Binary Image Analysis II

5. Binary Image Morphology

- **Definition ---** The word *morphology* 形態學 refers to form and structure; in computer vision it can be used to refer to the shape of a region.
  - The operations of *mathematical morphology* were originally defined as set operations and shown to be useful for processing sets of 2D points.
  - In computer vision, we define the operations of binary morphology and show how they can be useful in processing the regions derived from the connected components labeling operation.

Structuring Elements (SE)

- The structuring element represents a shape; it can be of any size and have arbitrary structure that can be represented by a binary image

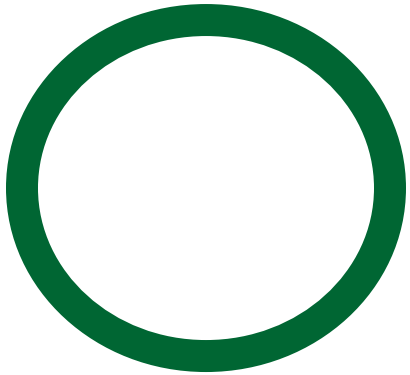
- **Examples** - Some common structuring elements

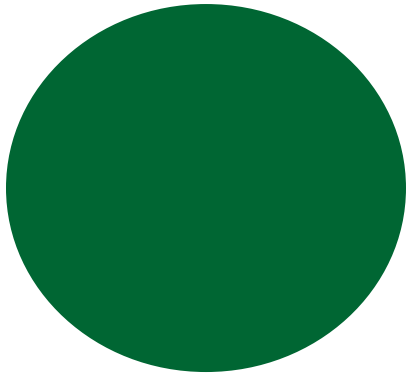
0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
0	1	1	1	0

RING(5)

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

DISK (5)





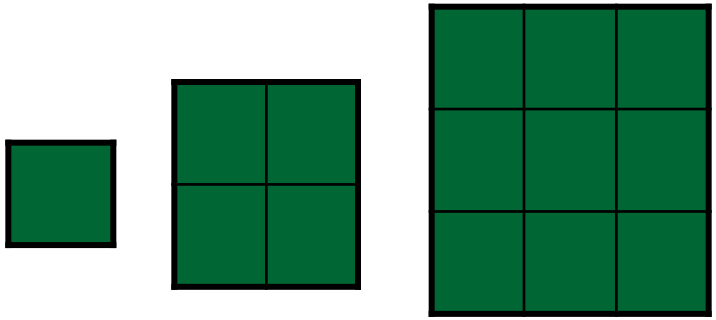
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

BOX (3,5)

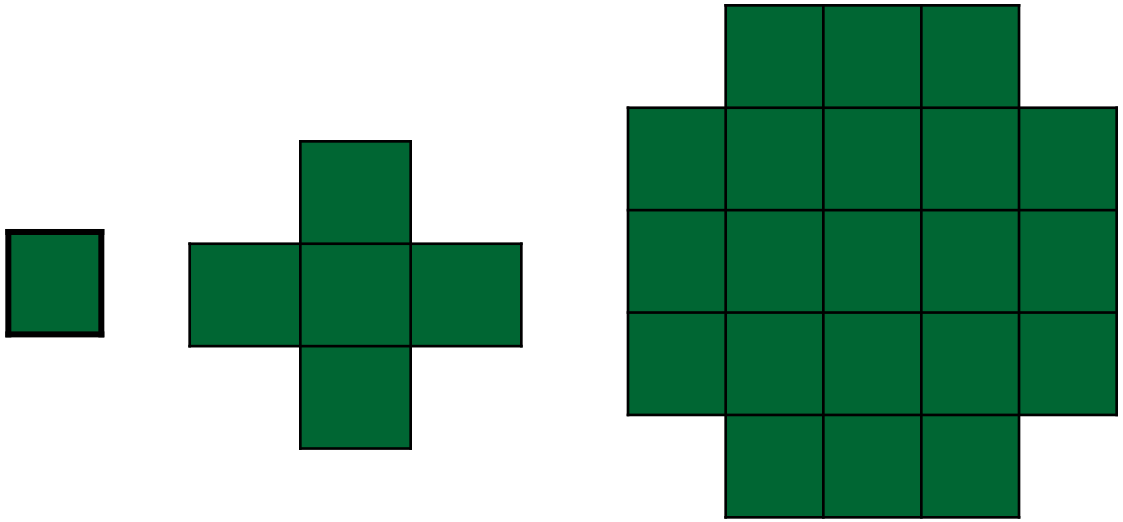
1	1		
1	1		
1	1	1	1

1	1	1	1	1	1
1		1	1		1
1		1	1		1

1
1
1
1



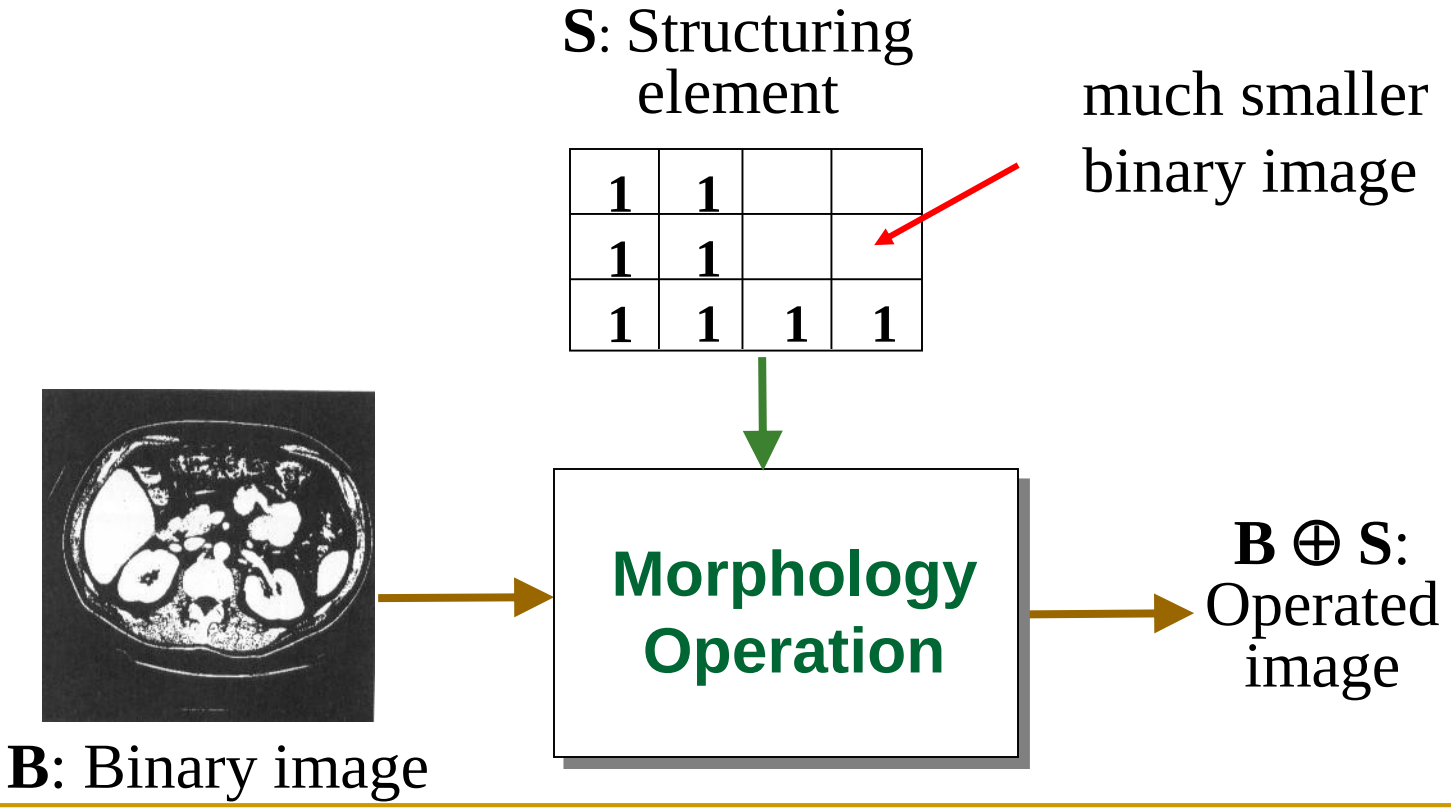
***SE*: rectangle with R=1,2,3**



***SE*: circle with**

R=1,3,5

- **Operations** - The operations of binary morphology input a binary image **B** and a *structuring element* ***S***, which is another, usually much smaller, binary image

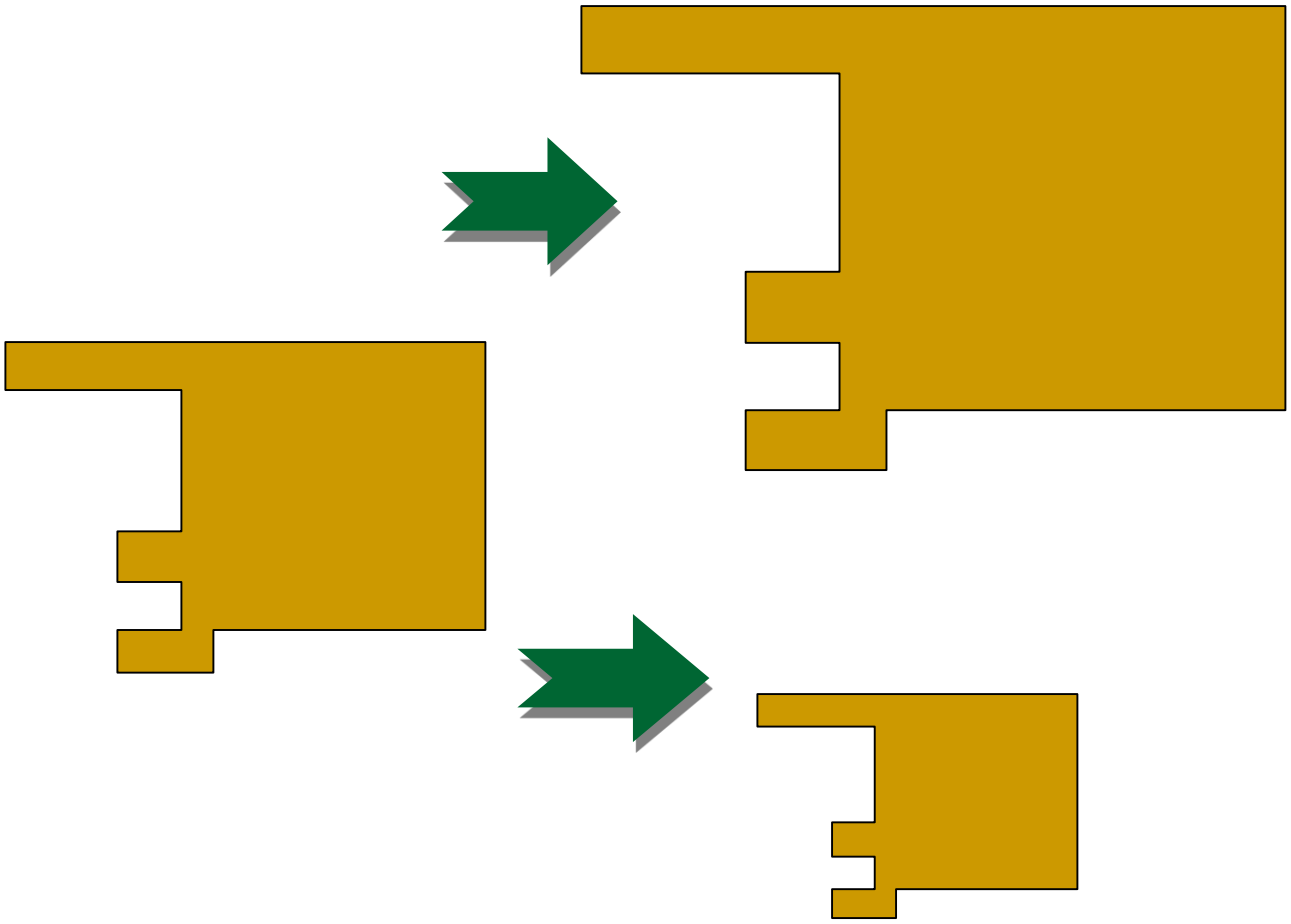


Basic Operations

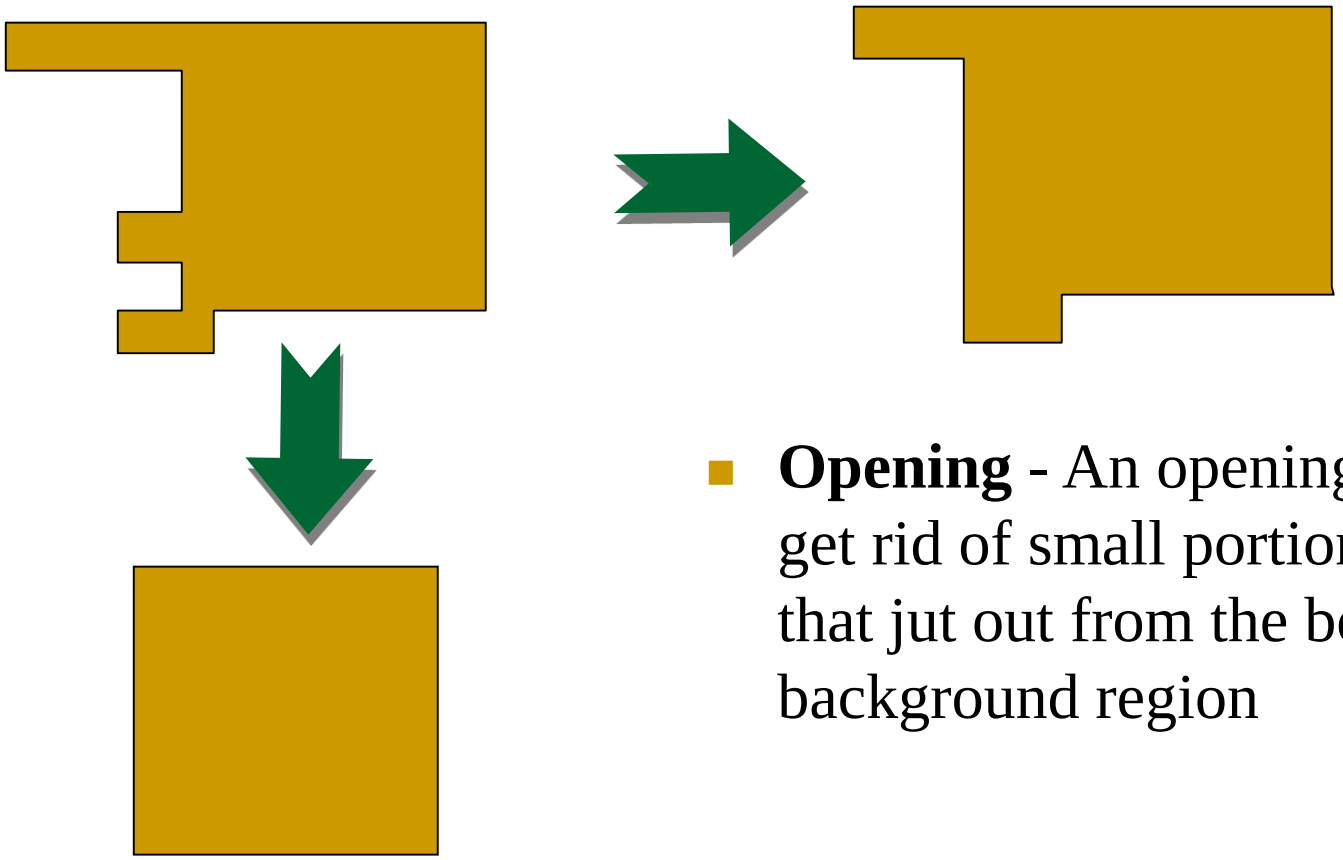
- The basic operations of binary morphology are ***dilation, erosion, opening*** and ***closing*** 膨脹、腐蝕、開、閉運算

- **Dilation -** A dilation operation enlarges a region

- **Erosion -** An erosion makes the region smaller.



- **Closing -** A closing operation can close up internal holes in a region and eliminate *bays* along the boundary



- **Opening -** An opening operation can get rid of small portions of the region that jut out from the boundary into the background region

Dilation Operation

- The **dilation** of binary image B by structuring element S is denoted by **B ⊕ S** and is defined by

$$B \oplus S = \bigcup_{b \in B} S_b$$

- *Dilation*: if at least one pixel of the SE is inside the region, *f[i,j] = 1*

  –dilation expands the region

Implementation

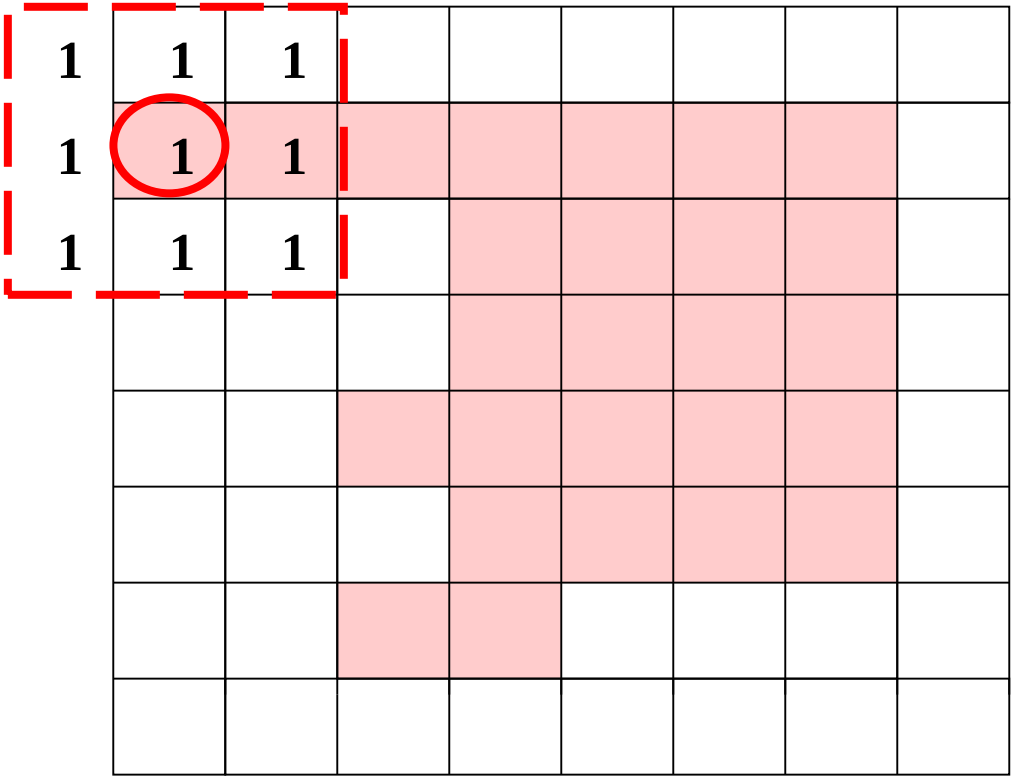
1. One pixel of the structuring element **S** is denoted as its origin; this is often the central pixel of a symmetric structuring element, but may in principle be any chosen pixel. Using the origin as a reference point.

1	1	1
1	**1**	1
1	1	1

Using the origin as a reference point

Structuring element S

2. The structuring element **S** is swept over the image. Each time the origin of the structuring element touches a binary 1-pixel of image **B**, the entire translated structuring element shape fills to the output image, which has been initialized to all zeros.



1	1	1
1	**1**	1
1	1	1

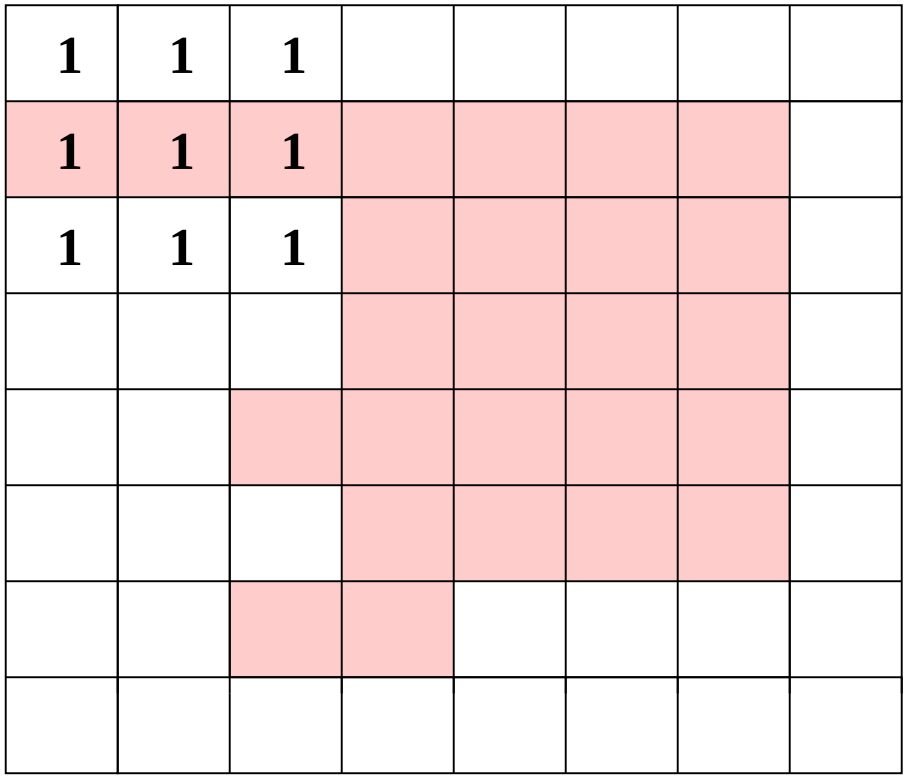
Structuring element SE

Binary image B

3. As a result of this operation, the output image (initially all 0) has 1-pixels at positions [0,0], [0,1], [1,0], [1,1], [2,0], and [2,1], which are real positions, and at positions [0,-1], [1,-1], and [2,-1], which are virtual positions and are ignored

4. For the next pixel [1, 1] of B, the structuring element S is added to the output so that its origin (which is its center) coincides with position (1,1).

5. As a result of this operation, the output image has 1-pixels at positions [0,0], [0,1], [0,2], [1,0], [1,1], [1,2], [2,0], [2,1], [2,2].



1	1	1
1	**1**	1
1	1	1

Structuring element S

Binary image B

Output image B⊕S

6. This continues until a copy of the structuring element **S** has been added into the output image for every pixel of the input image, producing the final result.

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	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
	1	1	1	1			

1	1	1
1	**1**	1
1	1	1

Structuring element S

Binary image B

Output image B⊕S

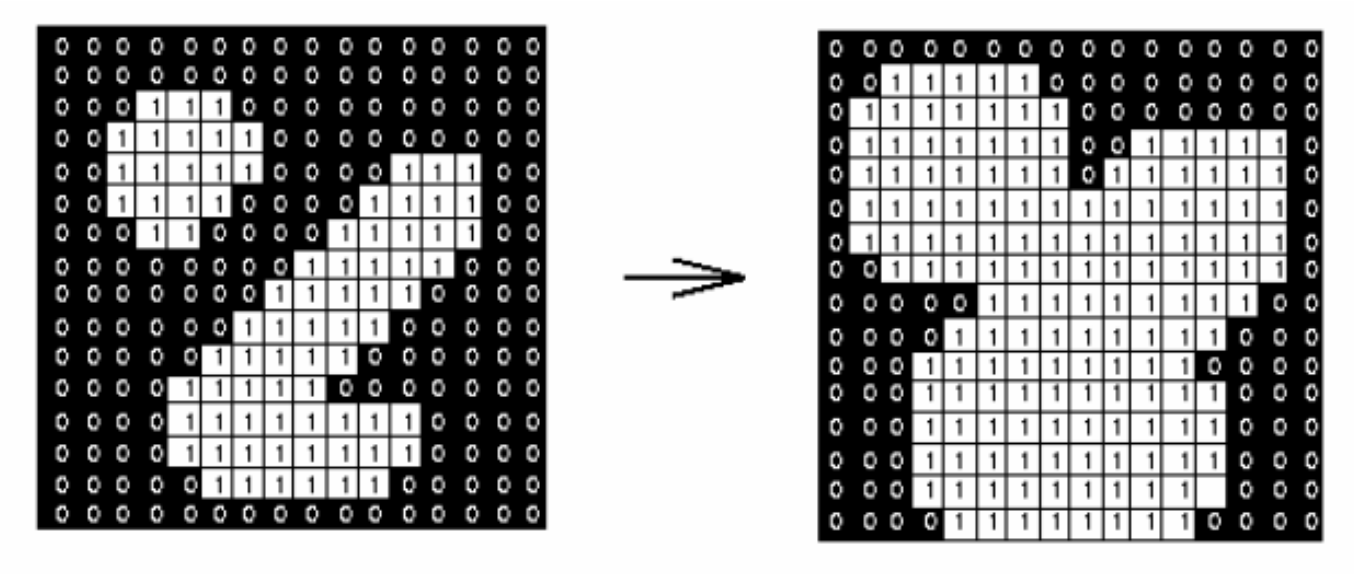


Figure 16: Effect of dilation using a 3×3 square structuring element

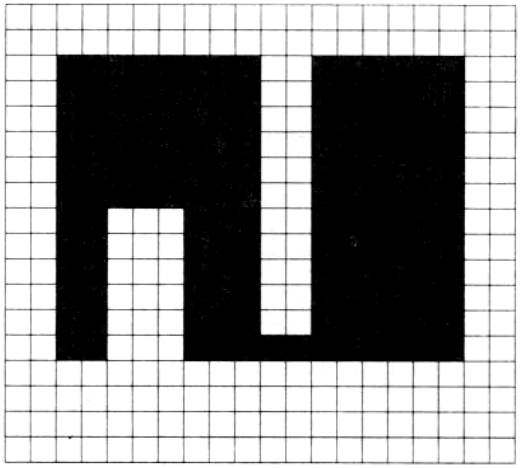
Figure 17: Effect of dilation on the binary image produced by two dilation passes using a disc shaped structuring element of 11 pixels radius.

Erosion Operation

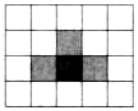
- The **erosion** of binary image **B** by structuring element **S** is denoted by **B⊖S** and is defined by

$$B \Theta S = \{ b \mid b + s \in B \ \ \forall s \in S \}$$

- *Erosion*: apply the *SE* on every pixel *(i,j)* of the image *f*
  - *(i,j)* is the center of the *SE*
  - if the whole SE is included in the region then, *f[i,j] = 1*
  - otherwise, *f[i,j] = 0*
  - erosion shrinks the object

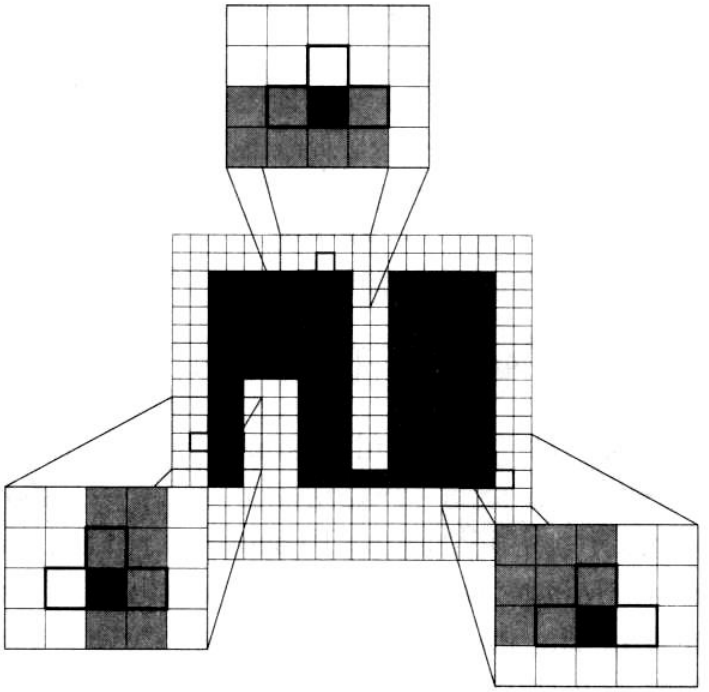


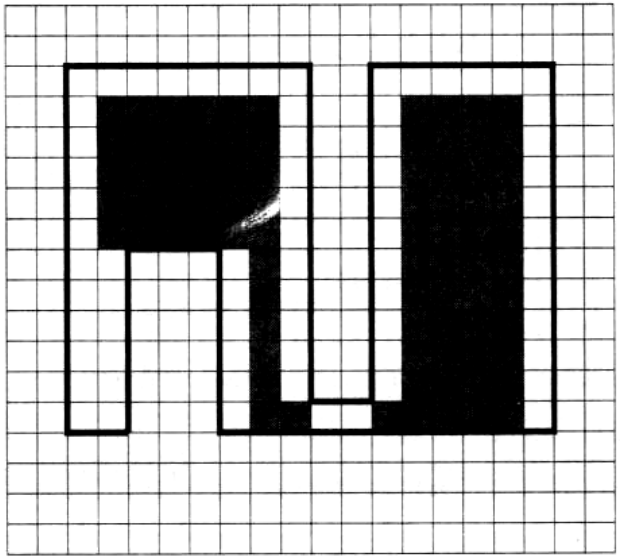
Structuring Element (*SE*)



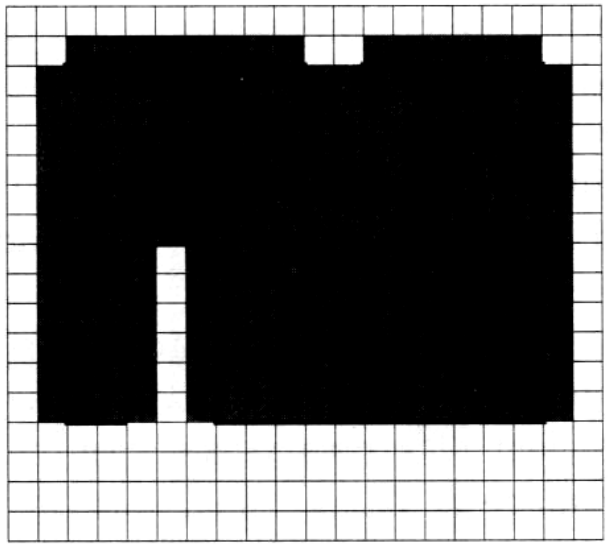
original image

Erosion with the *SE* at various positions of the original image





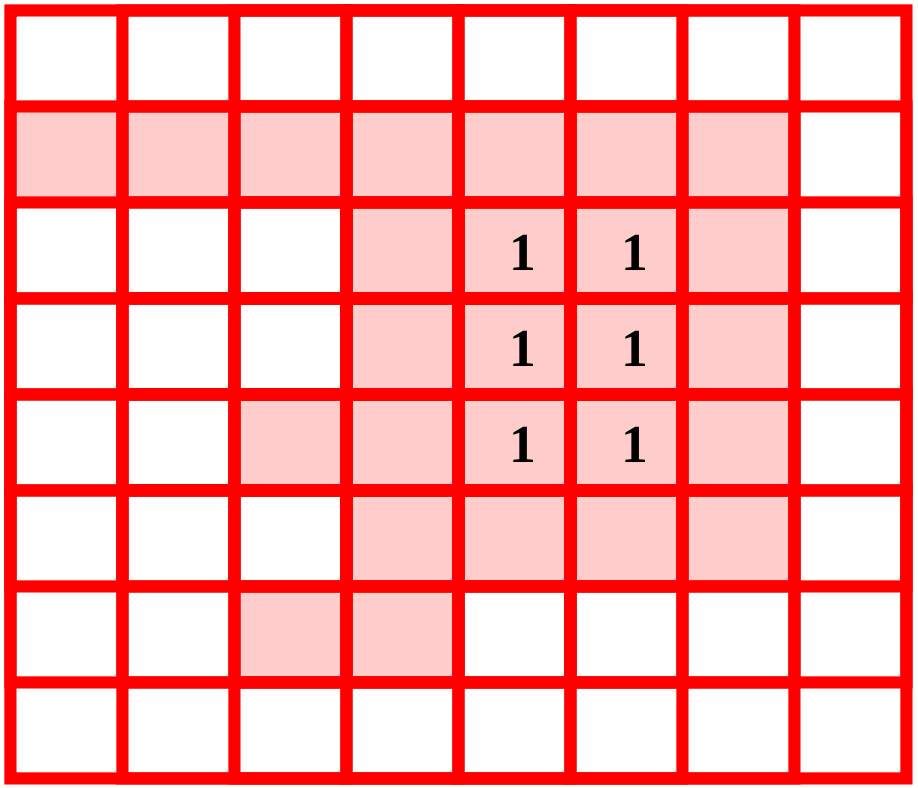
The erosion of the original image with the *SE*
The bold line shows the border of the original image



The dilation of the original image with the *SE*

- **Implementation**
  - The erosion operation also sweeps the structuring element **S** over the entire image **B**.
  - At each position where every 1-pixel of the structuring element **S** covers a 1-pixel of the binary image B, the binary image pixel corresponding to the origin of the structuring element S is added to the output image.

Example of Erosion



1	1	1
1	1	1
1	1	1

Structuring element **S**

Binary image **B**

Output image **B⊖S**

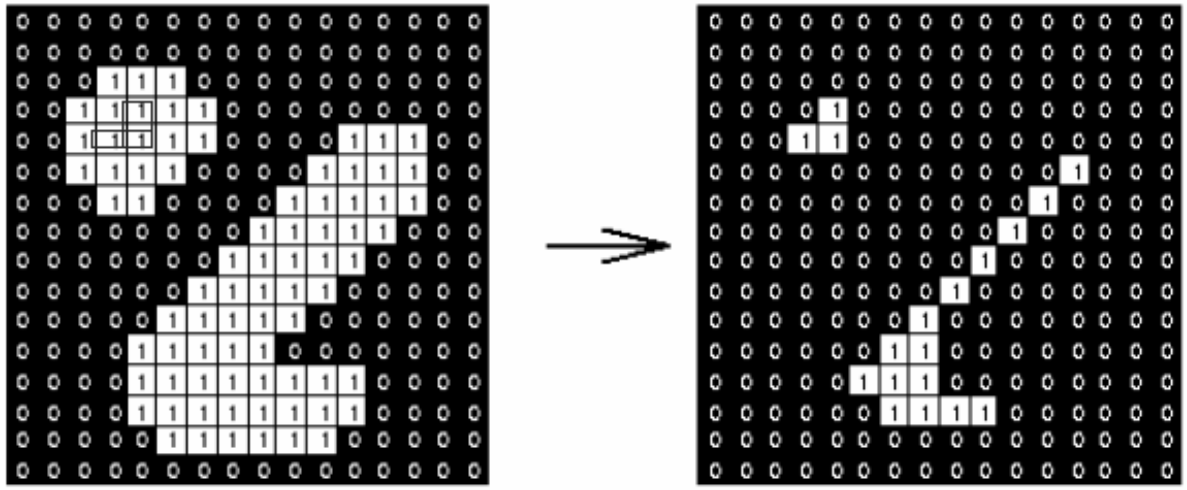


Figure 14: Effect of erosion using a 3×3 square structuring element

Figure 15: Effect of erosion using disc shaped structuring element 11 pixels in diameter.

Closing and Opening Operations

- The **closing** of binary image **B** by structuring element **S** is denoted by **B • S** and is defined by

$$B \bullet S = (B \oplus S) \Theta S$$

- *Closing*: dilation followed by erosion with the same *SE*

  – smooth by expansion, fills gaps and holes smaller than the *SE*

- The **opening** of binary image **B** by structuring element **S** is denoted **B∘S** and is defined by

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

- *Opening*: erosion followed by dilation with the same *SE*

  – filters out "positive" detail, shrinks the region

Example of Closing

	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	
		1	1				

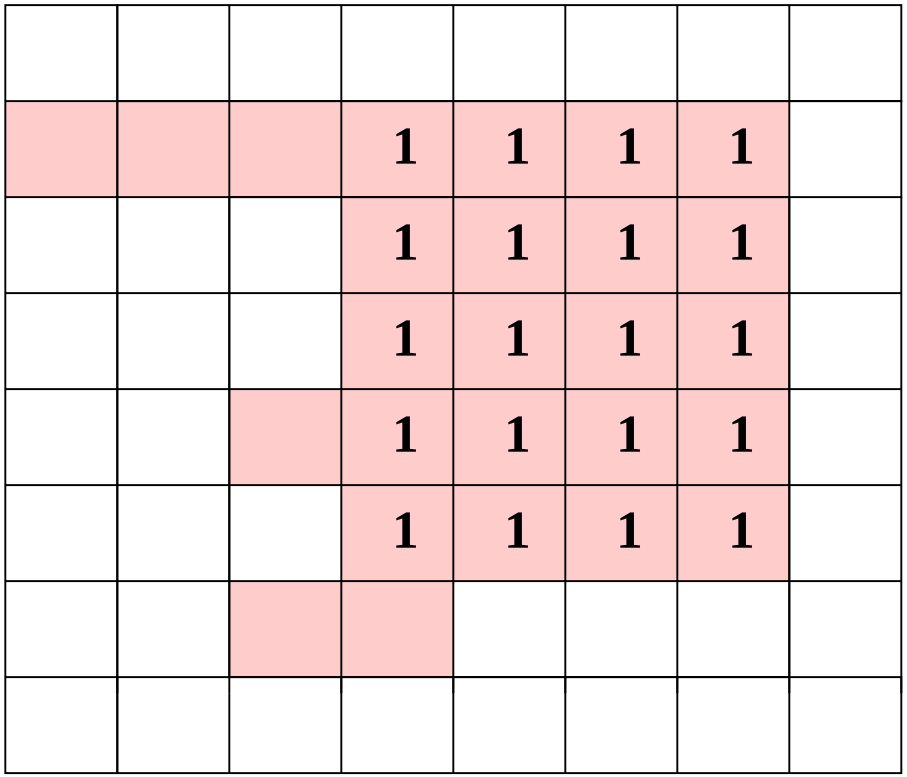
1	1	1
1	1	1
1	1	1

Structuring element **S**

Binary image **B**

Output image **B•S**

Example of Opening



1	1	1
1	1	1
1	1	1

Structuring element S

Binary image **B**

Output image **B∘S**

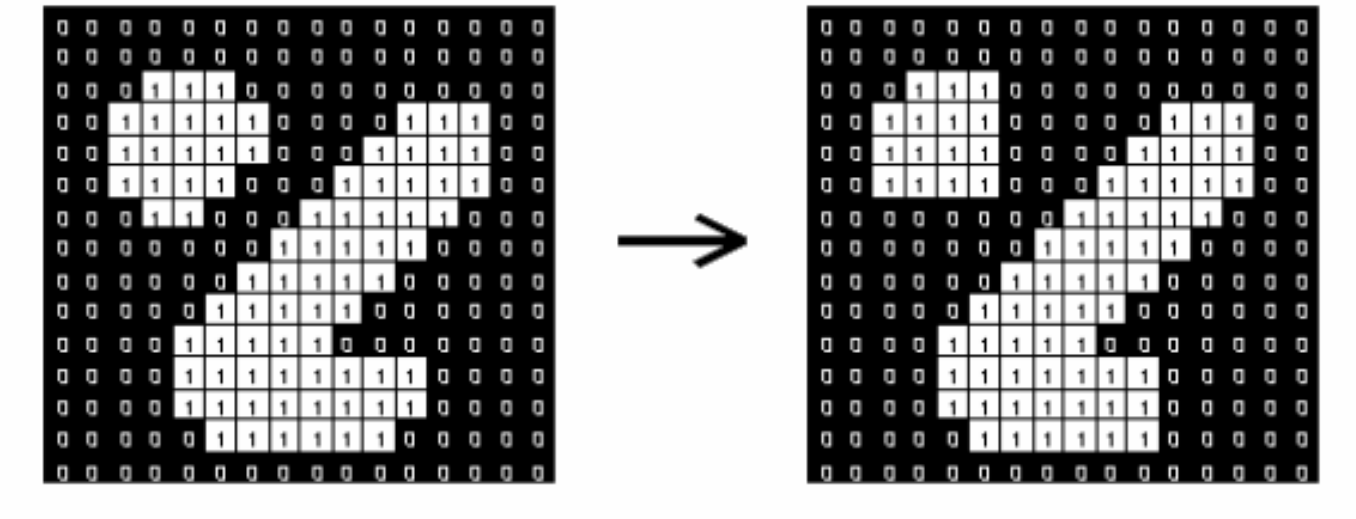


Figure 18: Effect of opening using a 3×3 square structuring element

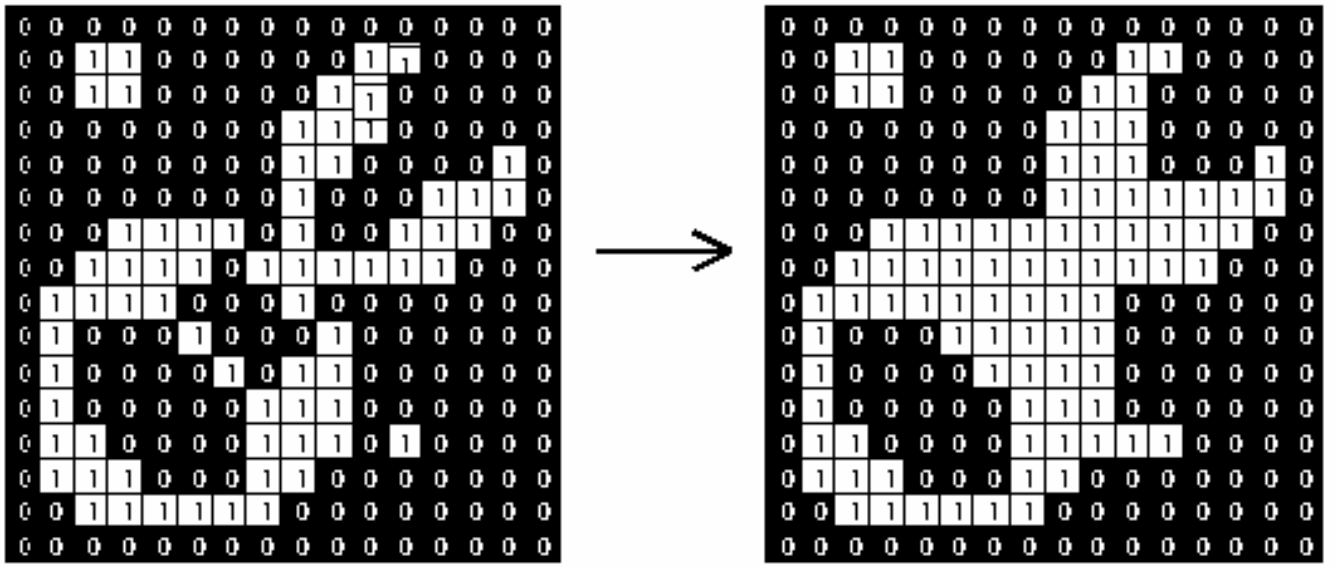
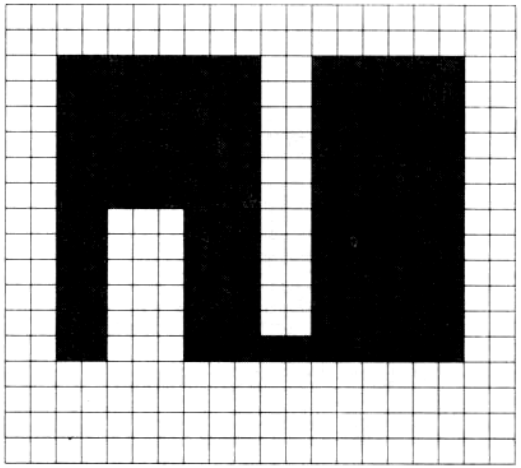
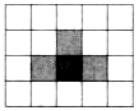


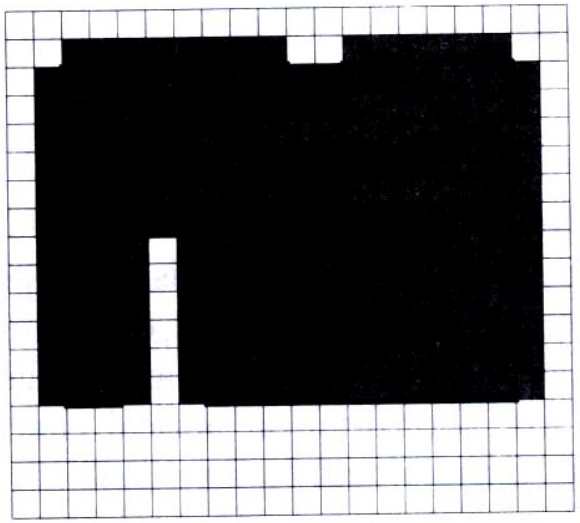
Figure 20: Effect of closing using a 3×3 square structuring element

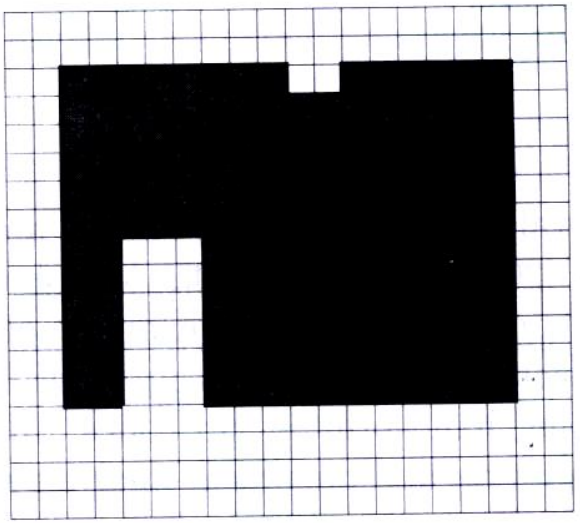




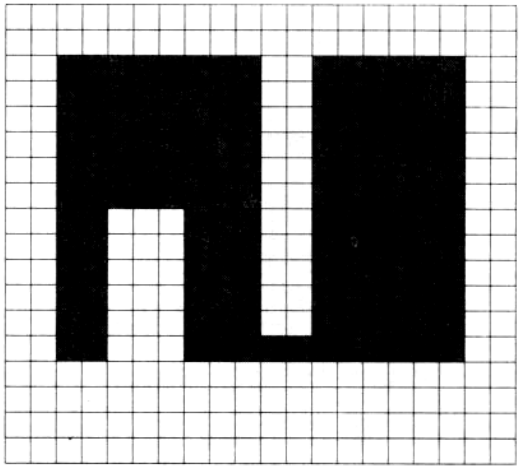
Closing

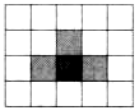
initial dilation





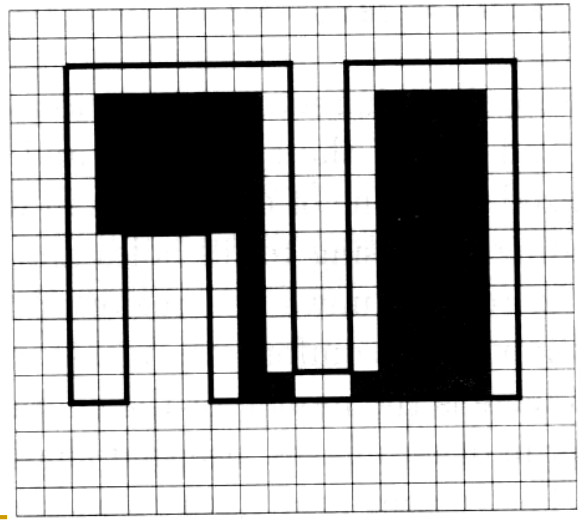
succeeding erosion

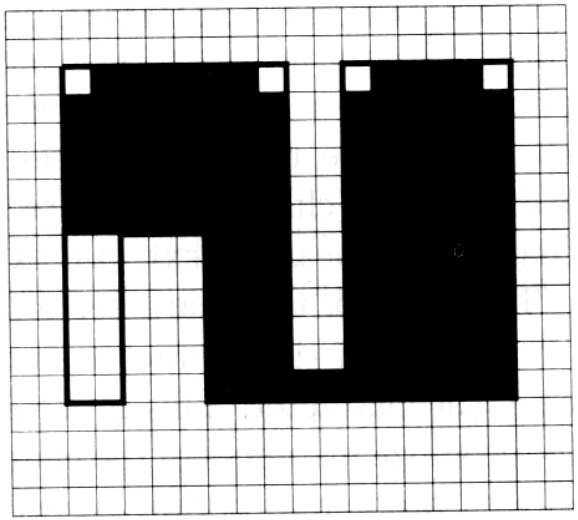




Opening

initial erosion





succeeding dilation

Closing

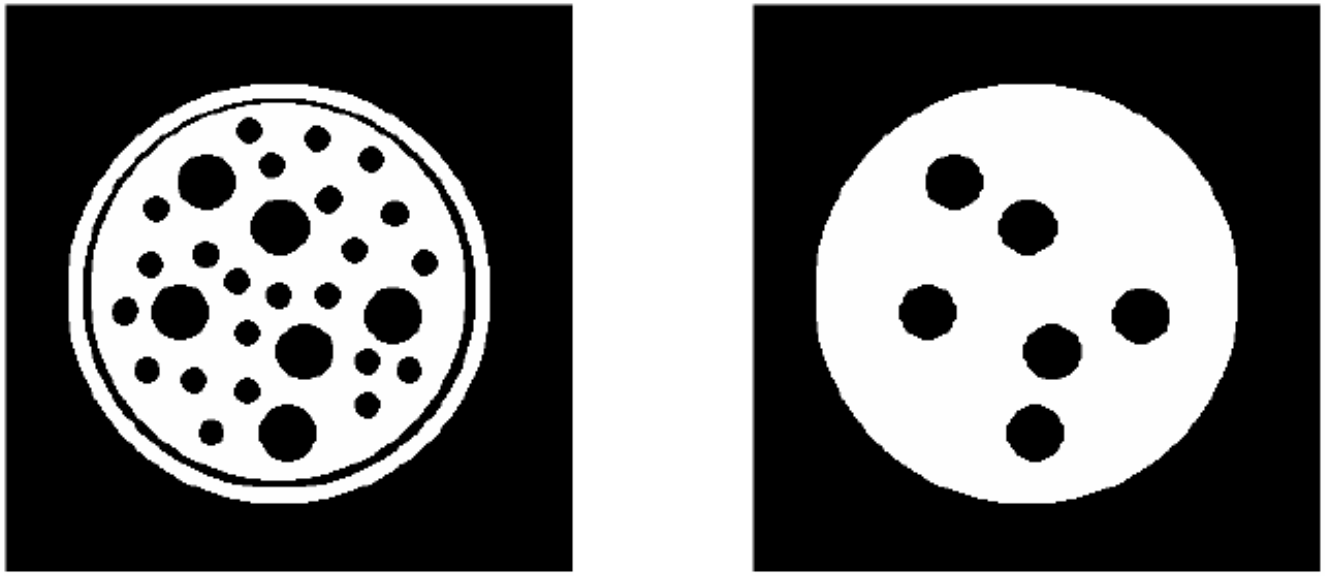


Figure 21: To remove the small holes while retaining the large holes, perform a closing with a disc-shaped structuring element with a diameter larger than the smaller holes, but smaller than the large holes.

Opening

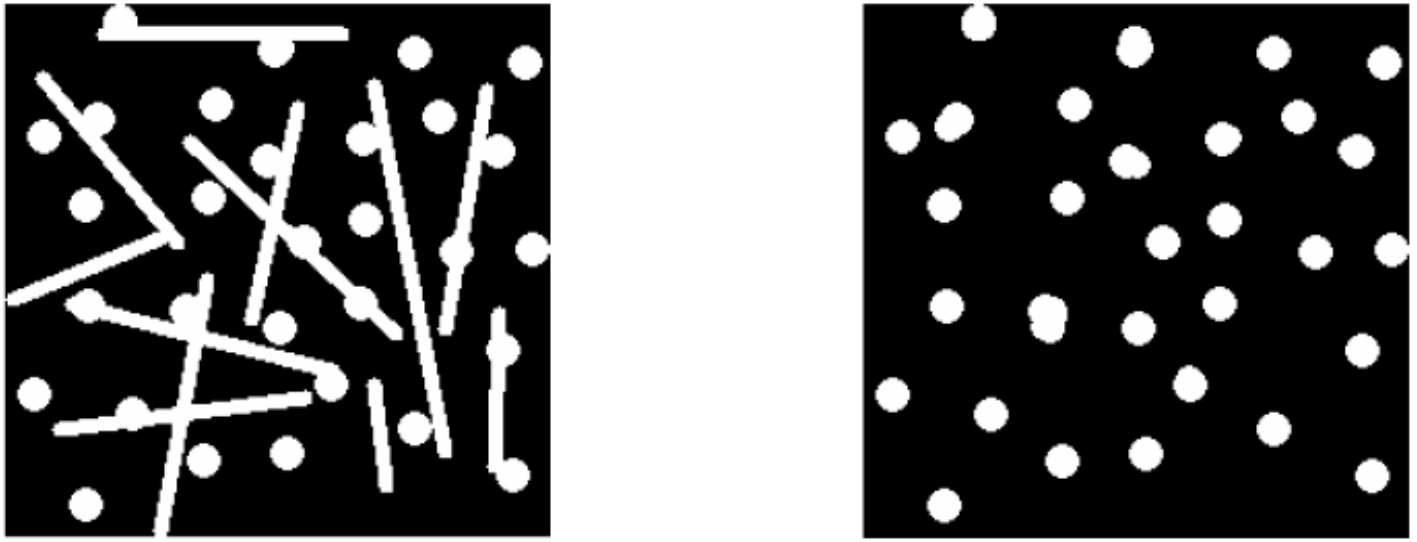
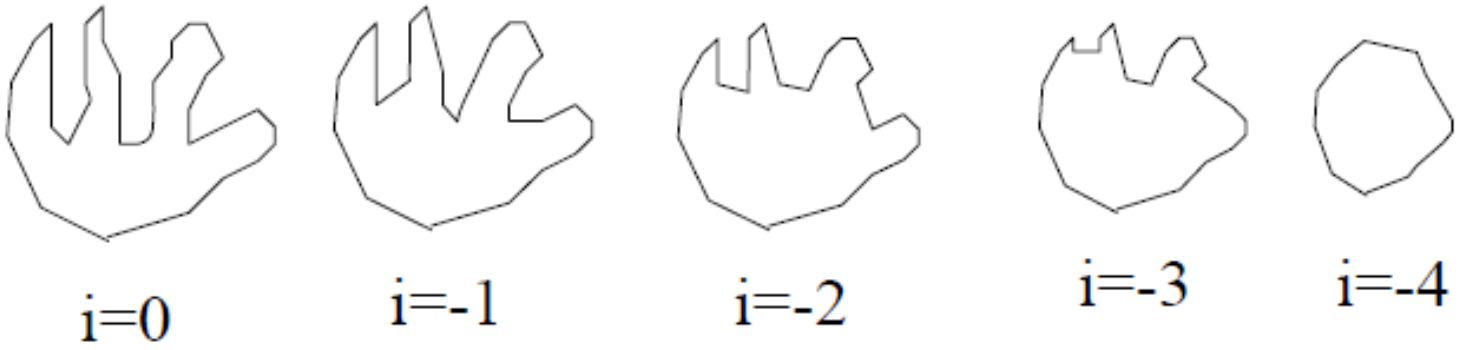
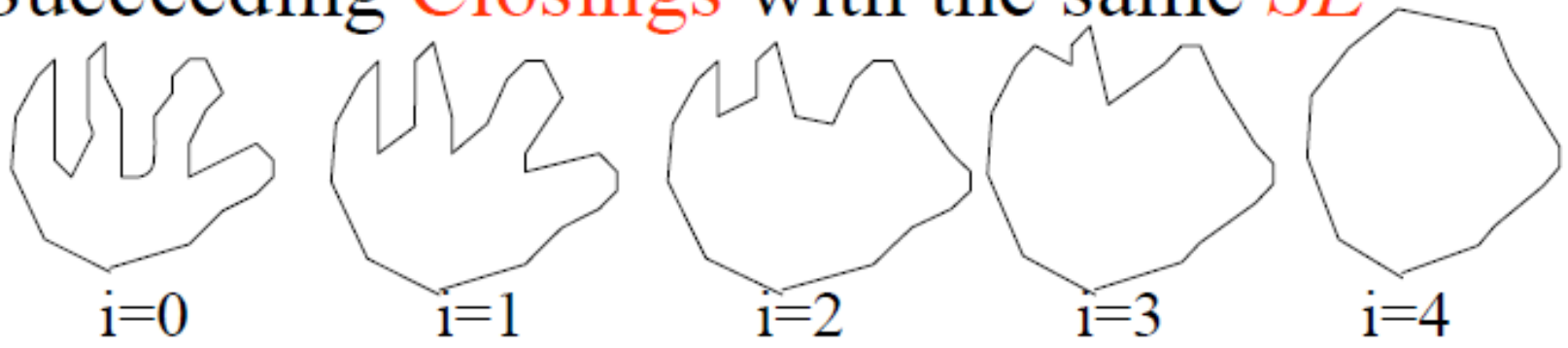


Figure 19: To separate out the circles from the lines so that they can be counted, open with a disc shaped structuring element 11 pixels in diameter

- Succeeding Openings with the same *SE*



Succeeding Closings with the same *SE*



Applications of Binary Morphology

- Closings and openings are useful in imaging applications where thresholding - or some other initial process - produces a binary image with tiny holes in the connected components or with a pair of components that should be separate joined by a thin region of foreground pixels.

- **Example:** Use of binary morphology in medical imaging

512 x 512 16-bit medical image

The original image is thresholded to produce the binary image

Opening with a DISK(13) structuring element and closing with a DISK(2) gives the results

Applications

- ***Character recognition:*** Mail sorting, Label reading, Super-market product labeling, Bank check processing, text reading.
- ***Medical image processing:*** Tumor (腫瘤) detection, measurement of size and shape of internal organs, chromosome (染色體) analysis, blood cell count.
- ***Industrial automation:*** Parts identification on assembly lines, Defect and fault inspection.

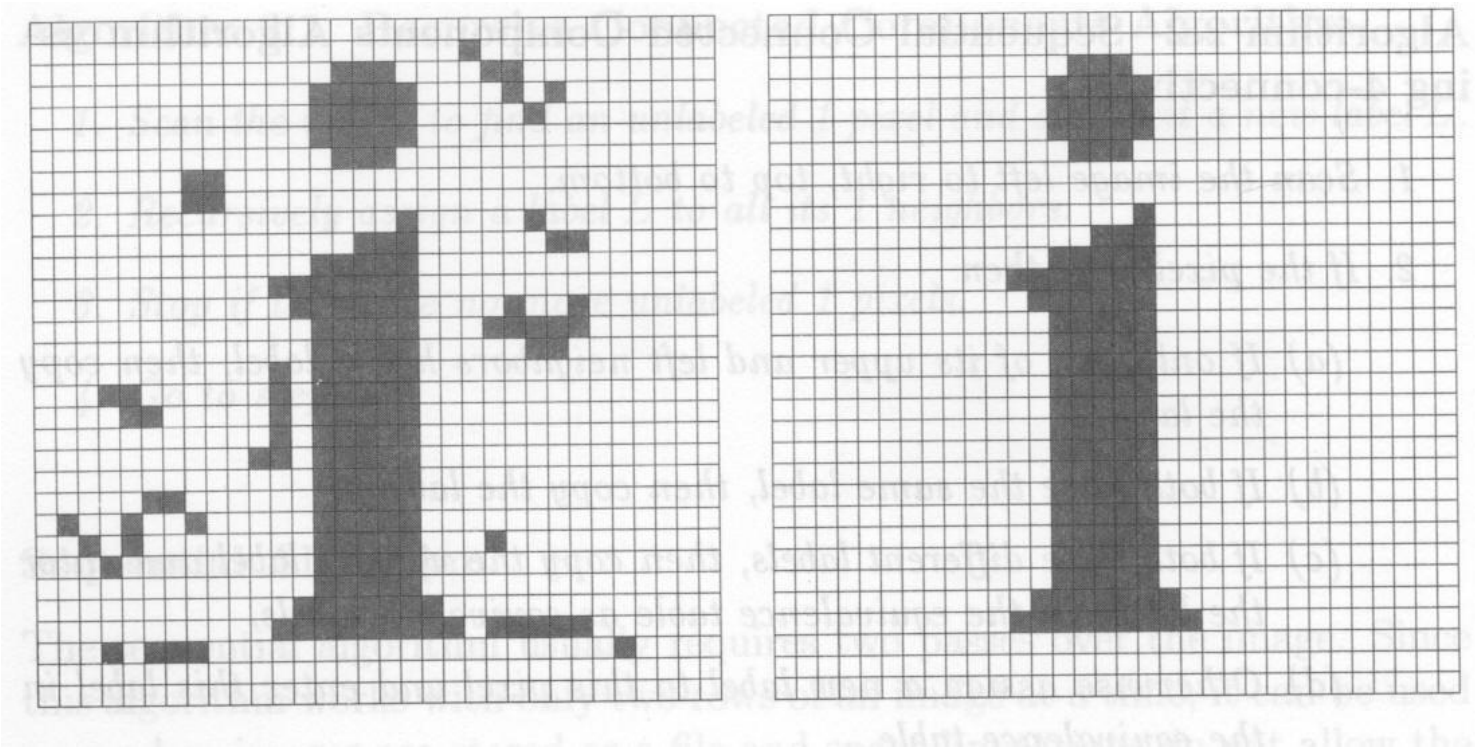
- ***Robotics:*** Recognition and interpretation of objects in a scene, Motion control and execution through visual feedback.
- ***Forensics*** (法醫學): Finger-print, iris-print matching and analysis for automated security systems.
- ***Entertainment, arts and web applications***: Image and video compression, Content-base image retrieval

6. Other applications

1. **Noise suppression**

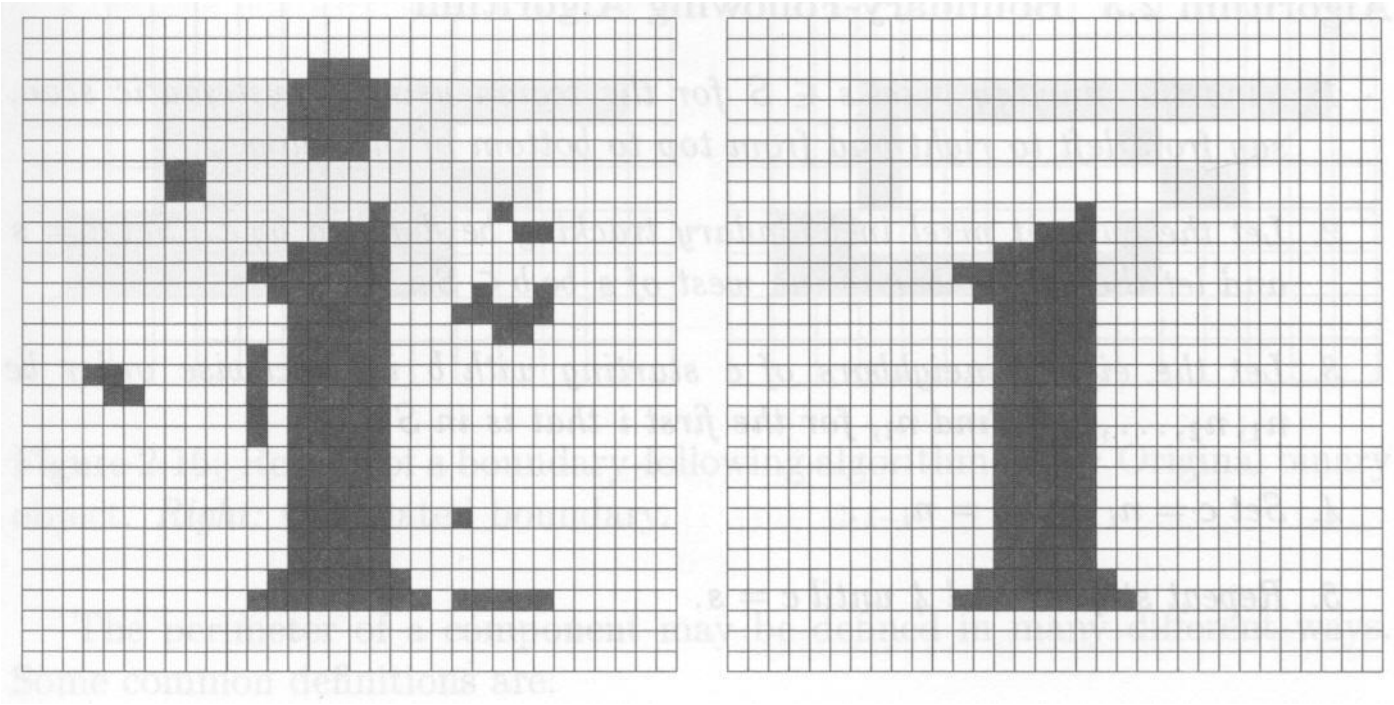
- Small regions are not useful information
  - –apply “size filter” to remove such regions
  - –all regions below *T* pixels in size are removed by changing the value of their pixels to *0* (background)

- It is generally difficult to find a good value of T
  - –if T is small, some noise will remain
  - –if T is large, useful information will be lost



Original noisy image

Filtered Image T=10



Filtered Image T=5

Filtered Image T=25

2. Run-Length encoding

Binary image:

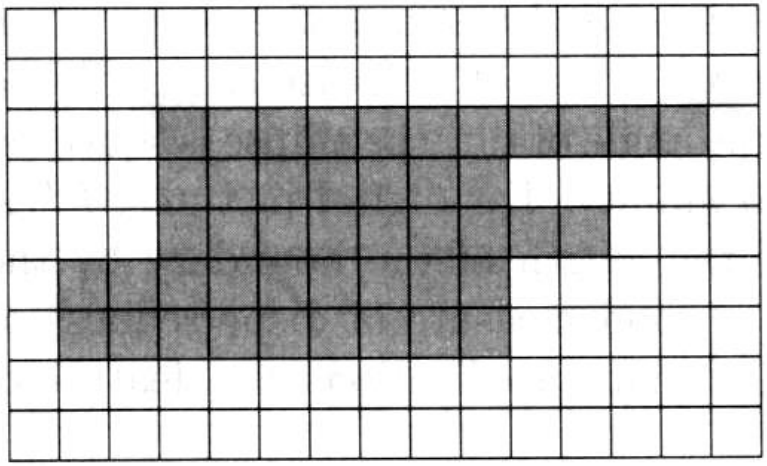
1	1	1	0	0	0	1	1	0	0	0	1	1	1	1	0	1	1	0	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Start and length of 1 runs: $(1,3)\ (7,2)\ (12,4)\ (17,2)\ (20,3)$
$(5,13)\ (19,4)$
$(1,3)\ (17,6)$

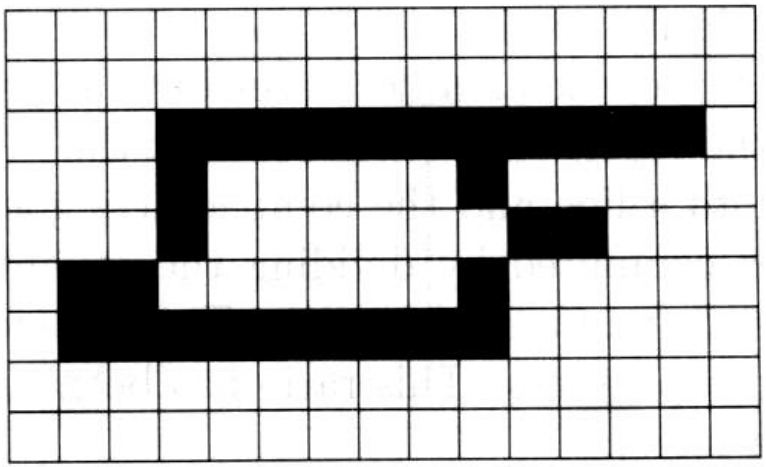
Length of 1 and 0 runs: $3,3,2,3,4,1,2,1,3$
$0,4,13,1,4$
$3,13,6$

- Compact representation of a binary image
- Find the lengths of "*runs*" of *1* pixels sequences

3. Contour Extraction



Pixel binary region



Boundary

Contour Tracing (*border following* or *boundary following*)

- There are 2 kinds of boundary (or ***border***) pixels: ***4-border*** pixels and ***8-border*** pixels.
  - A black pixel is considered a ***4-border*** pixel if it shares an **edge** with at least one white pixel.
  - A black pixel is considered an ***8-border*** pixel if it shares an **edge or a vertex** with at least one white pixel
  - A 4-border pixel is also an 8-border pixel. An 8-border pixel may or may not be a 4-border pixel.

- Contour tracing: An **ordered sequence** of the boundary pixels from which we can extract the general **shape** of the pattern.

- Contour tracing is one of many preprocessing techniques performed on digital images in order to extract information about their general shape.
- Once the contour of a given pattern is extracted, it's different characteristics will be examined and used as features which will later on be used in pattern classification.

- Therefore, correct extraction of the contour will produce more accurate features which will increase the chances of correctly classifying a given pattern.
- **Contour tracing** is often a major contributor to the efficiency of the feature extraction process -an essential process in the field of pattern recognition

- The following algorithms will ignore any ***"holes"*** present in the pattern.

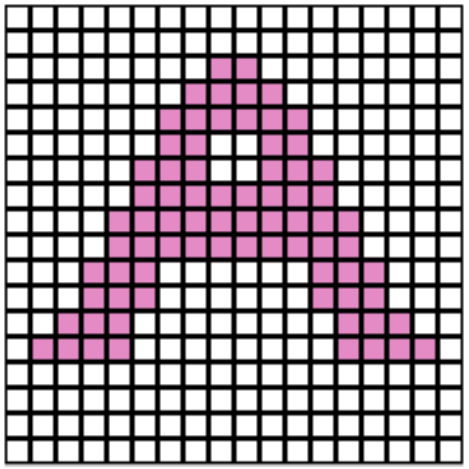


Figure 1

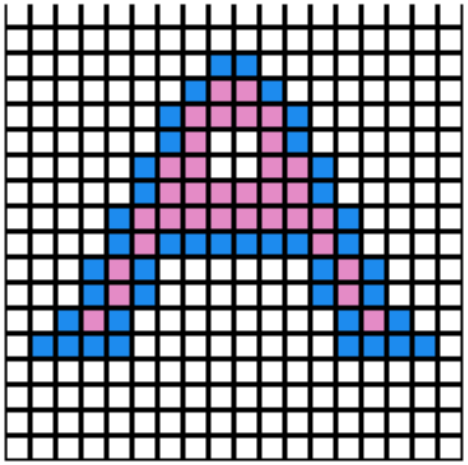


Figure 2

- This could be acceptable in some applications but in other applications, like character recognition, we would want to trace the interior of the pattern as well in order to capture any holes which identify a certain character.

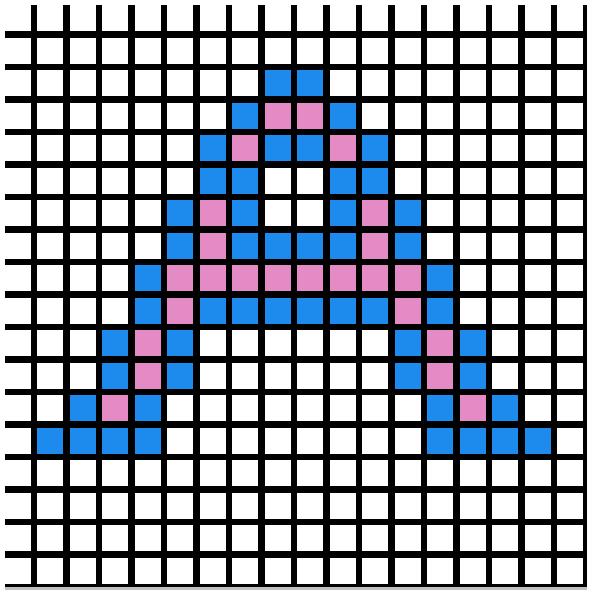


Figure 3

- A *"**hole searching**"* algorithm should be used to first extract the holes in a given pattern and then apply a contour tracing algorithm on each hole in order to extract the complete contour.

1. Square Tracing Algorithm

- Given a digital pattern i.e. a group of black pixels, on a background of white pixels i.e. a grid; locate a black pixel and declare it as your "**start**" pixel.
  - Locating a "**start**" pixel can be done in a number of ways
  - we'll start at the bottom left corner of the grid
  - scan each column of pixels from the bottom going upwards -starting from the leftmost column and proceeding to the right- until we encounter a black pixel. We'll declare that pixel as our "**start**" pixel.

- ***Algorithm***
- Imagine that you are a bug (ladybird) standing on the **start** pixel as in **Figure Square Tracing Algorithm.** In order to extract the contour of the pattern, you have to do the following:

1. **every time you find yourself standing on a black pixel, turn left, and**
2. **every time you find yourself standing on a white pixel, turn right,**
3. **until you encounter the start pixel again.**

The black pixels you walked over will be the contour of the pattern.

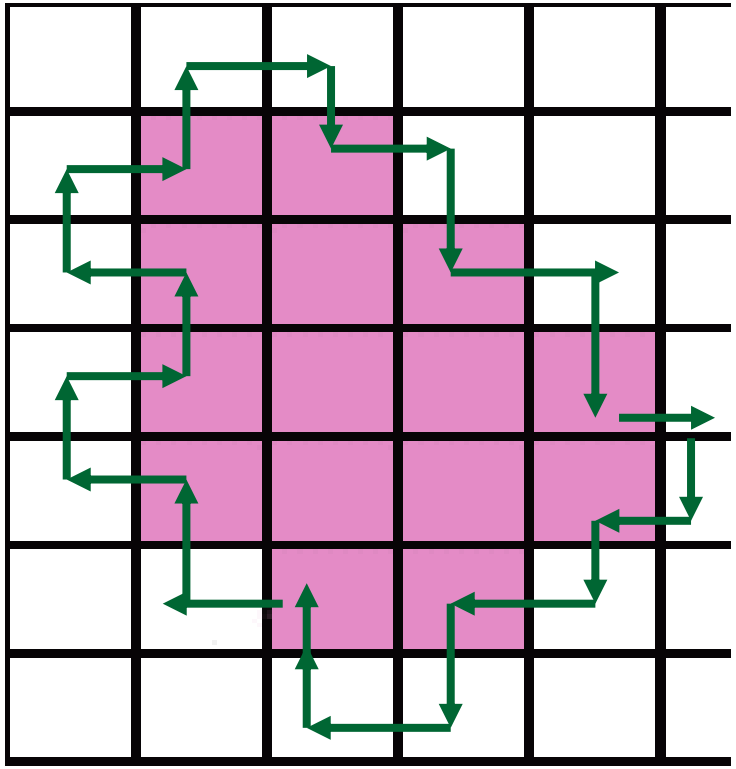
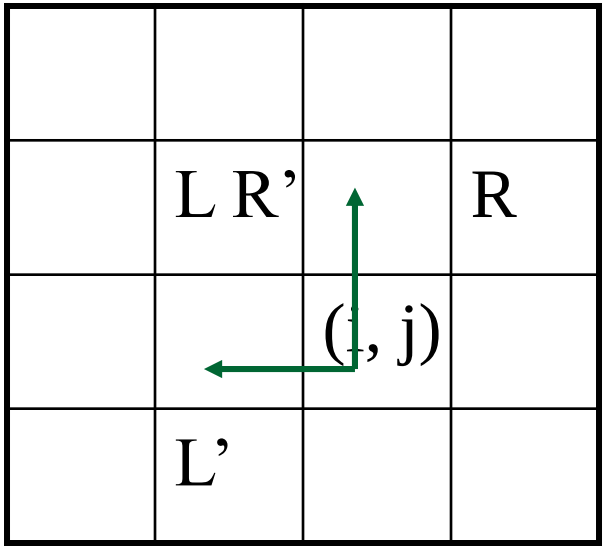


Figure Square Tracing Algorithm



- The notion of left and right in the algorithm is not to be interpreted with respect to the page or the reader but rather with respect to the direction of entering the "current" pixel during the execution of the scan.

Input: A binary image, **T**, containing a connected component **P** of black cells.

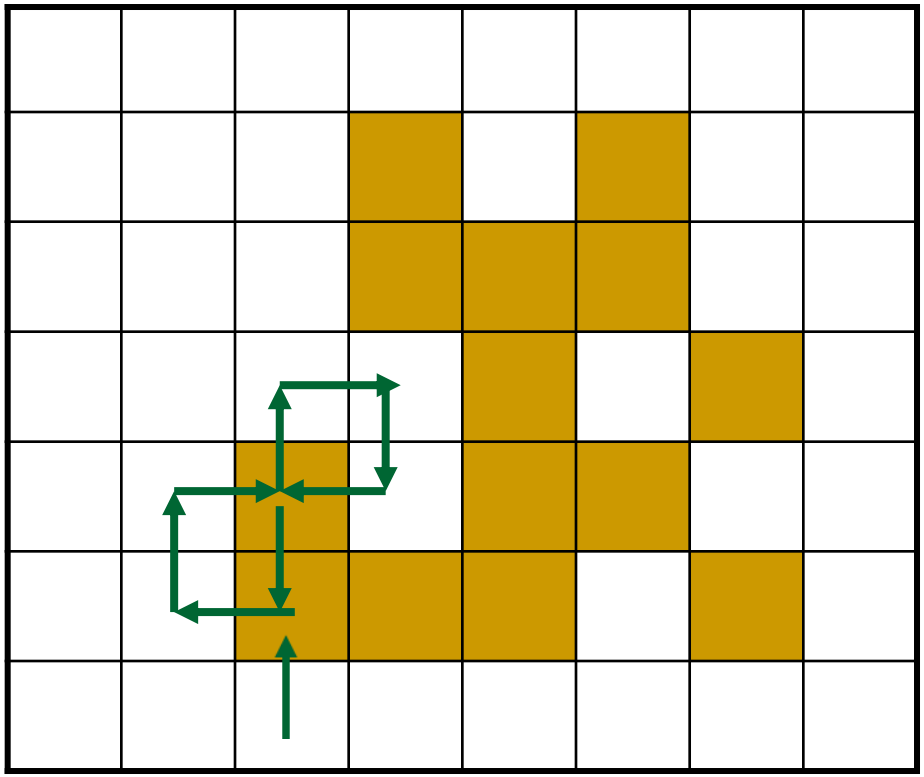
Output: A sequence **B (b1, b2 ,..., bk)** of boundary pixels i.e. the contour.

Begin

- Set ***B*** to be empty.
- From bottom to top and left to right scan the cells of **T** until a black pixel, **s**, of **P** is found.
- Insert **s** in **B**.
- Set the current pixel, **p**, to be the starting pixel, **s**.
- Turn left i.e. visit the left adjacent pixel of **p**.
- Update **p** i.e. set it to be the current pixel.

- While **p** not equal to **s** do
  If the current pixel **p** is black
    insert **p** in **B** and turn left (visit the left adjacent pixel of **p**).
    Update **p** i.e. set it to be the current pixel.
  else
    turn right (visit the right adjacent pixel of **p**).
    Update **p** i.e. set it to be the current pixel.
  end While

End



Weakness of the algorithm

- The algorithm fails to extract the contour of a large family of patterns which frequently occur in real life applications.

- This is largely attributed to the left and right turns which tend to miss pixels lying “diagonally”（對角線的） with respect to a given pixel.

- Another weakness of the square tracing algorithm lies in the choice of the stopping criterion.

- Improvement:
  - Stop after entering the **start** pixel a second time **in the same manner you entered it initially.** This criterion was proposed by Jacob Eliosoff and we will therefore call it ***Jacob's stopping criterion.***

Summary

- Given a 4-connected pattern and background, the square tracing algorithm will trace the whole boundary i.e. contour, of the pattern and will stop after tracing the boundary once.

2. Moore neighborhood tracing algorithm

- **Moore Neighborhood**

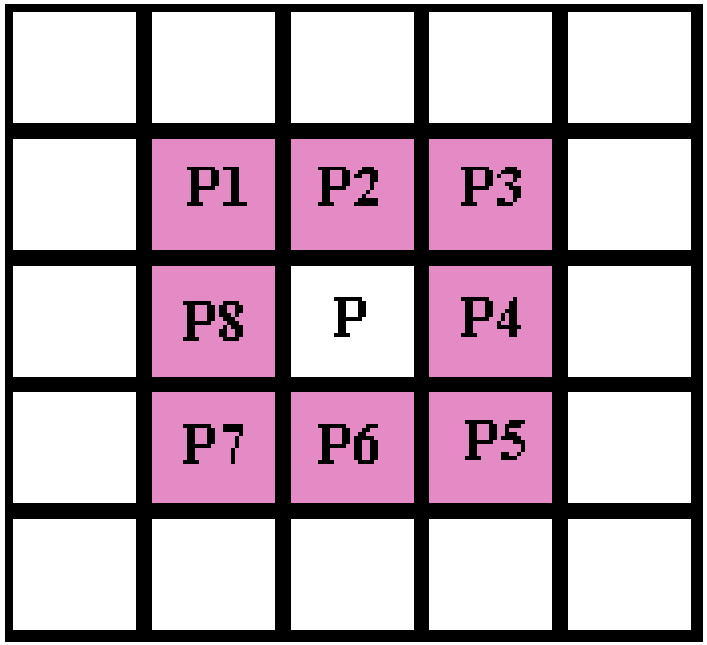


Figure 1

a. Given a digital pattern i.e. a group of black pixels, on a background of white pixels i.e. a grid
b. locate a black pixel and declare it as your "**start**" pixel. (we'll start at the bottom left corner of the grid.)
c. extract the contour by going around the pattern in a clockwise direction.

- every time you hit a black pixel, **P**, backtrack i.e. go back to the white pixel you were previously standing on,
- then, go **around** pixel **P** in a clockwise direction, visiting each pixel in its Moore neighborhood, until you hit a black pixel.
- The algorithm terminates when the start pixel is visited for a second time.

The black pixels you walked over will be the contour of the pattern.

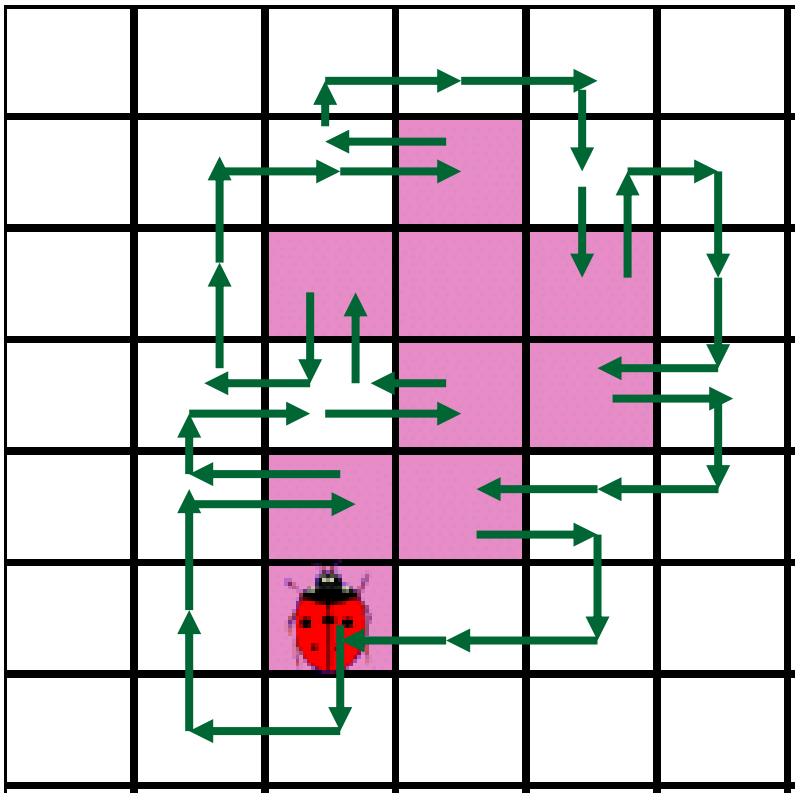


Figure 2

Input: A binary image, **T**, containing a connected component **P** of black cells.

Output: A sequence **B (b1, b2 ,..., bk)** of boundary pixels i.e. the contour.

Define **M(a)** to be the Moore neighborhood of pixel **a**.
Let **p** denote the current boundary pixel.
Let **c** denote the current pixel under consideration i.e. **c** is in **M(p)**.

Begin

- Set B to be empty.
- From bottom to top and left to right scan the cells of **T** until a black pixel, **s,** of **P** is found

- Insert **s** in **B**.
- Set the current boundary point **p** to **s** i.e**. p=s**
- Backtrack i.e. move to the pixel from which **s** was entered.
- Set **c** to be the next clockwise pixel in **M(p)**.
- While **c** not equal to **s** do

```
            If c is black
                insert c in B
                set p=c
                backtrack (move the current pixel c to
                the pixel from which p was entered)
             else
                advance the current pixel c to the next
                clockwise pixel in M(p)
    end While
End
```

Weakness of the algorithm

- The main weakness of Moore-Neighbor tracing lies in the choice of the stopping criterion
- Using Jacob's stopping criterion will greatly improve the performance of Moore-Neighbor tracing making it the best algorithm for extracting the contour of any pattern no matter what its connectivity.