

Morphological Image Processing

Dr. Mongkol Ekpanyapong

What is mathematical morphology?

- **Basic principle:** the extraction of geometrical and topological information from an unknown set (an image) through transformations using another, well-defined, set, known as *structuring element* (SE).
- In morphological image processing, the design of SEs, their shape and size, is crucial to the success of the morphological operations that use them.

Preliminaries

- The language of mathematical morphology is set theory

- Set terminology:

Let A be a set of \mathbb{Z}^2 . if $w(x,y)$ is an element of A , we write: $w \in A$

If w is not an element of A , we write: $w \notin A$

A decorative graphic in the top-left corner consisting of a blue square above a grid of smaller squares in various colors.

Fundamental concepts and operations

- Basic set operations:

- Complement $A^c = \{z | z \notin A\}$

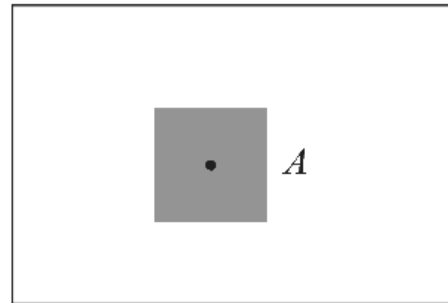
- Difference $A - B = \{z | z \in A, z \notin B\} = A \cap B^c$

- Translation $A_w = \{c | c = a + w, \text{ for } a \in A\}$

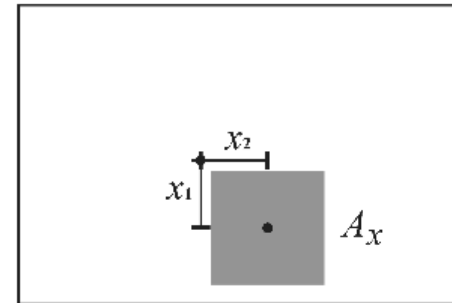
- Reflection $\hat{A} = \{z | z = -a \text{ for } a \in A\}$



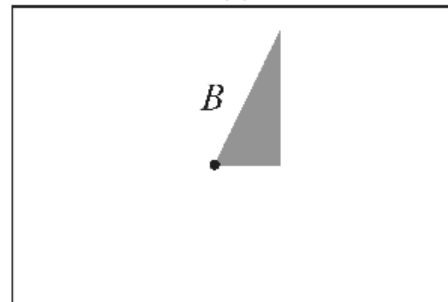
Example



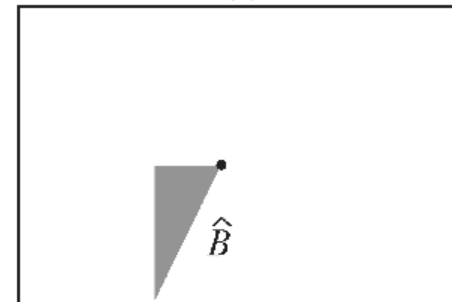
(a)



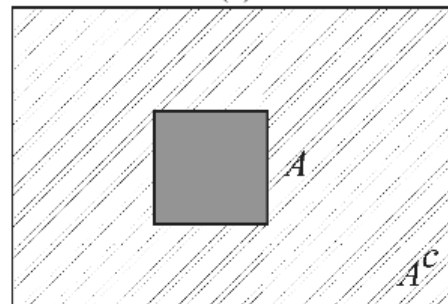
(b)



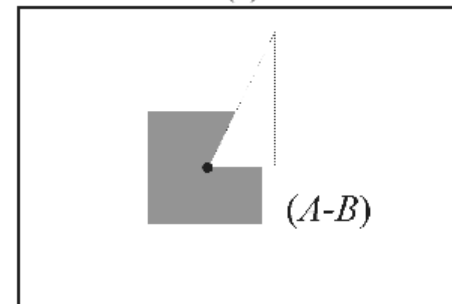
(c)



(d)



(e)



(f)



Fundamental concepts and operations

- Logical equivalents of set theory operations

- Intersection ~ logical AND

$$C = \{(x, y) | (x, y) \in A \text{ and } (x, y) \in B\}$$

$$C(x, y) = \begin{matrix} \longrightarrow \\ 1 & \text{if } A(x, y) \text{ and } B(x, y) \text{ are both } 1 \\ 0 & \text{otherwise} \end{matrix}$$

- Similarly:

- Complement ~ logical NOT
- Union ~ logical OR
- Difference ~ A AND (NOT B)

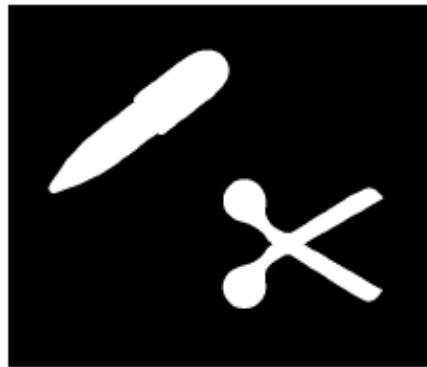


Examples

- Logical equivalents of set theory operations



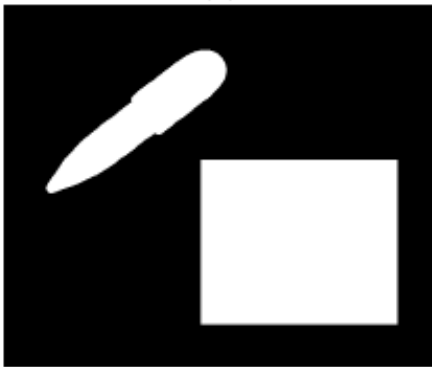
(a)



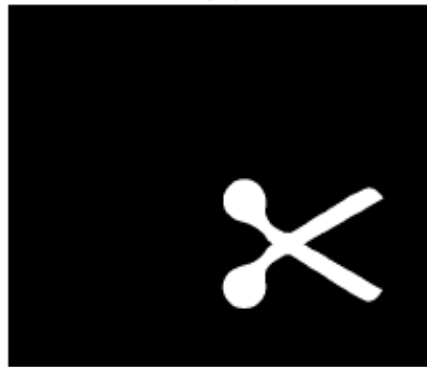
(b)



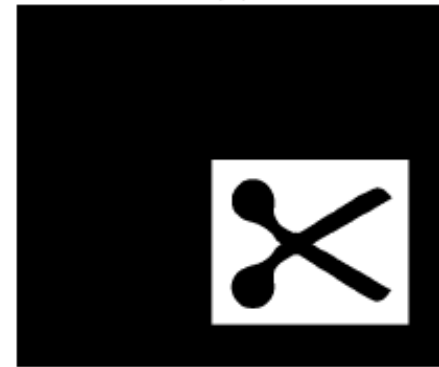
(c)



(d)



(e)



(f)

Operations

c) A^c

d) $A \cup B$

e) $A \cap B$

f) $A - B$

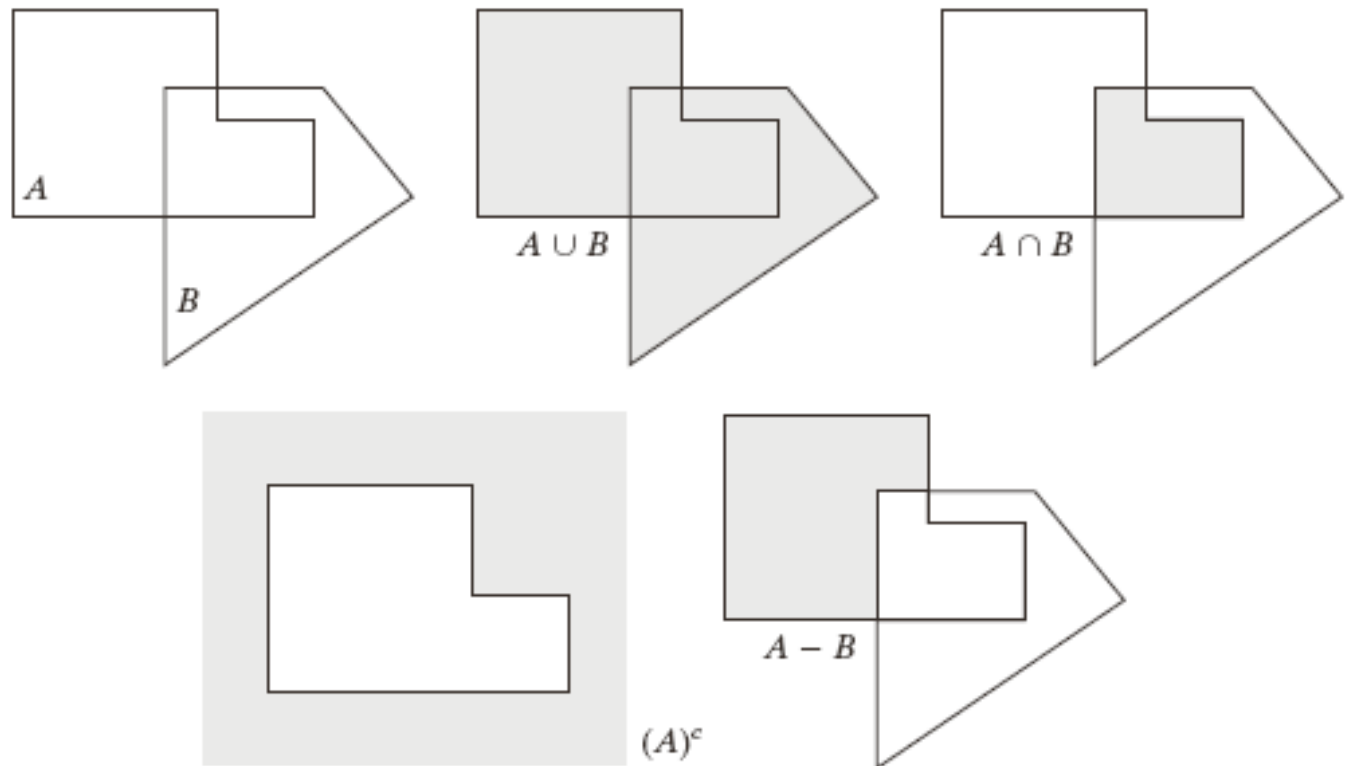


Set Operation on Images

a b c
d e

FIGURE 10.1

(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 .

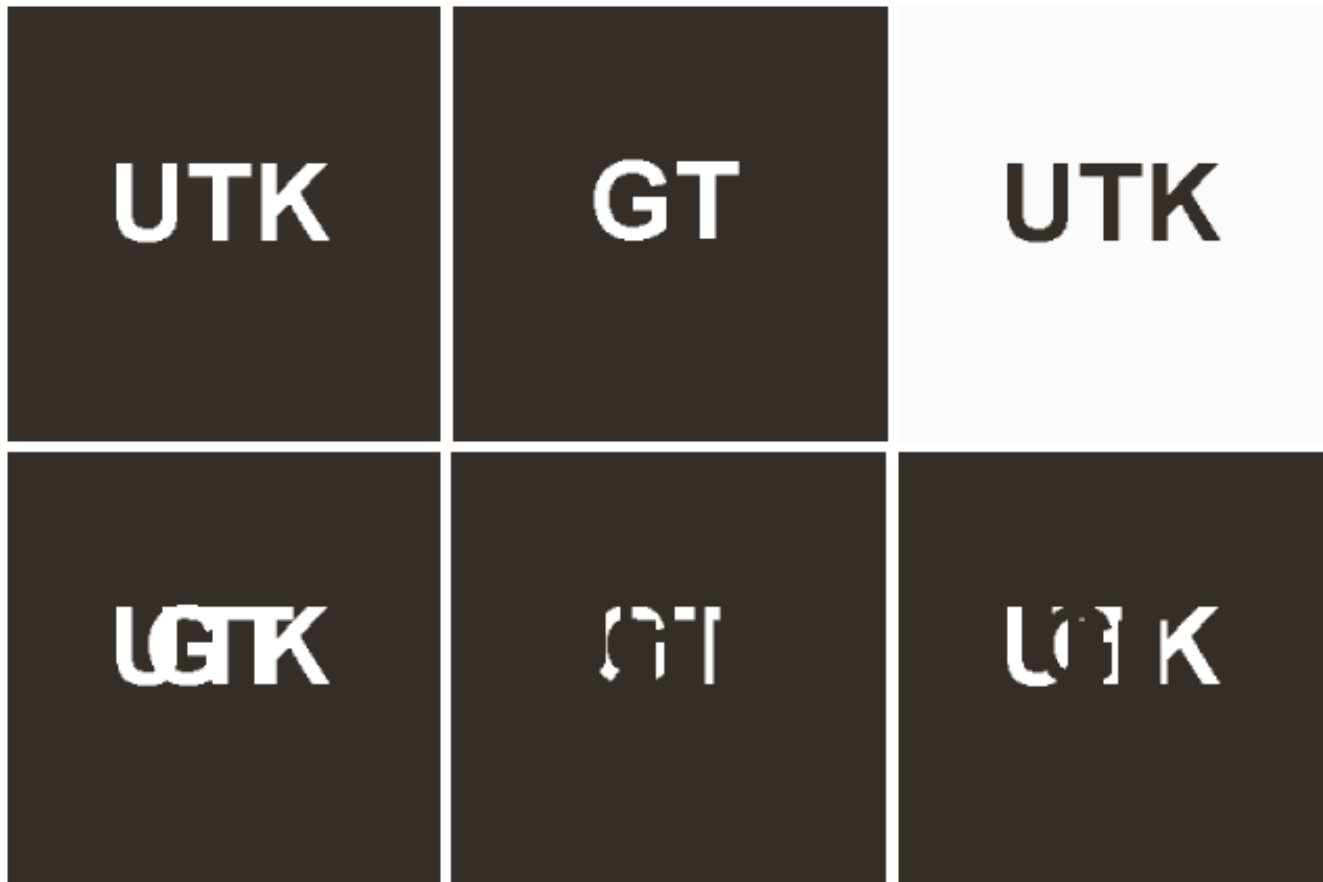


Set operations

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

TABLE 10.1
Using logical
expressions in
MATLAB to
perform set
operations on
binary images.

Set operations on Binary Images



a b c
d e f

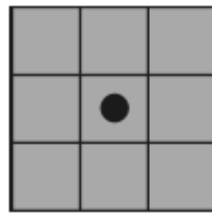
FIGURE 10.3 (a) Binary image A. (b) Binary image B. (c) Complement $\sim A$. (d) Union $A \cup B$. (e) Intersection $A \cap B$. (f) Set difference $A \setminus B$.

Structure Element

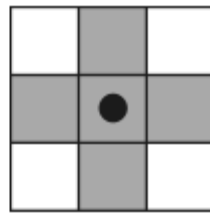
- The structuring element (SE) is the basic neighborhood structure associated with morphological image operations.
- It is usually represented as a small matrix, whose shape and size impact the results of applying a certain morphological operator to an image.
- Although a structuring element can have any shape, its implementation requires that it be converted to a rectangular array.
 - For each array, the shaded squares correspond to the members of the SE whereas the empty squares are used for padding, only.



Structuring Element Example



square



cross

Matlab's `strel` function is used to generate SE structure, e.g., `strel('square',3);`

Structuring Element

MORPH_RECT (0) - a rectangular structuring element:

MORPH_ELLIPSE (2) - an elliptic structuring element, that is, a filled ellipse inscribed into the rectangle

MORPH_CROSS (1)- a cross-shaped structuring element:

CV_SHAPE_CUSTOM (100) - custom structuring element



Dilation

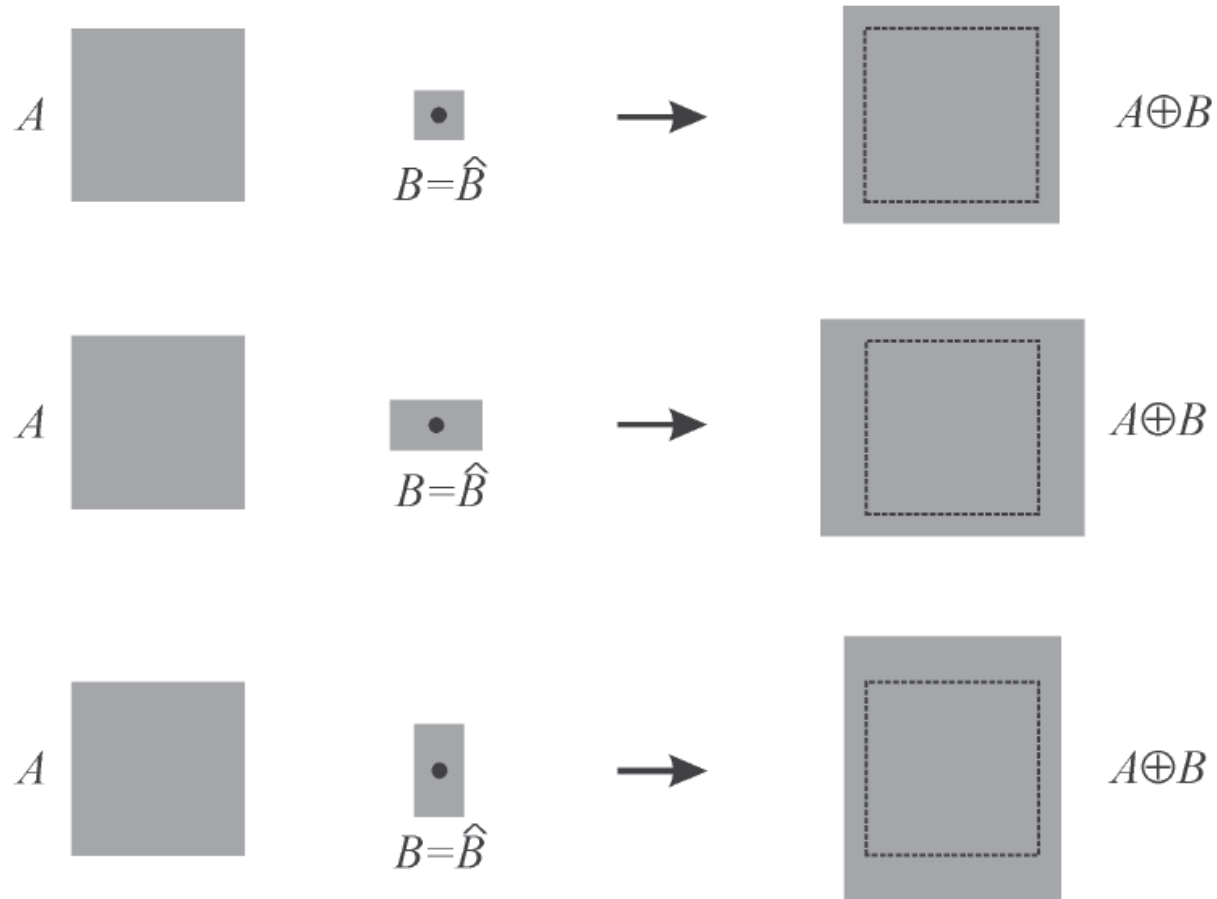
- The two fundamental morphological image operations.
 - **Dilation**: a morphological operation whose effect is to “grow” or “thicken” objects in a binary image.
 - The extent and direction of this thickening is controlled by the size and shape of the structuring element.
 - Mathematically:

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

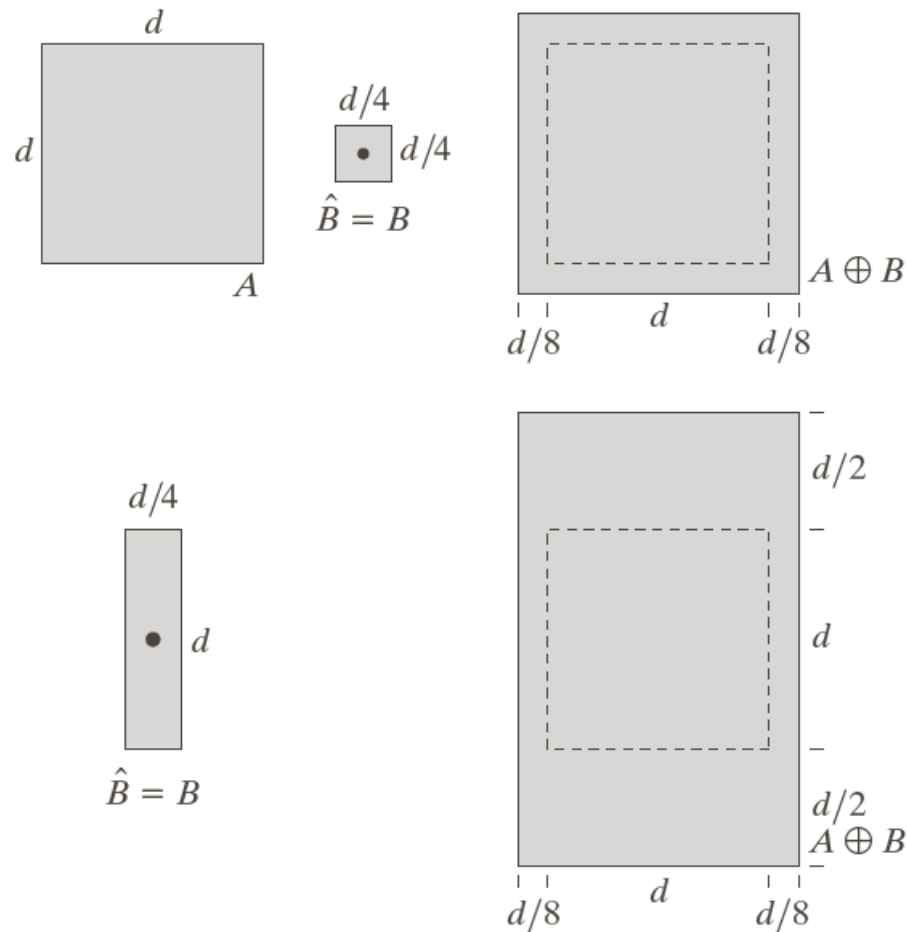
Dilation

In words, this is the set of all displacements z such that B_z and A are overlapped at least by one element

Example



Example of Dilation

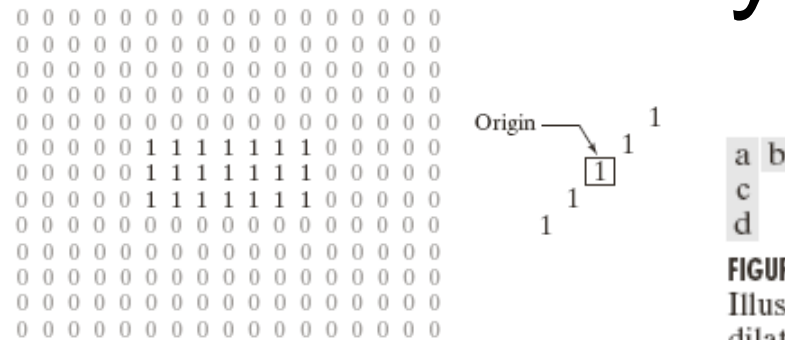


a	b	c
d		e

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

Example of Binary Data



The structuring element translated to these locations does not overlap any 1-valued pixels in the original image.

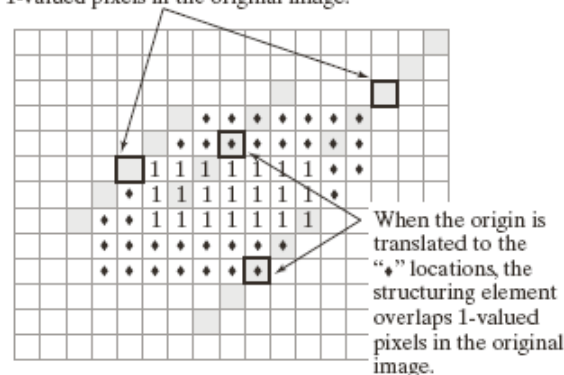


FIGURE 10.4

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.

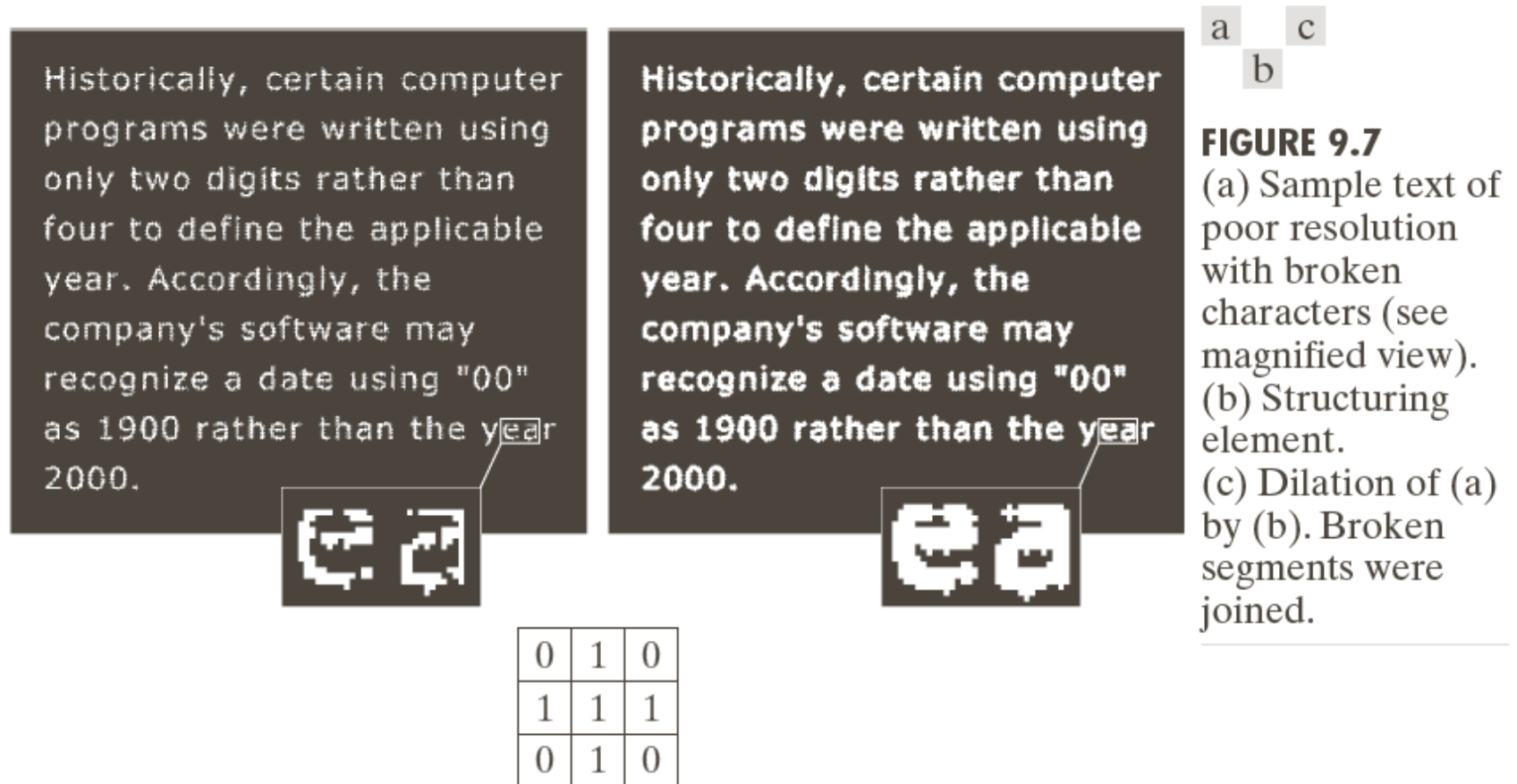


OpenCV code

```
int main(int argc, char** argv){  
    Mat src = imread("d:/image/broken_text.tif",  
    IMREAD_GRAYSCALE);  
    Mat dst;  
    Mat element = getStructuringElement(MORPH_CROSS,  
    Size(3, 3));  
    imshow("src", src);  
    dilate(src, dst, element);  
    imshow("result", dst);  
    waitKey(0); //  
}
```



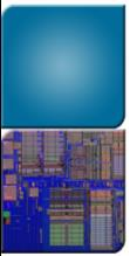
An example of Dilation's application



Erosion

- **Erosion:** a morphological operation whose effect is to “shrink” or “thin” objects in a binary image.
 - The direction and extent of this thinning is controlled by the shape and size of the structuring element.
 - Mathematically:

$$A \ominus B = \left\{ z \mid (\hat{B})_z \cup A^c \neq \emptyset \right\}$$

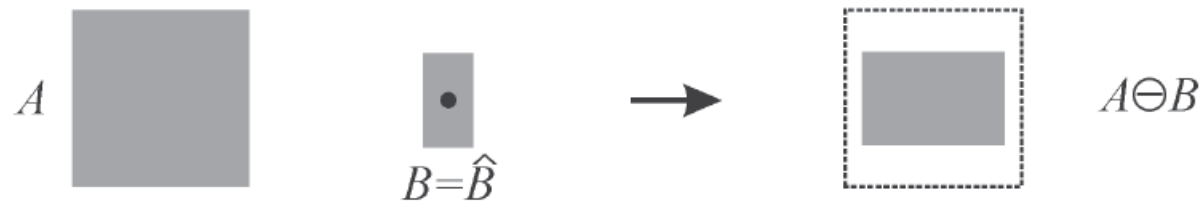


Erosion

In words, this is the set of all point z , such that B , translate by z is contained in A



Example



Example of Erosion

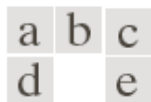
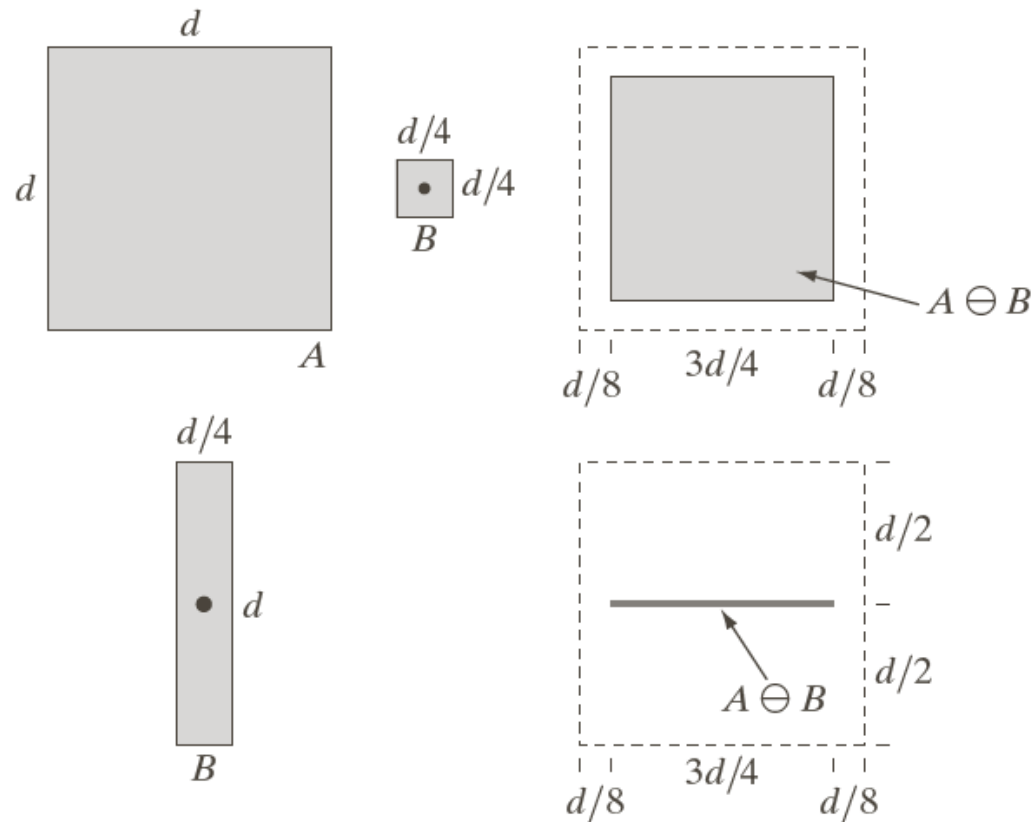


FIGURE 9.4 (a) Set A . (b) Square structuring element, B . (c) Erosion of A by B , shown shaded. (d) Elongated structuring element. (e) Erosion of A by B using this element. The dotted border in (c) and (e) is the boundary of set A , shown only for reference.

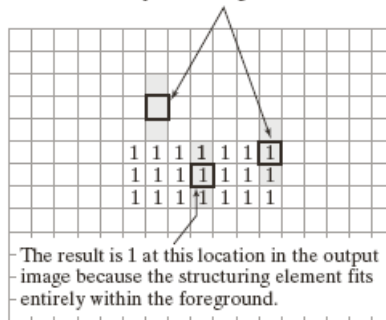
Example of Binary Data

```

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

```

The result is 0 at these locations in the output image because all or part of the structuring element overlaps the background.



```

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

```

```

1
1
1

```

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.

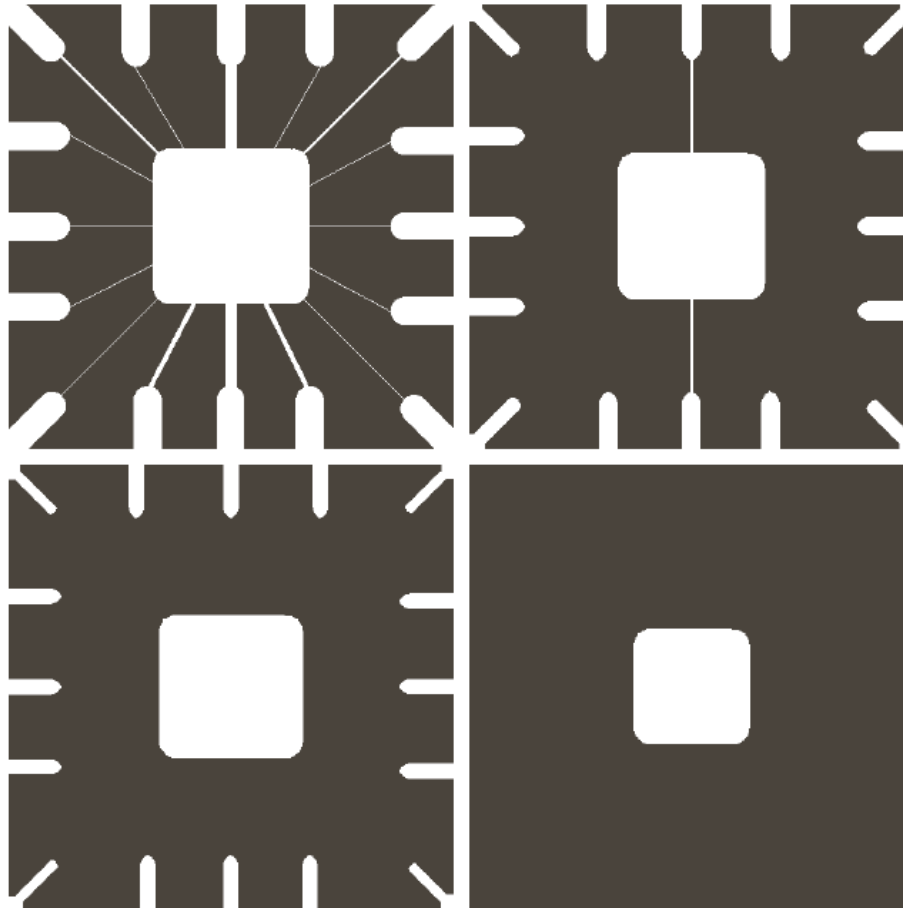


OpenCV

```
int main(int argc, char** argv){  
    Mat src = imread("d:/image/wirebond-mask.tif",  
        IMREAD_GRAYSCALE);  
    Mat dst;  
    Mat element = getStructuringElement(MORPH_ELLIPSE, Size(11,  
        11));  
    imshow("src", src);  
    erode(src, dst, element);  
    imshow("result", dst);  
    waitKey(0); //  
}
```



An example of Erosion's application



a	b
c	d

FIGURE 9.5 Using erosion to remove image components. (a) A 486×486 binary image of a wire-bond mask. (b)–(d) Image eroded using square structuring elements of sizes 11×11 , 15×15 , and 45×45 , respectively. The elements of the SEs were all 1s.

Dilation and Erosion

- Erosion and dilation are dual operations

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

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

Dilation and Erosion

- Erosion and dilation can be interpreted in terms of whether a SE *fits* or *hits* an image (region)
- Erosion:

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

- Dilation:

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



Opening and Closing

- We have seen, dilation expands the components of an image and erosion shrinks them
- Opening of A by B is simply erosion of A by B followed by the dilation of the result by B
- Closing of A by B is simply dilation of A by B followed by the erosion of the result by B



Opening and Closing

- The opening of A by B denoted $A \circ B$ is defined as

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

- The closing of A by B denoted $A \bullet B$ is defined as

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

Opening

- **Opening**: erosion followed by dilation
- Mathematically:

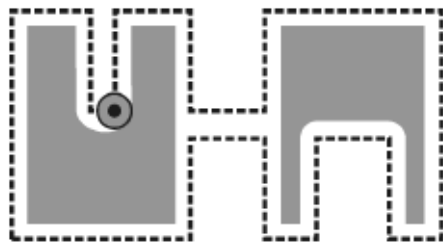
$$A \circ B = (A \ominus B) \oplus B$$

or:
$$A \circ B = \bigcup \{(B)_z \mid (B)_z \subseteq A\}$$

- In OpenCV: `morphologyEx(src, dst, MORPH_OPEN, element);`



Example

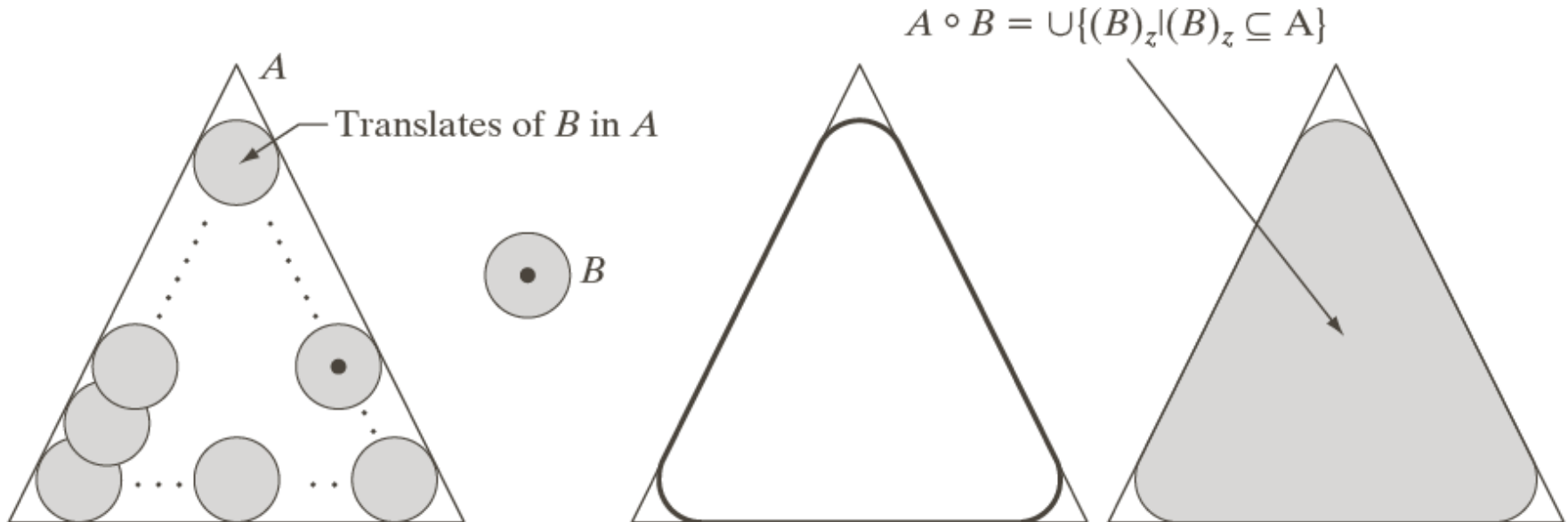


$A \ominus B$



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

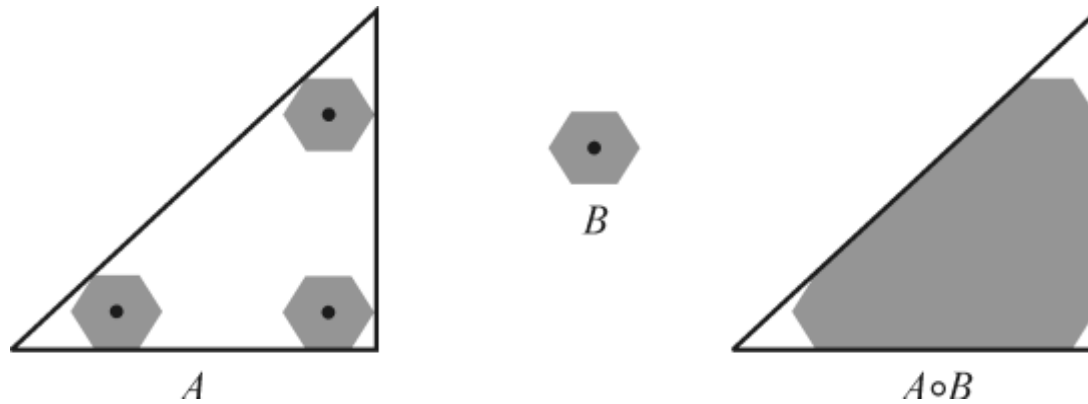
Opening



a b c d

FIGURE 9.8 (a) Structuring element B “rolling” along the inner boundary of A (the dot indicates the origin of B). (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.

Opening Example



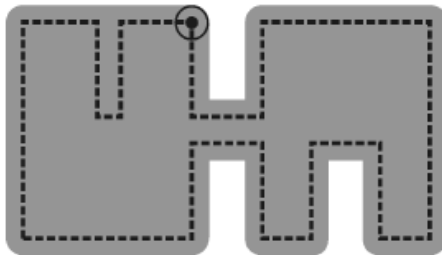
Closing

- **Closing**: dilation followed by erosion
- Mathematically:

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

- In OpenCV: `morphologyEx(src, dst, MORPH_CLOSE, element);`

Closing Example



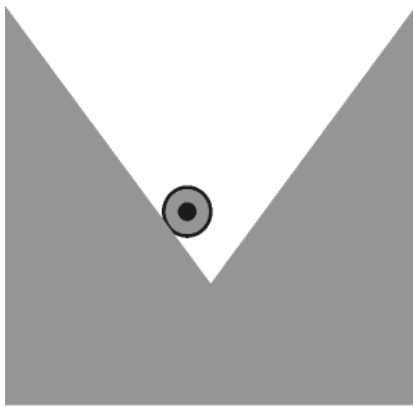
$$A \oplus B$$



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



Closing Example



A



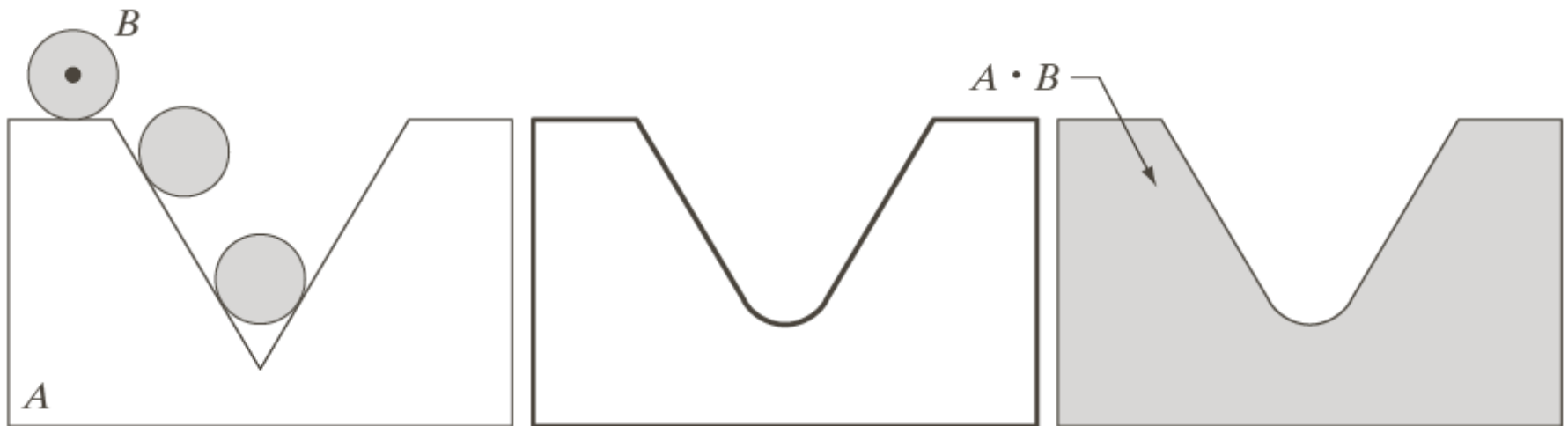
B



$A \bullet B$



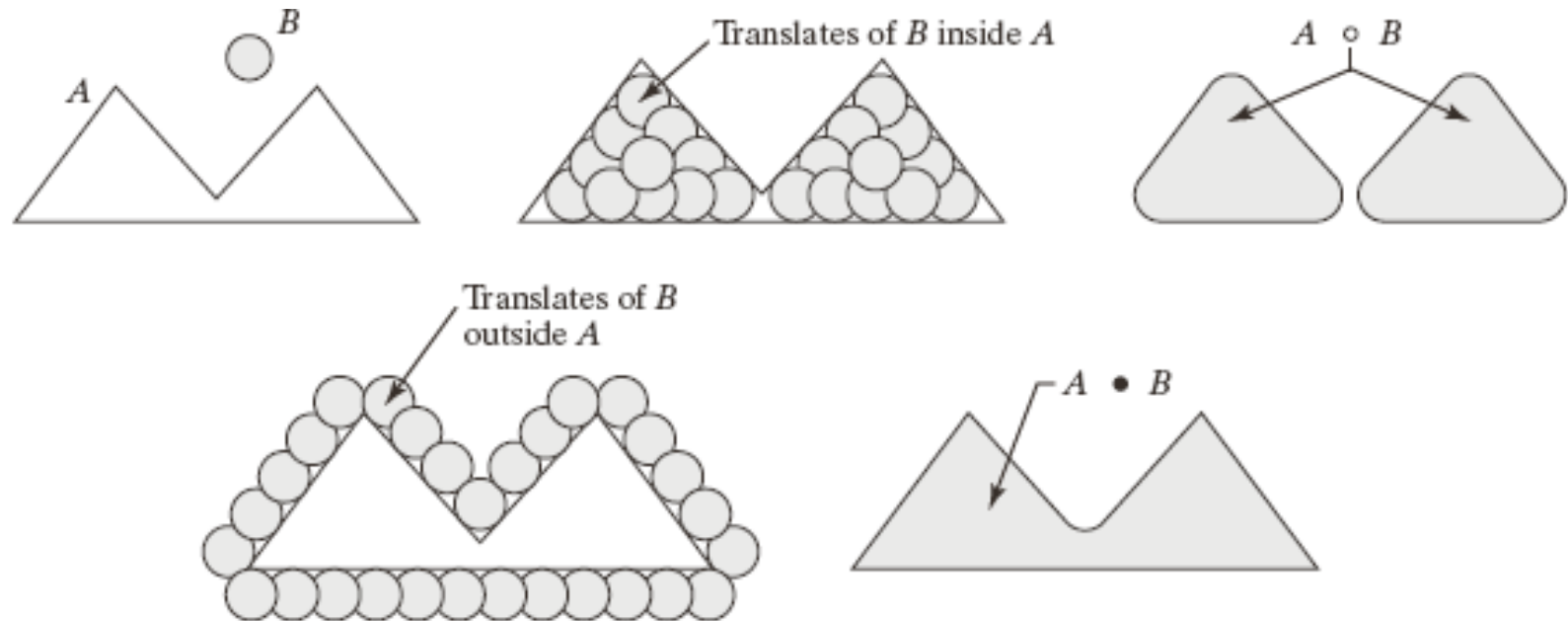
Closing



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.

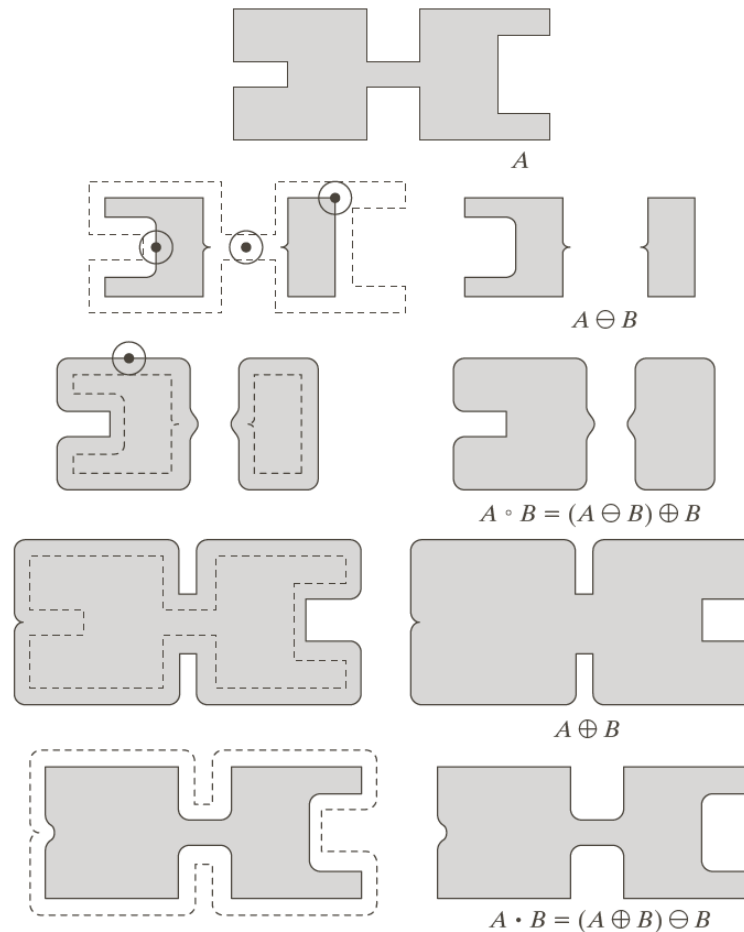
Opening and Closing



a b c
d e

FIGURE 10.9 Opening and closing as unions of translated structuring elements. (a) Set A and structuring element B . (b) Translations of B that fit entirely within set A . (c) The complete opening (shaded). (d) Translations of B outside the border of A . (e) The complete closing (shaded).

Comparison Opening and Closing



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 SE was not shaded here for clarity. The dark dot is the center of the structuring element.



OpenCV Morph Operation

`cv::morphologyEx` to apply Morphological Transformation such as:

Opening


Closing

Morphological Gradient

Top Hat

Black Hat



A decorative image in the top-left corner consisting of a blue square above a grid of smaller squares in various colors.

```
int main(int argc, char** argv){  
    Mat src = imread("d:/image/blobs.png", IMREAD_GRAYSCALE);  
    Mat dst1,dst2;  
    Mat element = getStructuringElement(MORPH_RECT, Size(5, 5));  
    imshow("src", src);  
    morphologyEx(src, dst1, MORPH_OPEN, element);  
    imshow("result1", dst1);  
  
    morphologyEx(src, dst2, MORPH_CLOSE, element);  
    imshow("result2", dst2);  
    waitKey(0); //  
}
```



Hit or Miss Transformation

- The Hit-or-Miss transformation is to identify specific configuration of pixels such as isolated foreground pixels or pixels that are end-point of line segment

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



Hit or Miss Transform

- **Hit-or-miss (HoM) transform:** a combination of morphological operations that uses two structuring elements (B_1 and B_2) designed in such a way that the output image will consist of all locations that match the pixels in B_1 (a *hit*) and that have none of the pixels in B_2 (a *miss*).

– Mathematically:

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

or:

$$A \otimes B = (A \ominus B_1) - (A \oplus \hat{B}_2)$$

- In OpenCV: `morphologyEx(src, dst1, MORPH_HITMISS, element);`



Example

- To identify the location of cross-shape pixels configuration:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Example using this:

$$A \otimes B = (A \ominus B_1) - (A \oplus \hat{B}_2)$$





```

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 0 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 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

```

  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 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

```

  1   1
1   [ ]   1
  1   1

```

```

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 0 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 0 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 b
c
d e
f
g

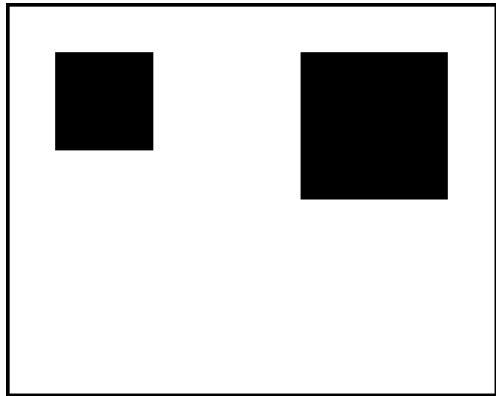
```

FIGURE 10.12

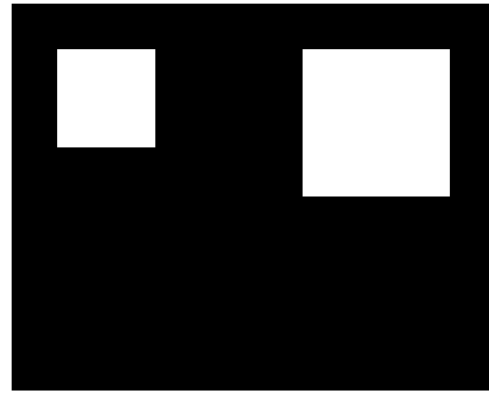
(a) Original image A . (b) Structuring element B_1 . (c) Erosion of A by B_1 . (d) Complement of the original image, A^c . (e) Structuring element B_2 . (f) Erosion of A^c by B_2 . (g) Output image.



Hit or Miss Example



(a)



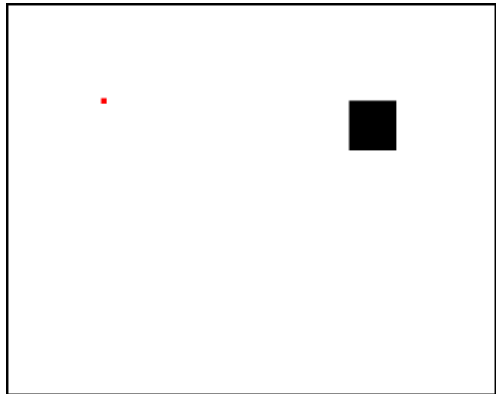
(b)



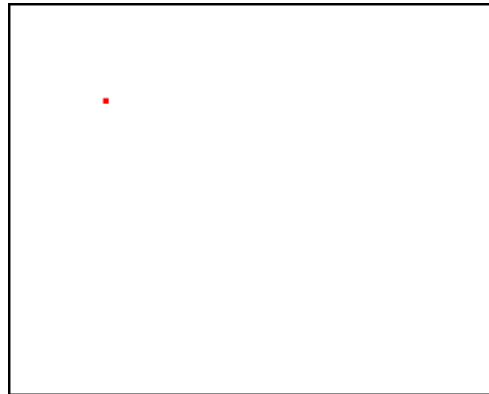
(c)



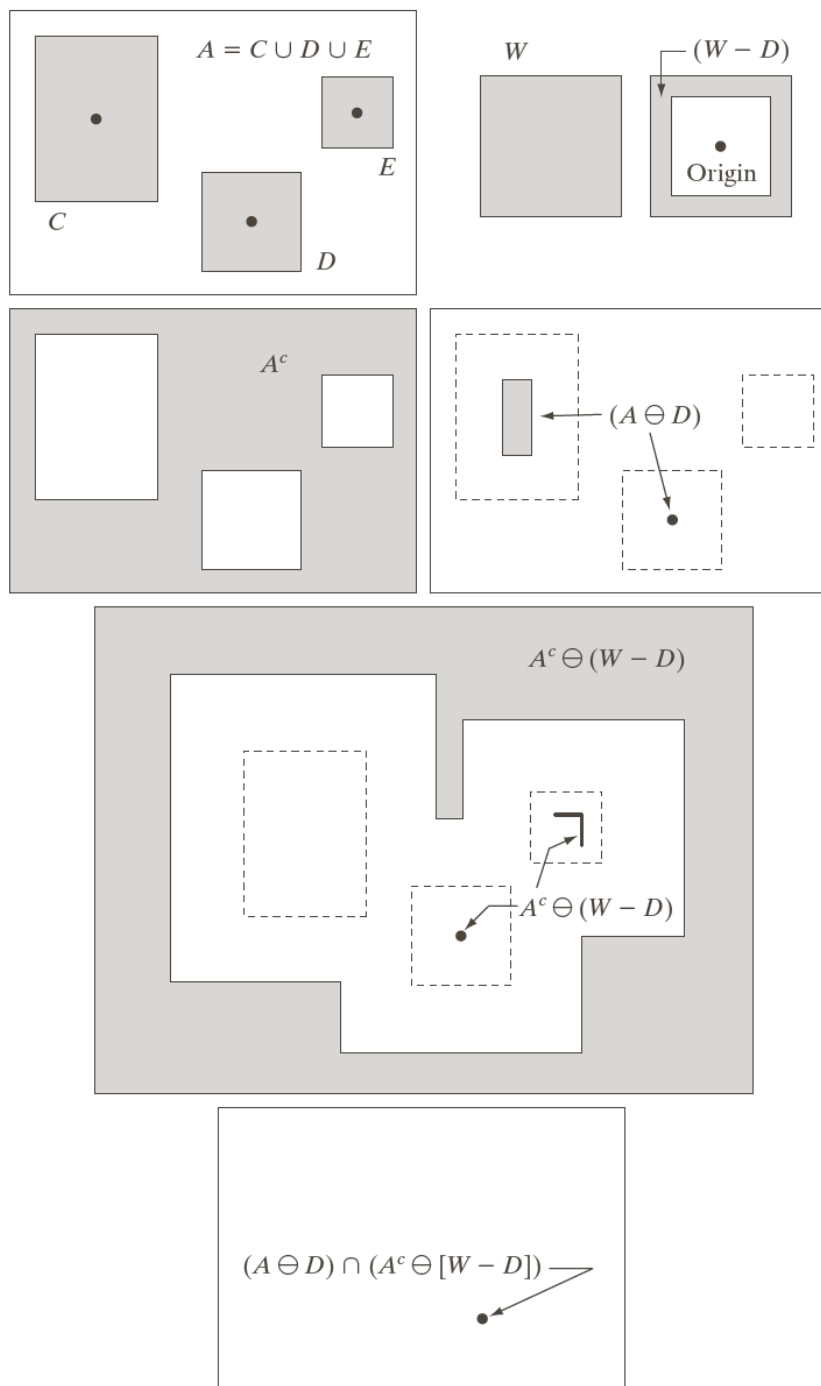
(d)



(e)




(f)




a	b
c	d
e	
f	

FIGURE 9.12

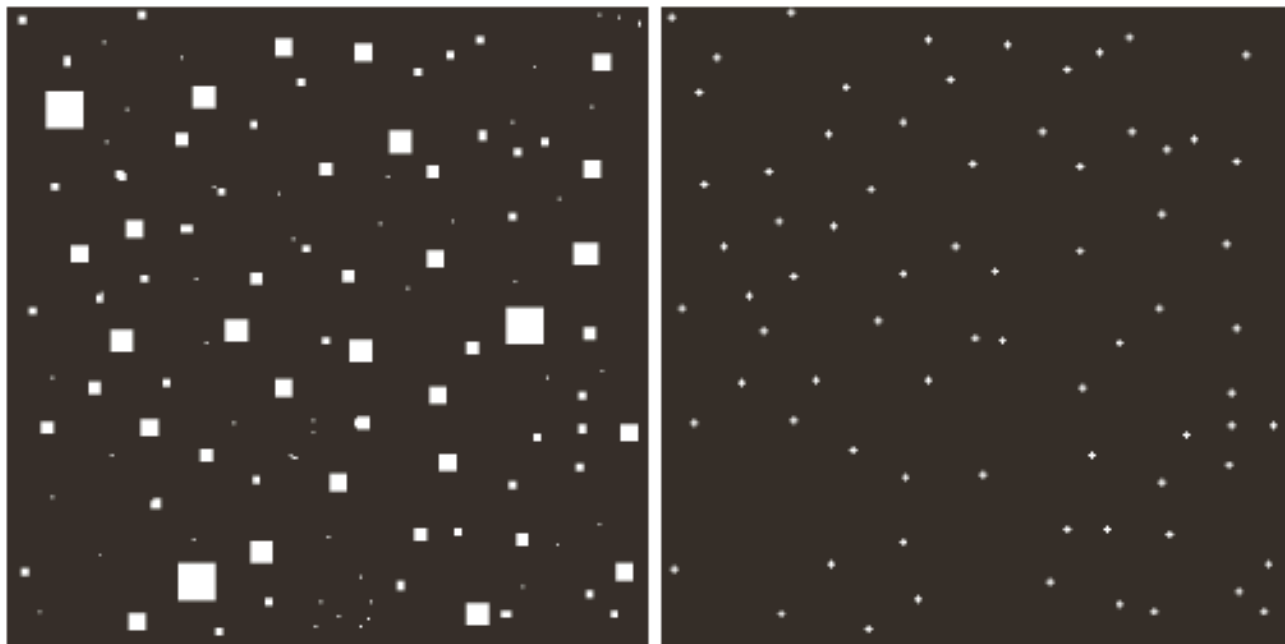
(a) Set A . (b) A window, W , and the local background of D with respect to W , $(W - D)$. (c) Complement of A . (d) Erosion of A by D . (e) Erosion of A^c by $(W - D)$. (f) Intersection of (d) and (e), showing the location of the origin of D , as desired. The dots indicate the origins of C , D , and E .

A decorative image in the top-left corner consisting of a blue square above a colorful, abstract pattern.

```
int main(int argc, char** argv){  
    Mat src = imread("d:/image/hit_miss.tif",  
        IMREAD_GRAYSCALE);  
    Mat dst1, dst2;  
    Mat element = getStructuringElement(MORPH_RECT,  
        Size(5, 5));  
    Mat kernel = (Mat_<int>(3, 3) <<  
        0, 0, 0,  
        0, 1, 1,  
        0, 1, 1);  
    imshow("src", src);  
    morphologyEx(src, dst1, MORPH_HITMISS, kernel);  
    imshow("result1", dst1);  
    waitKey(0); //  
}
```

A decorative image in the bottom-left corner consisting of a yellow square followed by a colorful, abstract pattern.

Result



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).

Morphological Filtering

- Morphological filters are Boolean filters that apply a many-to-one binary (or Boolean) function h within a window W in the binary input image $f(x,y)$, producing at the output an image $g(x,y)$ given by:

$$g(x, y) = h [W f(x, y)]$$



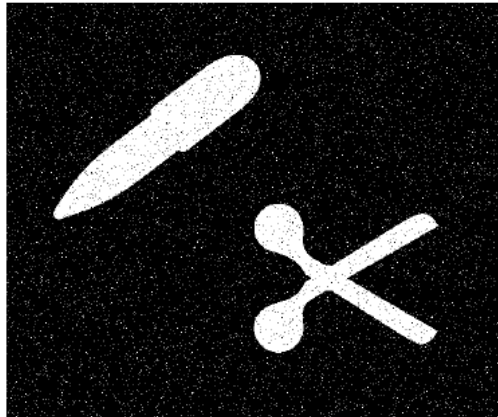
Example

- Examples of Boolean operations (h):
 - OR: equivalent to a morphological dilation with a square SE of the same size as W .
 - AND: equivalent to a morphological erosion with a square SE of the same size as W .
 - MAJ (majority): the morphological equivalent to a median filter applicable to binary images.



Morphological filtering

- Application: noise removal $C = ((A \circ B) \bullet B)$



(a)



(b)




(c)



(d)



A decorative image in the top-left corner consisting of a blue square above a grid of smaller squares in various colors.

```
int main(int argc, char** argv){  
    Mat src = imread("d:/image/y_noise.tif",  
IMREAD_GRAYSCALE);  
    Mat dst1, dst2;  
    Mat element = getStructuringElement(MORPH_RECT,  
Size(2, 2));  
    morphologyEx(src, dst1, MORPH_OPEN, element);  
    morphologyEx(dst1, dst2, MORPH_CLOSE, element);  
    imshow("result1", dst2);  
    waitKey(0); //  
}
```



A small abstract image in the top-left corner, showing a blue gradient transitioning into a purple and blue pattern.

Boundary extraction

- Internal: pixels in A that sit at the edge of A .

$$\mathcal{BE}(A) = A - (A \ominus B)$$

- External: pixels outside A that sit immediately next to A .

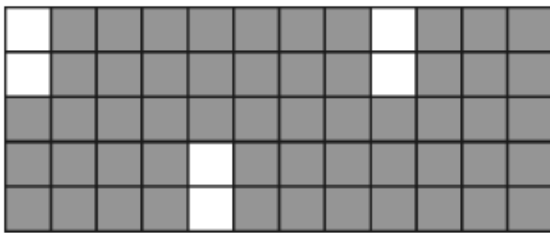
$$\mathcal{BE}(A) = (A \oplus B) - A$$

- Morphological gradient: combination of internal and external boundaries.

$$\mathcal{BE}(A) = (A \oplus B) - (A \ominus B)$$

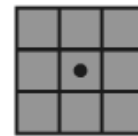


Example



A

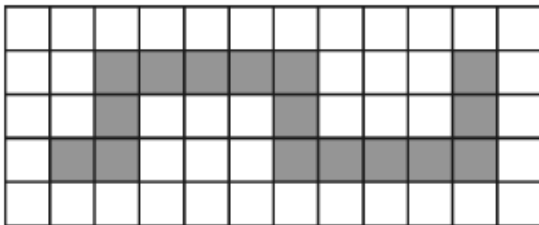
(a)



B

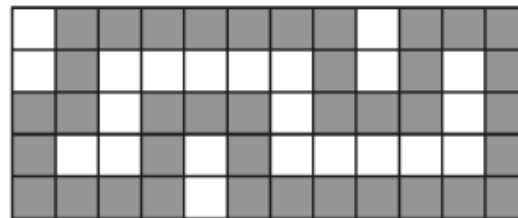
(b)

```
a = ones(5,12)
a(1:2,1)=0
a(1:2,9)=0
a(4:5,5)=0
b = bwperim(a,8)
```



$A \oplus B$

(c)

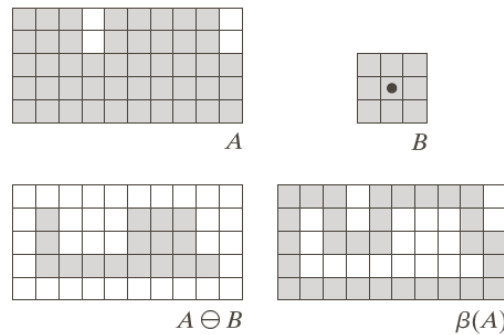


$A - (A \oplus B)$

(d)

Boundary Extraction

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



a	b
c	d

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.

Example



a b

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

Region Filling

Let p be a pixel in a region surrounded by an 8-connected boundary, A . The goal of a region filling algorithm is to fill up the entire region with 1s using p as a starting point (i.e., setting it as 1). Region filling can be accomplished using an iterative procedure, mathematically expressed as follows:

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3, \dots \quad (13.27)$$

where: $X_0 = p$ and B is the cross-shaped structuring element. The algorithm stops at the k^{th} iteration if $X_k = X_{k-1}$. The union of X_k and A contains the original boundary (A) and all the pixels within it labeled as 1.



A decorative image in the top-left corner consisting of a blue square above a grid of smaller squares in various colors.

Region/Hole Filling

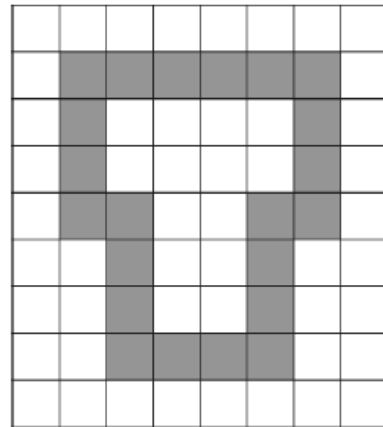
- Hole filling

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

- At iteration $X_k = X_{k-1}$ then we fill all the holes

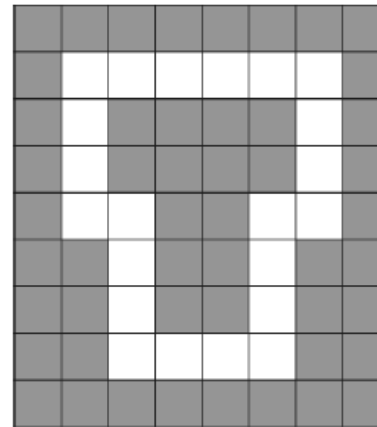


Example



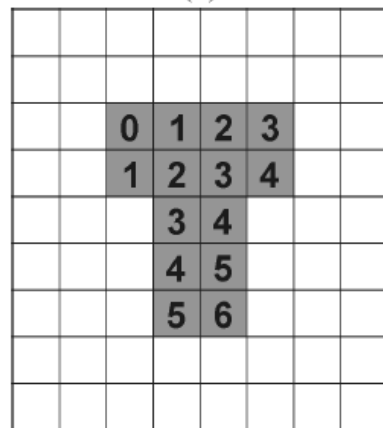
A

(a)



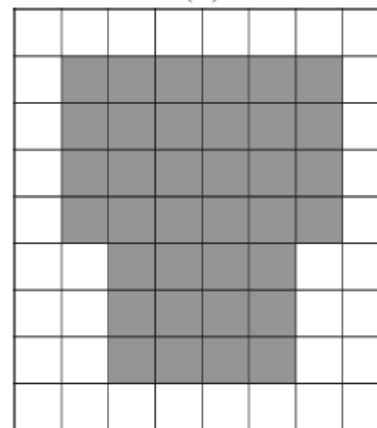
A^c

(b)



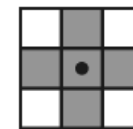
X_6

(c)



$X_6 \cup A$

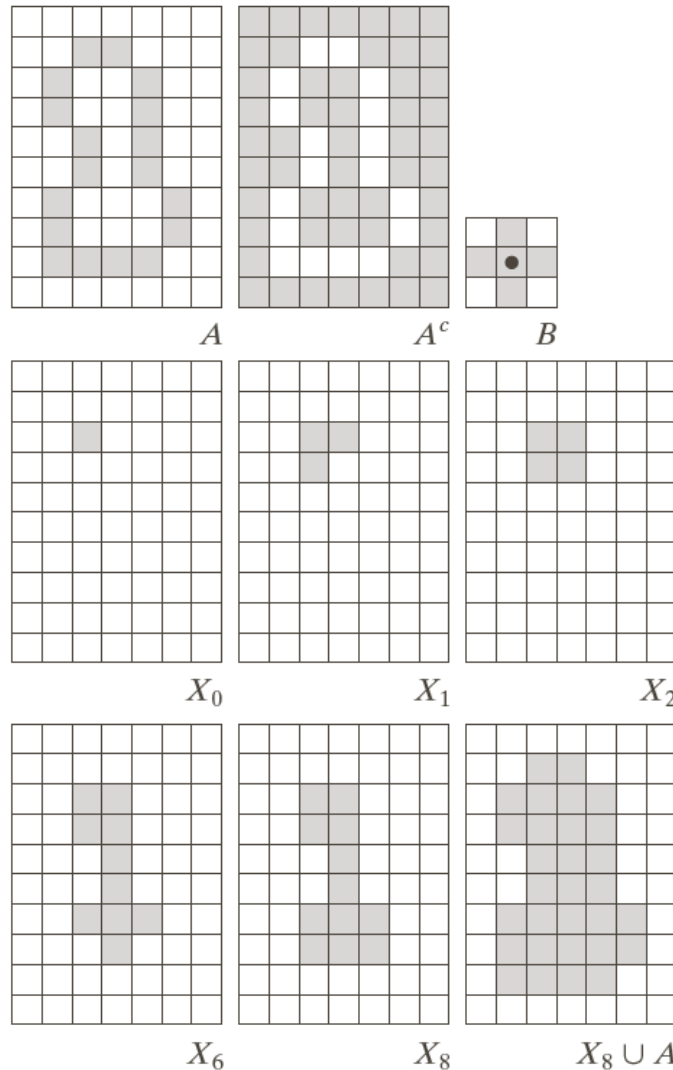
(d)



B

(e)

Example



a	b	c
d	e	f
g	h	i

FIGURE 9.15 Hole filling. (a) Set A (shown shaded). (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)].

Example



a b c

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

Connected Components

- Iterative procedure, similar to region filling

$$X_k = (X_{k-1} \oplus B) \cap A \quad k = 1, 2, 3, \dots \quad (13.28)$$

where: $X_0 = p$ and B is a suitable structuring element: cross-shaped for 4-connectivity, 3×3 square for 8-connectivity.

The algorithm stops at the k^{th} iteration if $X_k = X_{k-1}$.



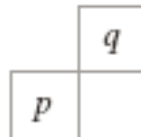
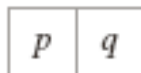
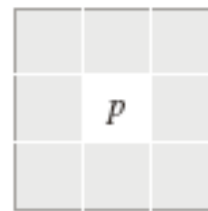
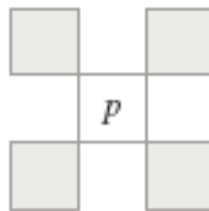
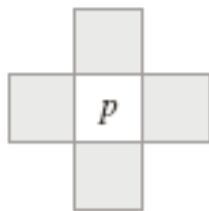
Connected Components

- Let S represent a subset of pixels in an image. Two pixels p and q are said to be connected in S if there exists a path between them consisting entirely of pixels in S .
- For any pixel p in S , the set of pixels that are connected to it is called a connected component of S .
- If it has only one connected component, the set S is called a connected set.

4-neighbors and 8-neighbors

- Given a pixel p at coordinates (x,y) , 4-neighbors of p denoted $N_4(p)$ are $(x+1,y)$, $(x-1,y)$, $(x,y+1)$, and $(x,y-1)$
- The four diagonal neighbors of p denoted $N_D(p)$ are $(x+1,y+1)$, $(x+1,y-1)$, $(x-1,y+1)$, and $(x-1,y-1)$
- The 8-neighbors of p are the union of $N_4(p)$ and $N_D(p)$

4-neighbors and 8-neighbors



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

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

a	b	c
d	e	
f	g	

FIGURE 10.18

(a) Pixel p and its 4-neighbors,

(b) Pixel p and its diagonal neighbors,

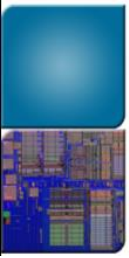
(c) Pixel p and its 8-neighbors,

(d) Pixels p and q are 4-adjacent and 8-adjacent.

(e) Pixels p and q are 8-adjacent but not 4-adjacent.

(f) The shaded pixels are both 4-connected and 8-connected.

(g) The shaded pixels are 8-connected but not 4-connected.



Finding Connected Components

- Finding connected components

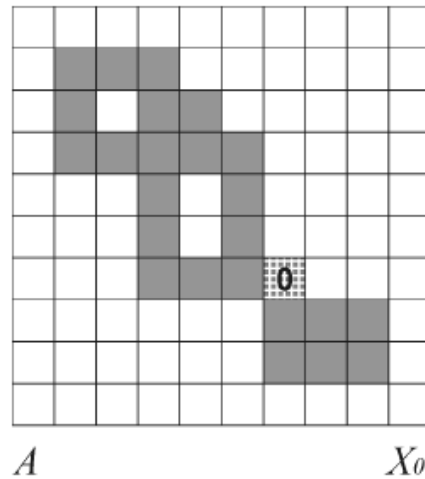
$$X_k = (X_{k-1} \oplus B) \cap A \quad k = 1, 2, 3, \dots$$

- At iteration $X_k = X_{k-1}$ then the algorithm terminates

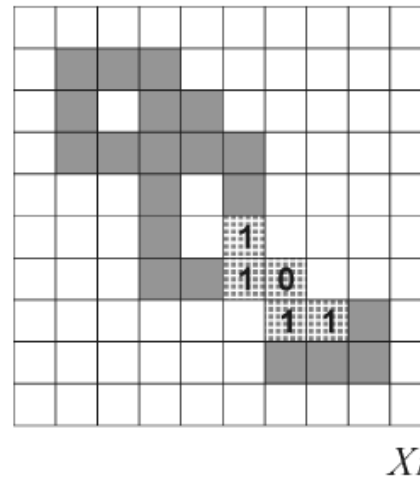




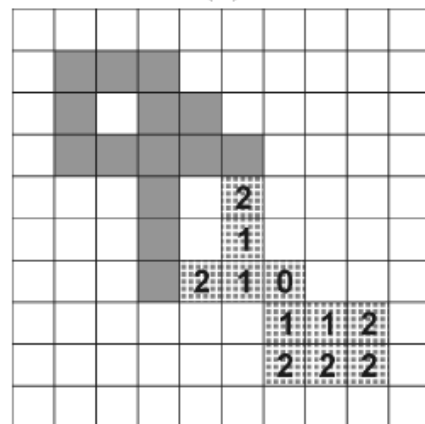
Example



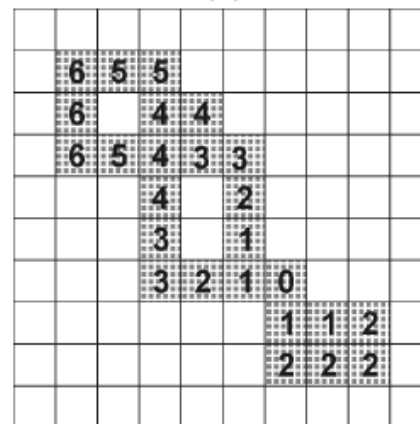
(a)



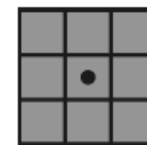
(b)



(c)



(d)



B

(e)



Example

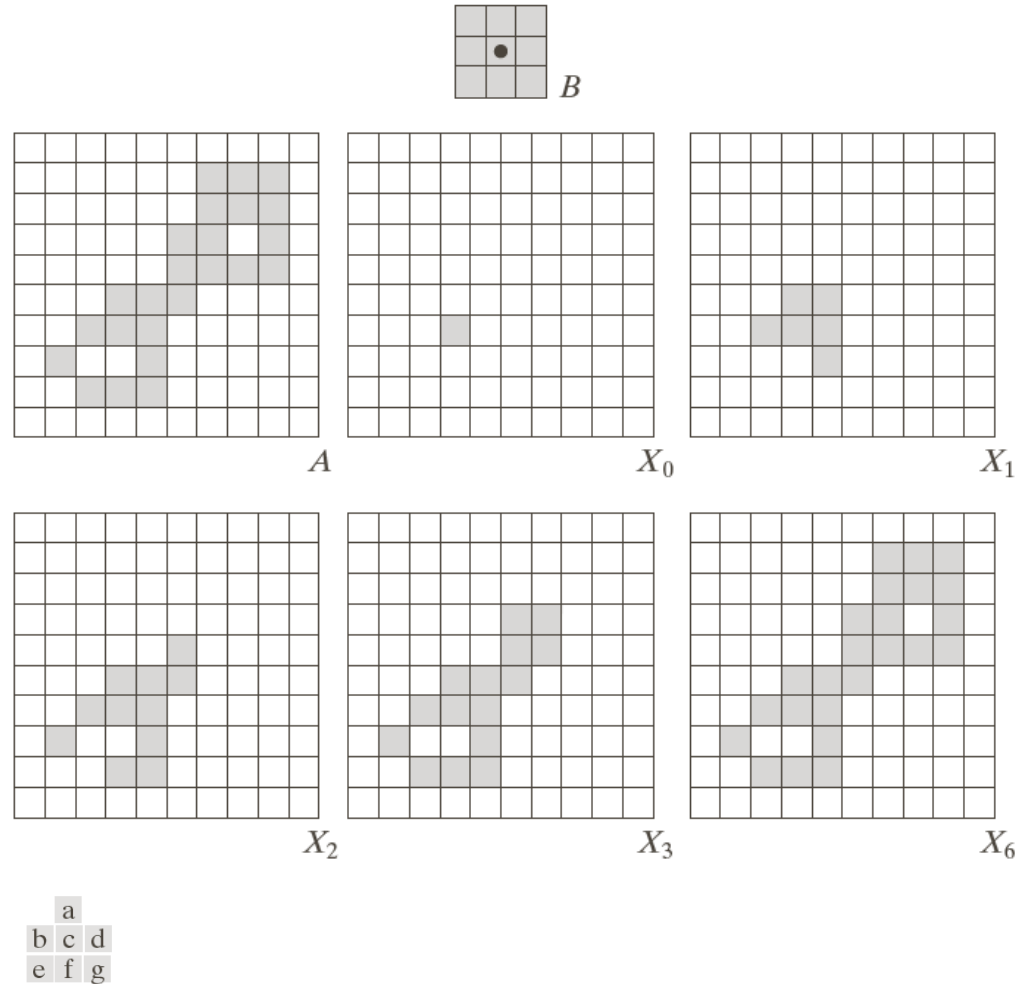
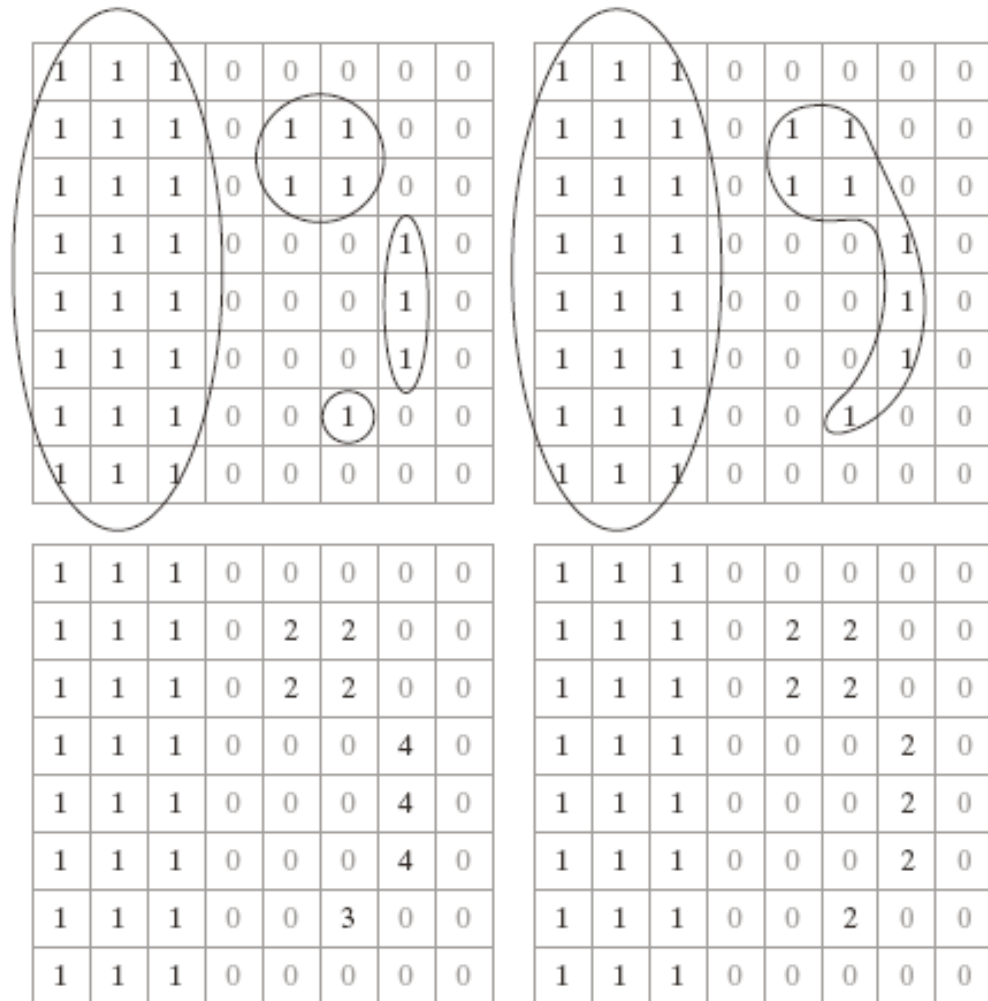


FIGURE 9.17 Extracting connected components. (a) Structuring element. (b) Array containing a set with one connected component. (c) Initial array containing a 1 in the region of the connected component. (d)–(g) Various steps in the iteration of Eq. (9.5-3).

Example

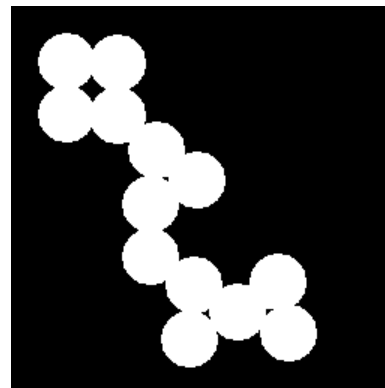
a b
c d

FIGURE 10.19
Connected components.
(a) Four
4-connected components.
(b) Two
8-connected components.
(c) Label matrix
obtained using
4-connectivity
(d) Label matrix
obtained using
8-connectivity.

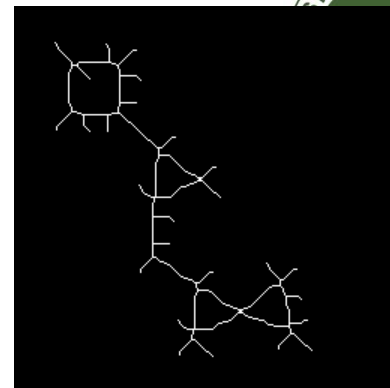


Example

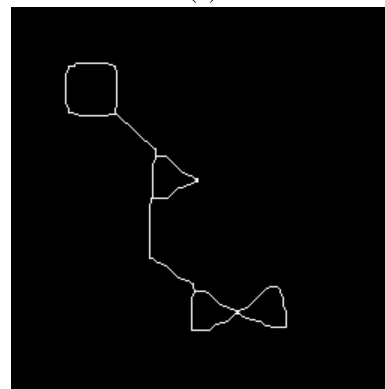
```
A = imread('circles.png');  
B = bwmorph(A,'skel', Inf);  
C = bwmorph(B,'spur',Inf);  
D = bwmorph(A,'remove');  
E = bwmorph(D,'thicken',3);  
F = bwmorph(E,'thin',3);
```



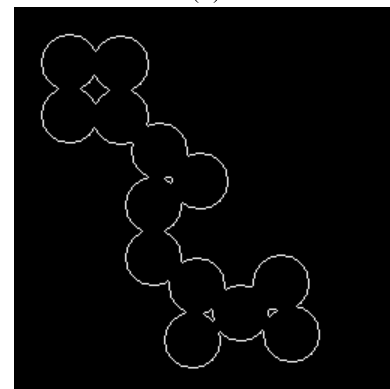
(a)



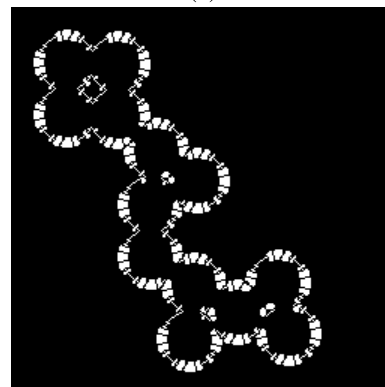
(b)



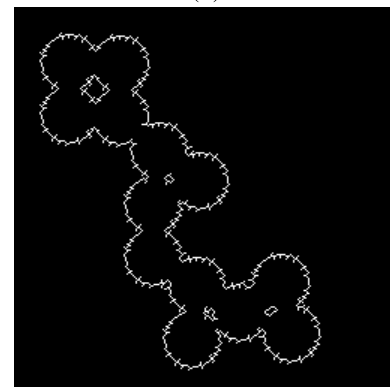
(c)




(d)



(e)




(f)

A blue square and a small image of a circuit board.

```
Mat src_gray;
int thresh = 100;
int max_thresh = 255;
RNG rng(12345);
void thresh_callback(int, void*){
    Mat canny_output;
    vector<vector<Point> > contours;
    vector<Vec4i> hierarchy;
    Canny(src_gray, canny_output, thresh, thresh * 2, 3);
    findContours(canny_output, contours, hierarchy, CV_RETR_TREE,
CV_CHAIN_APPROX_SIMPLE, Point(0, 0));
    Mat drawing = Mat::zeros(canny_output.size(), CV_8UC3);
    for (int i = 0; i< contours.size(); i++){
        Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255),
rng.uniform(0, 255));
        drawContours(drawing, contours, i, color, 2, 8, hierarchy, 0, Point());
    }
    namedWindow("Contours", CV_WINDOW_AUTOSIZE);
    imshow("Contours", drawing);
}
```



A decorative image in the top-left corner consisting of a blue square above a grid of smaller squares in various colors.

```
int main(int argc, char** argv){
    src_gray = imread("d:/image/Test3.png", IMREAD_GRAYSCALE);
    char* source_window = "Source";
    namedWindow(source_window, CV_WINDOW_AUTOSIZE);
    imshow(source_window, src_gray);

    createTrackbar(" Canny thresh:", "Source", &thresh, max_thresh,
thresh_callback);
    thresh_callback(0, 0);

    waitKey(0);
    return(0);

}
```





Questions?



```
int main(int argc, char** argv){
    Mat src = imread("d:/image/eight_pepper.tif", IMREAD_GRAYSCALE);
    // Apply the filters
    Mat dst, dst2, dst3;
    inRange(src, Scalar(0, 0, 100), Scalar(40, 30, 255), dst);
    Mat element = getStructuringElement(MORPH_ELLIPSE, Size(15, 15));
    dilate(dst, dst2, element);
    erode(dst2, dst3, element);
    // Show the results
    namedWindow(" ORIGINAL ", WINDOW_AUTOSIZE);
    imshow(" ORIGINAL ", src);
    namedWindow(" SEGMENTED ", WINDOW_AUTOSIZE);
    imshow(" SEGMENTED ", dst);
    namedWindow(" DILATION ", WINDOW_AUTOSIZE);
    imshow(" DILATION ", dst2);
    namedWindow(" EROSION ", WINDOW_AUTOSIZE);
    imshow(" EROSION ", dst3);
    waitKey();
    return 0;
}
```

