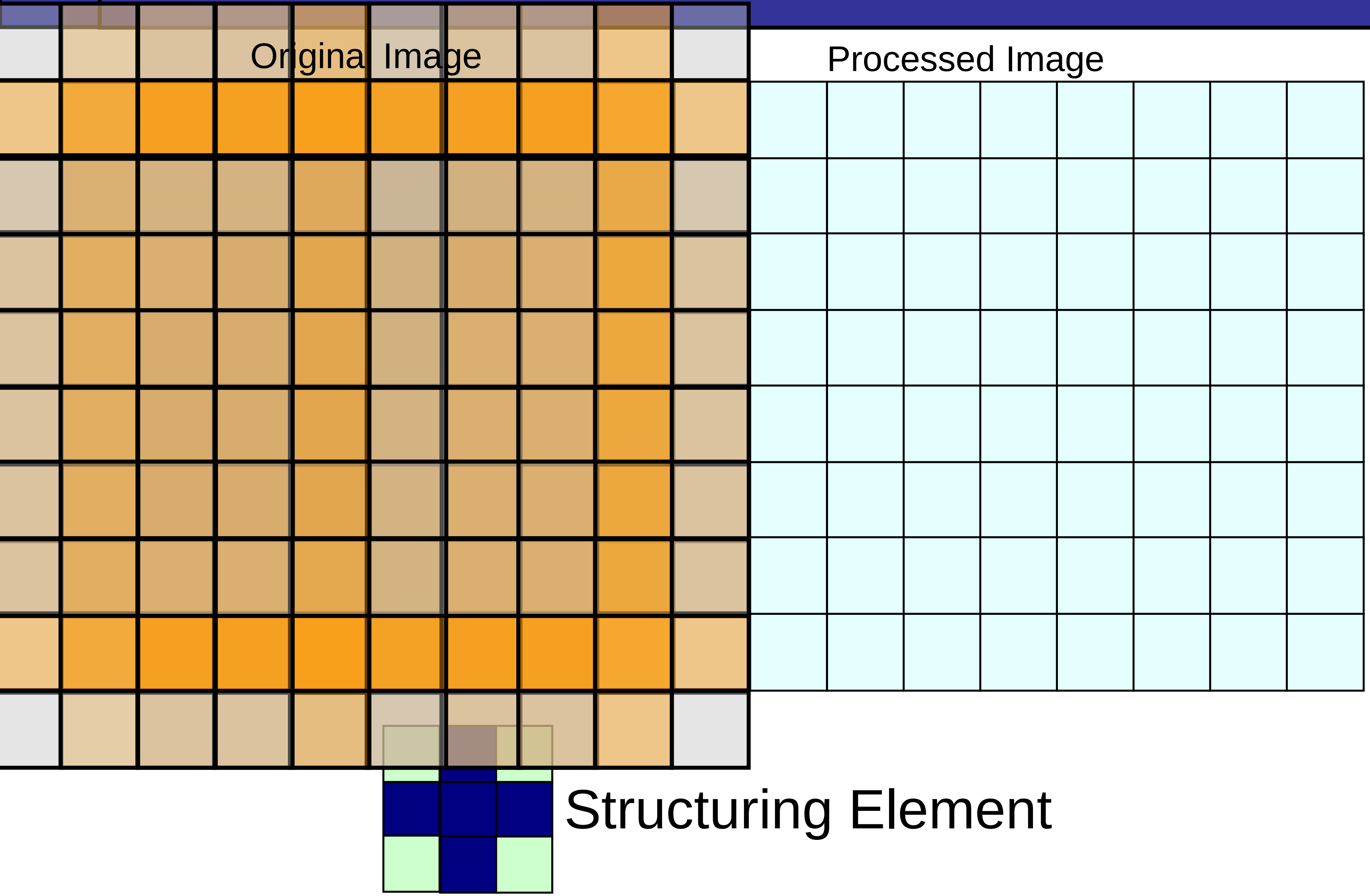
$$A \ominus B = \{ z \mid (B)_Z \subseteq A \}$$

The set of all points z such that B, translated by z, is contained by A.

Informally, the structuring element B is positioned with its origin at (x, y) and the new pixel value is determined using the rule:

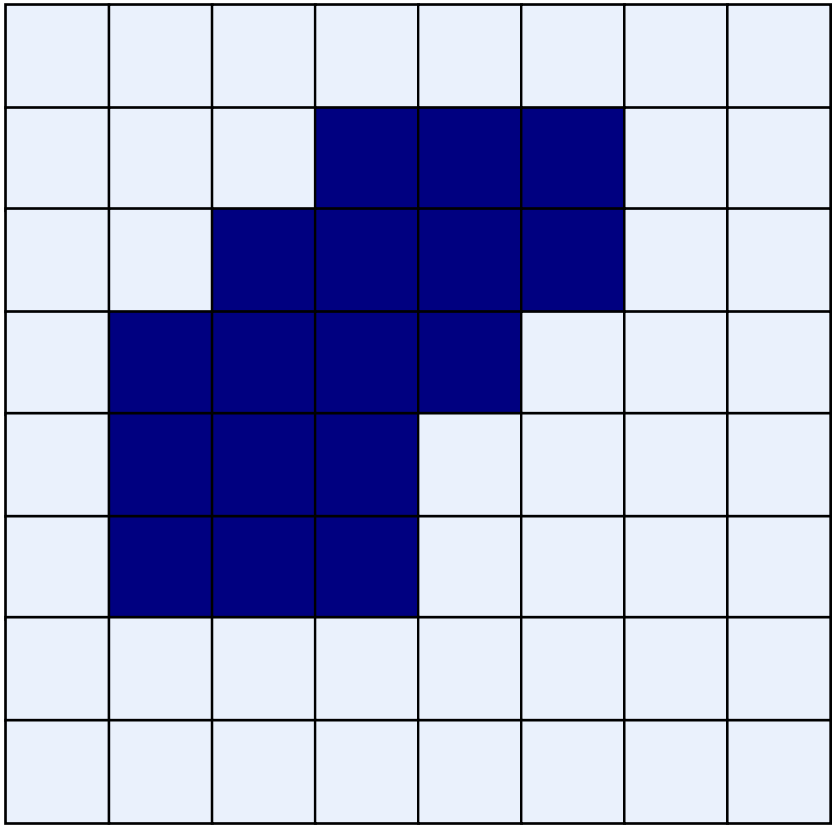
$$g(x, y) = \begin{cases} 1 \text{ if } B \text{ fits } A \\ 0 \text{ otherwise} \end{cases}$$

Erosion Example

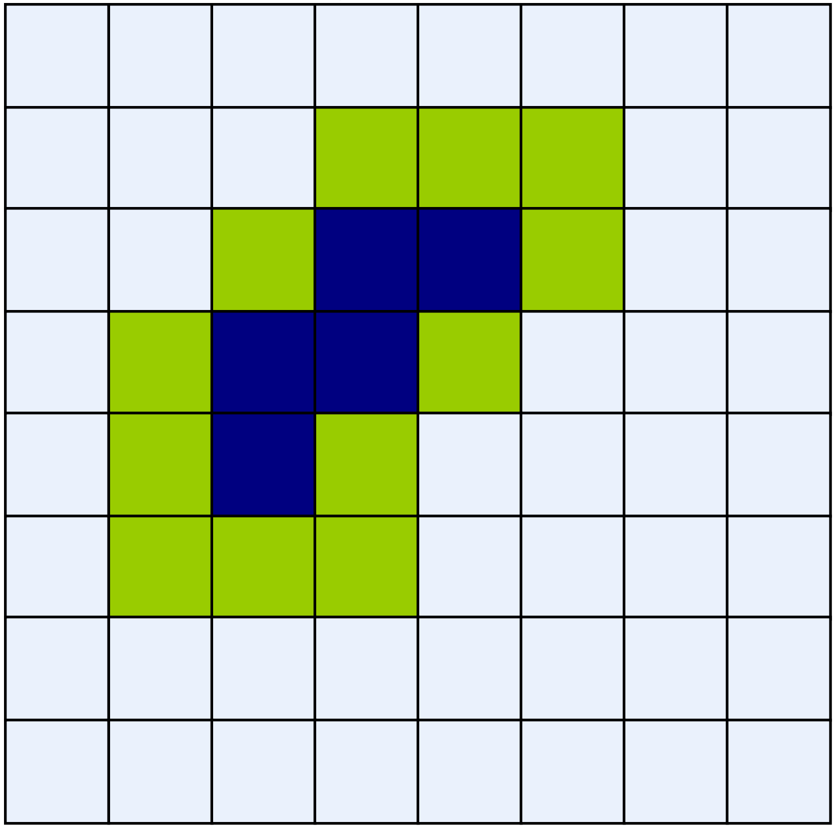


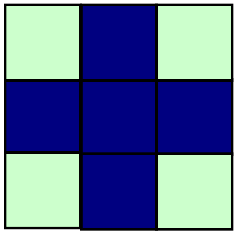
Erosion Example

Original Image



Processed Image With Eroded Pixels

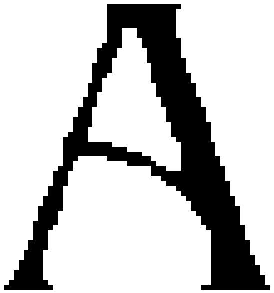




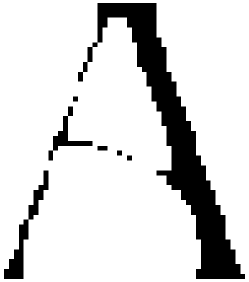
Structuring Element

Erosion Example 1

Original image



Erosion by 3*3 square structuring element

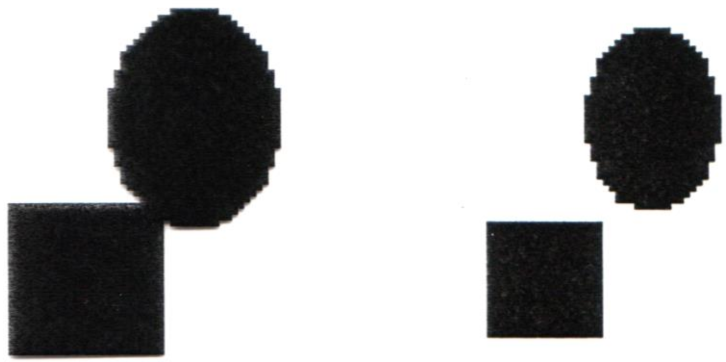


Erosion by 5*5 square structuring element

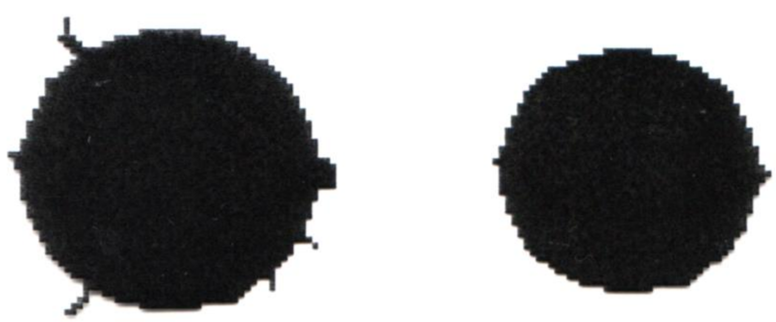
Watch out: In these examples a 1 refers to a black pixel!

What Is Erosion For?

Erosion can split apart joined objects

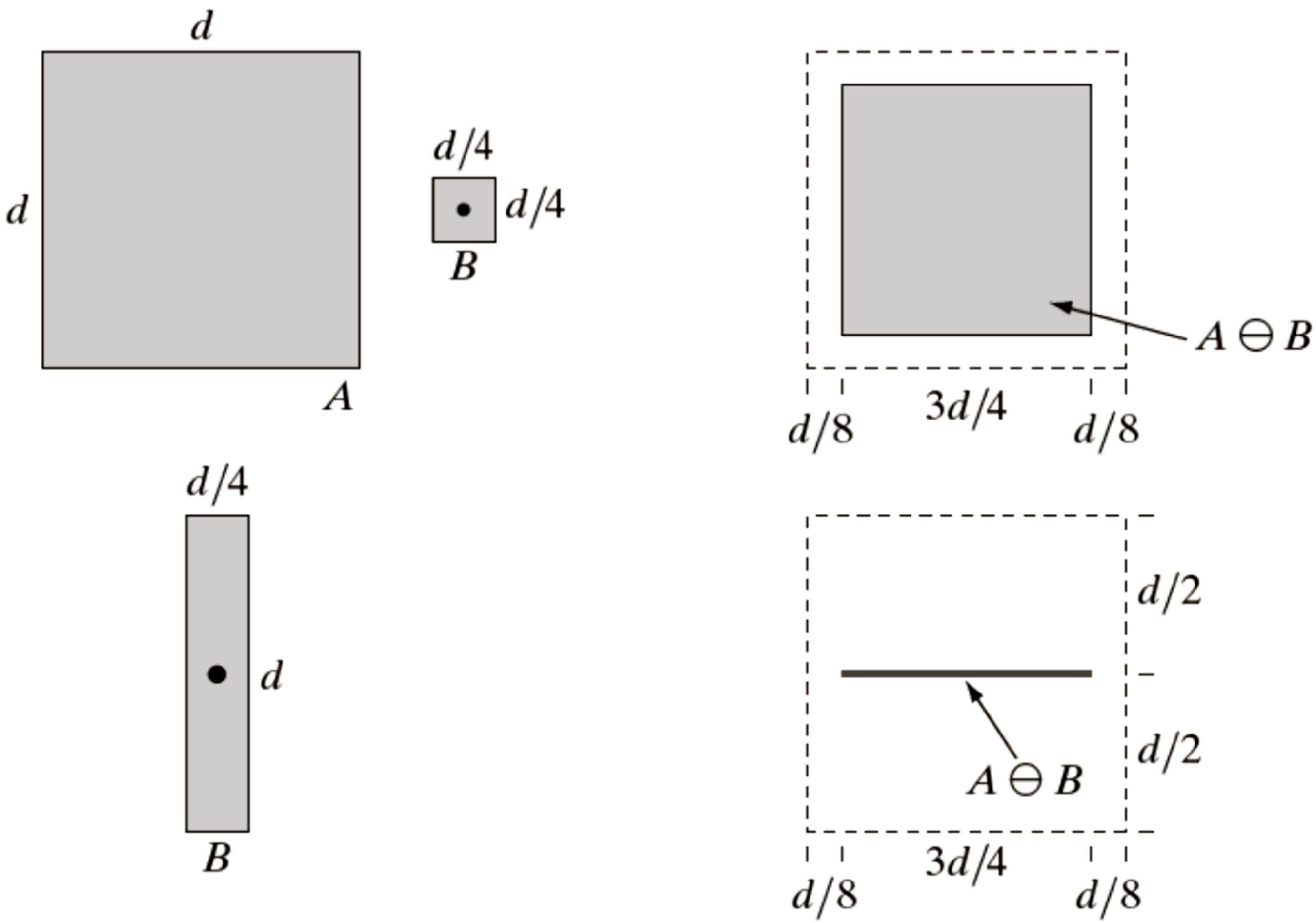


Erosion can strip away extrusions



Watch out: Erosion shrinks objects

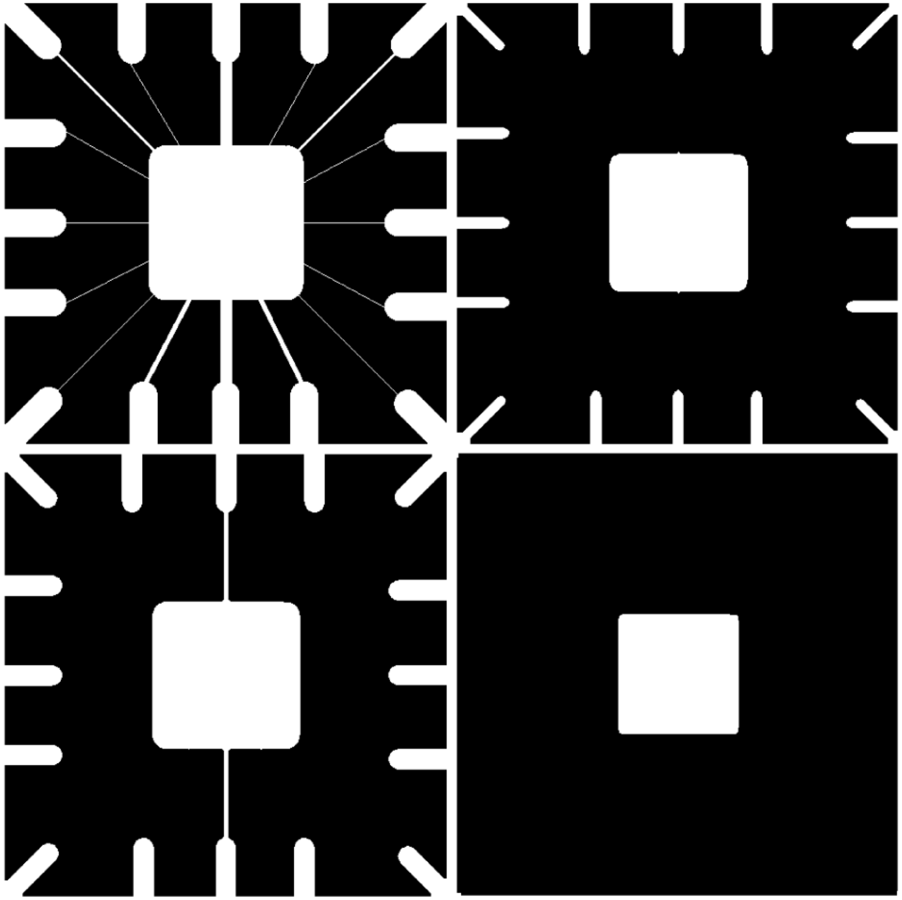
Example 2



Erosion Example 3

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Original image



After erosion with a disc of radius 15

After erosion with a disc of radius 11

After erosion with a disc of radius 45

Dilation

Dilation of image A by structuring element B is given by:

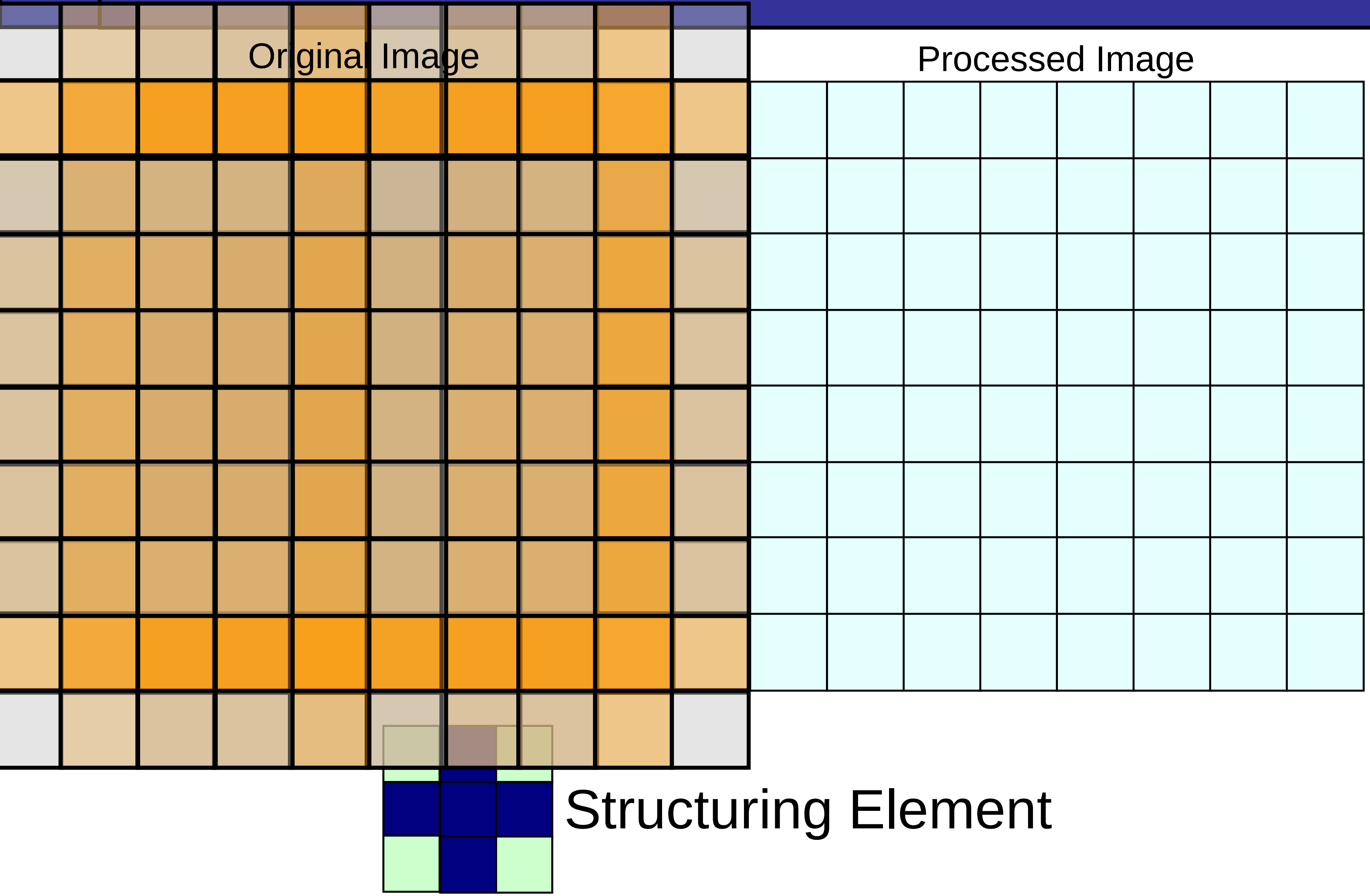
$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

The set of all displacements z, such that $\hat{B}$ and A overlap by at least one element.

Informally: The structuring element B (if symmetric with origin at center, else reflected) is positioned with its origin at (x, y) and the new pixel value is determined using the rule:

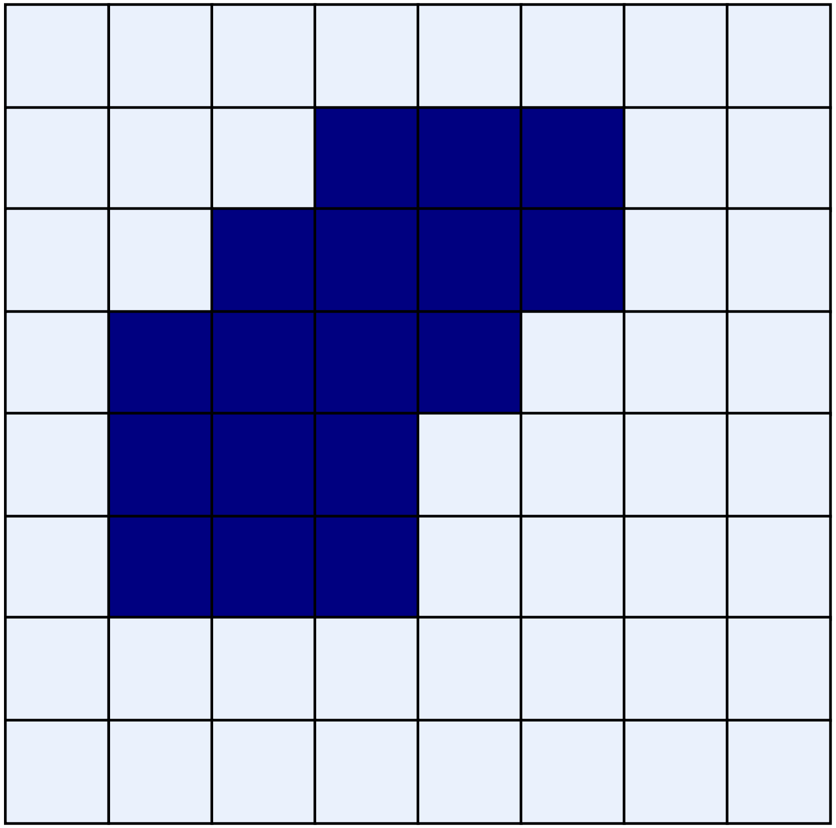
$$g(x,y) = \begin{cases} 1 & \text{if } B \text{ hits } A \\ 0 & \text{otherwise} \end{cases}$$

Dilation Example

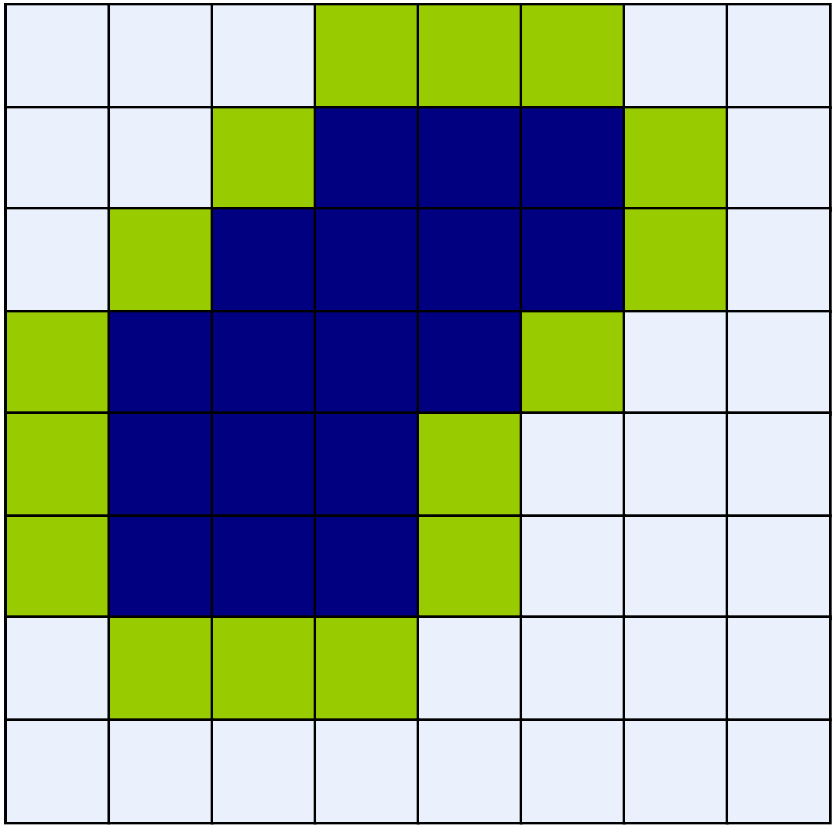


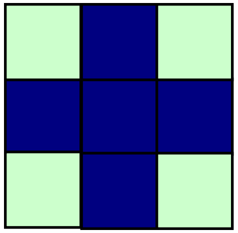
Dilation Example

Original Image



Processed Image With Dilated Pixels





Structuring Element

Dilation Example 1

Original image

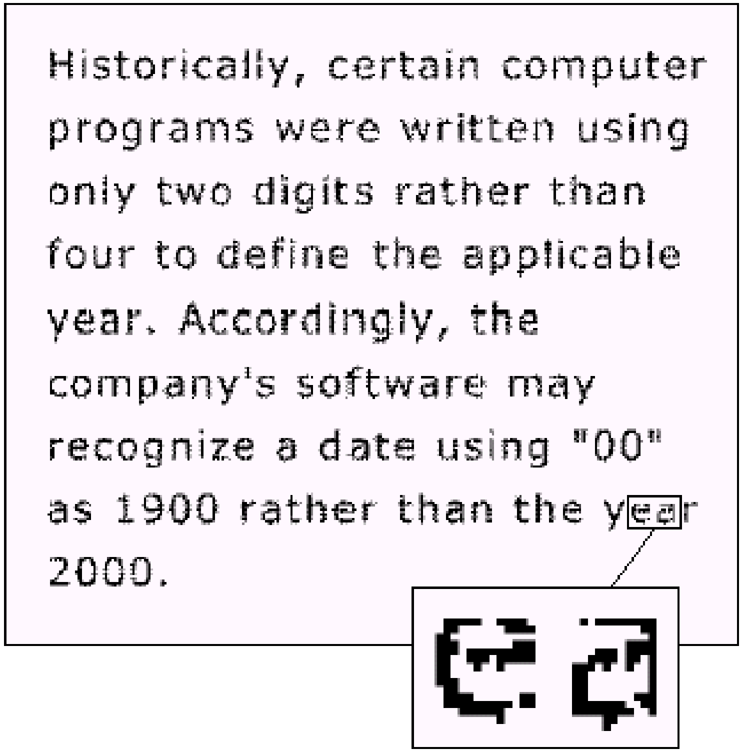
Dilation by 3*3 square structuring element

Dilation by 5*5 square structuring element

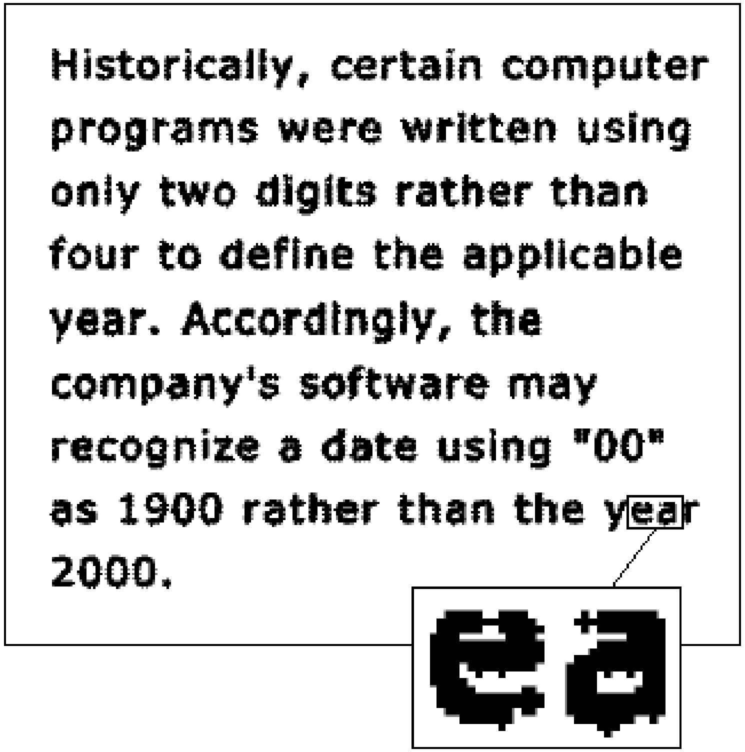
Watch out: In these examples a 1 refers to a black pixel!

Dilation Example 2

Original image



After dilation



0	1	0
1	1	1
0	1	0

Structuring element

Dilation Example 3

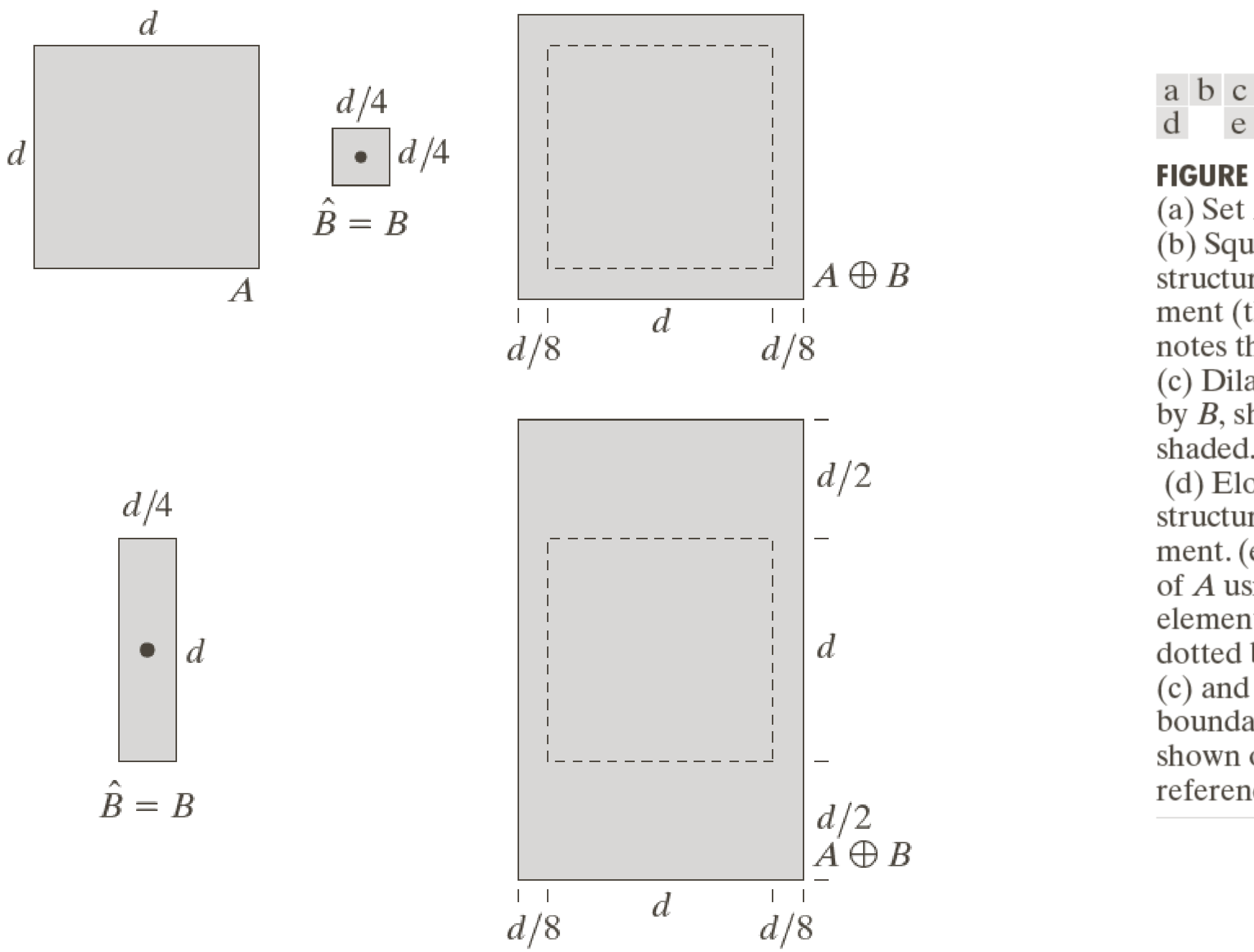
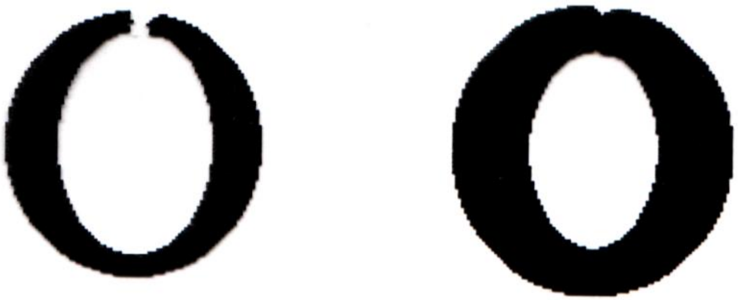


FIGURE 9.6
(a) Set A.
(b) Square structuring element (the dot denotes the origin).
(c) Dilation of A by B, shown shaded.
(d) Elongated structuring element. (e) Dilation of A using this element. The dotted border in (c) and (e) is the boundary of set A, shown only for reference

What Is Dilation For?

Dilation can repair breaks

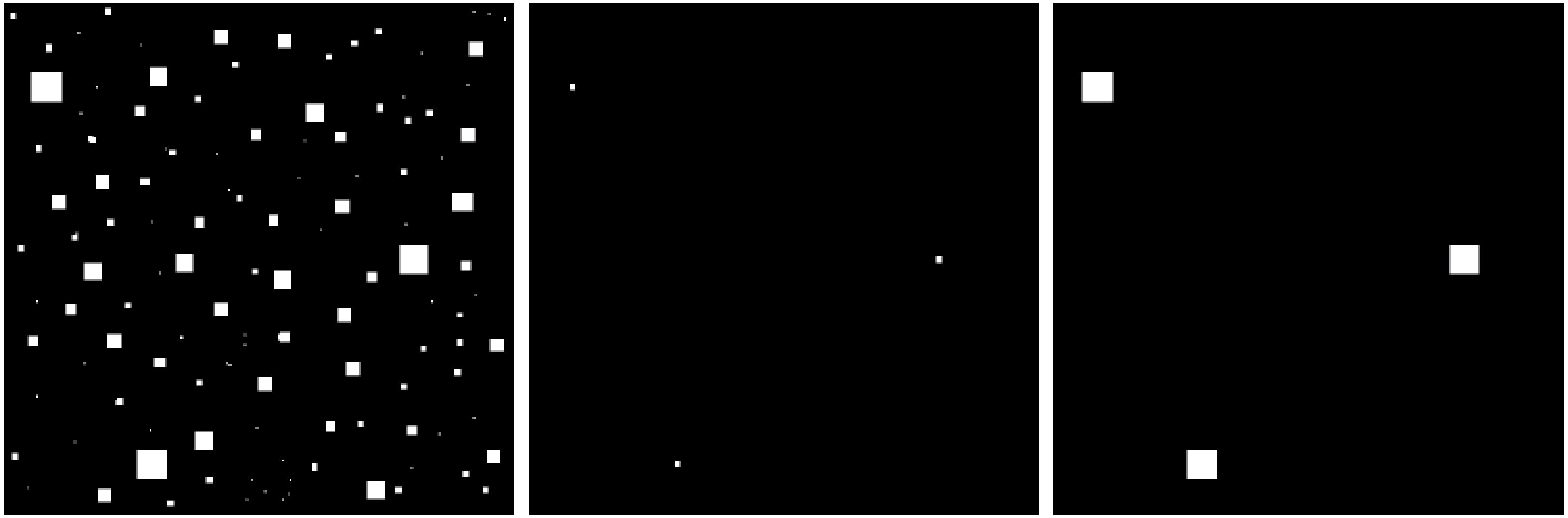


Dilation can repair intrusions

Watch out: Dilation enlarges objects

What Is Dilation For?

Combined with erosion: eliminate irrelevant details



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.

structuring element B = 13x13 pixels of gray level 1

Compound Operations

More interesting morphological operations can be performed by performing combinations of erosions and dilations

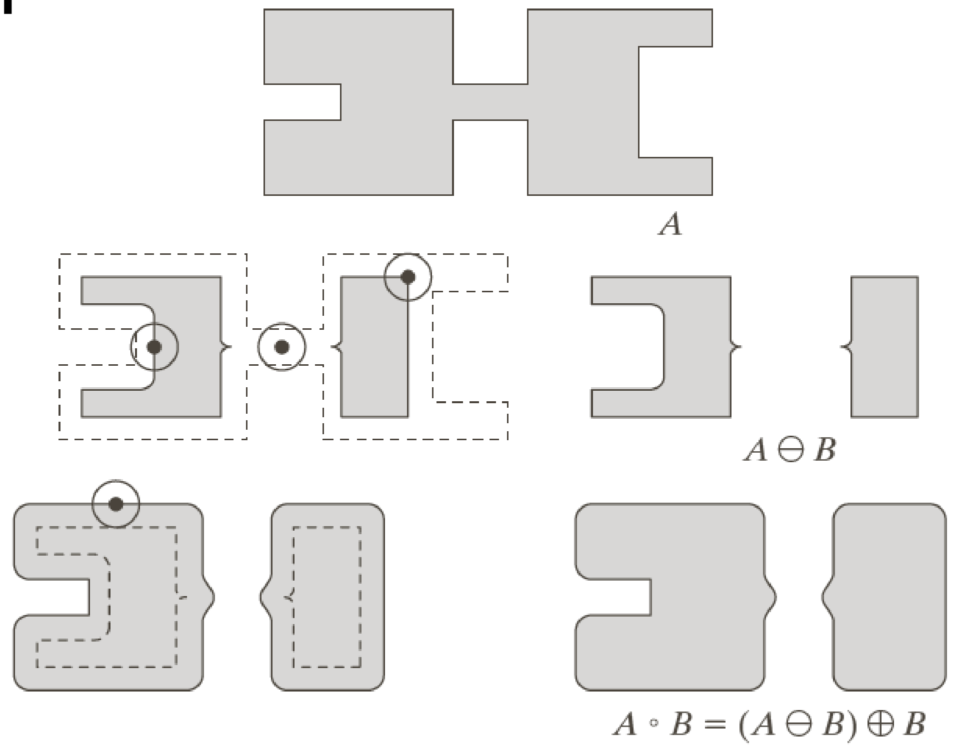
The most widely used of these *compound operations* are:

- Opening
- Closing

Opening

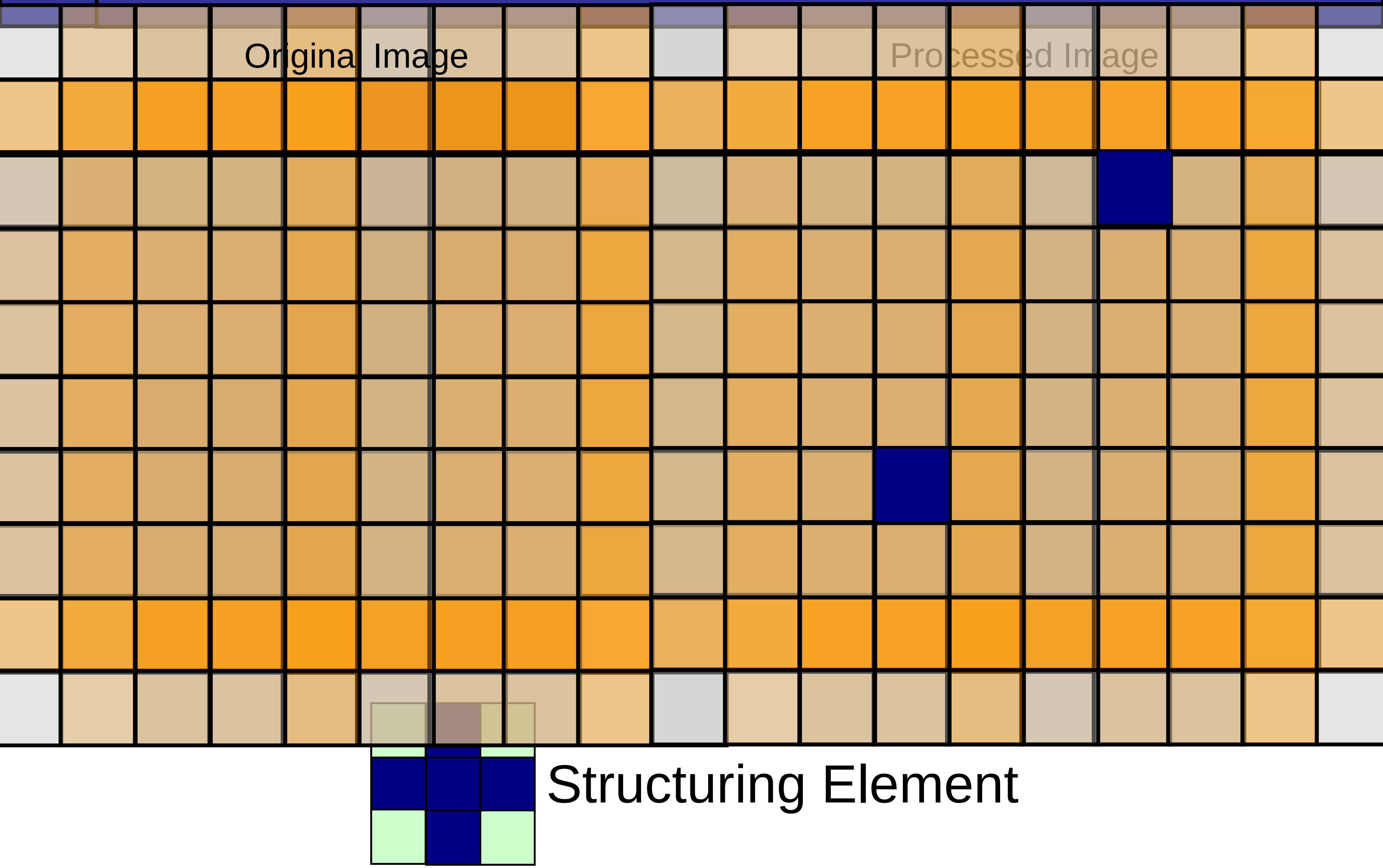
The opening of image f by structuring element s, denoted $f \circ s$ is simply an erosion followed by a dilation

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



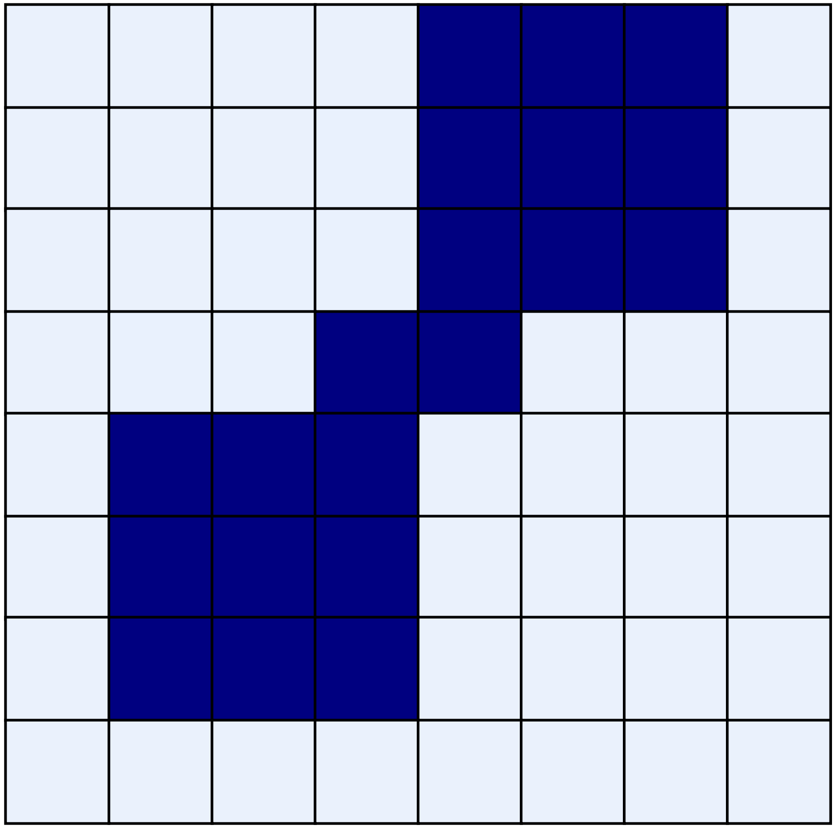
Note a disc shaped structuring element is used

Opening Example

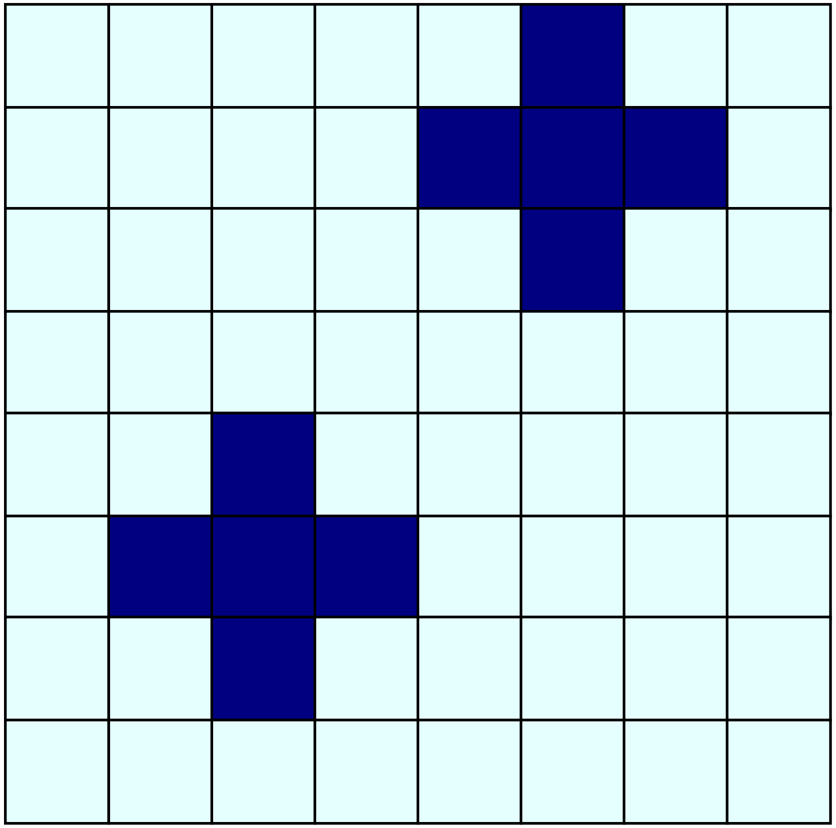


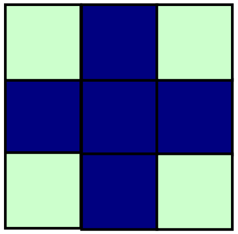
Opening Example

Original Image



Processed Image





Structuring Element

Opening Examples

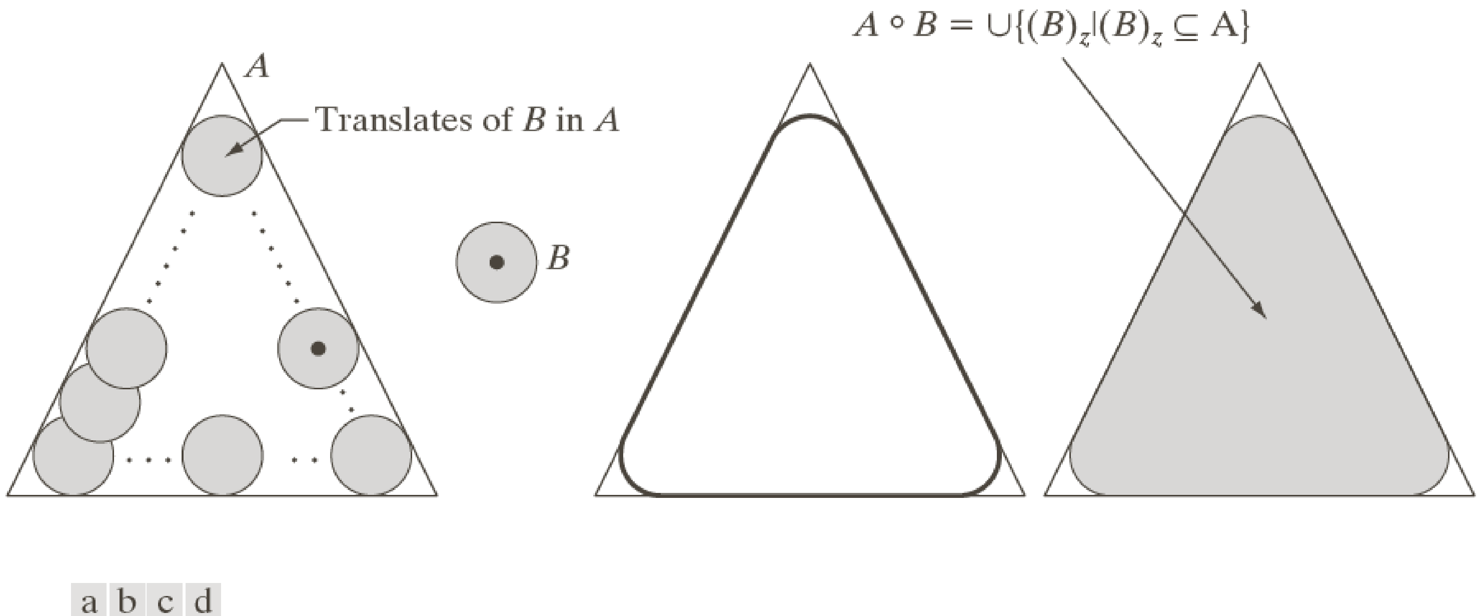
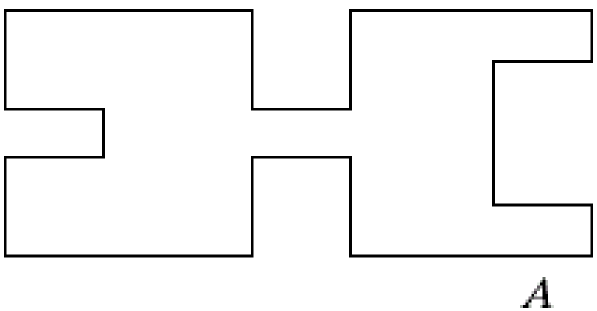


FIGURE 9.8 (a) Structuring element B "rolling" along the inner boundary of A (the dot indicates the origin of B). (b) Structuring element. (c) The heavy line is the outer boundary of the opening. (d) Complete opening (shaded). We did not shade A in (a) for clarity.

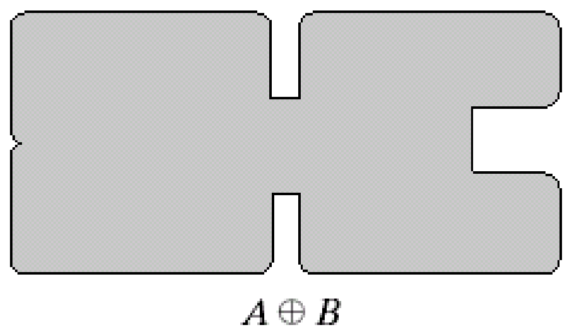
Closing

The closing of image f by structuring element s, denoted $f \bullet s$ is simply a dilation followed by an erosion

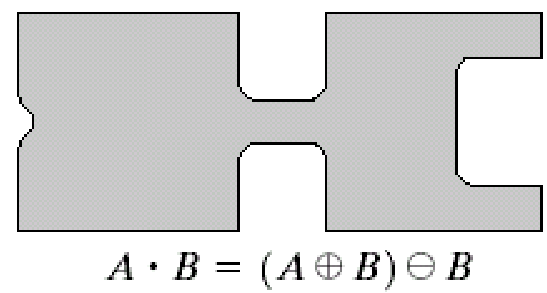
$$f \bullet s = (f \oplus s) \ominus s$$



Original shape



After dilation

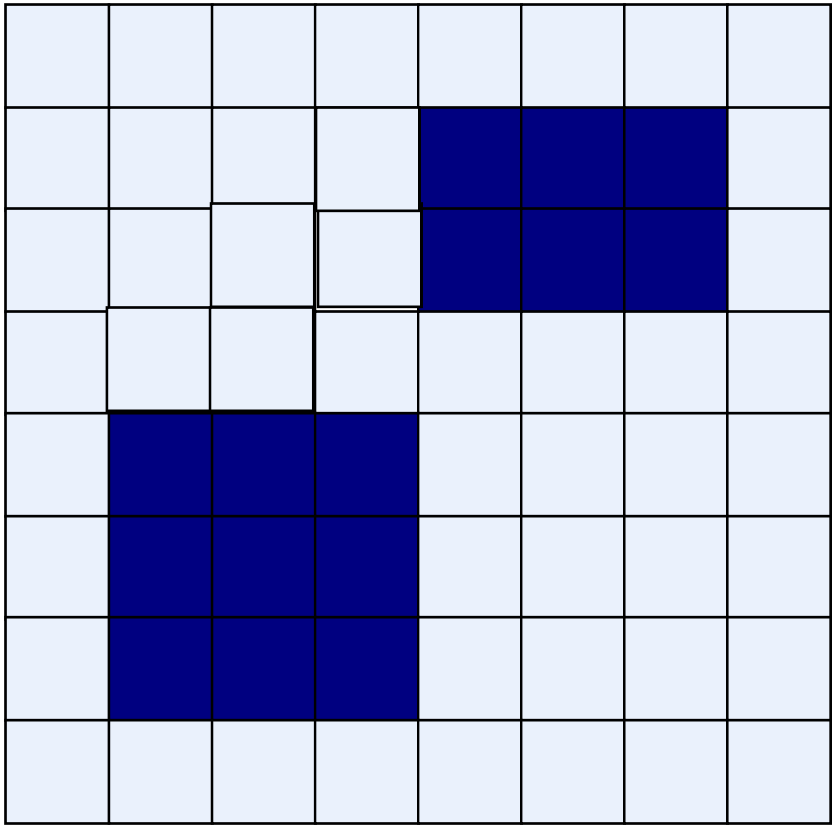


After erosion (closing)

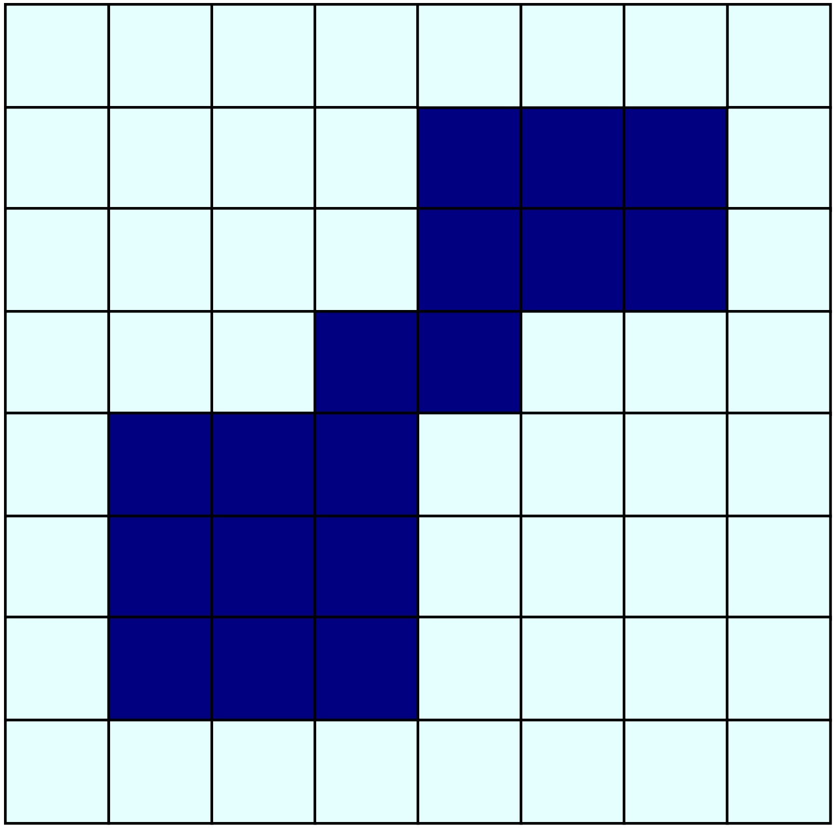
Note a disc shaped structuring element is used

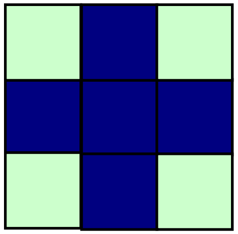
Closing Example

Original Image



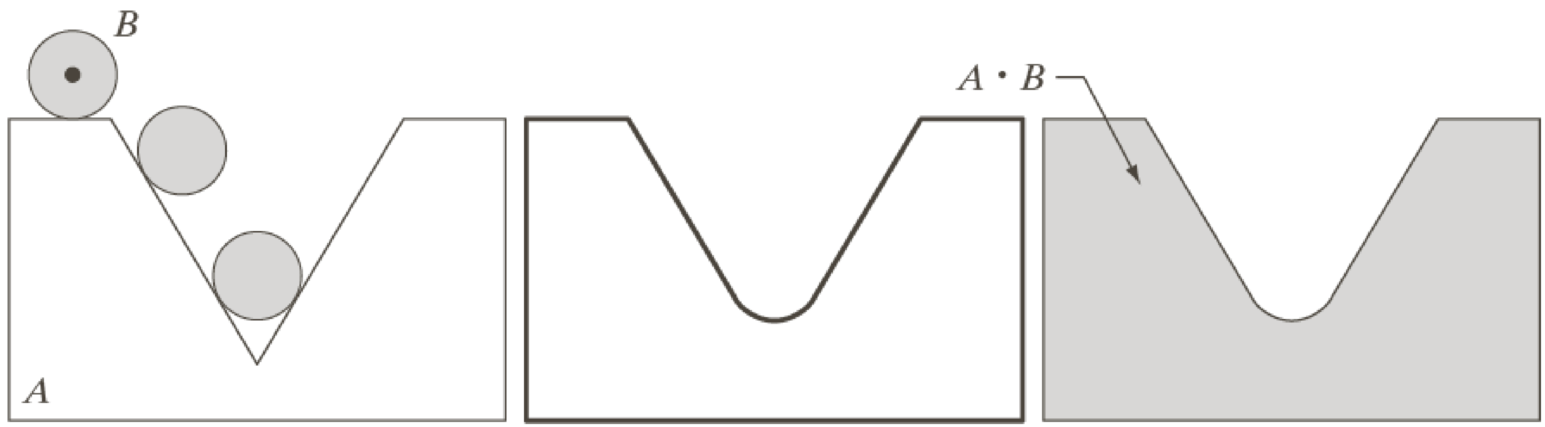
Processed Image





Structuring Element

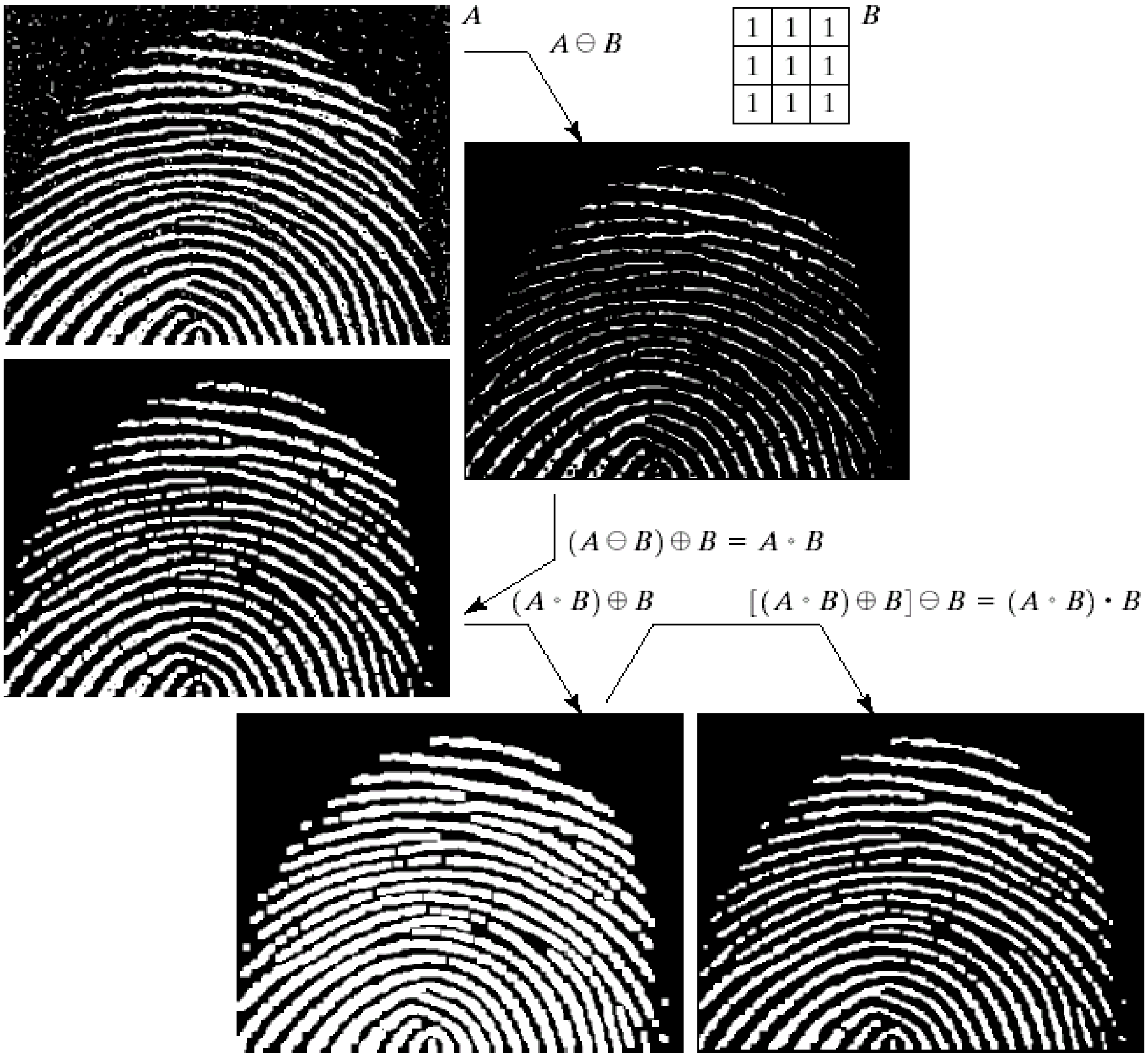
Closing Example



a b c

FIGURE 9.9 (a) Structuring element B "rolling" on the outer boundary of set A. (b) The heavy line is the outer boundary of the closing. (c) Complete closing (shaded). We did not shade A in (a) for clarity.

Morphological Processing Example



Morphological Algorithms

Using the simple technique we have looked at so far we can begin to consider some more interesting morphological algorithms

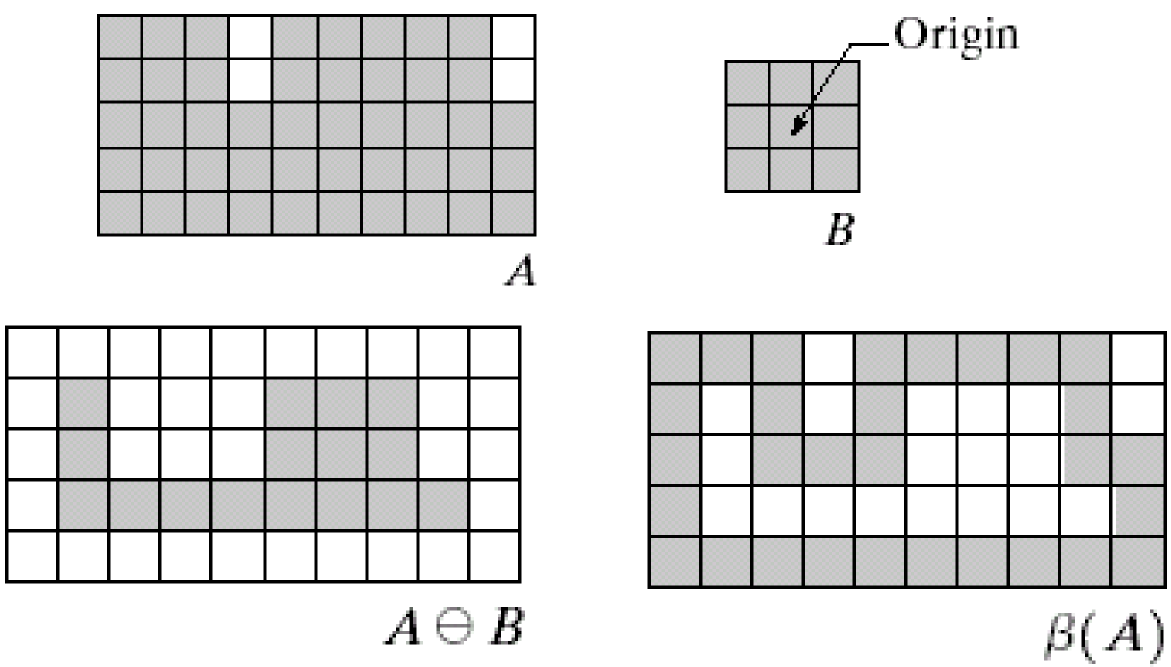
Such as:

- Boundary extraction
- Region filling
- Extraction of connected components
- Hit or Miss Transformation
- Convex Hull
- Thinning/thickening

Boundary Extraction

- Extracting the boundary (or outline) of an object is often extremely useful
- The boundary can be given simply as

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



Boundary Extraction Example

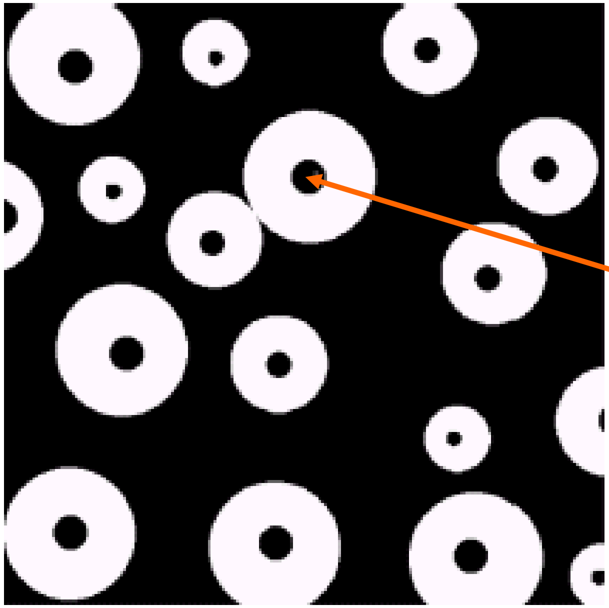
A simple image and the result of performing boundary extraction using a square 3*3 structuring element

Original Image

Extracted Boundary

Region Filling

Given a pixel inside a boundary, *region filling* attempts to fill that boundary with object pixels (1s)



Given a point inside here, can we fill the whole circle?

Region Filling (cont…)

The key equation for region filling is

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

Where X_0 is simply the starting point inside the boundary, B is a simple structuring element and A^c is the complement of A

This equation is applied repeatedly until X_k is equal to X_{k-1}

Finally the result is unioned with the original boundary

Region Filling Step By Step

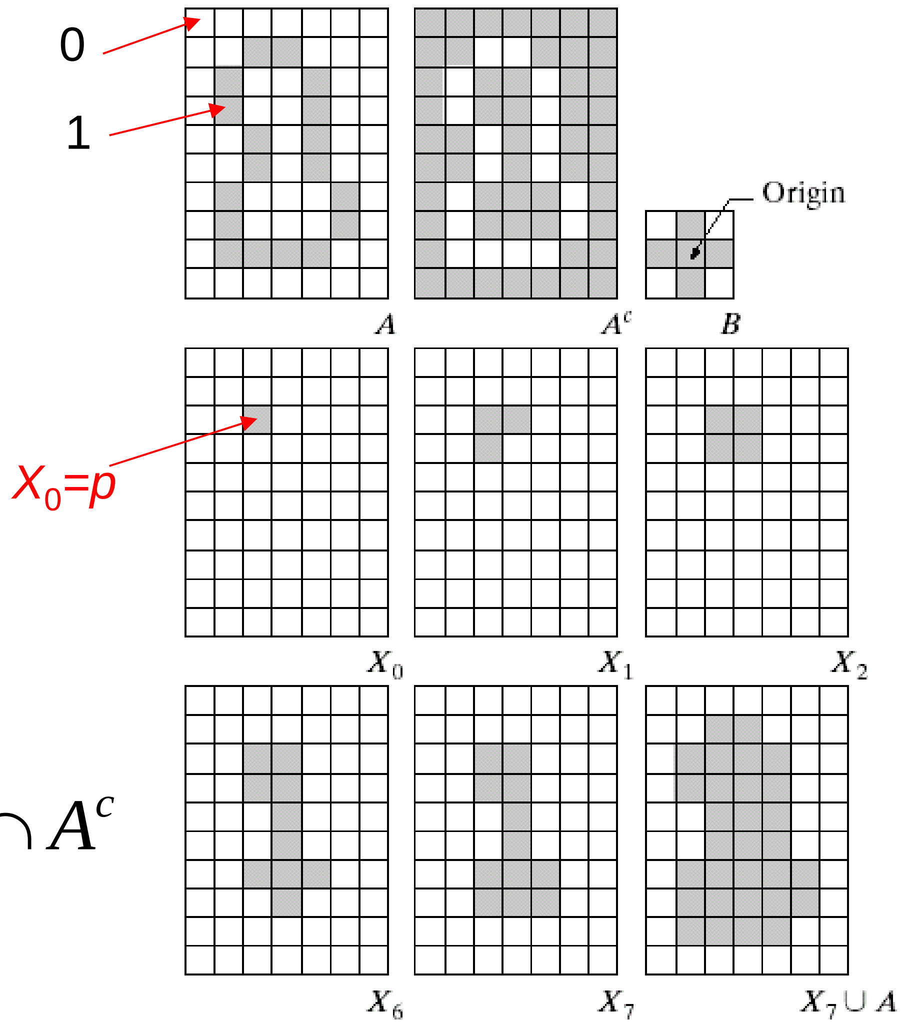
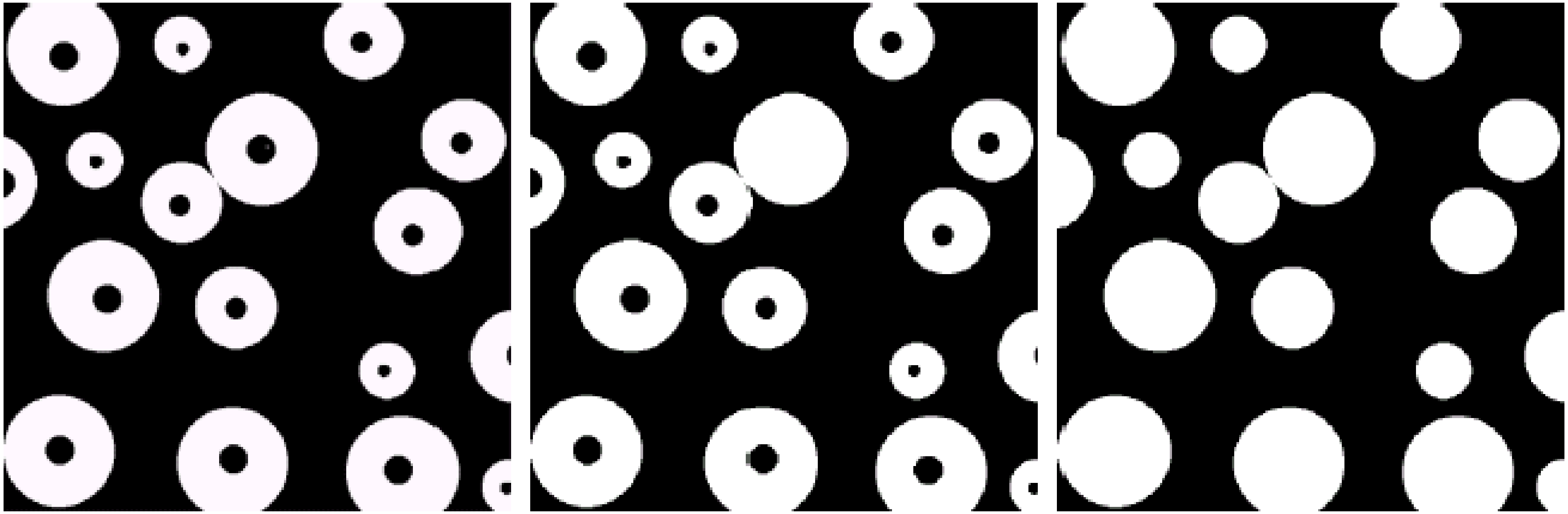


FIGURE 9.15 Region filling. (a) Set A. (b) Complement of A. (c) Structuring element B. (d) Initial point inside the boundary. (e)–(h) Various steps of Eq. (9.5-2). (i) Final result [union of (a) and (h)].

$$X_k = (X_{k-1} \oplus B) \cap A^c$$

Region Filling Example

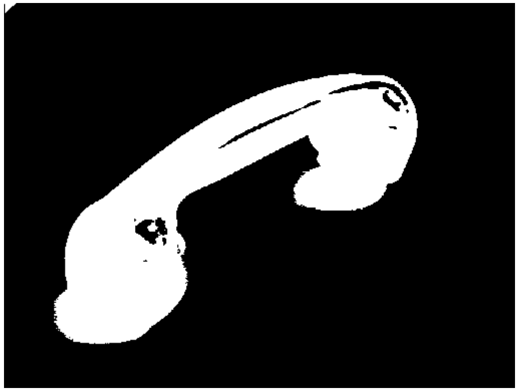
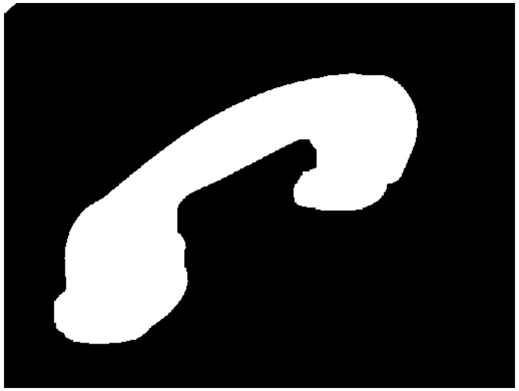


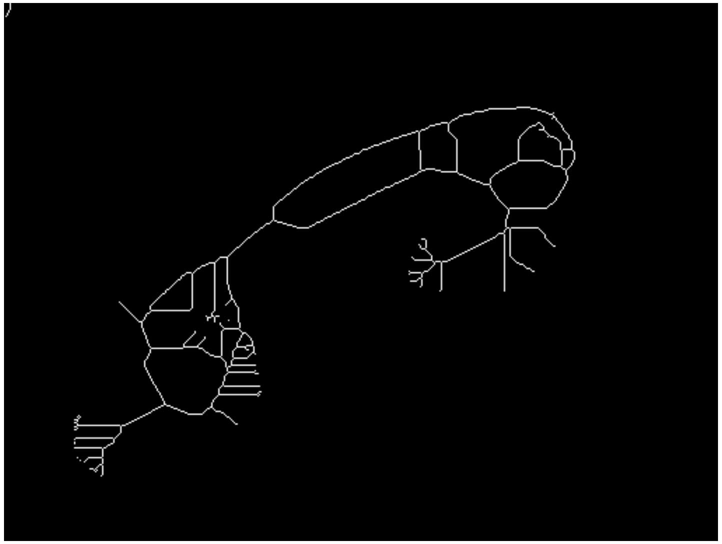
Original Image

One Region Filled

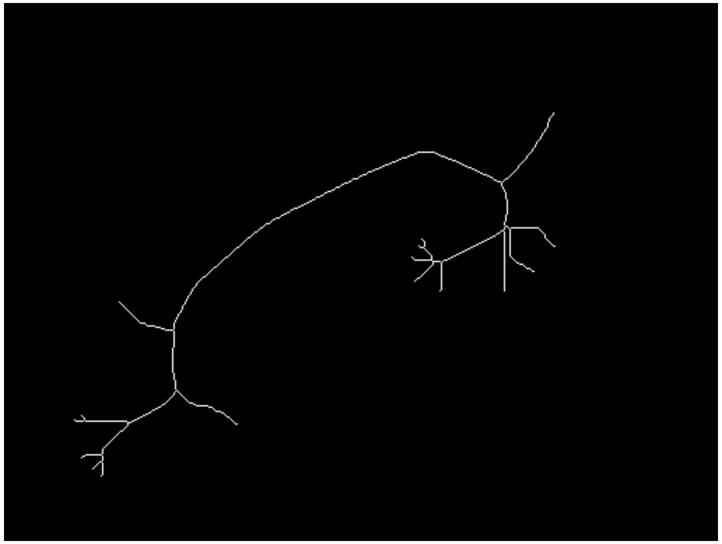
All Regions Filled

Utility for further processing



skeleton before closing



skeleton after closing

Extraction of connected components

$$X_k = (X_{k-1} \oplus B) \cap A$$

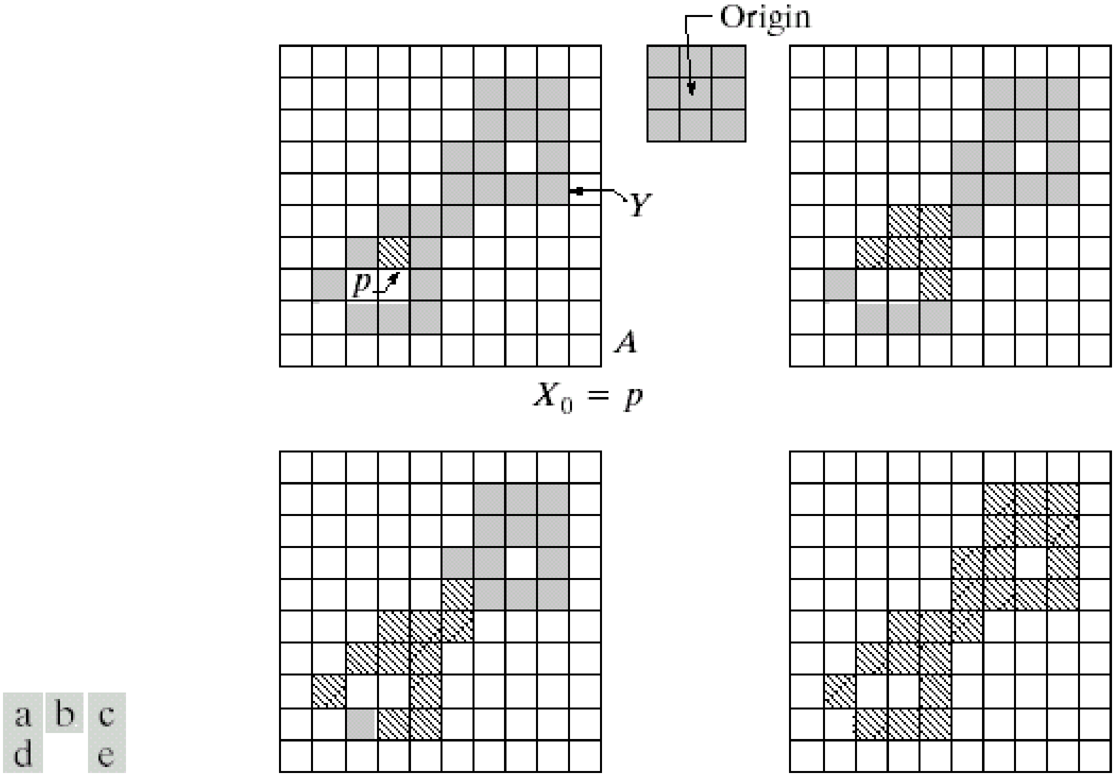


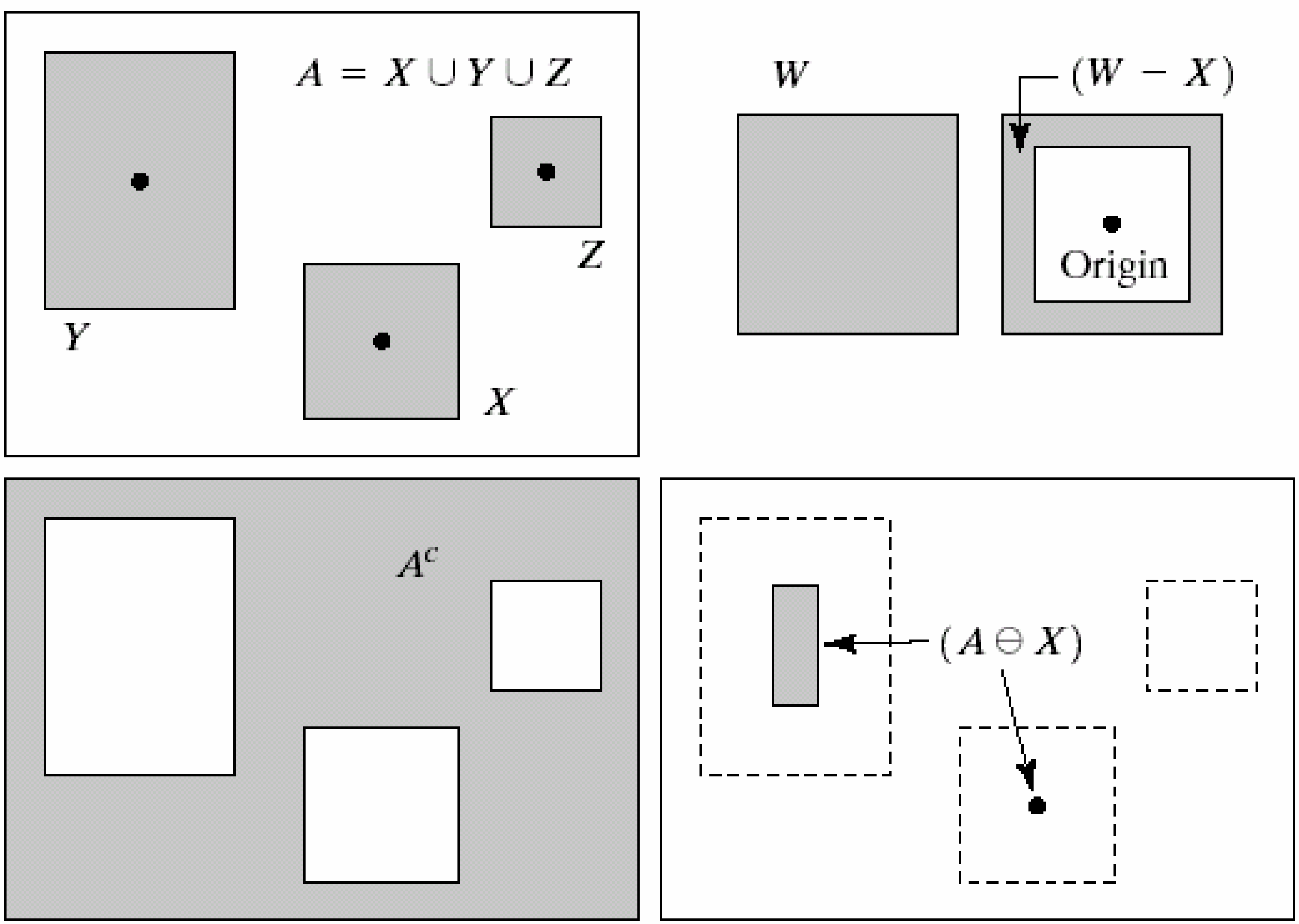
FIGURE 9.17 (a) Set A showing initial point p (all shaded points are valued 1, but are shown different from p to indicate that they have not yet been found by the algorithm). (b) Structuring element. (c) Result of first iterative step. (d) Result of second step. (e) Final result.

The Hit-or-Miss Transformation

- Used to look for particular patterns of foreground and background pixels
- Very basic tool for shape detection
- Input:
  - Binary Image
  - Group of Structuring Elements, containing 0s, 1s and don't cares(!)

The Hit-or-Miss Transformation

$$A \circledast B = (A \ominus X) \cap \left[A^c \ominus (W - X)\right] \quad B = (X, W - X)$$



a b
c d
e
f

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.

The Hit-or-Miss Transformation

$$A \circledast B = (A \ominus X) \cap \left[A^c \ominus (W - X)\right] \quad B = (X, W - X)$$

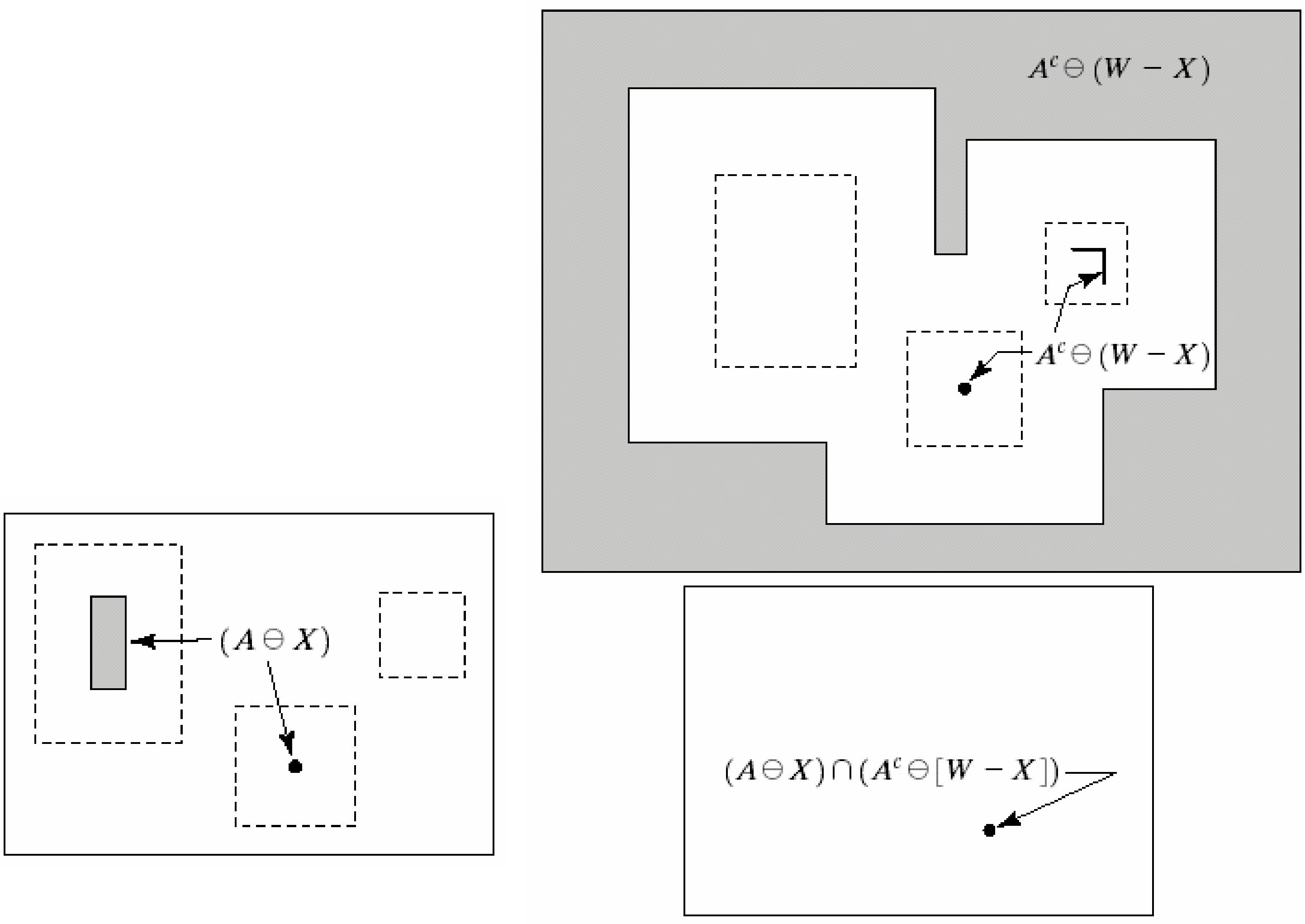
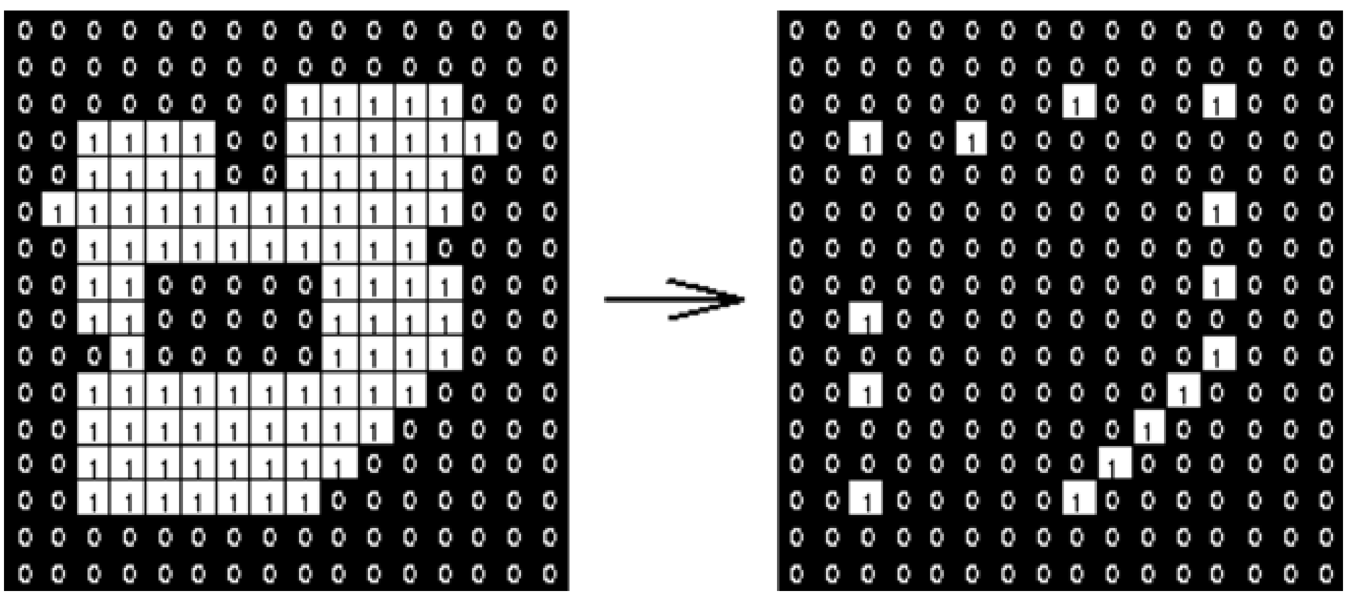


FIGURE 9.12 (a) Set A. (b) A window, W, and the local background of X with respect to W, $(W - X)$. (c) Complement of A. (d) Erosion of A by X. (e) Erosion of A^c by $(W - X)$. (f) Intersection of (d) and (e), showing the location of the origin of X, as desired.

Corner Detection

- To find all the right angle convex corners of a region in a given image as shown below?



Corner Detection

- Structuring Elements representing four corners
- Contains 0s, 1s and *don't care*'s

x	1	x
0	1	1
0	0	x

x	1	x
1	1	0
x	0	0

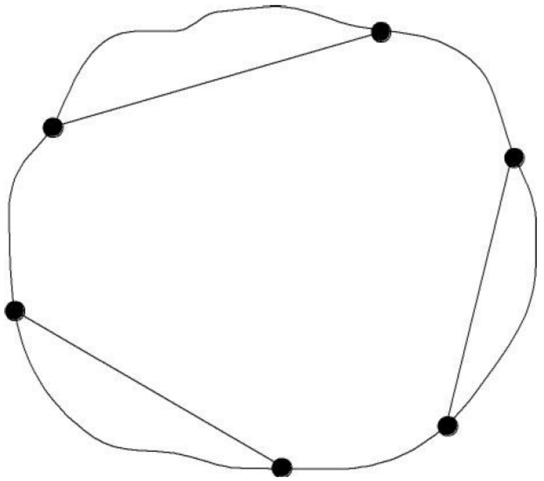
x	0	0
1	1	0
x	1	x

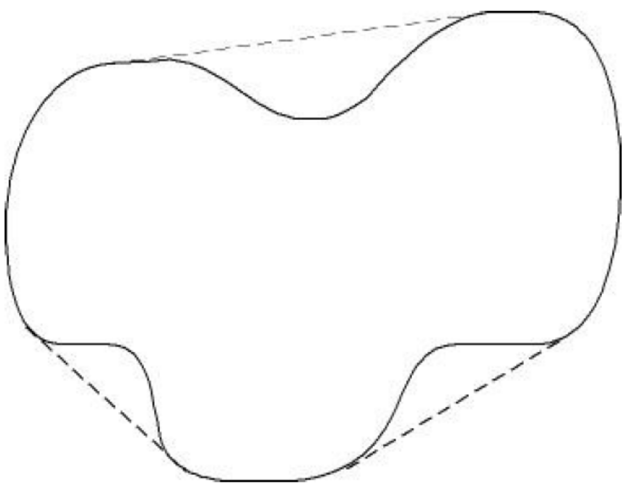
0	0	x
0	1	1
x	1	x

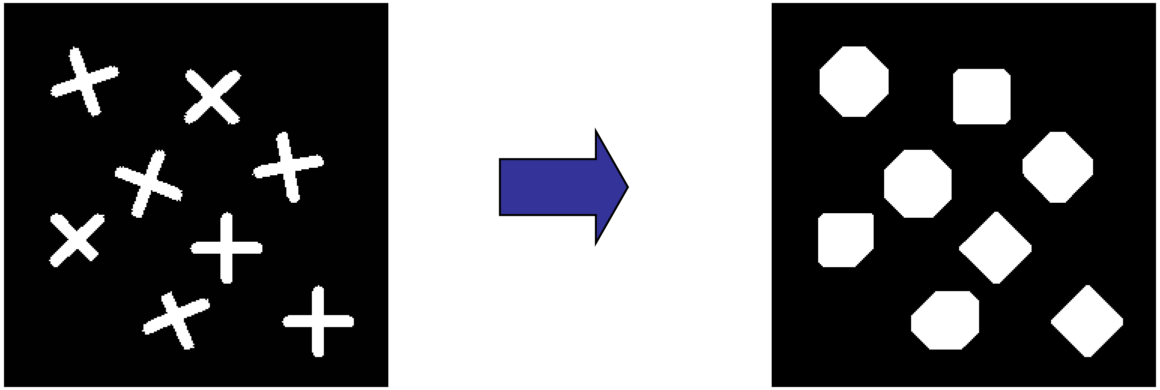
- Apply each Structuring Element
- Use OR operation to combine the four results

Convex Hull

- 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.







Convex Hull

Let B^i, $i = 1,\ 2,\ 3,\ 4$, represent the four structuring elements.

The procedure consists of implementing the equation:

$$X_k^i = (X_{k-1} \circledast B^i) \cup A$$

$$i = 1, 2, 3, 4 \ \text{ and } \ k = 1, 2, 3, \ldots$$

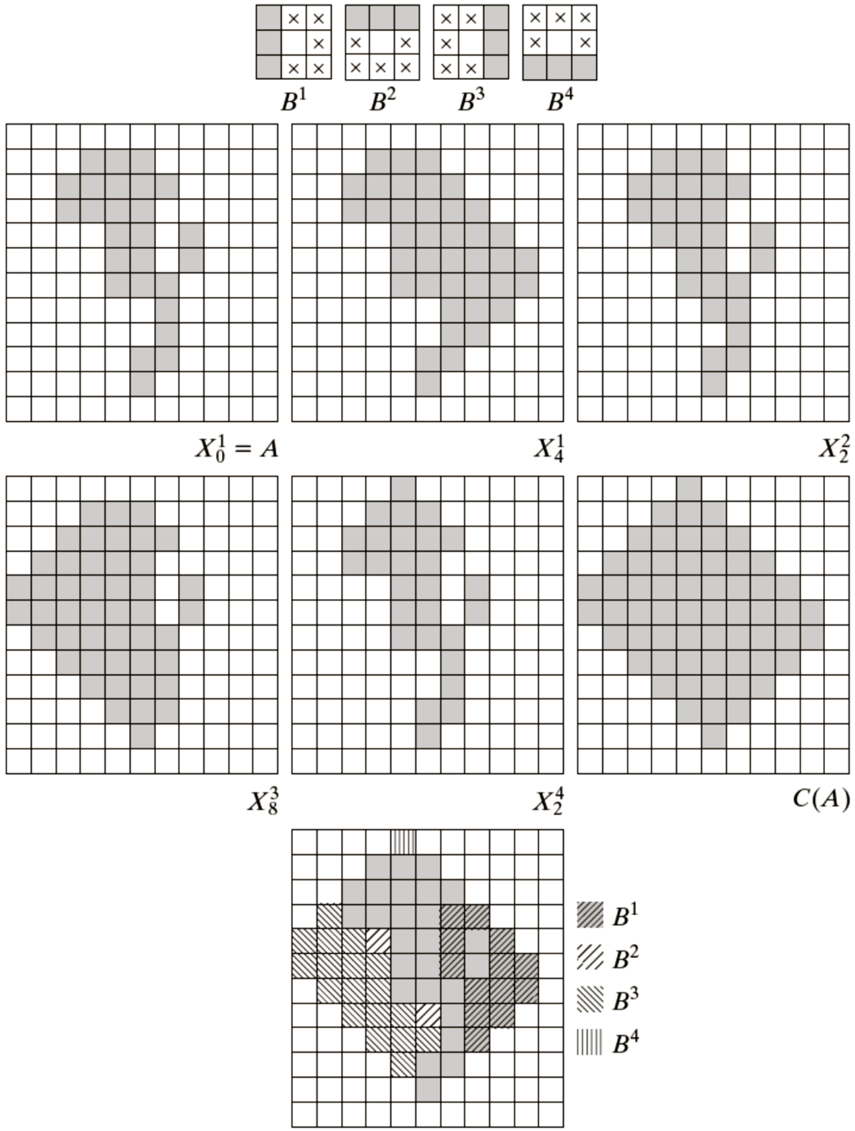
with $X_0^i = A$.

When the procedure converges, or $X_k^i = X_{k-1}^i$, let $D^i = X_k^i$,

the convex hull of A is

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

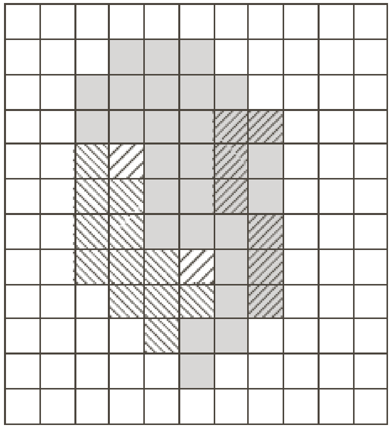
Convex Hull



a
b c d
e f g
h

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.

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



Thining

1. Used to remove selected foreground pixels from binary images
2. After edge detection, lines are often thicker than one pixel.
3. Thinning can be used to thin those line to one pixel width.
4. Several applications, but is particularly useful for skeletonization

Thining

- The thinning of a set A by a structuring element B, defined: $A \otimes B = A - (A \circledast B)$

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

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

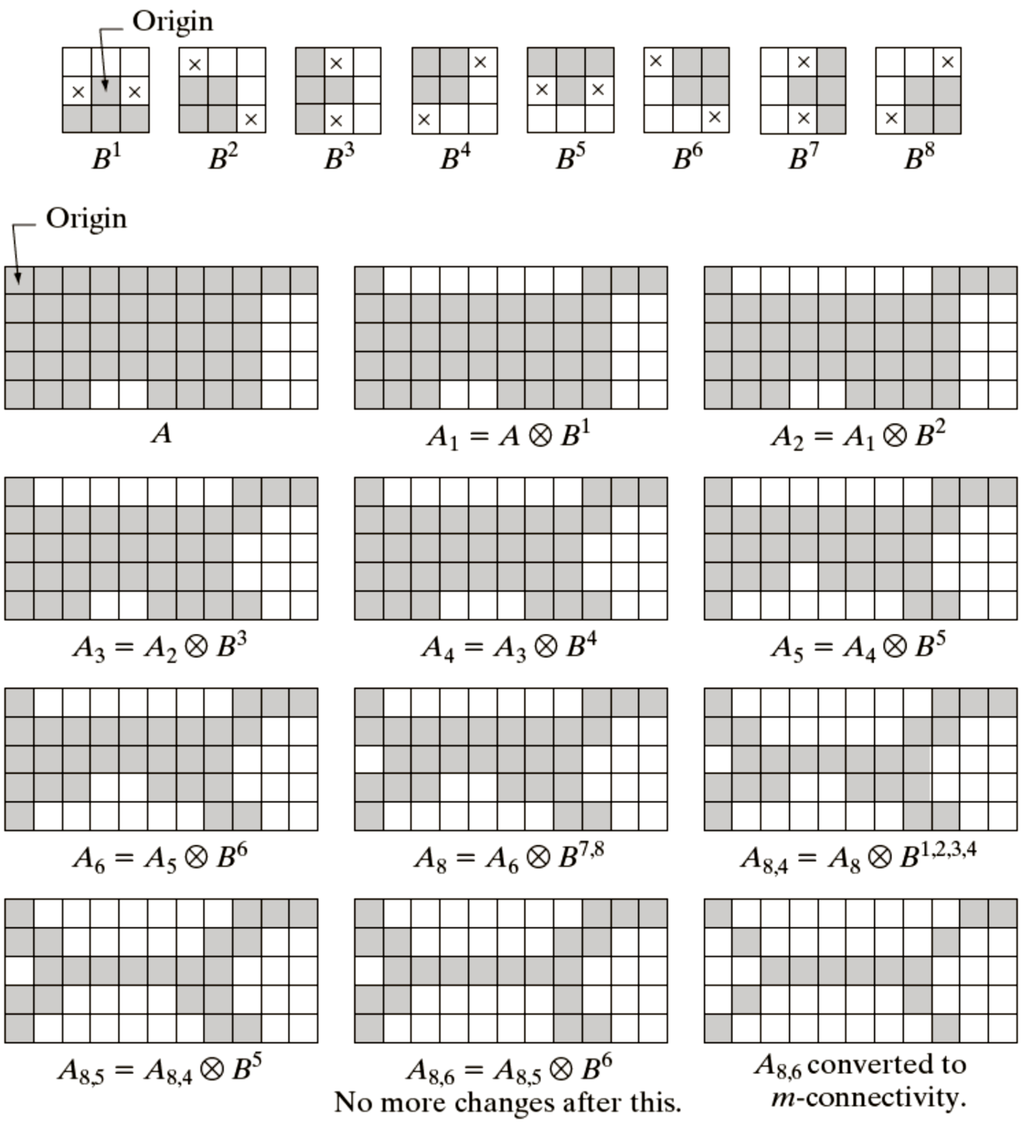
$$\{B\} = \left\{B^1, B^2, B^3, \ldots, B^n\right\}$$

where B^i is a rotated version of $B^{i\text{-}1}$

The thinning of A by a sequence of structuring element $\{B\}$

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

Thining



	a	
b	c	d
e	f	g
h	i	j
k	l	m

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 four elements again. (l) Result after convergence. (m) Conversion to m-connectivity.

Summary

The purpose of morphological processing is primarily to remove imperfections added during segmentation

The basic operations are *erosion* and *dilation*

Using the basic operations we can perform *opening* and *closing*

More advanced morphological operation can then be implemented using combinations of all of these