National University of Sciences and Technology (NUST)

SEECS

Digital Image Processing

Khawar Khurshid

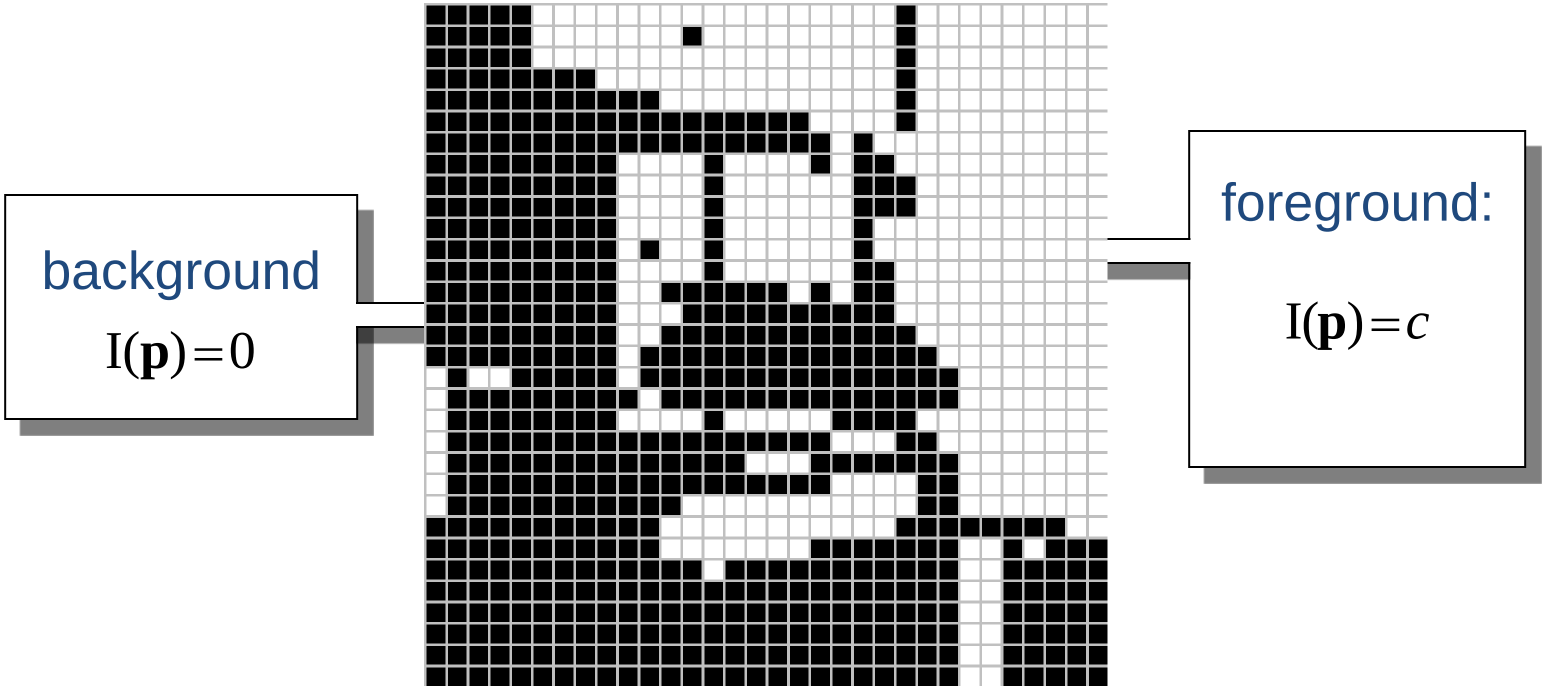
Morphological Operations

Introduction

A tool for extracting image components that are useful in the representation and description of region shapes.

The language of mathematical morphology is Set Theory.

Introduction



This represents a digital image. Each square is one pixel.

Quick Example

Image after thresholding

After morphological operations

Set Theory

The set space of binary image is Z^2

Each element of the set is a 2D vector whose coordinates are the *(x,y)* of a black (or white, depending on the convention) pixel in the image

The set space of gray level image is Z^3

Each element of the set is a 3D vector: *(x,y)* and intensity level.

NOTE:

Set Theory and Logical operations are covered in:
Section 9.1, Chapter # 9, 2nd Edition DIP by Gonzalez
Section 2.6.4, Chapter # 2, 3rd Edition DIP by Gonzalez

Set Theory

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

$$a \in A$$

If a is not an element of A, we write

$$a \notin A$$

Set representation

$$A = \{(a_1, a_2), (a_3, a_4)\}$$

Empty or Null set

$$A = \varnothing$$

Set Theory

Subset: if every element of A is also an element of another set B, the A is said to be a subset of B

$$A \subseteq B$$

Union: The set of all elements belonging either to A, B or both

$$C = A \bigcup B$$

Intersection: The set of all elements belonging to both A and B

$$D = A \bigcap B$$

Set Theory

Two sets A and B are said to be disjoint or mutually exclusive if they have no common element

$$A \bigcap B = \varnothing$$

Complement: The set of elements not contained in A

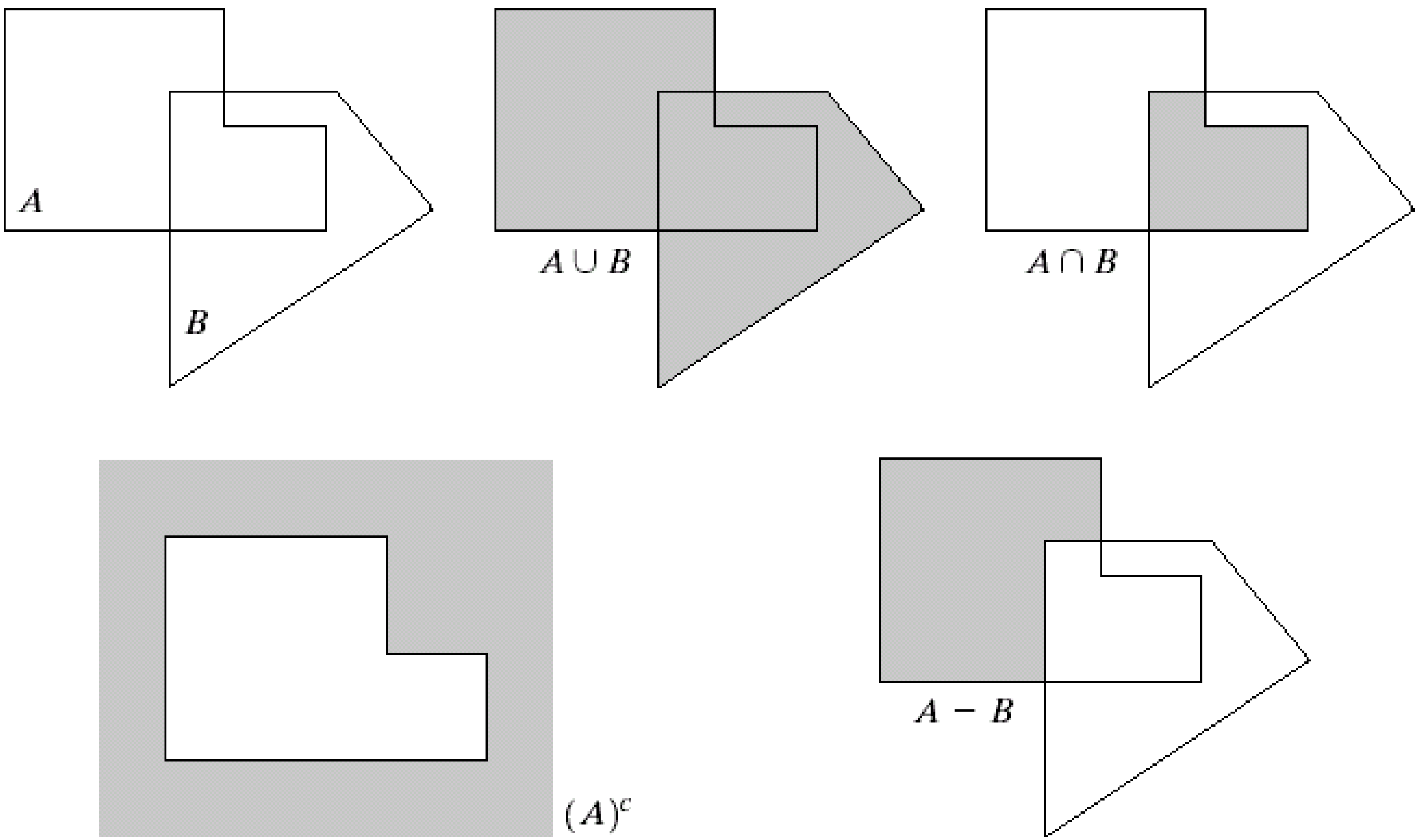
$$A^c = \{w \mid w \notin A\}$$

Difference of two sets A and B, denoted by A – B, is defined as

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

i.e. the set of elements that belong to A, but not to B

Set Theory



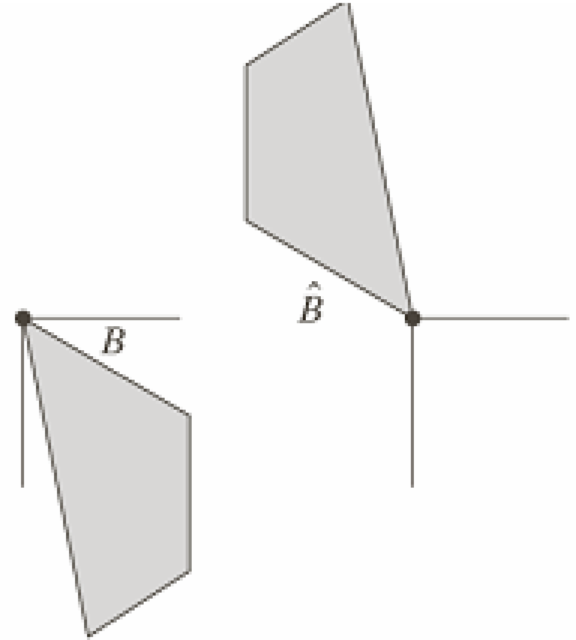
a b c
d e

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.

Reflection of set B

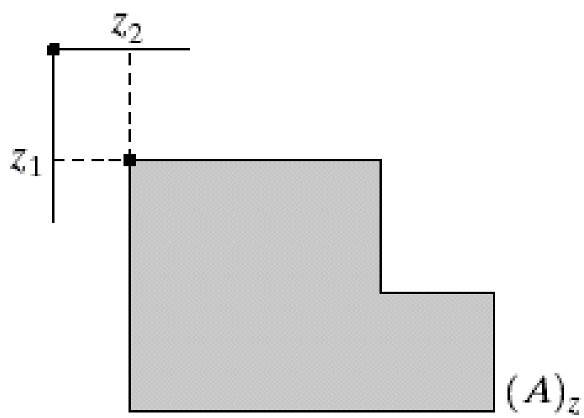
$$B = \{w \mid w = -b, \quad \text{for } b \in B\}$$

i.e. the set of element *w*, such that *w* is formed by multiplying each of two coordinates of all the elements of set *B* by -1



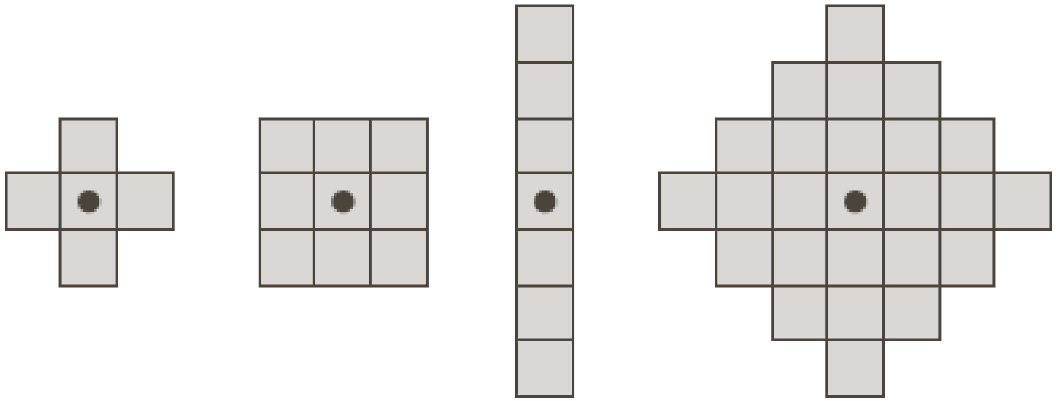
Translation of set A by point $z = (z_1, z_2)$, denoted $(A)_z$, is defined as

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

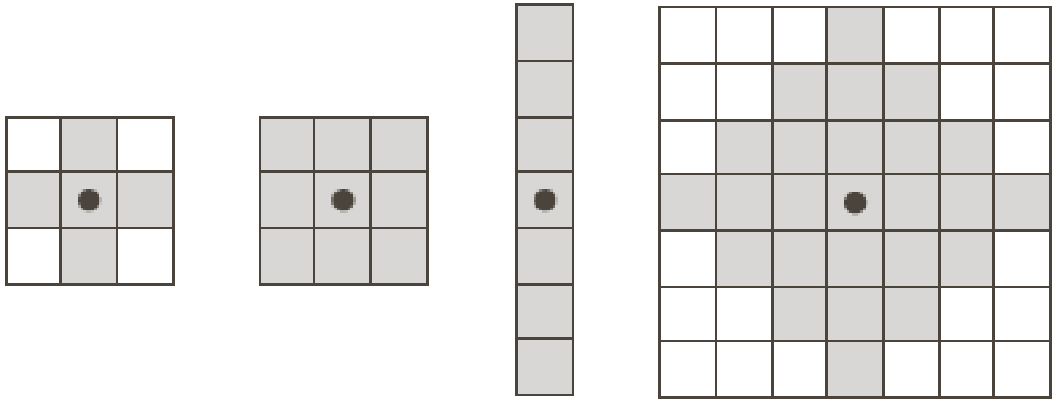


Structuring Elements

A structuring element is a small image – used as a moving window.

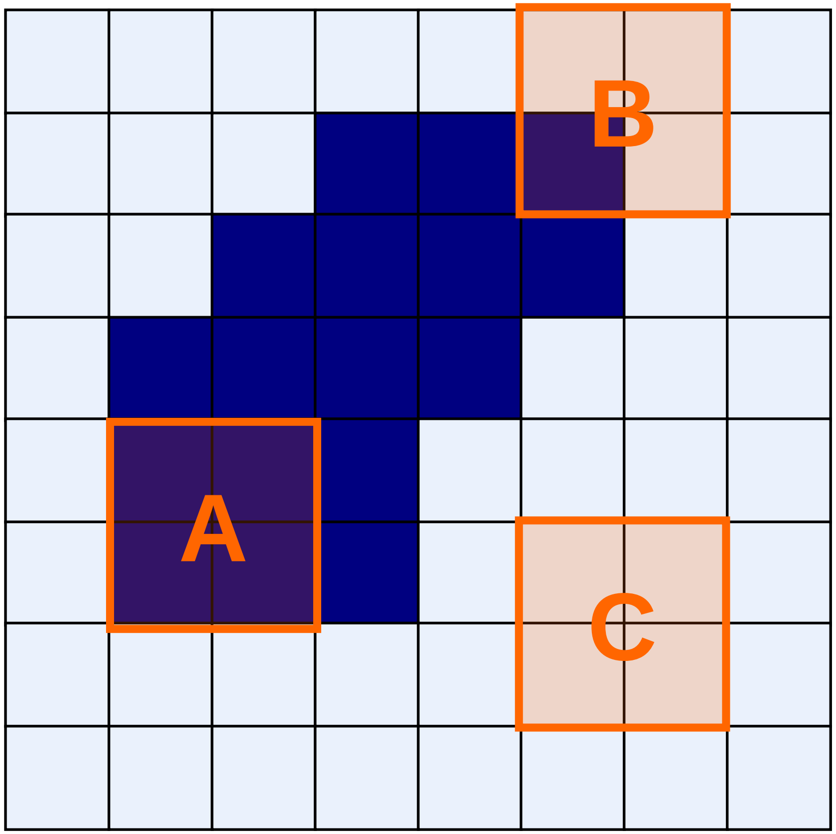


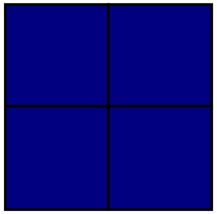
Example Structuring Elements



Structuring Elements Converted to Rectangular Arrays

Structuring Elements





Structuring Element

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

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

All morphological processing operations are based on these simple ideas

Fundamental Morphology

Fundamentally morphological image processing is very 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 ***f*** by structuring element ***s*** is given by
$f \ominus s$

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

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

Erosion - Steps

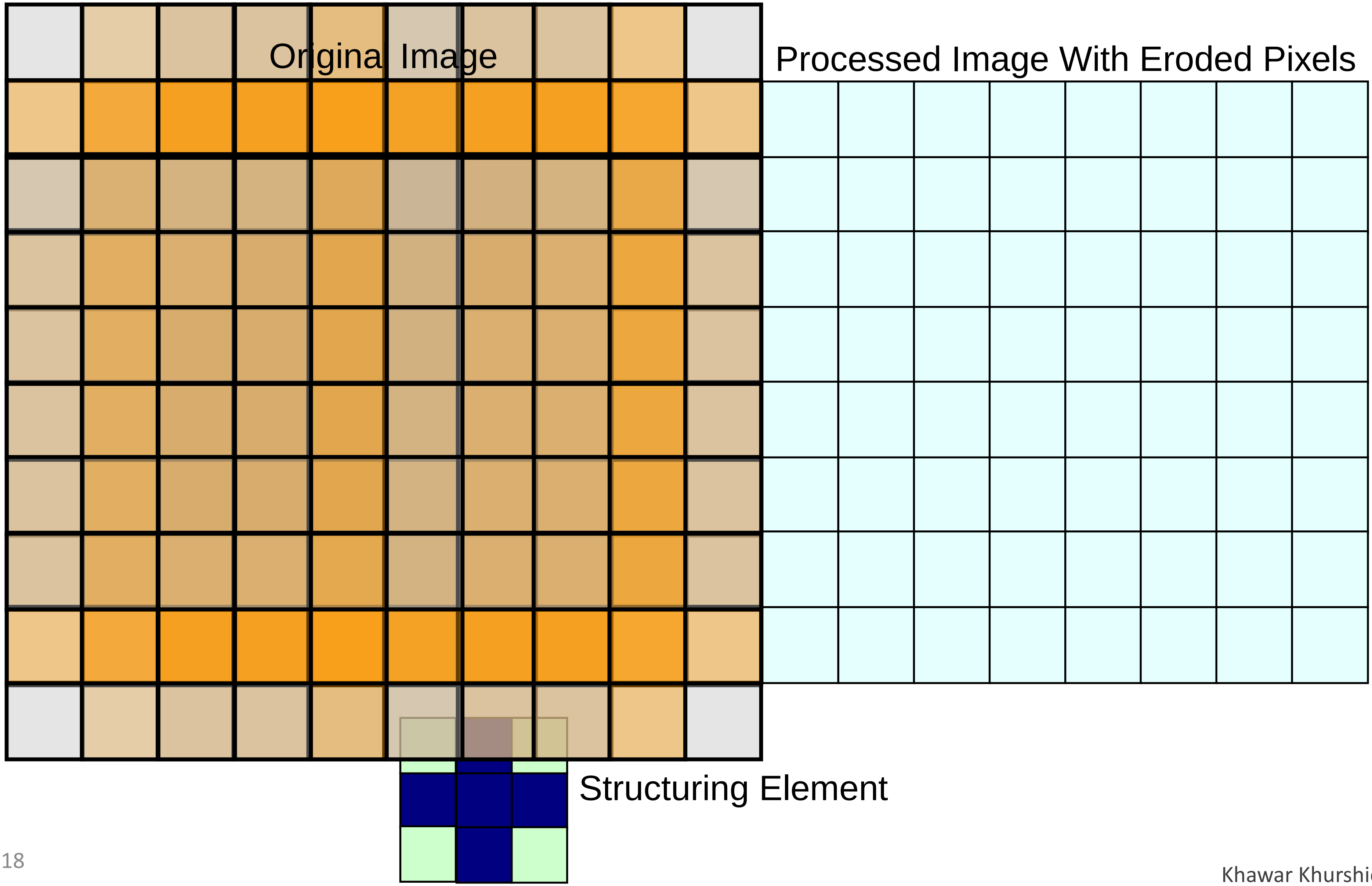
For each foreground pixel (which we will call the *input pixel*)

Superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position.

If *for every* pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is.

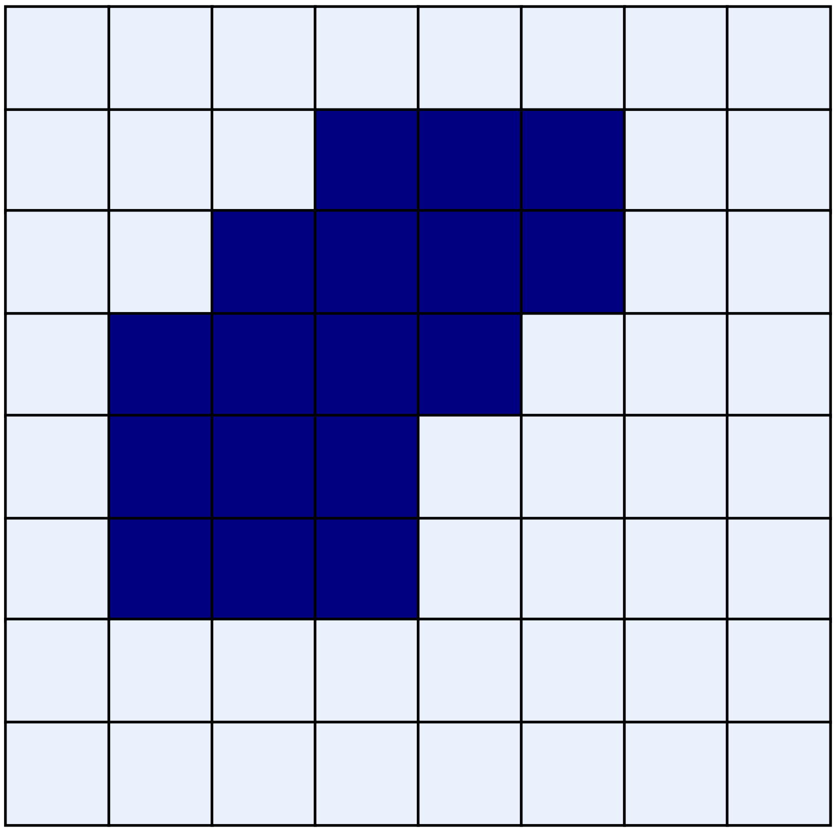
If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value.

Erosion - Example

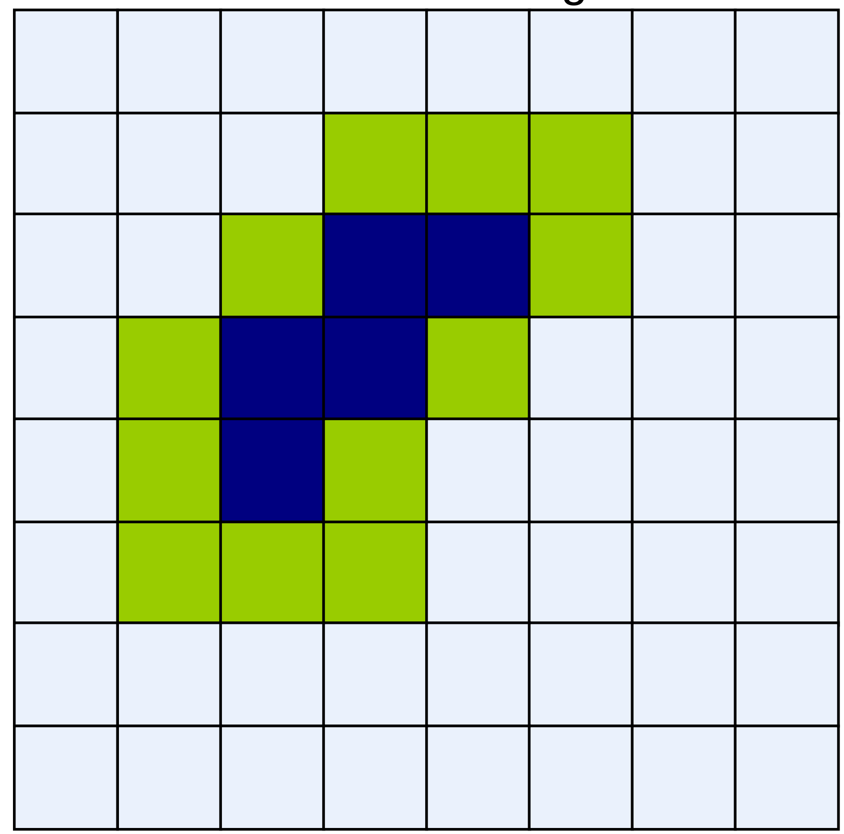


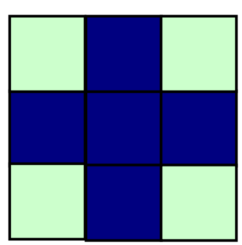
Erosion - Example

Original Image



Processed Image





Structuring Element

Erosion - Basics

Effects

Shrinks the size of foreground (1-valued) objects
Smoothes object boundaries
Removes small objects

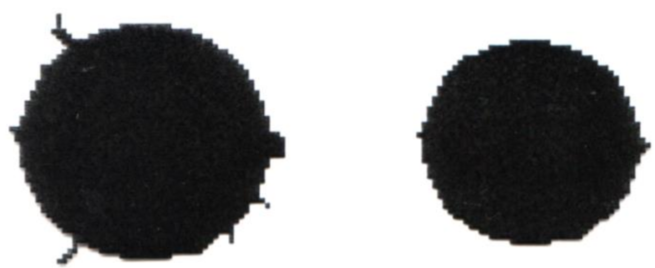
Rule for Erosion

In a binary image, if any of the pixel (in the neighborhood defined by structuring element) is 0, then output is 0

Erosion - Basics

Erosion can split apart joined objects

Erosion can strip away extrusions



Watch out: Erosion shrinks objects

Erosion - Results

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

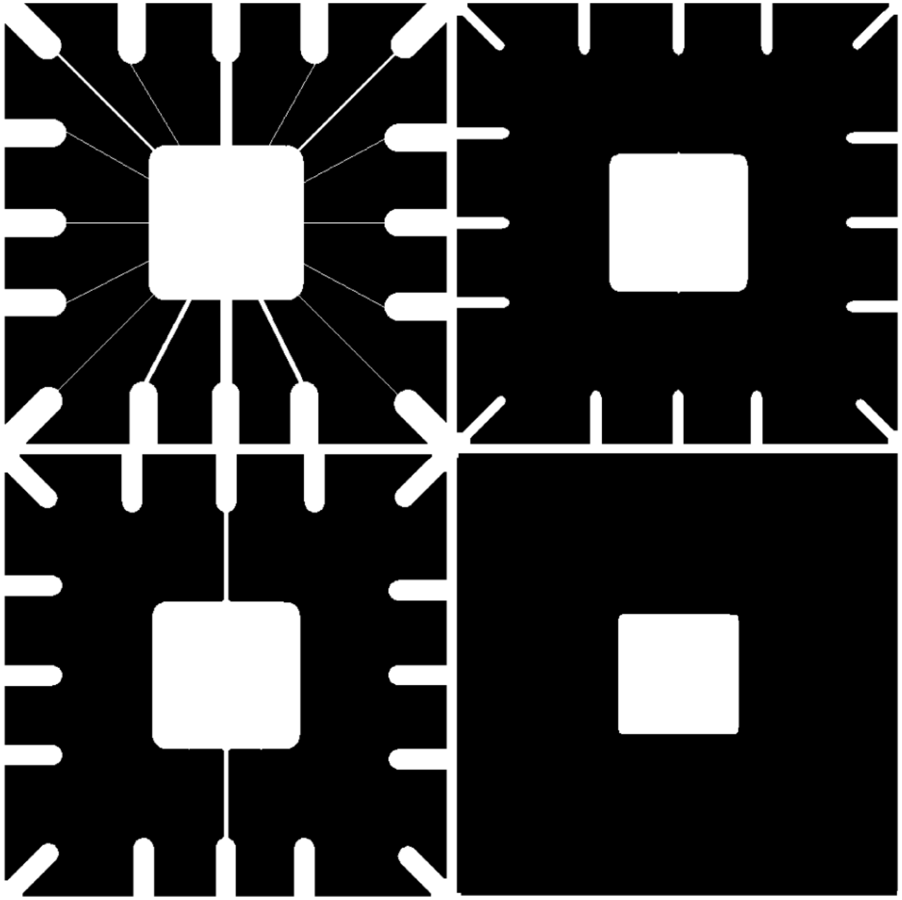
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0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

Erosion with a structuring element of size 3x3

Erosion - Results

Original image

After erosion with a disc of radius 10



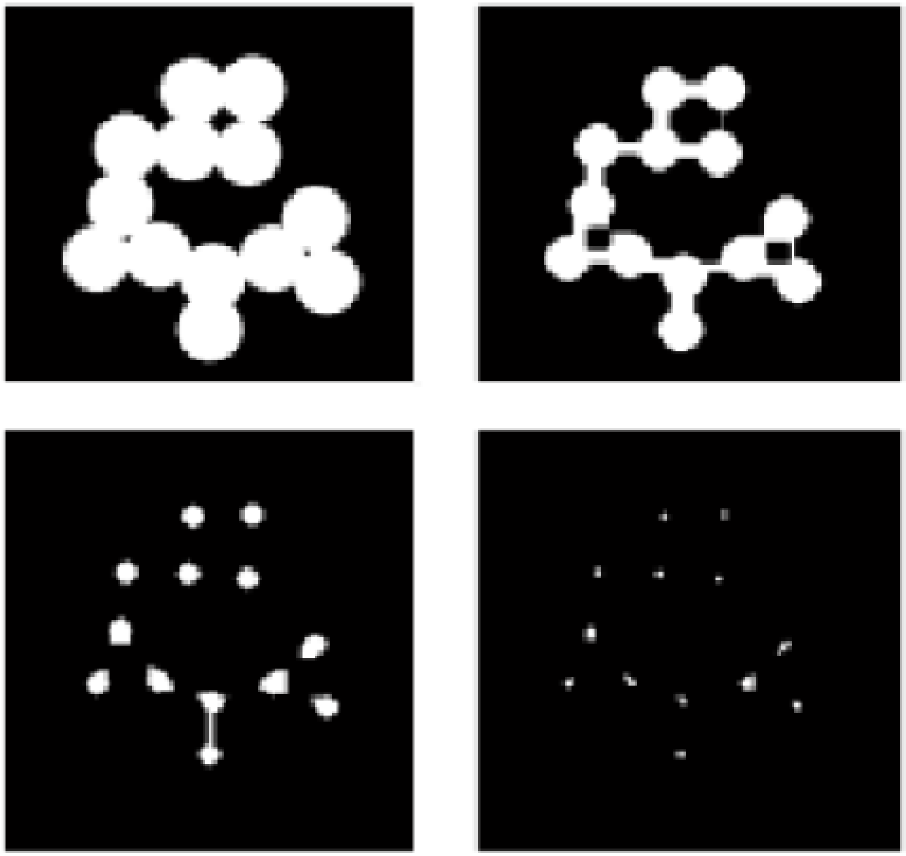
After erosion with a disc of radius 5

After erosion with a disc of radius 20

Erosion - Results

Original binary image *circles*

Erosion by 11x11 structuring element



Erosion by 21x21 structuring element

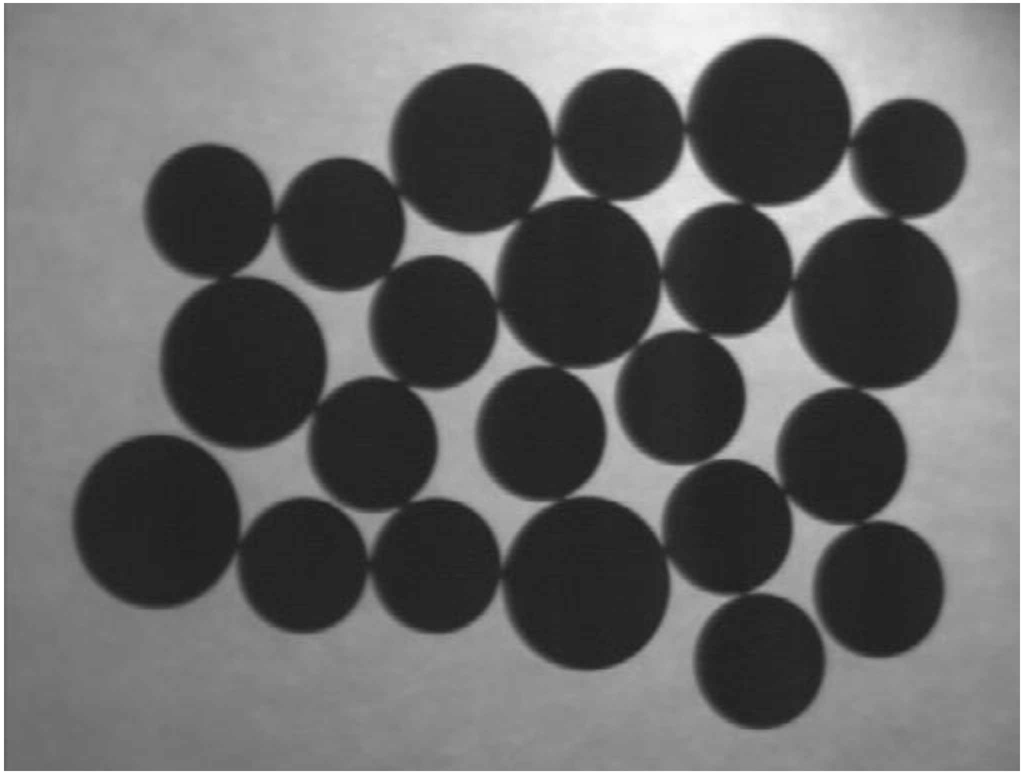
Erosion by 27x27 structuring element

Erosion – Exercise 1

$$
\begin{matrix}
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&1&1&1&0&0&0\\
0&0&0&1&1&1&1&1&1&1&0&0&0\\
0&0&0&1&1&1&1&1&1&1&0&0&0\\
0&0&0&1&1&1&1&1&1&1&0&0&0\\
0&0&0&1&1&1&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0
\end{matrix}
\;+\;
\begin{matrix}
1\\1\\1\\1\\1
\end{matrix}
\;=\;
\begin{matrix}
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0
\end{matrix}
$$

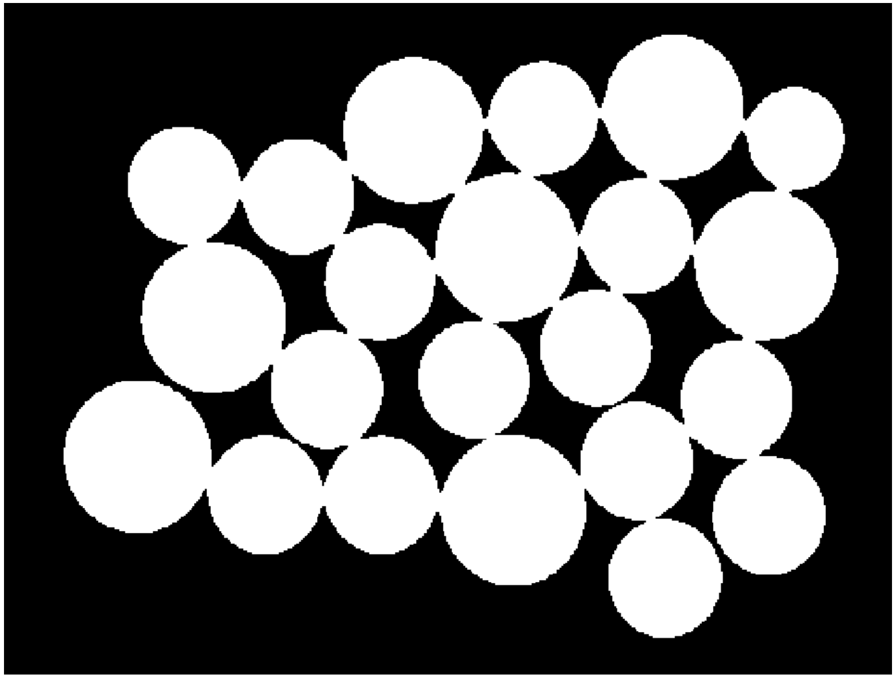
Erosion – Exercise 2

Using MATLAB, count the number or circles in the image below through erosion and object labeling.

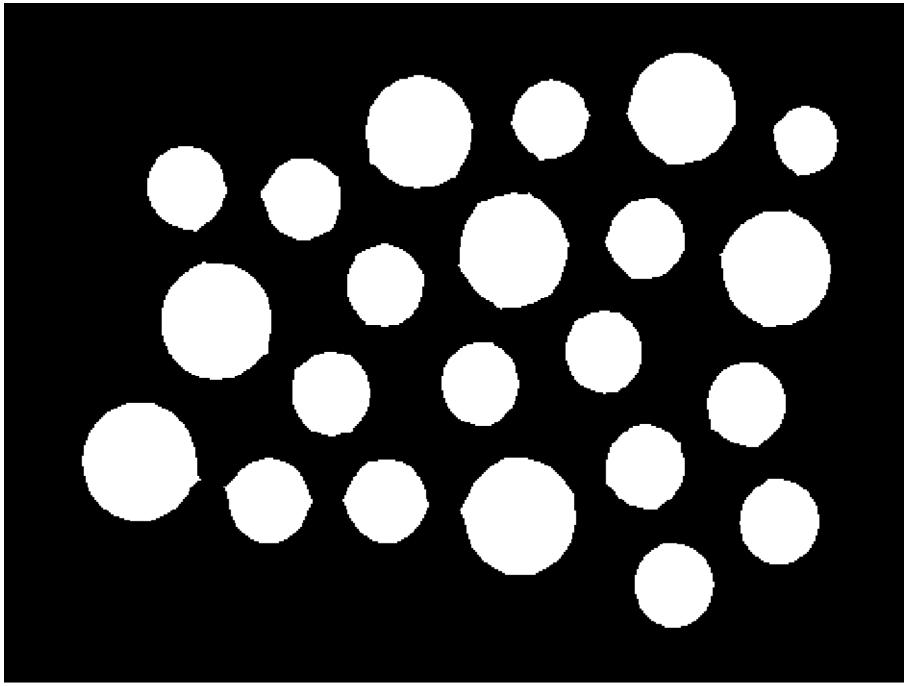


Erosion – Exercise 2 Results

Binarize the image



Perform Erosion



Use connected component labeling to count the number of coins

Dilation

Dilation of image *f* by structuring element *s* is given by

$f \oplus s$

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

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

Dilation - Steps

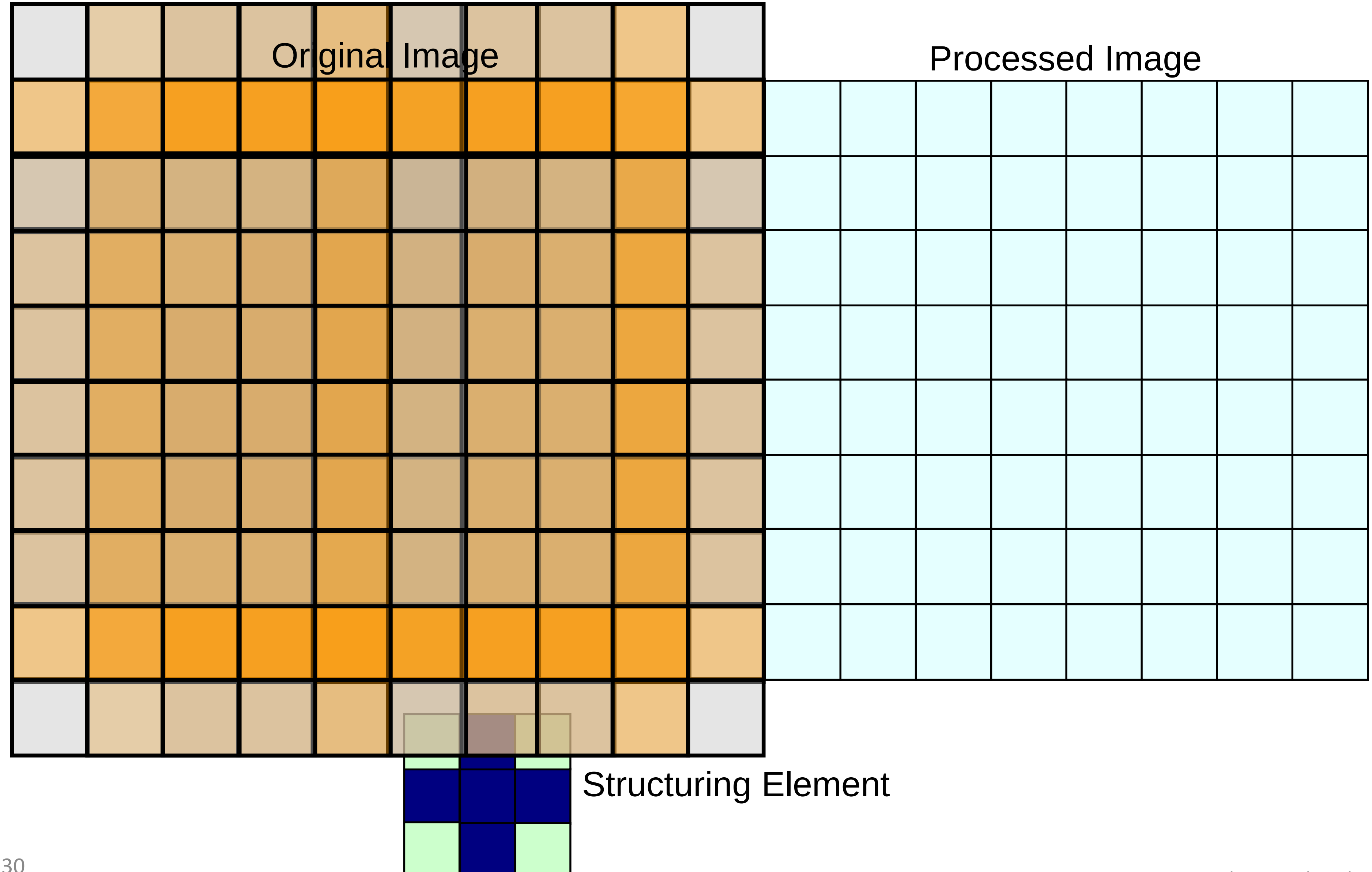
For each background pixel (which we will call the *input pixel*)

Superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position.

If *at least one* pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value.

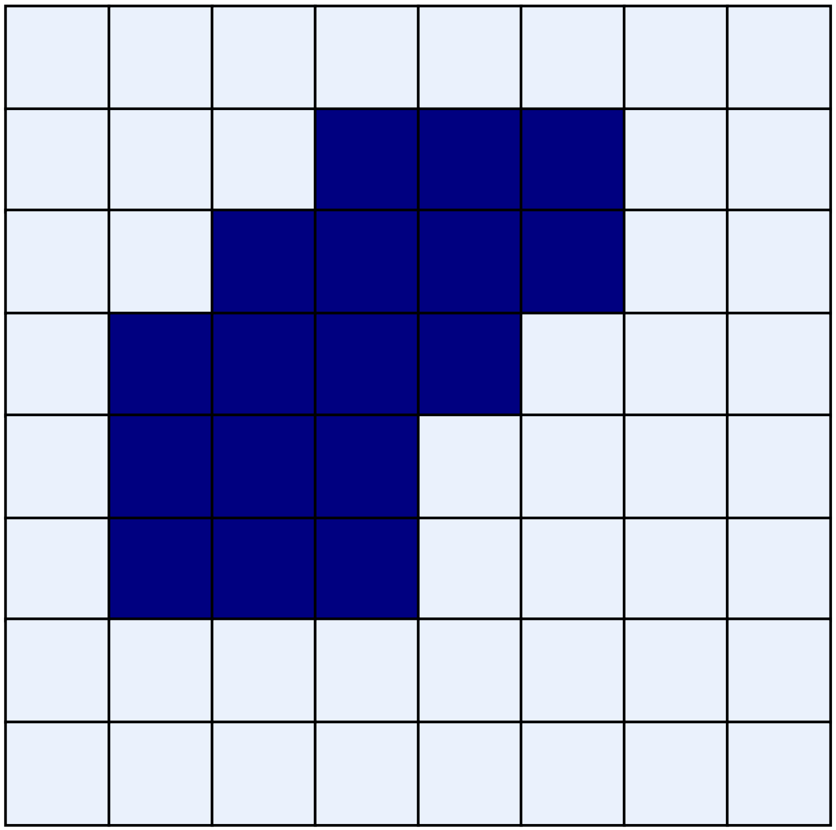
If all the corresponding pixels in the image are background, however, the input pixel is left at the background value.

Dilation - Example

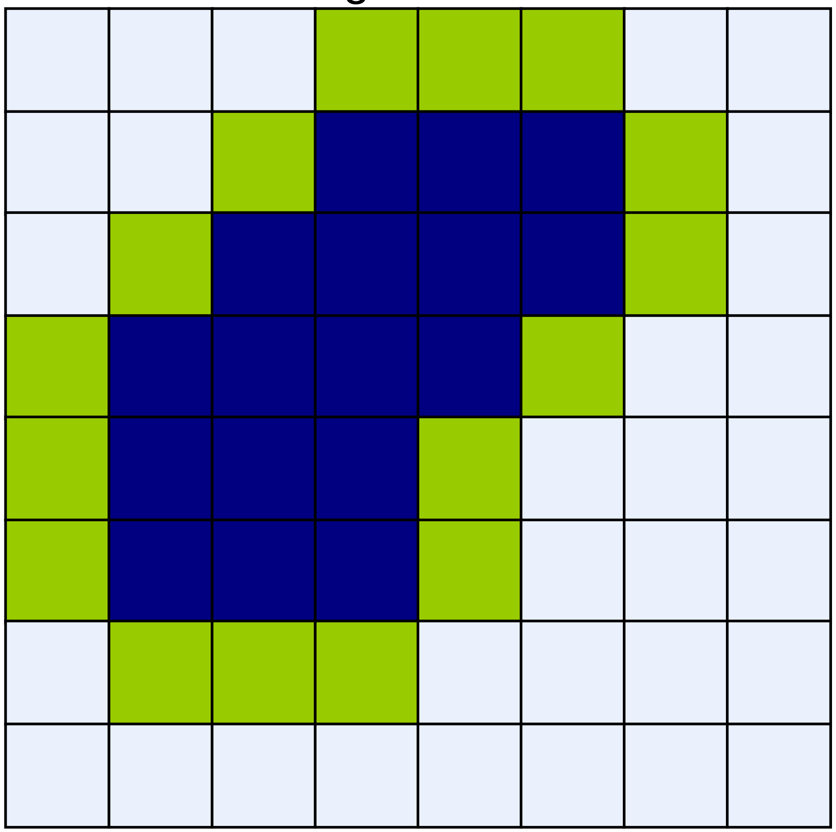


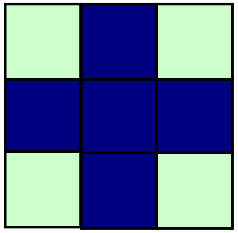
Dilation - Example

Original Image



Processed Image With Dilated Pixels





Structuring Element

Dilation - Basics

Effects

Expands the size of foreground (1-valued) objects.

Smoothes object boundaries.

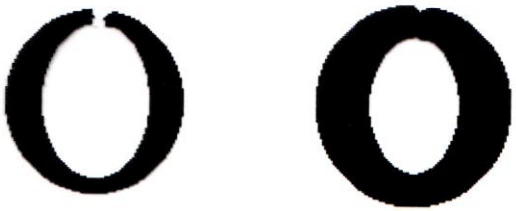
Closes holes and gaps.

Rule for Dilation

In a binary image, if any of the pixel (in the neighborhood defined by structuring element) is 1, then output is 1.

Dilation - Basics

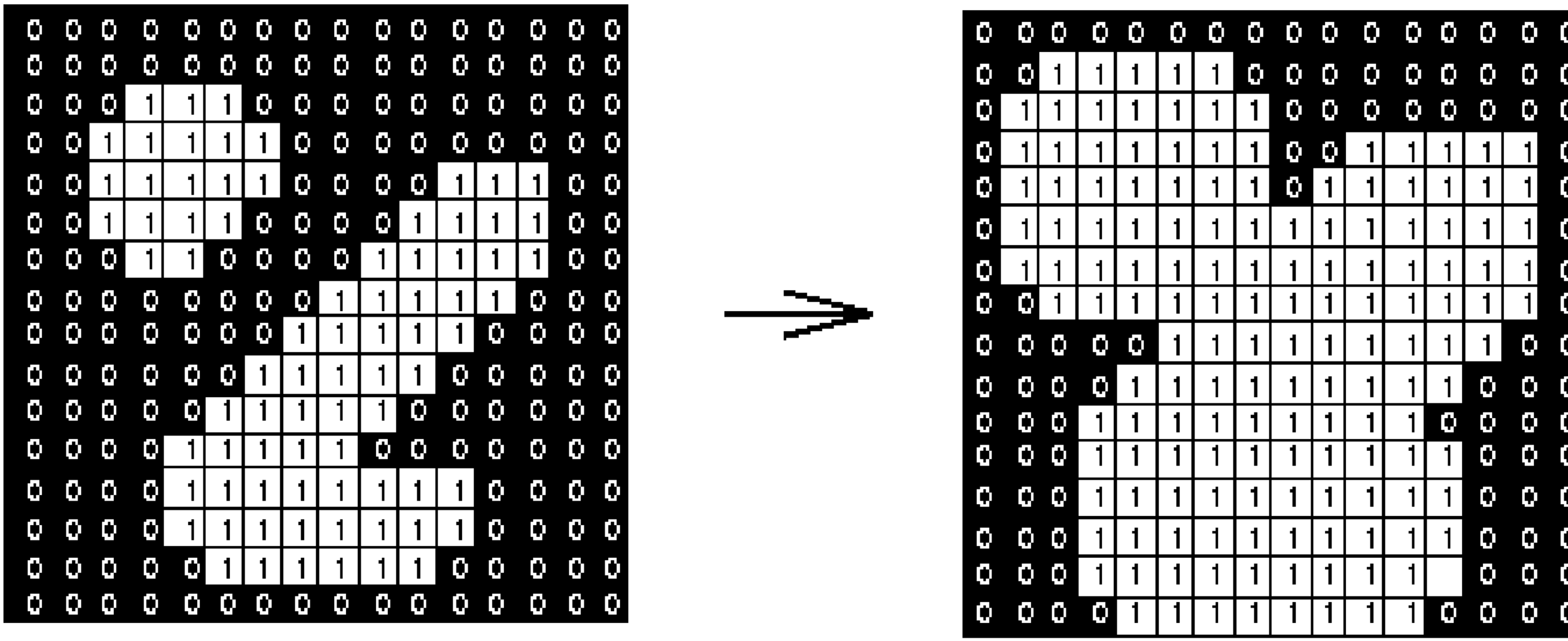
Dilation can repair breaks



Dilation can repair intrusions

Watch out: Dilation enlarges objects

Dilation – Example 1



Effect of dilation using a 3×3 square structuring element

Dilation – Example 2

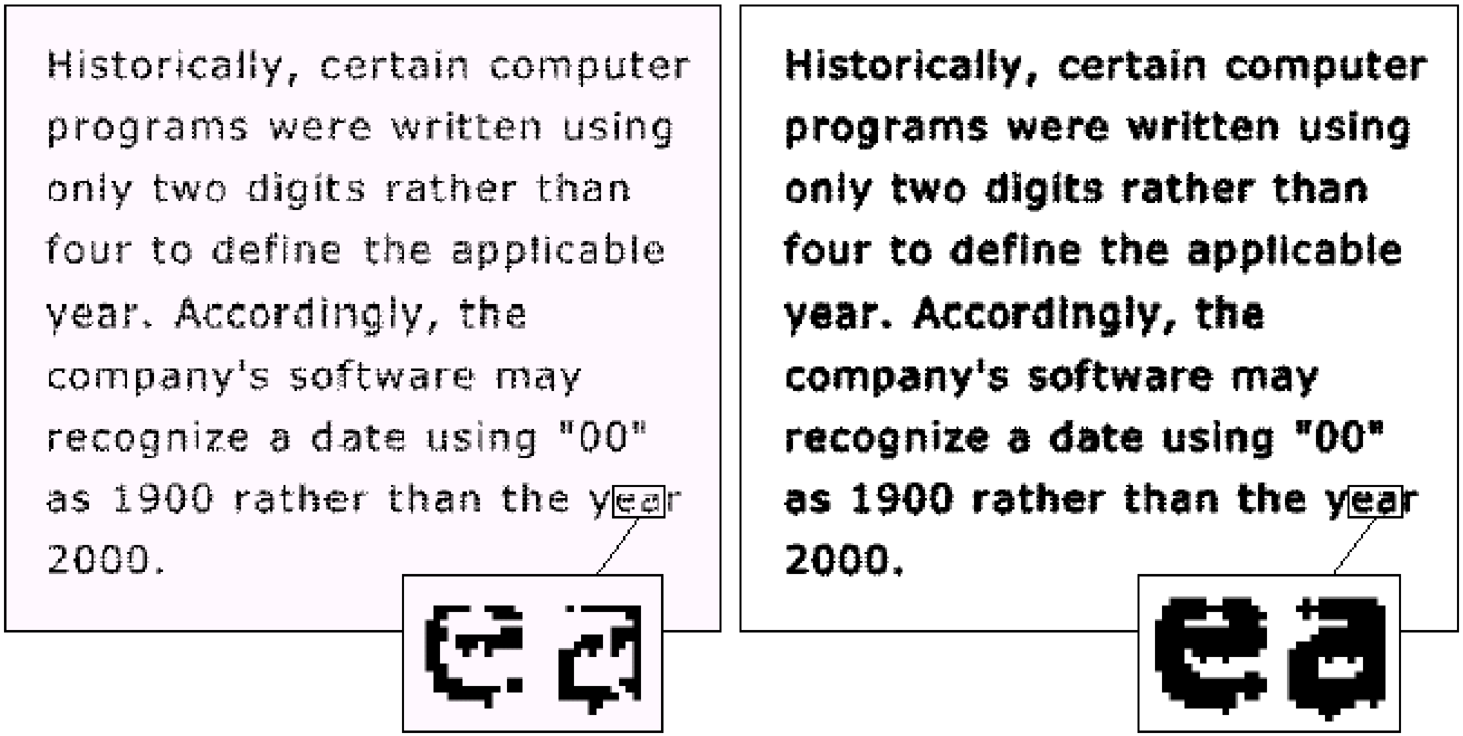
Original image

Dilation by 3*3 square structuring element

Dilation by 5*5 square structuring element

Note: In these examples a 1 refers to a black pixel!

Dilation – Example 3



0	1	0
1	1	1
0	1	0

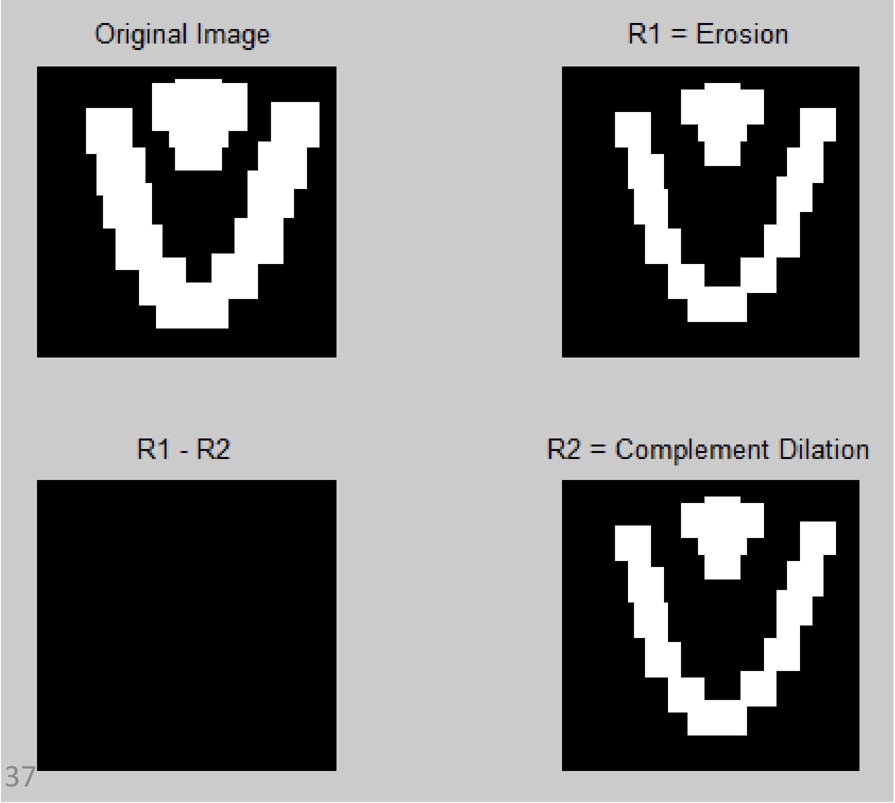
a c
b

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 & Dilation - Duality

Dilation and erosion are duals of each other:

It means that we can obtain erosion of an image A by B simply by dilating its background (i.e. A^c) with the same structuring element and complementing the result.




```
x = im2bw(rgb2gray(imread('a.bmp')));
se = ones(3,3);

r1 = imerode(x,se);
r2 = 1-imdilate((1-x),se);

figure;
subplot(2,2,1); imshow(x);  title('Original Image');
subplot(2,2,2); imshow(r1); title('R1 = Erosion');
subplot(2,2,4); imshow(r2); title('R2 = Complement Dilation');
subplot(2,2,3); imshow((r1-r2)); title('R1 - R2');
```

Compound Operations

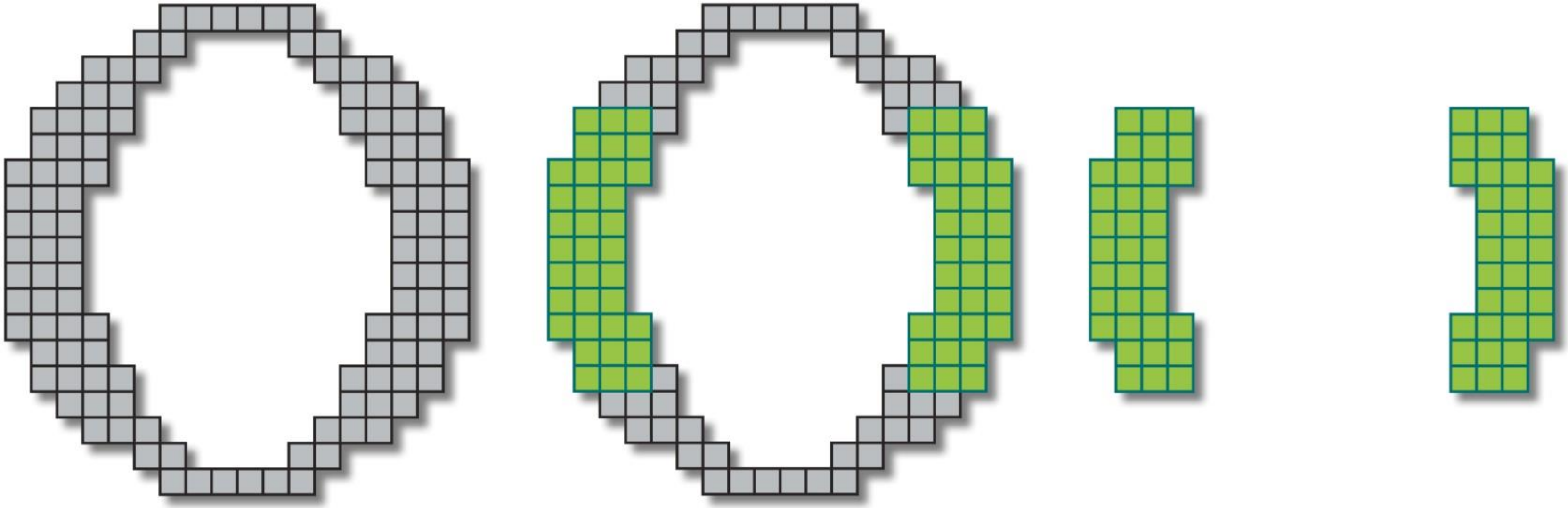
More interesting morphological operations can be performed by performing combinations of erosions and dilations

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

- Opening
- Closing

*Opening and Closing also hold the duality property.

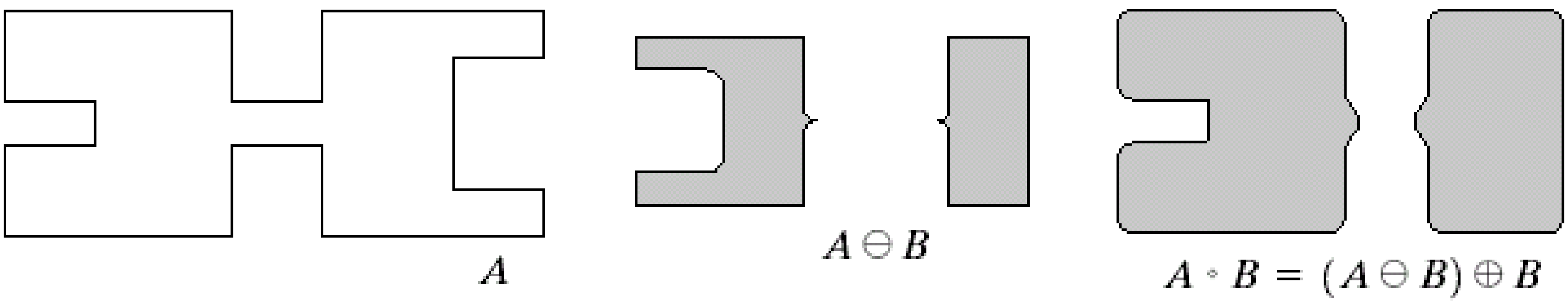
Opening



Opening

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

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



Original shape

After erosion

After dilation (opening)

Opening

Original Image

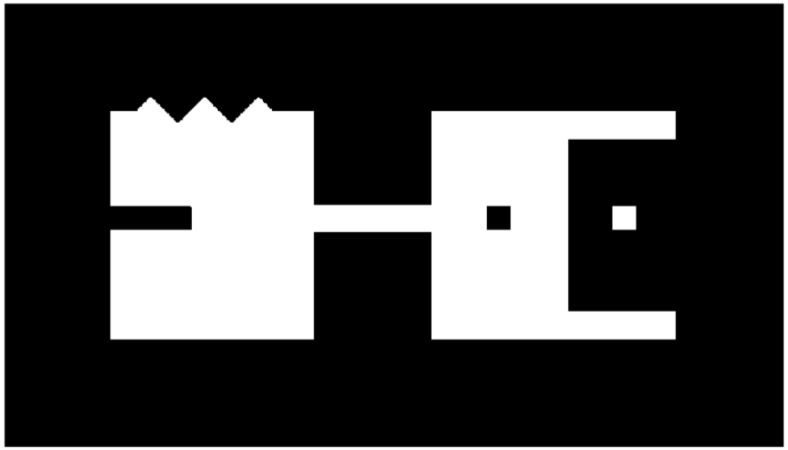


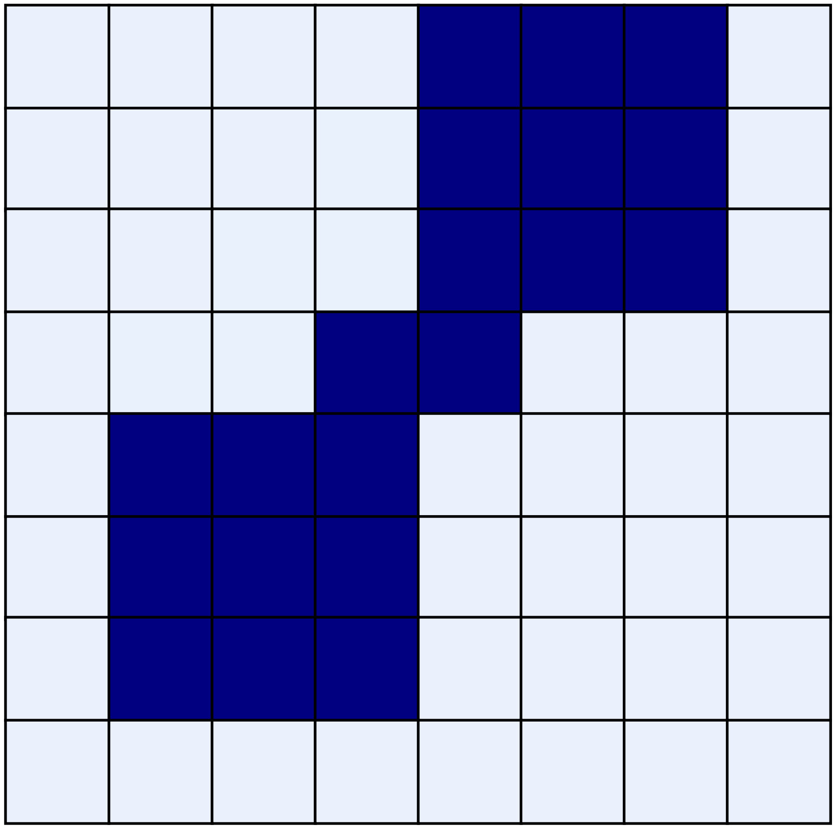
Image After Opening

Opening

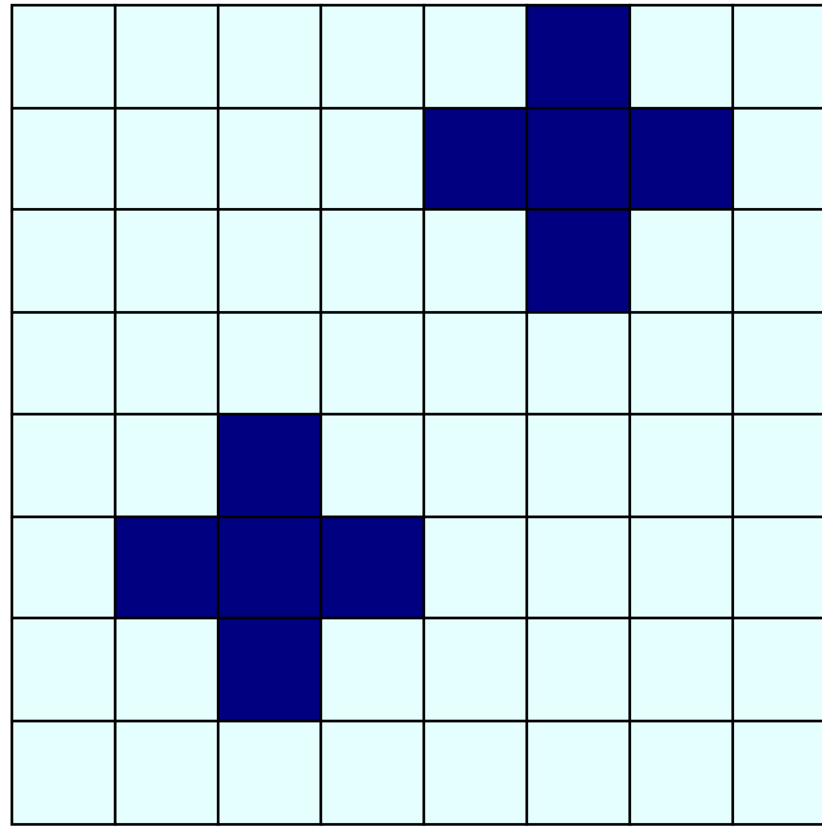
- Breaks narrow joints
- Removes ‘Salt’ noise

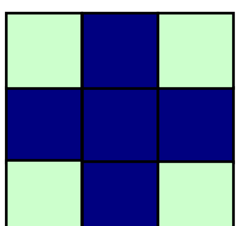
Opening

Original Image



Processed Image



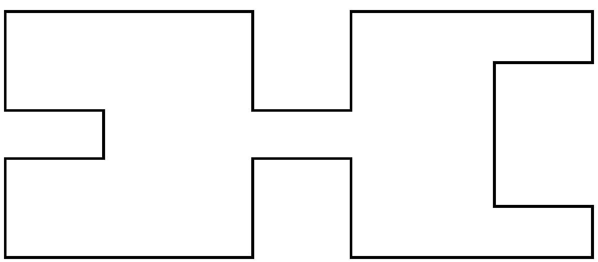


Structuring Element

Closing

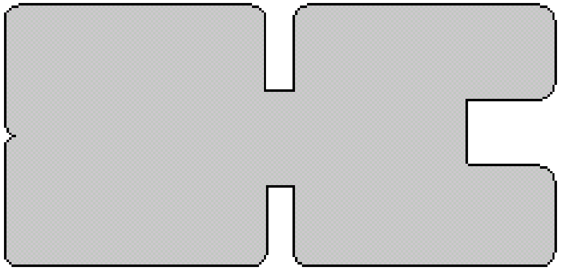
The closing of image *f* by structuring element *s*, denoted by $f \bullet s$ is simply a dilation followed by an erosion

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



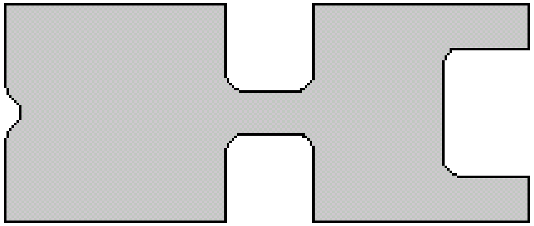
A

Original shape



$A \oplus B$

After dilation



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

After erosion (closing)

Closing

Original Image

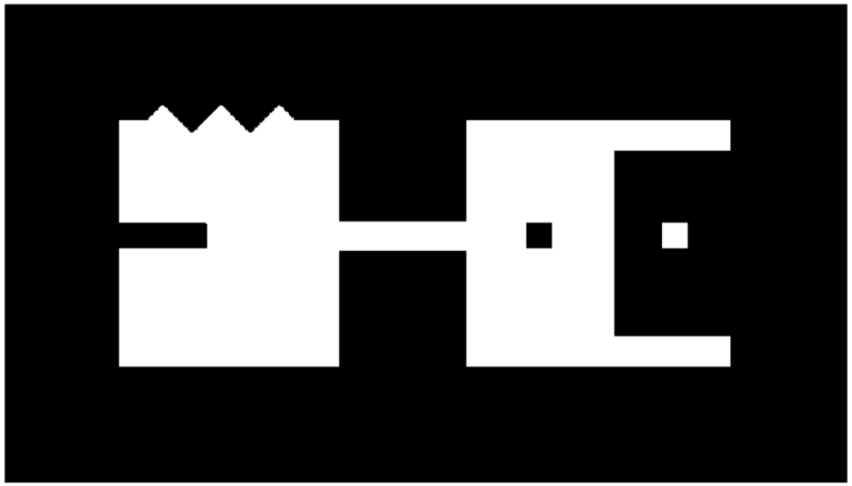


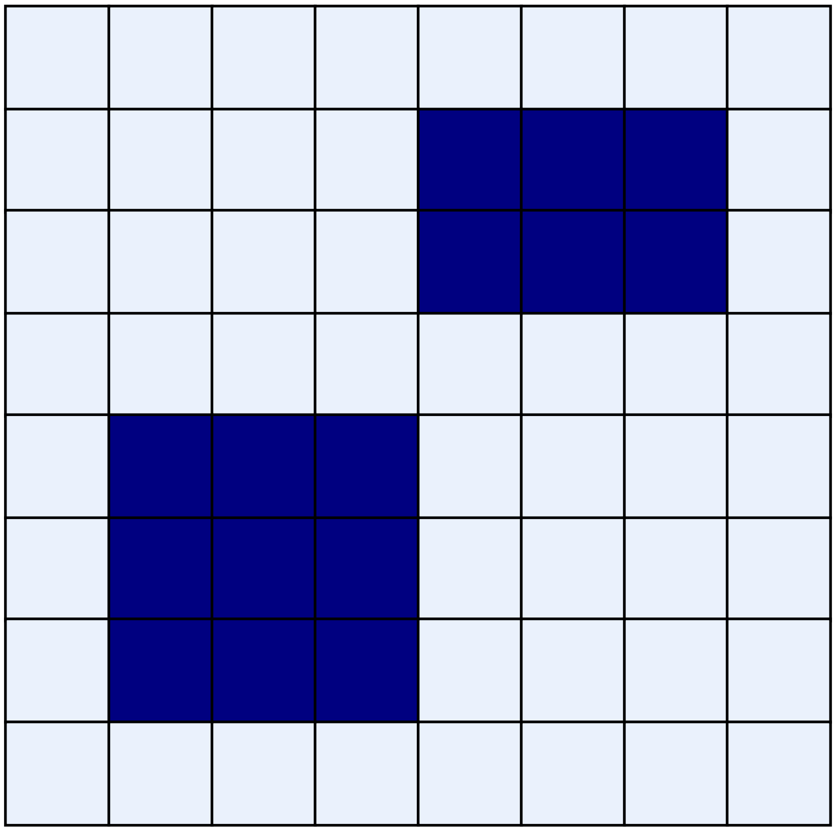
Image After Closing

Closing

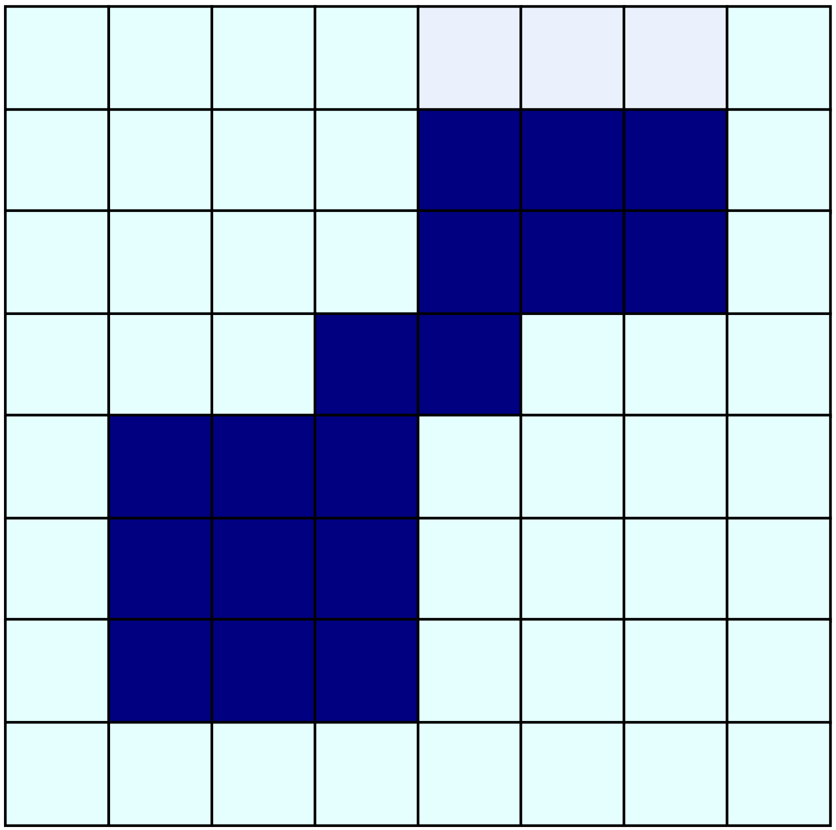
- Eliminates small holes
- Fills gaps
- Removes ‘Pepper’ noise

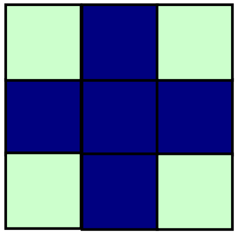
Closing

Original Image



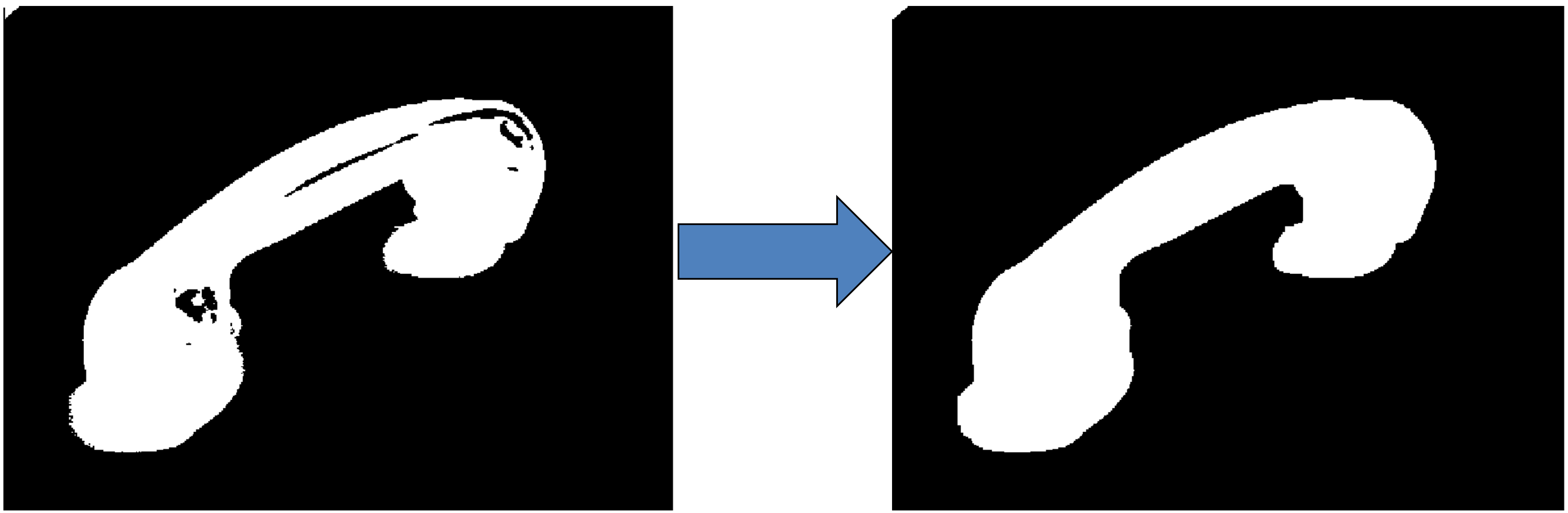
Processed Image



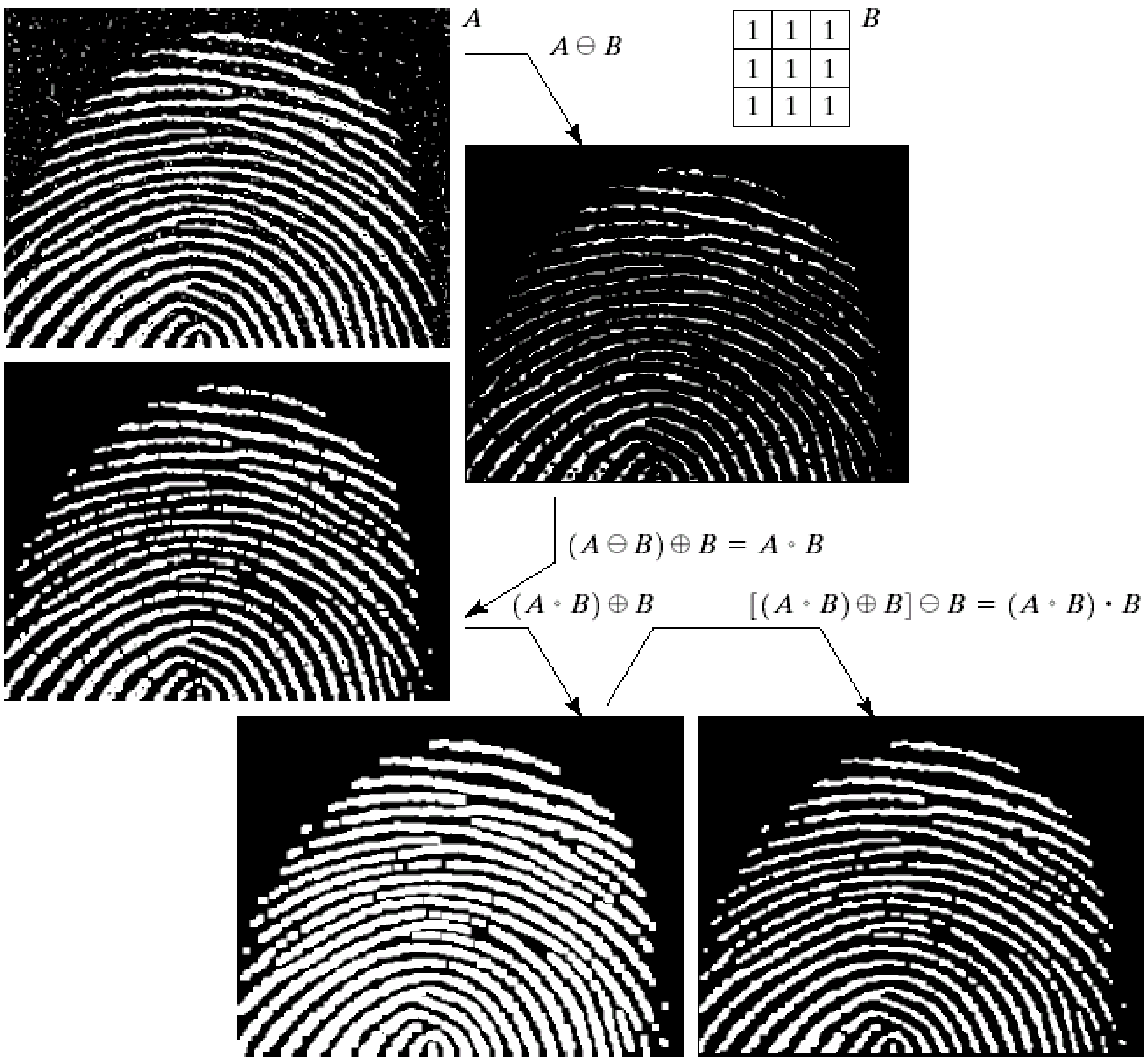


Structuring Element

Closing



Morphological Operations



a b
d c
e f

FIGURE 9.11
(a) Noisy image. (b) Structuring element. (c) Eroded image. (d) Opening of A. (e) Dilation of the opening. (f) Closing of the opening. (Original image courtesy of the National Institute of Standards and Technology.)

Hit-or-Miss Transform

A tool for shape detection or for the detection of a *disjoint region* in an image.

Basic Idea:

Suppose we have a binary image that contains certain shapes (circles, squares, lines, etc,….) called image A.

We use another image or matrix to search image A for a particular pattern of bits. We will call this pattern “shape B”.

We then search image A for shape B.

Whenever there is a ‘hit’, we indicate where the center of shape B was on image A.

Hit-or-Miss Transform

- A tool for shape detection or for the detection of a *disjoint region* in an image
- Idea
  - Sup
    (cir
  - We
    particular pattern of bits. We will call this pattern “shape B”
  - We then search image A for shape B
  - Whenever there is a ‘hit’, we indicate where the center of shape B was on image A.

Watch out: We actually look for ‘*fit*’ but we will be calling them ‘*hit*’ when talking about hit-or-miss transform

Hit-or-Miss Transform

Structuring Element

So far we have considered the SEs where 0s are treated as Don’t Cares i.e. we focus on the 1s only

	1	
1	**1**	1
	1	

0	1	0
1	**1**	1
0	1	0

Hit-or-Miss Transform

Extended Structuring Element

Now we will distinguish between the 0s and the Don’t cares

1	1	1
×	0	×
×	0	×

E.g. For a ‘fit’ the 0s of SE should match with 0s of the underlying image

Hit-or-Miss Transform

Extended Structuring Element: Example

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

$\circledast$

1	1	1
×	0	×
×	0	×

Erosion Recap: Slide the SE on the image and look for the 'fits'

Hit-or-Miss Transform

Extended Structuring Element: Example

0	0	0	0	0	0	0	0
0	0	**1**	**1**	**1**	0	0	0
0	0	0	**0**	1	1	1	0
0	1	0	**0**	1	1	1	1
0	1	1	1	1	1	0	1
0	0	1	1	1	1	0	1

‘Fit’ encountered

$\circledast$

1	1	1
×	0	×
×	0	×

Hit-or-Miss Transform

Extended Structuring Element: Example

0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0
0	1	0	0	1	**1**	**1**	**1**
0	1	1	1	1	1	**0**	1
0	0	1	1	1	1	**0**	1

$\circledast$

1	1	1
×	0	×
×	0	×

‘Fit’ encountered

Hit-or-Miss Transform

Extended Structuring Element: Example

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

$\circledast$

1	1	1
×	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	**1**	0
0	0	0	0	0	0	0	0

What we actually did???

Hit-or-Miss Transform

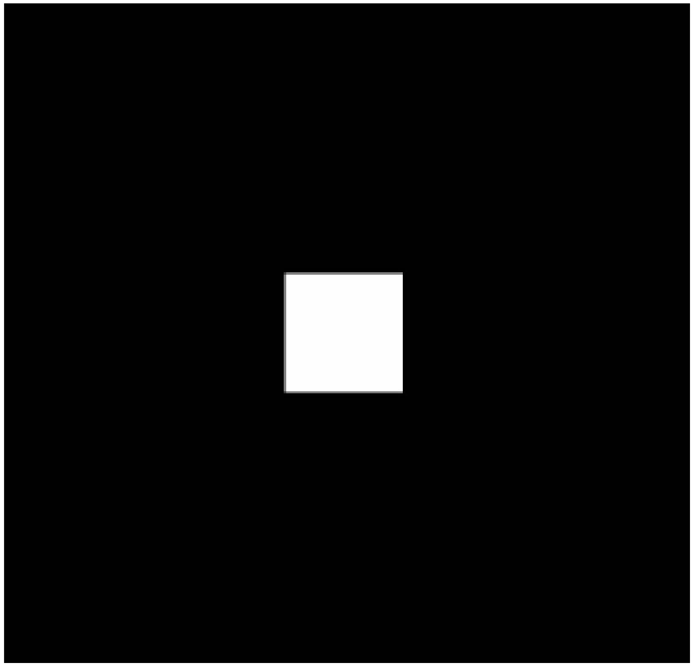
We have searched the pattern in the structuring element in the image.

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	**1**	0
0	0	0	0	0	0	0	0

Output: The center of the pattern is 1 and rest everything is 0

Hit-or-Miss Transform

Example: Search a 100x100 pixel square in an image



How do we search?

Take an image of size 100x100 (B) representing a white square

We search the pattern in the input image (A)

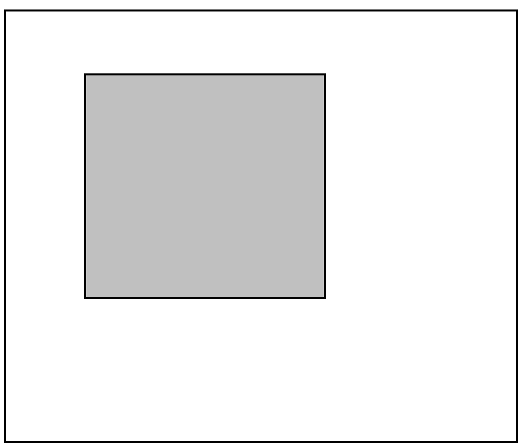
If found, we have a "fit". We mark the center of the "fit" with a white pixel

In the above example, there would be only 1 fit

Hit-or-Miss Transform

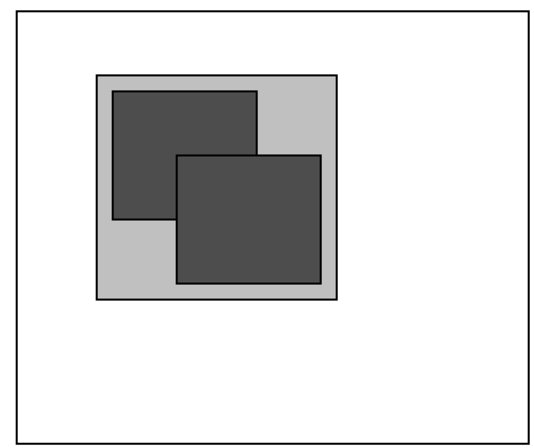
Do you find any problem with this?

If we search for a 100x100 pixel square in an image we will have a positive response for all squares greater than 100x100 as well



Image

Shape to pattern



The pattern will fit at different places

Morphological Operations

Algorithms

Morphological Algorithms

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

For example:

- Morphological gradients
- Region filling
- Extraction of connected components
- Thinning/thickening
- Convex Hull

Morphological Gradients

$$B(A) = A - A \ominus B$$ Internal Gradient

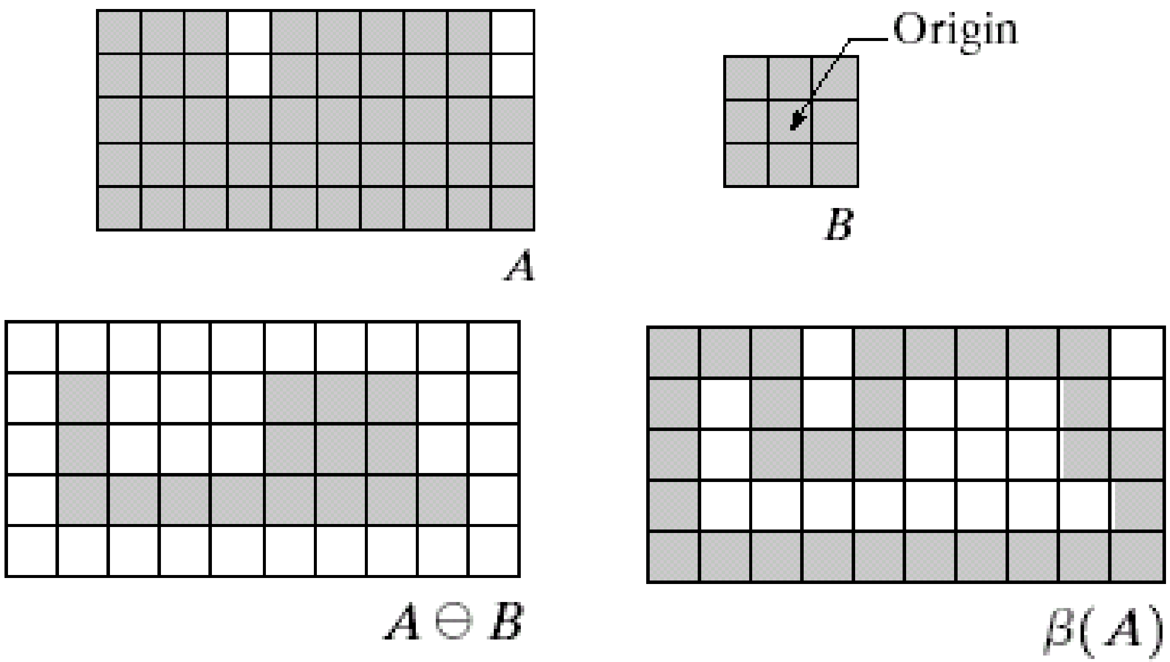
$$B(A) = A \oplus B - A$$ External Gradient

$$B(A) = A \oplus B - A \ominus B$$ Morphological Gradient

The boundary of set *A* denoted by *β(A)* is obtained by first eroding *A* by a suitable structuring element *B* and then taking the difference between *A* and its erosion.

Boundary Extraction

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



Boundary Extraction

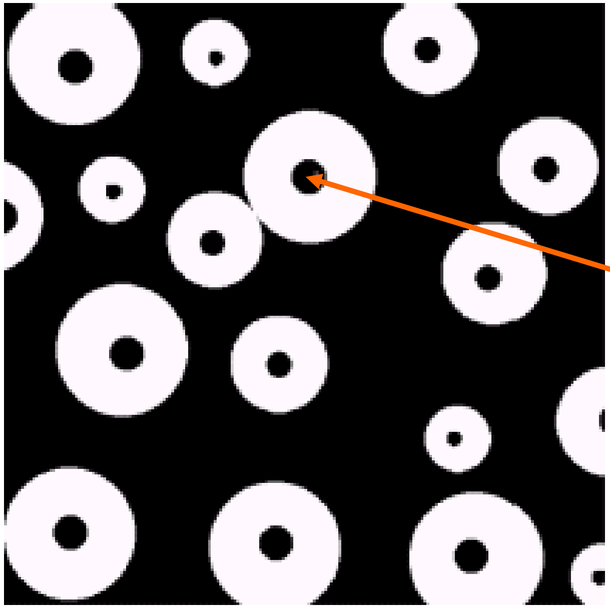
A simple image and the result of performing boundary extraction using a square 3*3 structuring element

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

Let A is a set containing a subset whose elements are 8-connected boundary points of a region, enclosing a background region i.e. hole

If all boundary points are labeled 1 and non boundary points are labeled 0, the following procedure fills the region:

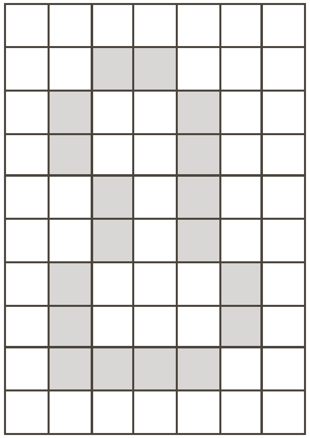
Inside the boundary

- Start from a known point p and taking $X_0 = p$,
- Then taking the next values of X_k as:

$$X_k = (X_k \oplus B) \bigcap A^c \qquad k = 1, 2, 3, \cdots$$

B is suitable structuring element

- Terminate iterations if $X_{k+1} = X_k$
- The set union of X_k and A contains the filled set and its boundaries.



A

Region Filling

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

A

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

A^c

0	1	0
1	1	1
0	1	0

B

Region Filling

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

X_0

0	1	0
1	1	1
0	1	0

B

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

$(X_0 \oplus B)$

$$X_k = (X_k \oplus B) \bigcap A^c$$

$$k = 1, 2, 3, \cdots$$

Region Filling

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

$X_1 = (X_0 \oplus B)$

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

A^c

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

$X_1 = (X_0 \oplus B) \bigcap A^c$

$$X_k = (X_k \oplus B) \bigcap A^c$$
$$k = 1, 2, 3, \cdots$$

Region Filling

$$X_k = (X_k \oplus B) \bigcap A^c$$
$$k = 1, 2, 3, \cdots$$

NOTE:
The intersection of dilation and the complement of A limits the result to inside the region of interest

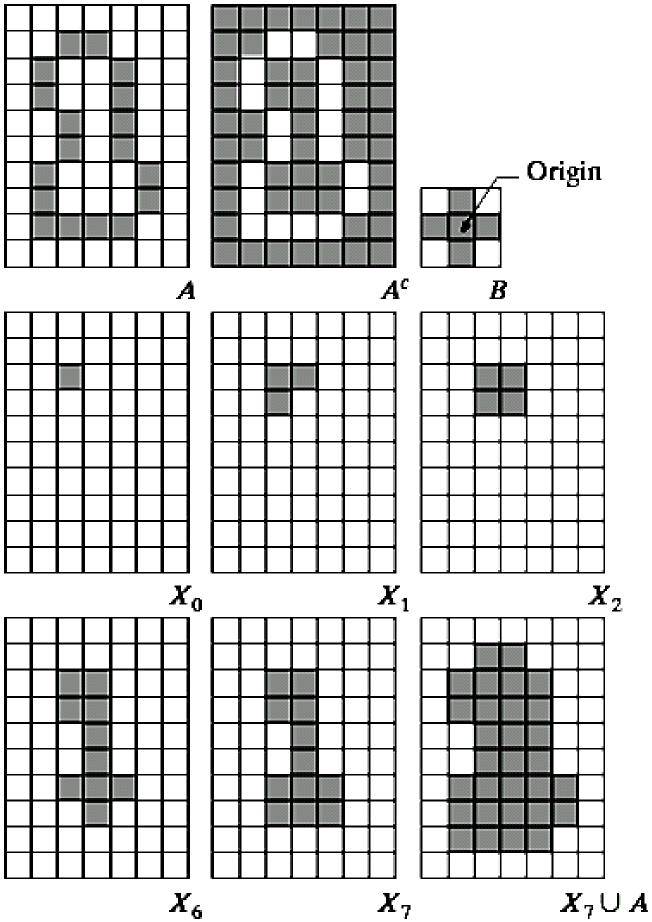
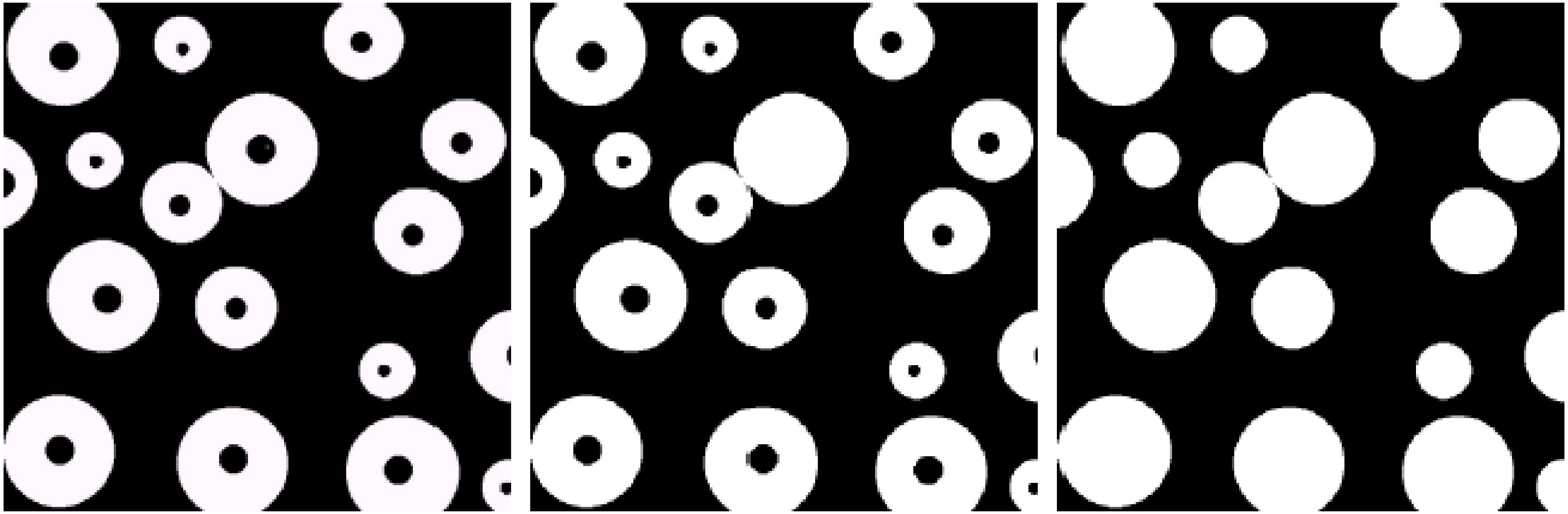


FIGURE 9.15 Hole filling. (a) Set A (shown shaded). (b) Complement of A. (c) Structuring element B. (d) Initial point inside the boundary. (e)–(h) Various steps of Eq. (9.5-2). (i) Final result [union of (a) and (h)].

Region Filling



Original Image | One Region Filled | All Regions Filled

Extraction of Connected Components

Let Y represents a connected component contained in *A* and the point *p* of the Y is known.

The following procedure iteratively finds all the elements of Y:

- Start from a known point p and taking $X_0 = p$,
- Then taking the next values of X_k as:

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

  B is a suitable structuring element
- Algorithm converges if $X_k = X_{k-1}$
- The component Y is given as $Y = X_k$

Extraction of Connected Components

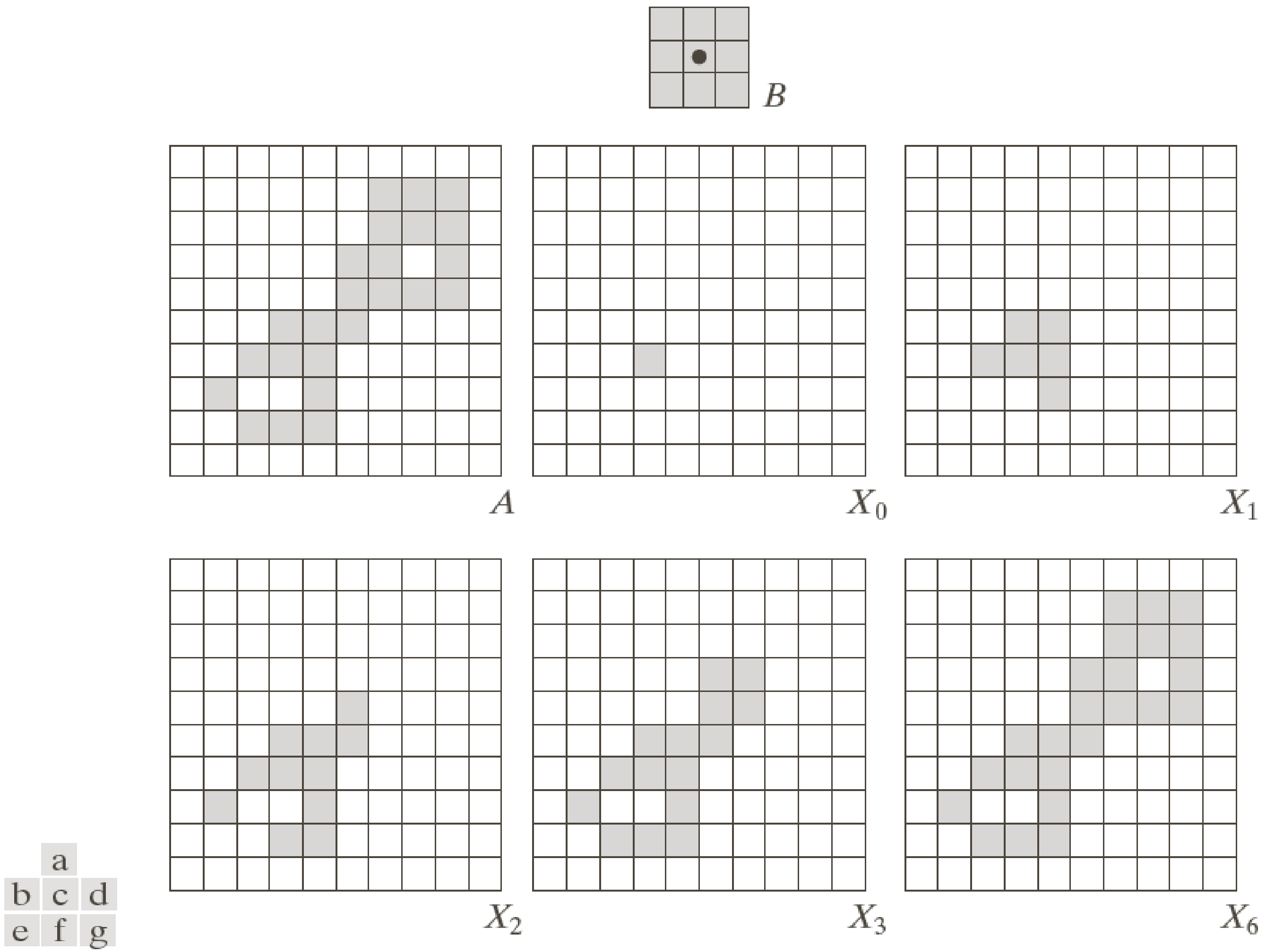


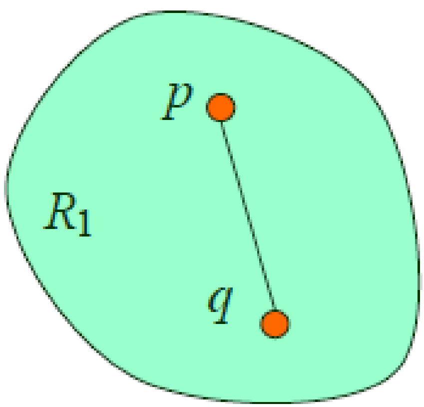
FIGURE 9.17 Extracting connected components. (a) Structuring element. (b) Array containing a set with one connected component. (c) Initial array containing a 1 in the region of the connected component. (d)–(g) Various steps in the iteration of Eq. (9.5-3).

Convex Hull

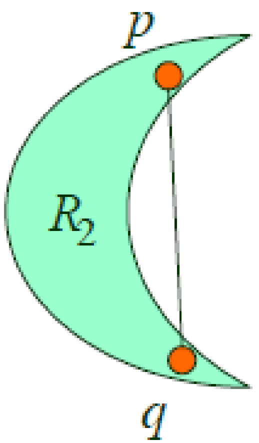
- Convex set

  A set *A* is said to be convex if any straight line segment joining two points of *A* lies within *A*

- Example: R_1 is convex as line segment *pq* lies within set R_1



Convex



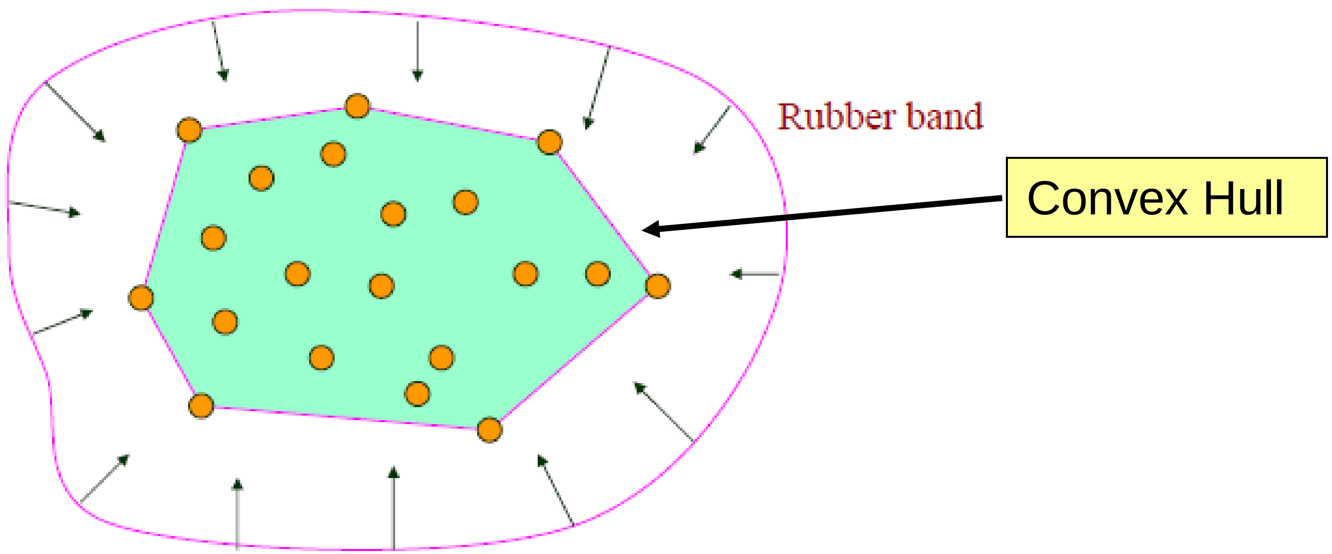
Concave

Convex Hull

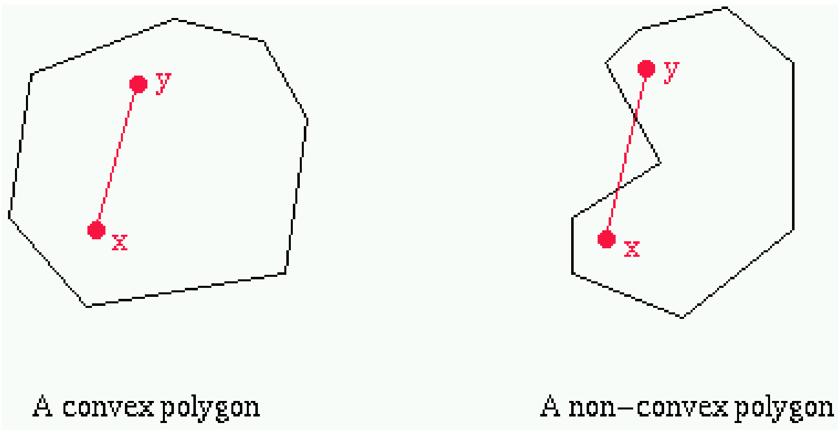
- Convex Hull

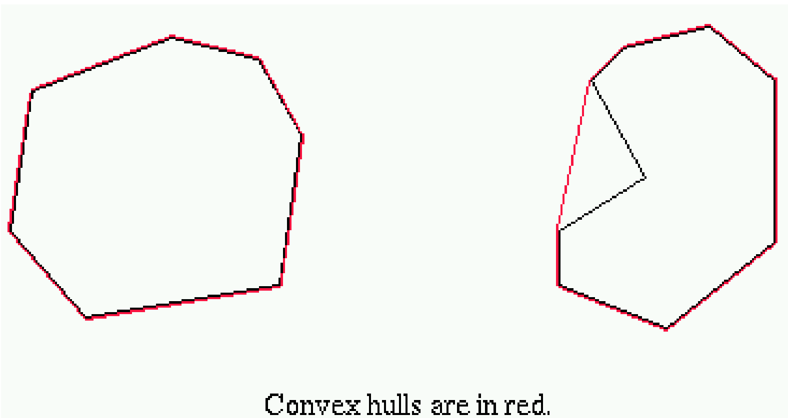
  Convex hull H of a set S is the smallest convex set containing S

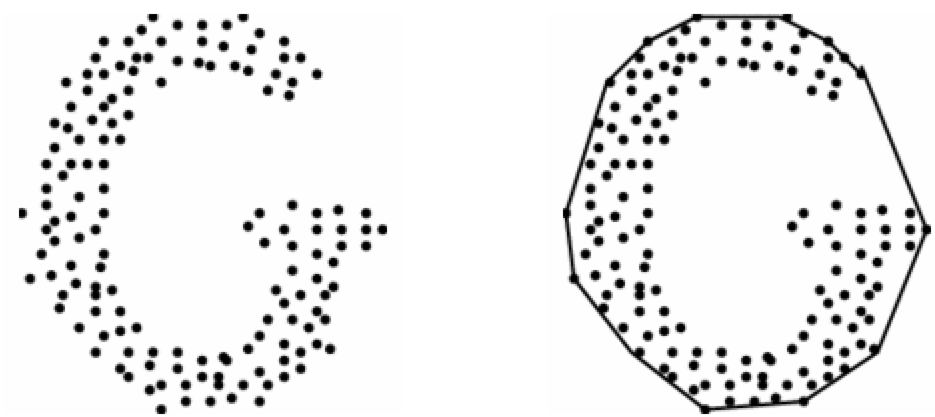
- Rubber band example:



Convex Hull


y
x
A convex polygon
y
x
A non−convex polygon


Convex hulls are in red.



Convex Hull

- To find the Convex Hull *C(A)* of a set *A* the following simple morphological algorithm can be used:

- Let B^i, where $i = 1, 2, 3, 4$, represent four structuring elements
- Implement:

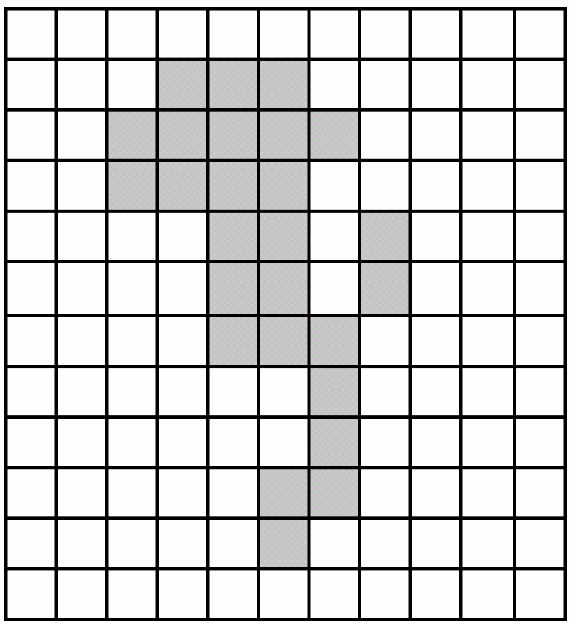
$$\boxed{X^i_k = (X^i_{k-1} \circledast B^i) \cup A \quad i = 1,2,3,4 \quad and \quad k = 1,2,3,\cdots}$$

- Starting with: $X^i_0 = A$
- Repeat 2nd step until convergence, i.e. $D^i = X^i_{conv} \to X^i_k = X^i_{k+1}$
- Convex Hull *C(A)* is given by:

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

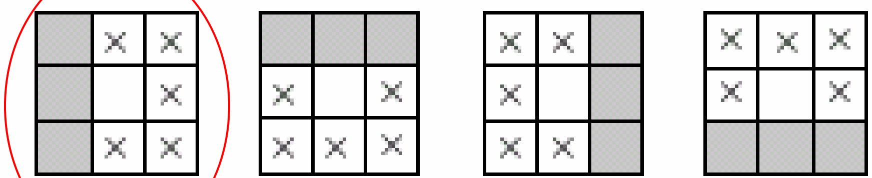
Convex Hull

Pick the first SE



A

Start At:



B^1 B^2 B^3 B^4

$$X_0^1 = A$$

Find:

$$X_1^1 = (X_0^1 \circledast B^1) \cup A$$

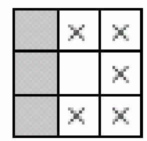
$$X_2^1 = (X_1^1 \circledast B^1) \cup A$$

$$\vdots$$

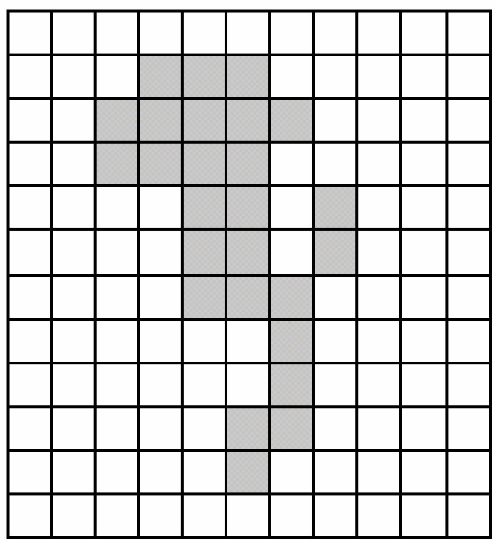
Until Convergence $X_k^1 = (X_{k-1} \circledast B^1) \cup A$

Convex Hull

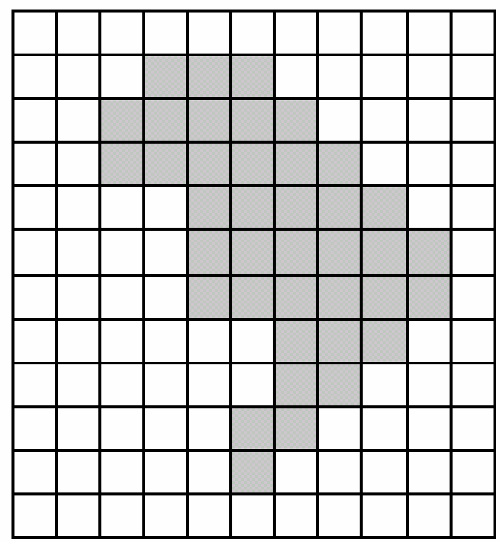
$$X^1_k = (X_{k-1} \circledast B^1) \cup A$$



B^1



$X_0^1 = A$



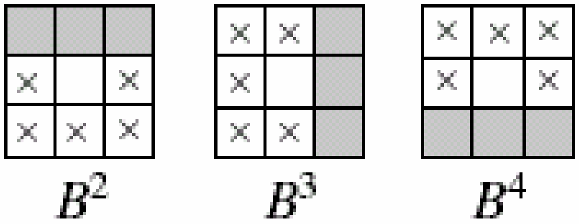
X_4^1

Call it D^1

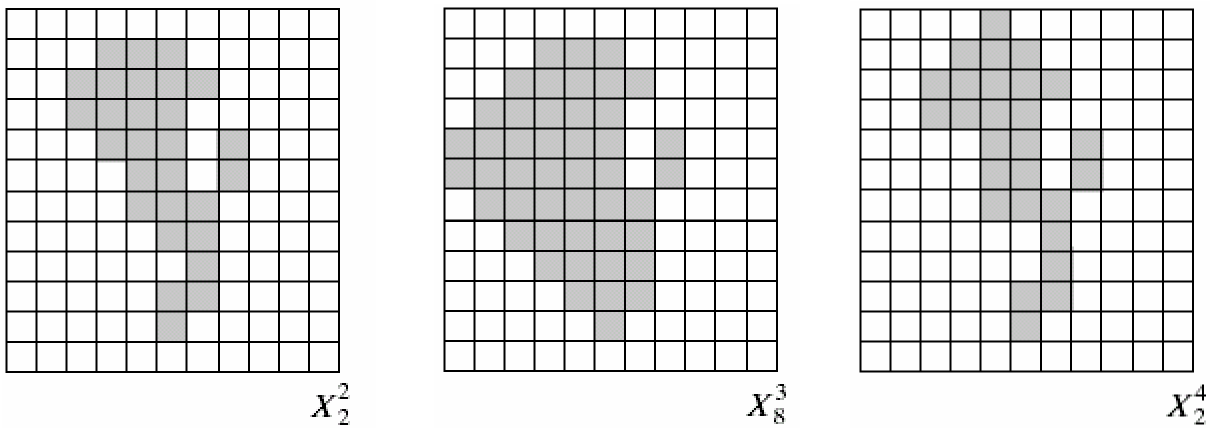
Convergence after four iterations

Convex Hull

Repeat the same process for B^2, B^3 and B^4



$$X^i_k = (X_{k-1} \circledast B^i) \cup A \quad i = 2,3,4$$

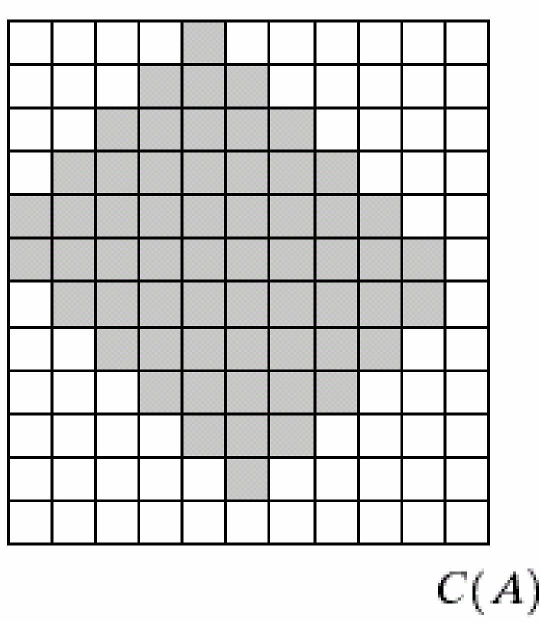


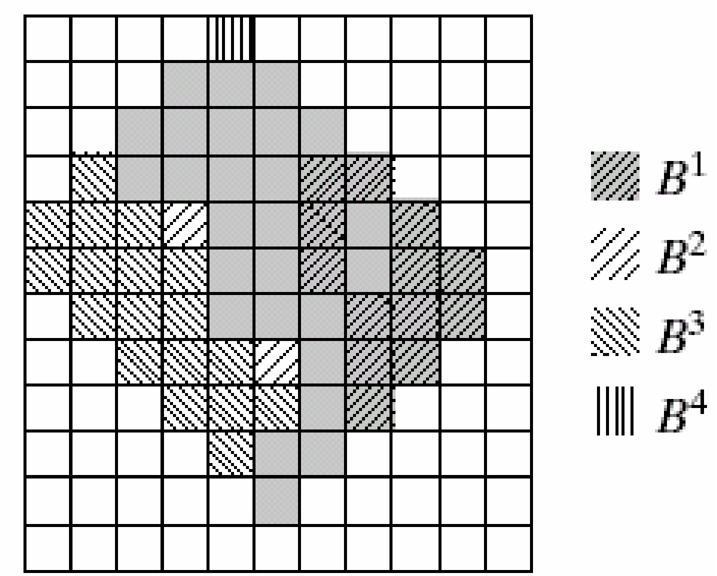
D2 D3 D4

Convex Hull

Take the union of all D^i to get the convex hull of A

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





End

Morphological Operations