BINARY IMAGE OPERATION

Duangpen jetpipattanapong

DIGITAL IMAGE OPERATION

- Addition

$$c[i,j] = a[i,j]+b[i,j]$$

- Subtraction

$$c[i,j] = a[i,j]-b[i,j]$$

- Multiplication by constant

$$b[I,j] = c \times a[i,j]$$

A

0	50	100	150	200	250
0	50	100	150	200	250
0	50	100	150	200	250
0	50	100	150	200	250
0	50	100	150	200	250
0	50	100	150	200	250

B

0	0	0	0	0	0
50	50	50	50	50	50
100	100	100	100	100	100
150	150	150	150	150	150
200	200	200	200	200	200
250	250	250	250	250	250

A+B

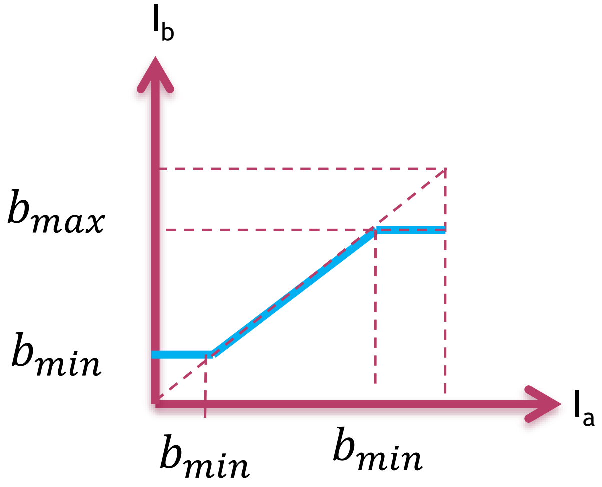
A-B

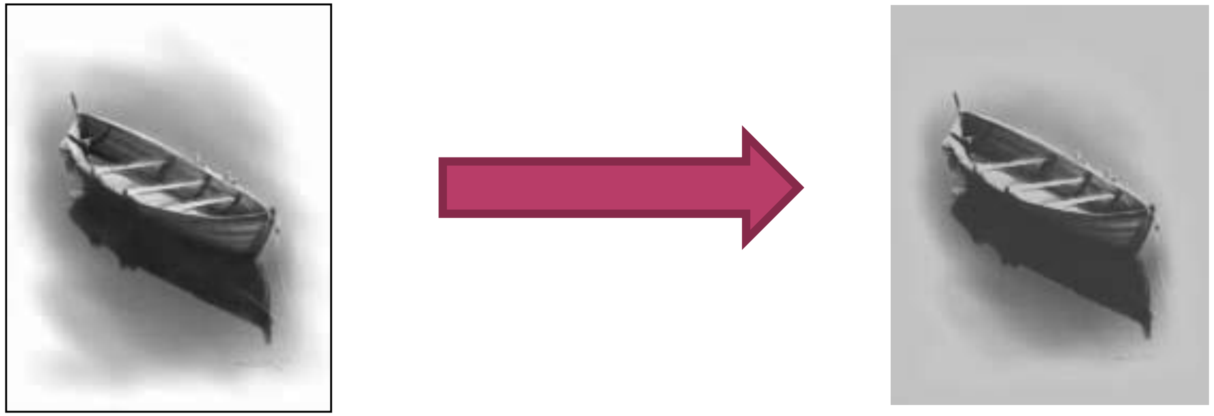
A*1.5

Clipping

- truncates all intensities below the value Min to Min and all intensities above the value Max to Max.

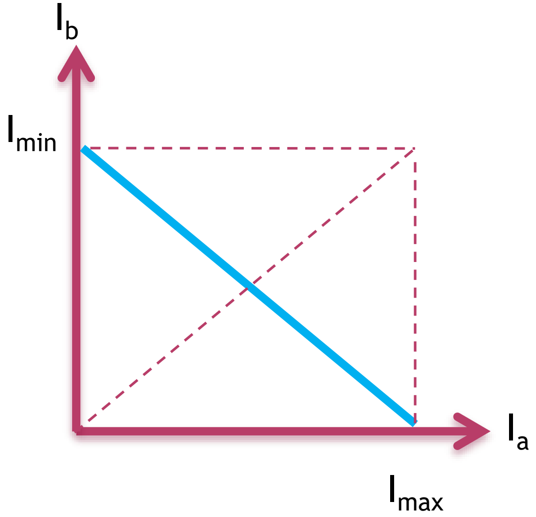
$$
\mathrm{b}[i,j] = \begin{cases} b_{max} & : a[i,j] > b_{max} \\ a[i,j] & : b_{min} \le a[i,j] \le b_{max} \\ b_{min} & : b[i,j] < b_{min} \end{cases}
$$

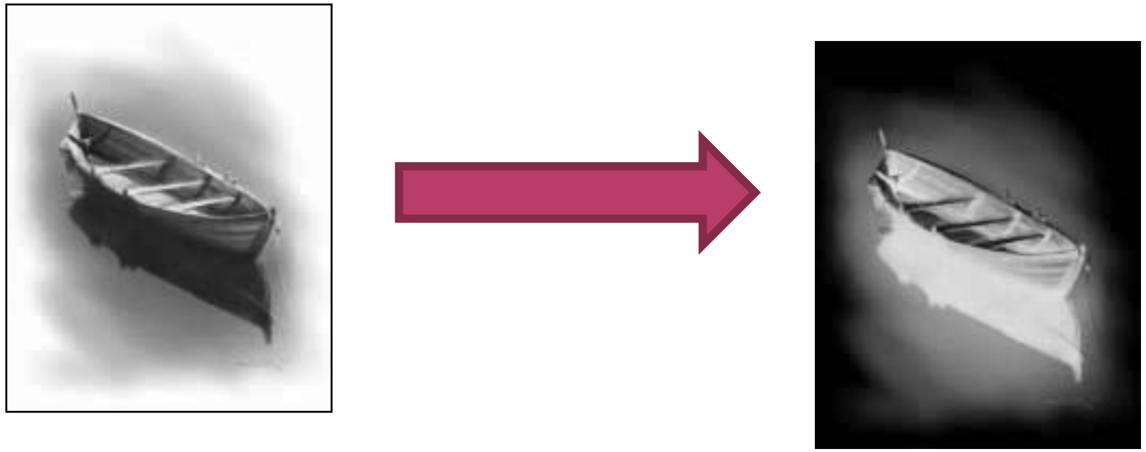




- Negative
  - Inverse intensity of image

$$b[i,j] = 255 - a[i,j]$$





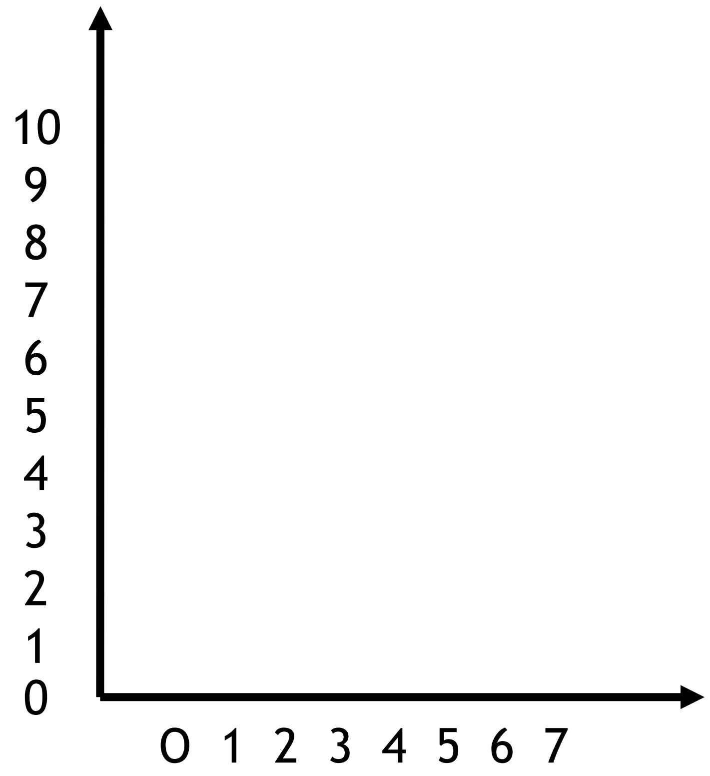
- Histogram
  - Show the number of pixels which have identical intensity

$$H(m) = \{|I[i,j]| \mid I[i,j] = m\}$$

H(m)

0	0	4	4	4	0
3	3	4	4	4	1
3	3	4	5	5	1
0	6	6	7	7	7
6	6	6	7	7	0
0	0	0	0	0	0

H(0) -
H(1) -
H(2) -
H(3) -
H(4) -
H(5) -
H(6) -
H(7) -

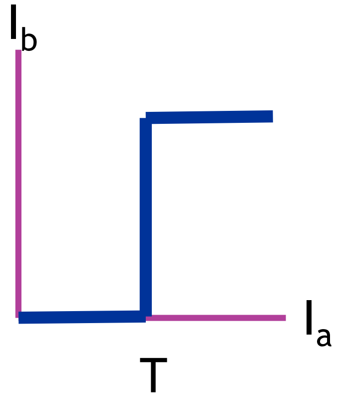


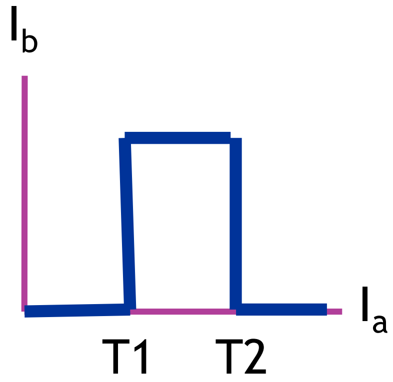
Thresholding

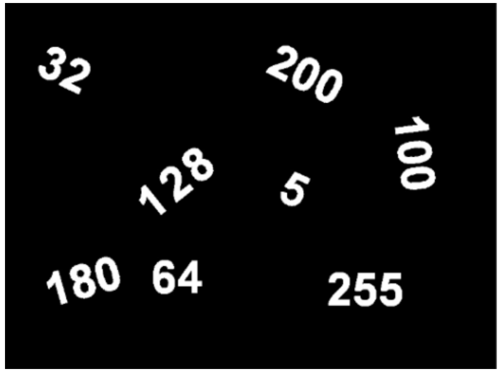
- Method to convert a grayscale image into a binary Image
- Cause objects of interest are separated from the background

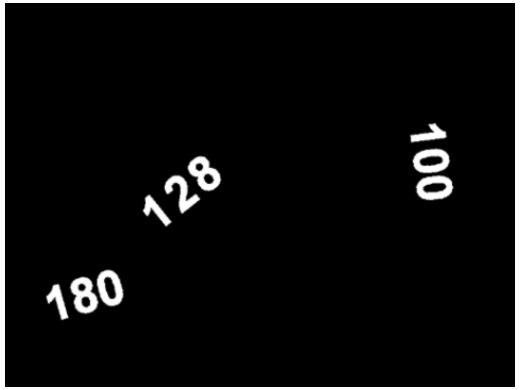
$$B[i,j] = \begin{cases} 1 & : a[i,j] <= T \\ 0 & : \text{otherwise} \end{cases}$$

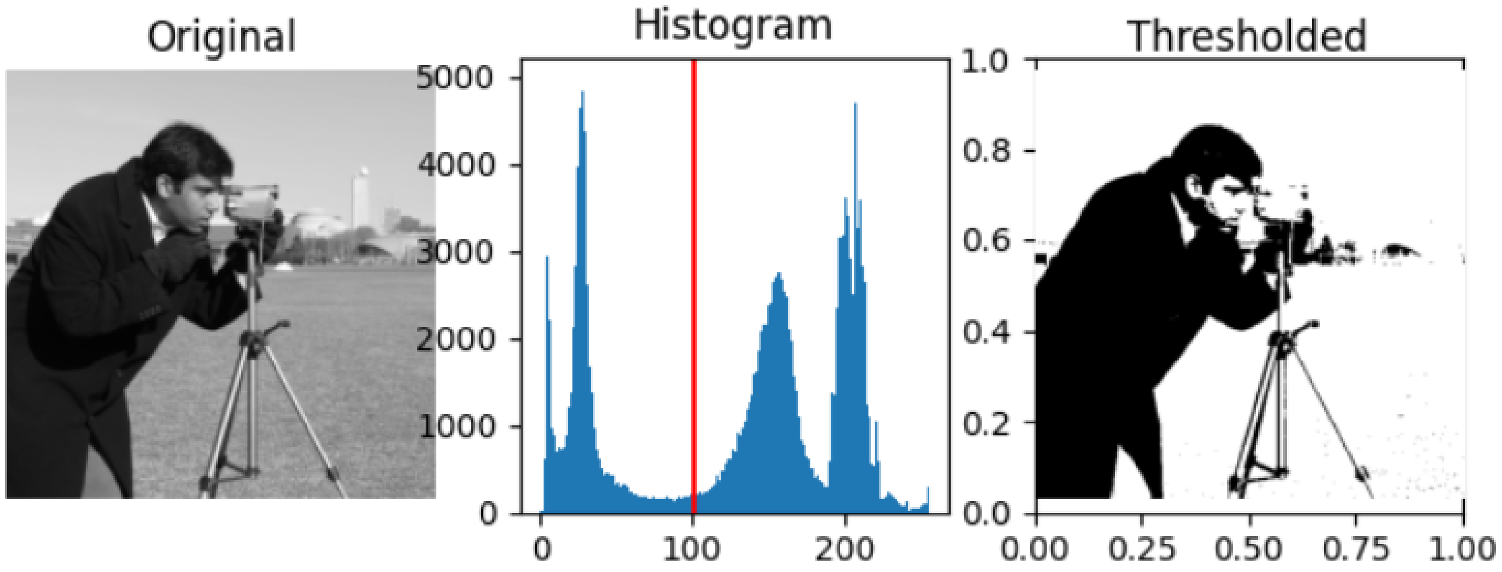
$$B[i,j] = \begin{cases} 1 & : T1 <= a[i,j] <= T2 \\ 0 & : \text{otherwise} \end{cases}$$










Original
Histogram
Thresholded
5000
4000
3000
2000
1000
0
0
100
200
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00

Automatic Thresholding

f(g) – number of pixel at gray level g

t(g) – actual number of pixels at gray level g or less

$$t(g) = \sum_{i=0}^{g} f(i)$$

P = number of pixel (MxN)

M(g) = mean gray level for only pixels with gray level between 0-g

$$m(g) = \frac{\sum_{i=0}^{g} i \times f(i)}{t(g)}$$

G – maximum number of gray level (0,1, …, G-1)

$$T = \max\left\{ \frac{t(g)}{P - t(g)} \times [m(g) - m(G-1)]^2 \right\} - 1$$

g	f(g)	t(g)	gxf(g)	sum(gxf(g))	m(g)	A	B	AxB
0	2							
1	4							
2	8							
3	9							
4	5							
5	4							
6	12							
7	8							
8	7							
9	1							

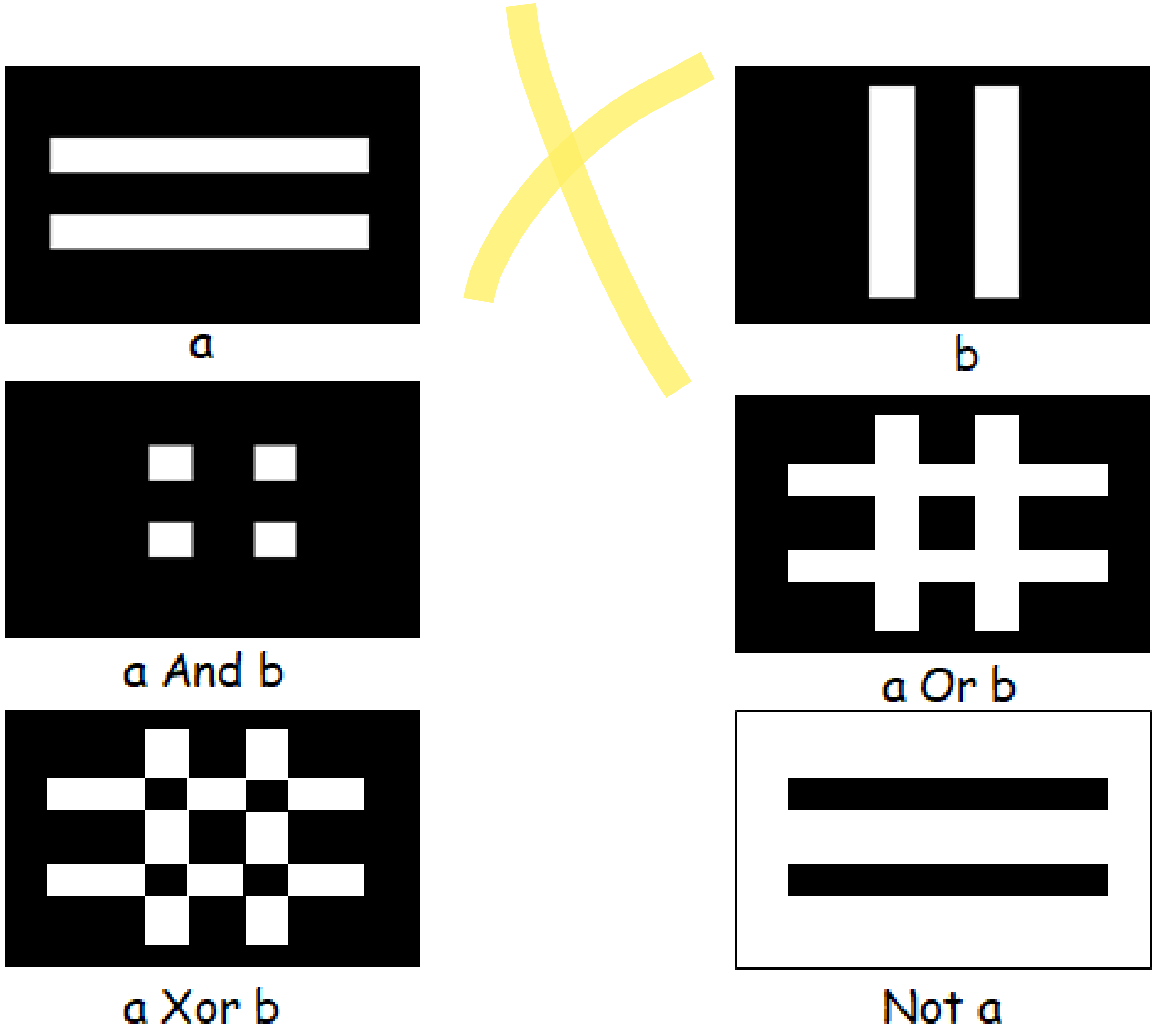
$$t(g) = \sum_{i=0}^{g} f(i)$$

$$m(g) = \frac{\sum_{i=0}^{g} i \times f(i)}{t(g)}$$

$$T = \max\left\{ \underbrace{\frac{t(g)}{P - t(g)}}_{A} \times \underbrace{[m(g) - m(G-1)]^2}_{B} \right\} - 1$$

BINARY OPERATION

- And
  - c[i,j] = a[i,j] & b[i,j]
- Or
  - c[i,j] = a[i,j] || b[i,j]
- Xor
  - c[i,j] = a[i,j] ^ b[i,j]
- Not
  - b[i,j] = ! a[i,j]



Projection

- Finding the number of '1' pixels that are on lines perpendicular to each row or column

  - Horizontal projection

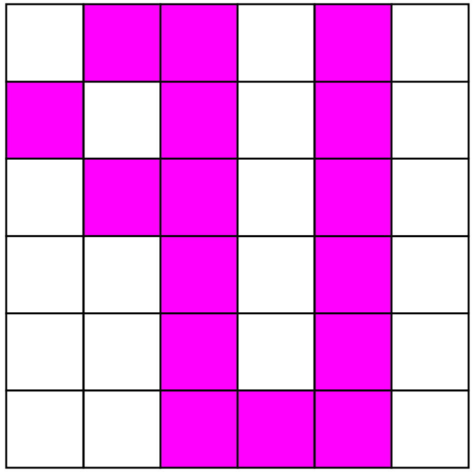
$$H[i] = \sum_{j=0}^{m-1} B[i, j]$$

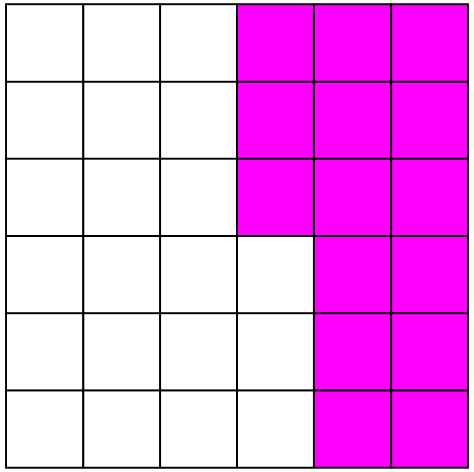
  - Vertical projection

$$V[j] = \sum_{j=0}^{n-1} B[i, j]$$

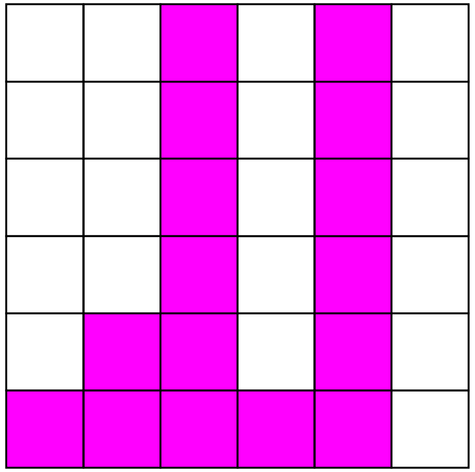
H[0] = 3
H[1] = 3
H[2] =
H[3] =
H[4] =
H[5] =

V[0] = 1
V[1] = 2
V[2] =
V[3] =
V[4] =
V[5] =





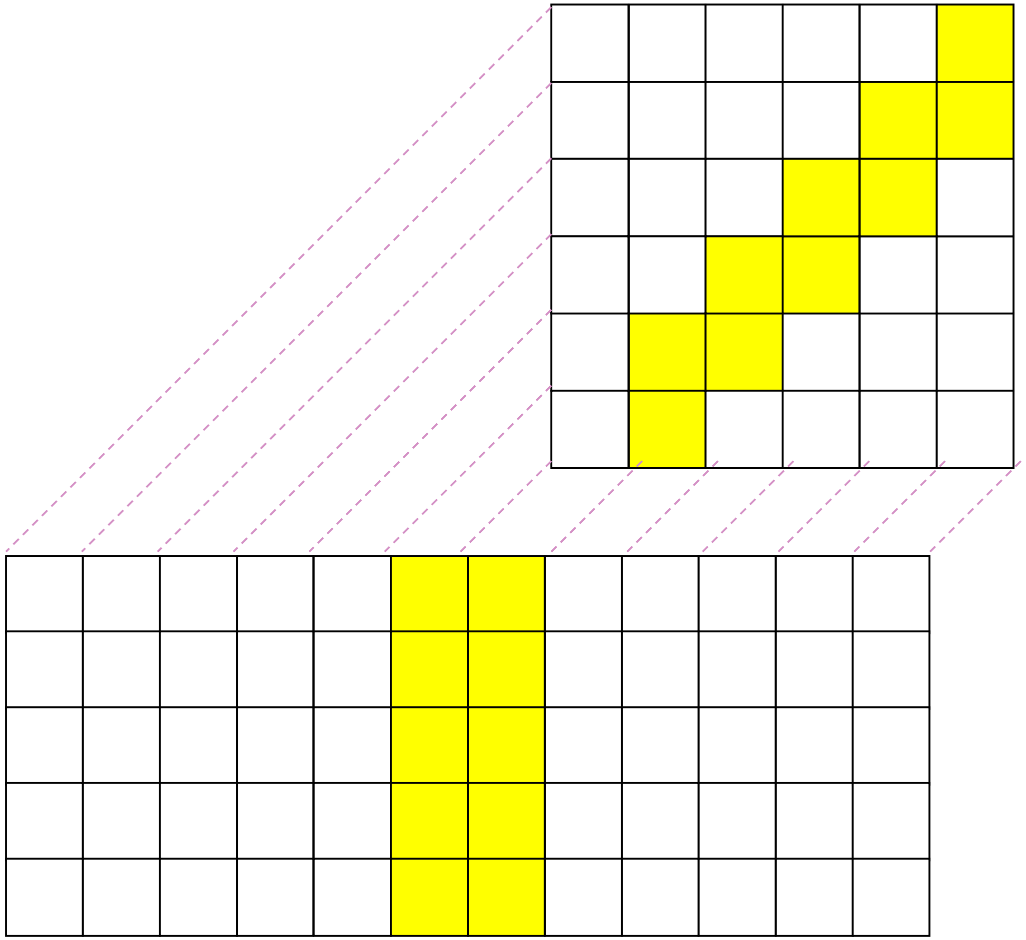
Horizontal Projection

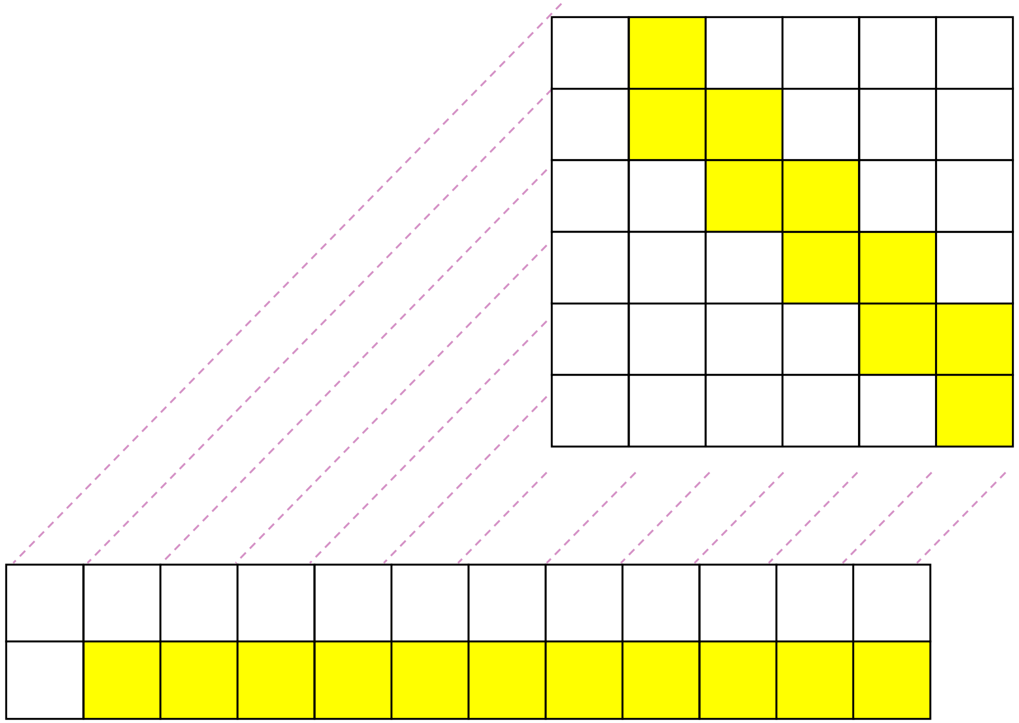


Vertical Projection

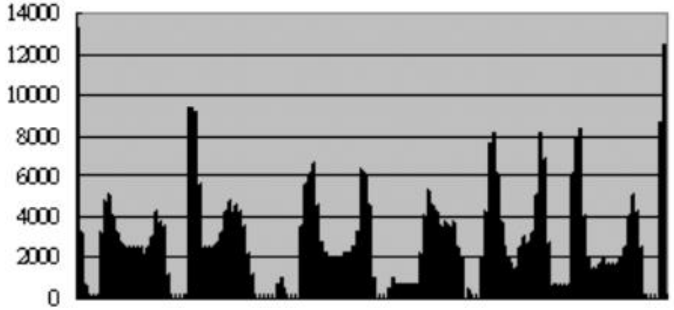
- Diagonal projection

$$D[k] = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} B[i,j] \,\Big|\, i + j = k$$

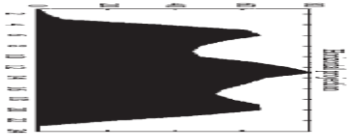


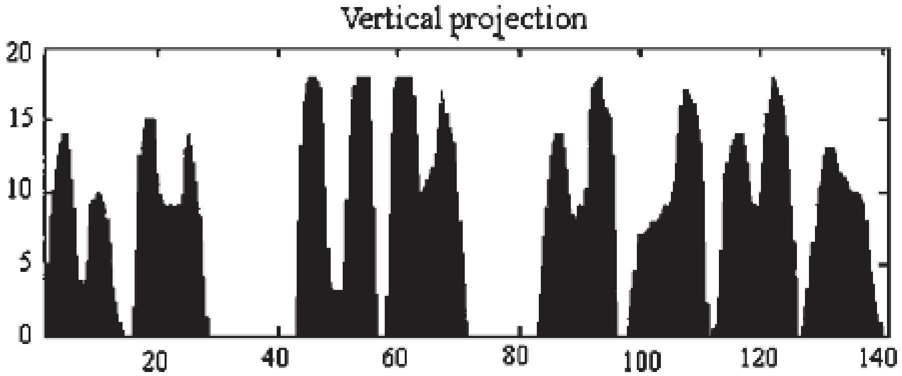



2K·5786


14000
12000
10000
8000
6000
4000
2000
0

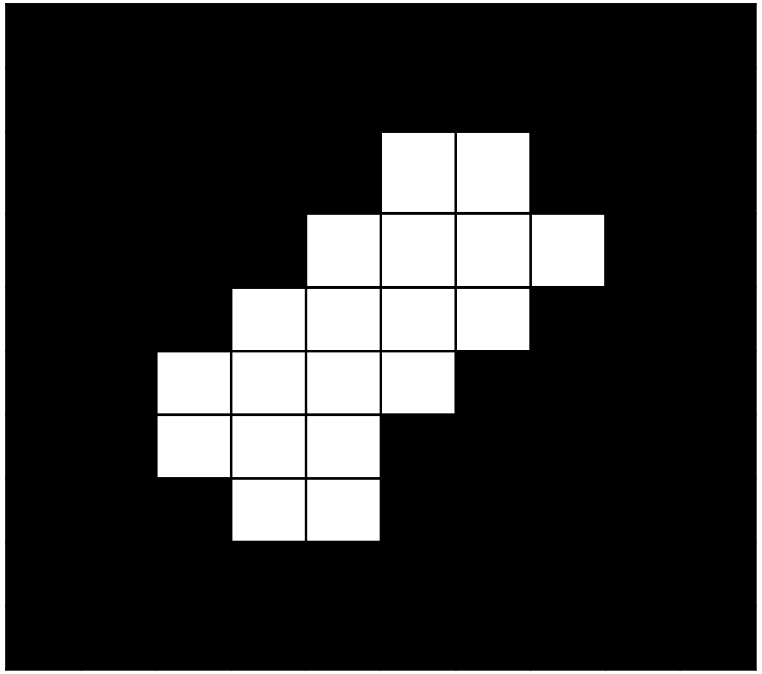

45 HB 9394




Vertical projection
20
15
10
5
0
20
40
60
80
100
120
140

GEOMETRIC PROPERTIES

size and shape features of object can help the system recognize them



- Size/Area
  - Area is a count of pixels in region R

$$A = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} B[i,j]$$

- Centroid
  - center of object area

$$\bar{x} = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} jB[i,j]}{A} \qquad \bar{y} = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} iB[i,j]}{A}$$

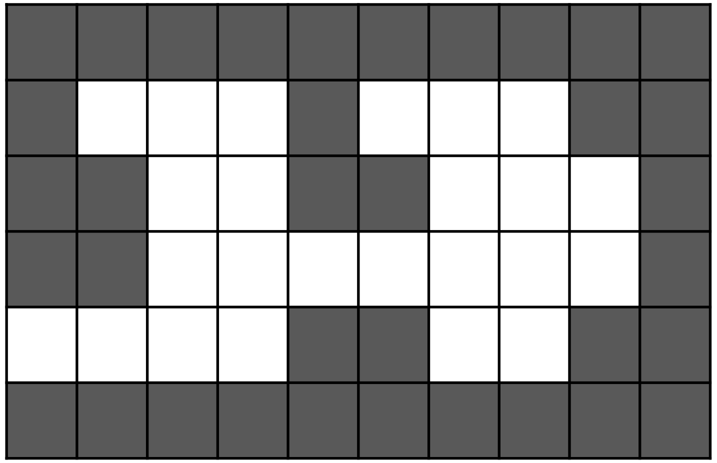
* Assume an image has only one object

RUN-LENGTH ENCODING

- Run-length encoding
  - compact representation of a binary image
  - Compose of start position and length of the runs of 1 pixels
  - Commonly use two approach

Start and length of 1 runs:

$r_{0.0}$ = (0,0)
$r_{1.0}$ =(1,3) $r_{1.1}$ = (5,3)
$r_{2.0}$ =(2,2) $r_{2.1}$ = (6,3)
$r_{3.0}$ =(2,7)
$r_{4.0}$ =(0,4) $r_{4.1}$ = (6,2)
$r_{5.0}$ =(0,0)



Length of 1 and 0 runs:

0 10
0 1 3 1 3 2
0 2 2 2 3 1
0 2 7 1
4 2 2 2
0 10

* $r_{i.k}$ is length of the *k*th run in the *i*th row

- Compute area with run-length encoding
  - Sum length of 1 runs from run-length encoding (in case of only one object)

Start and length of 1 runs:

$r_{0.0}$ = (0,0)

$r_{1.0}$ =(1,3) $r_{1.1}$ = (5,3)

$r_{2.0}$ =(2,2) $r_{2.1}$ = (6,3)

$r_{3.0}$ =(2,7)

$r_{4.0}$ =(0,4) $r_{4.1}$ = (6,2)

$r_{5.0}$ =(0,0)

Length of 1 and 0 runs:

0 10

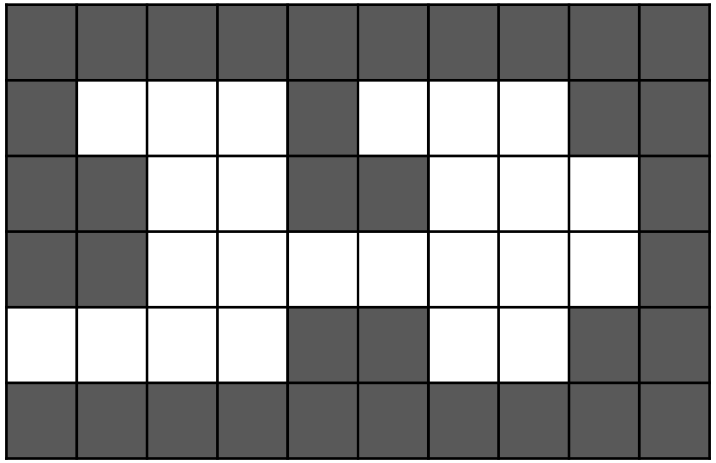
0 1 3 1 3 2

0 2 2 2 3 1

0 2 7 1

4 2 2 2

0 10



- Compute Horizontal projection from run-length code
  - Sum length of 1 runs from run-length encoding in each line

Start and length of 1 runs:

$r_{0.0} = (0,0)$

$r_{1.0} = (1,3)$ $r_{1.1} = (5,3)$

$r_{2.0} = (2,2)$ $r_{2.1} = (6,3)$

$r_{3.0} = (2,7)$

$r_{4.0} = (0,4)$ $r_{4.1} = (6,2)$

$r_{5.0} = (0,0)$

Length of 1 and 0 runs:

0 10

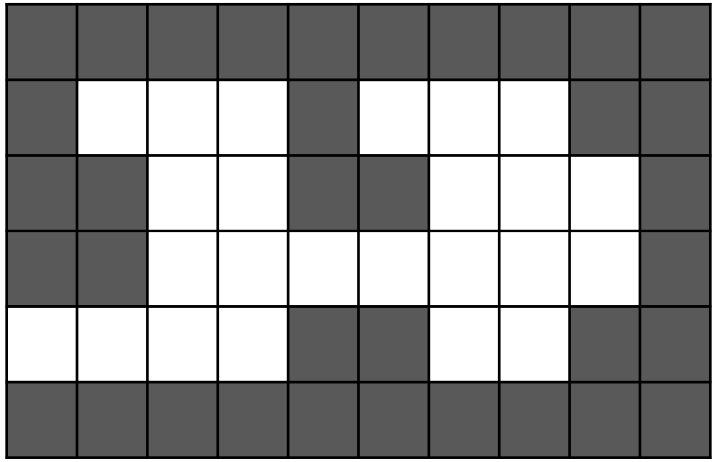
0 1 3 1 3 2

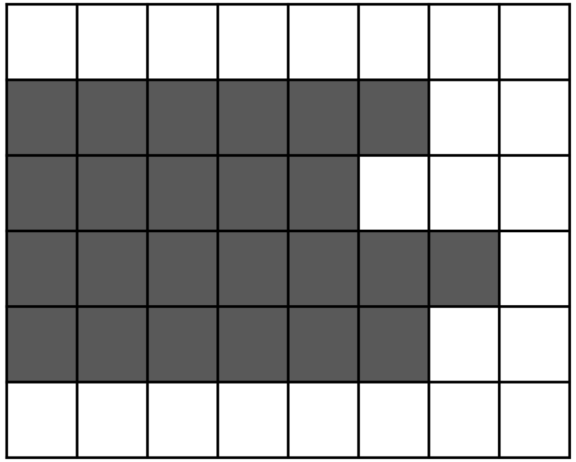
0 2 2 2 3 1

0 2 7 1

4 2 2 2

0 10





BINARY ALGORITHM

DEFINITION

- Neighbors
  - Pixels which are share border / corner together

	i, j+1	
i-1, j	i, j	i+1, j
	i, j-1	

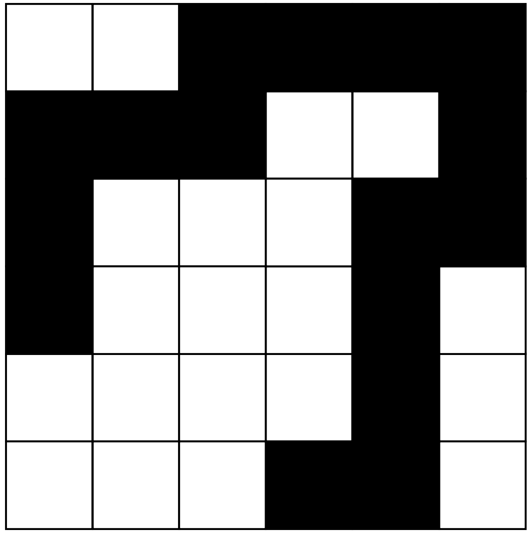
4 - neighbors

i-1, j+1	i, j+1	i+1, j+1
i-1, j	i, j	i+1, j
i-1, j-1	i, j-1	i+1, j-1

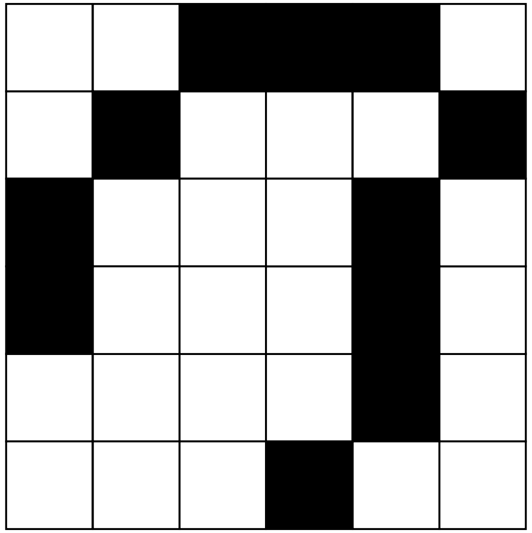
8 - neighbors

Two pixels are 4-neighbors if they share a common boundary
Two pixels are 8-neighbors if they share at lease one corner

- Path
  - A path from pixel $[i_0,j_0]$ to pixel $[i_n,j_n]$ is sequence of pixels indices $[i_0,j_0],[i_1,j_1]$,…, $[i_n,j_n]$ such that $[i_k,j_k]$ is a neighbor of $[i_{k+1},j_{k+1}]$ for all k with $0 \leq k \leq n-1$

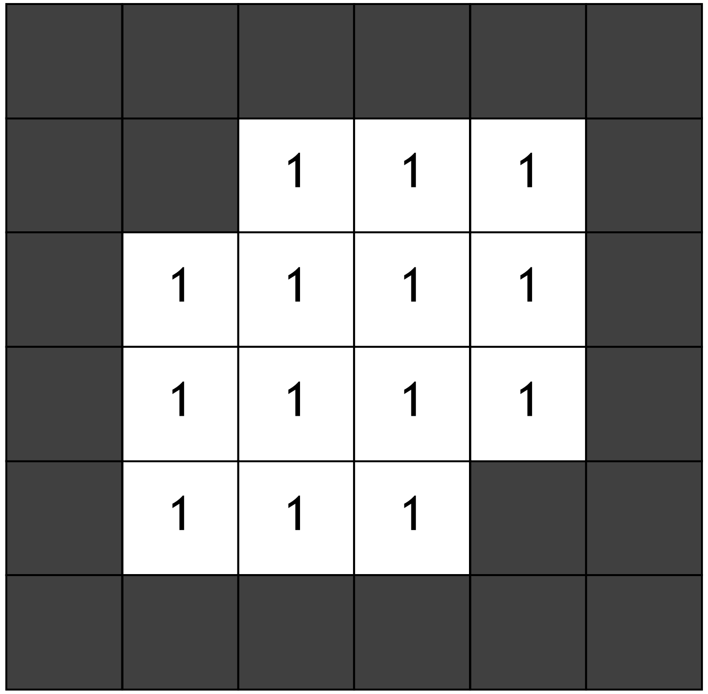


4-path (use 4 connection)



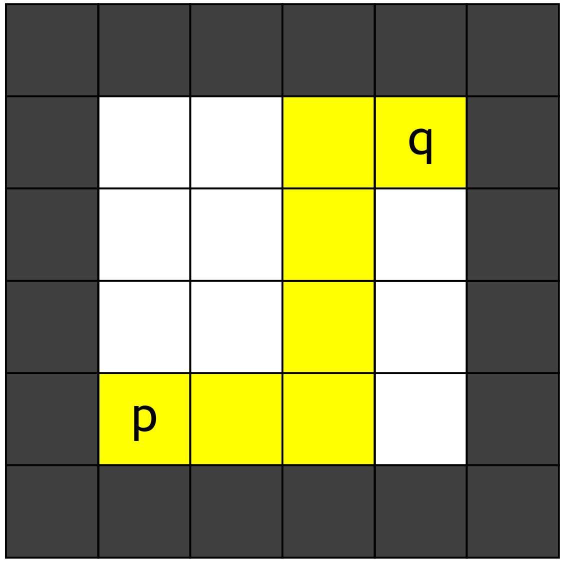
8-path (use 8 connection)

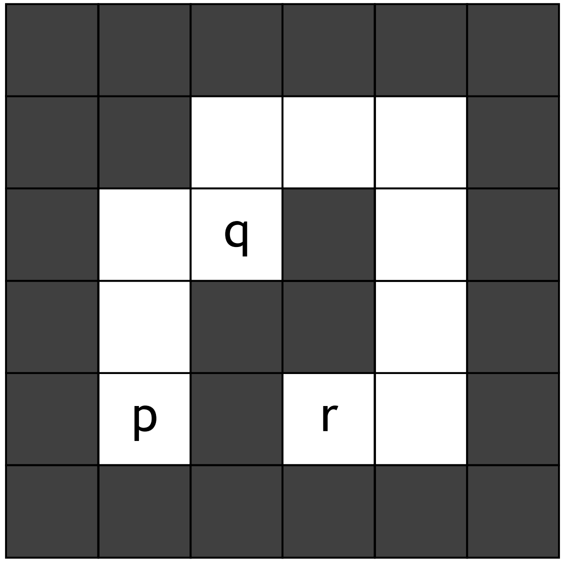
- Foreground
  - Set of all ‘1’ pixel in image denoted by **S**



Connectivity

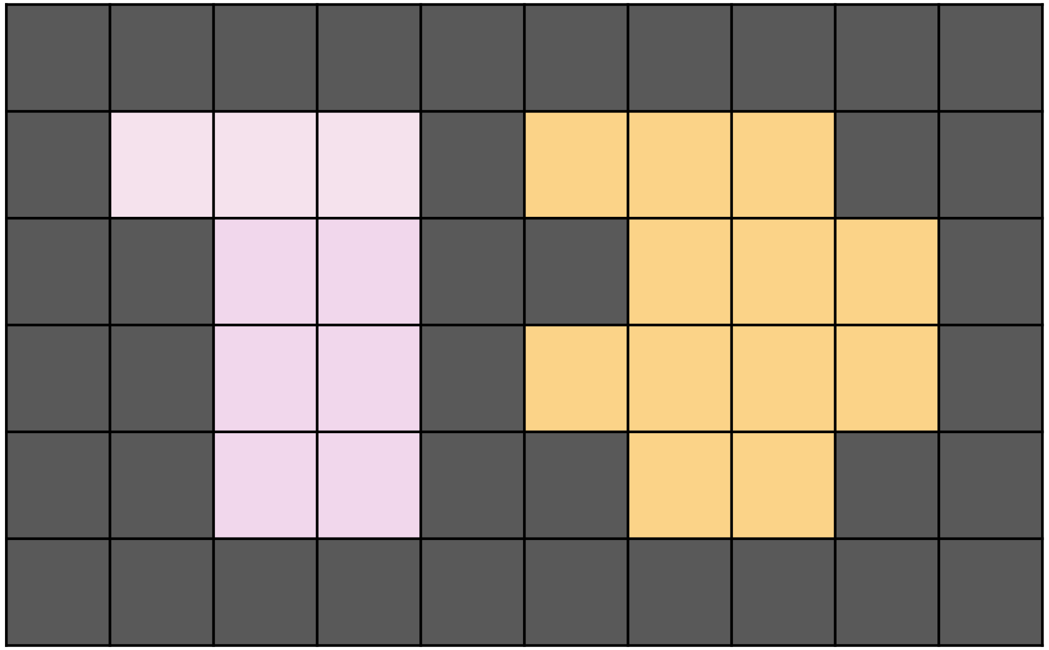
- Pixels pЄS is connected to q ЄS if there is a path from p to q consisting entirely of pixel of S





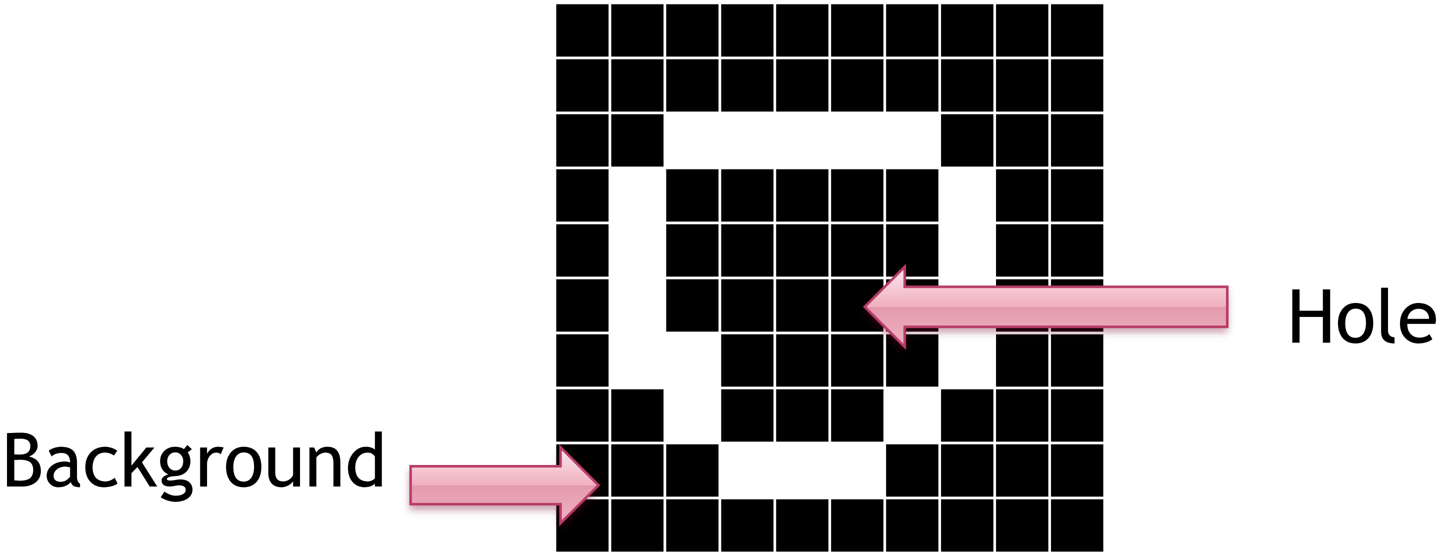
- For p,q,r in S
- If p is connected to q then q is connected to p
- If p is connected to q and q is connected to r then p is connected to r

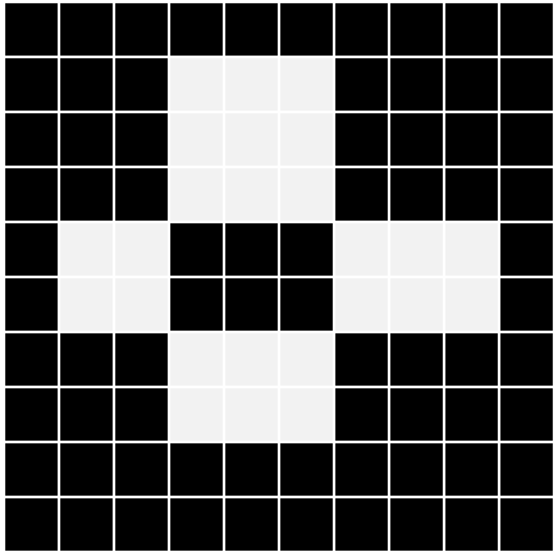
- Connected Component
  - Set of pixels in which each pixels are connected to all other pixels



2 components

- Background
  - Set of all connected components of $\bar{S}$ (the complement of S) that have points on border of image
  - The other connected components which have no points on border called holes



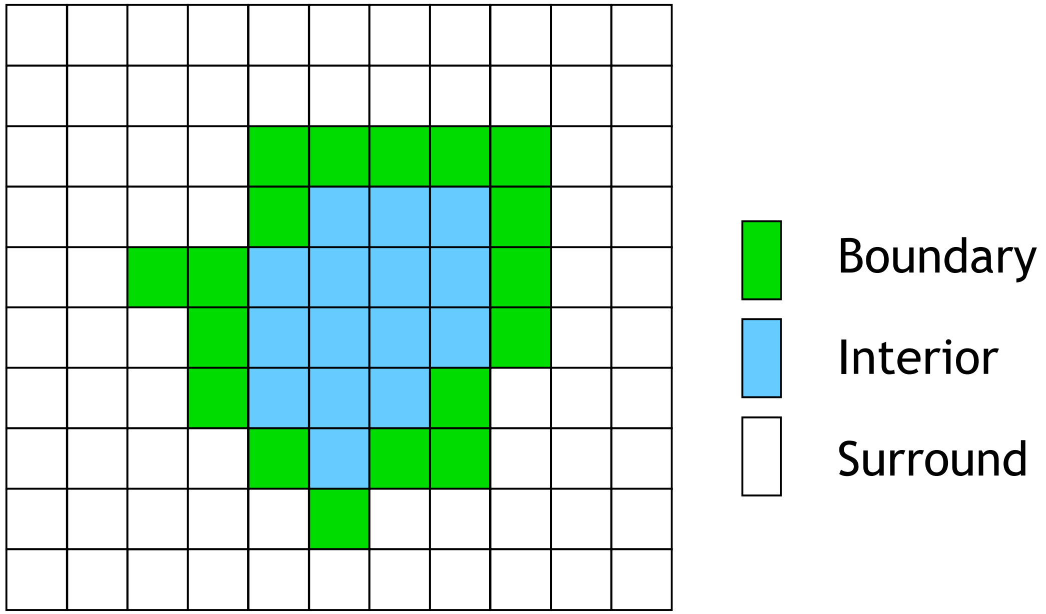


How many object?
How many hole?

4-connected for foreground and background
4 objects and 1 hole
8-connected for foreground and background
1 object and no hole

To avoid this, different connectedness should be used for objects and backgrounds

- Boundary
  - Set of pixels of S that have 4 neighbors in $\bar{S}$ (complement of S) denoted by S’
- Interior
  - Set of pixels of S that are not in its boundary (S-S’)
- Surrounds
  - Region T surrounds region S if any 4-path from any point of S to the border of the picture must intersect T (or S inside T)



COUNTING OBJECT

- Find the number of corner patterns occur in image

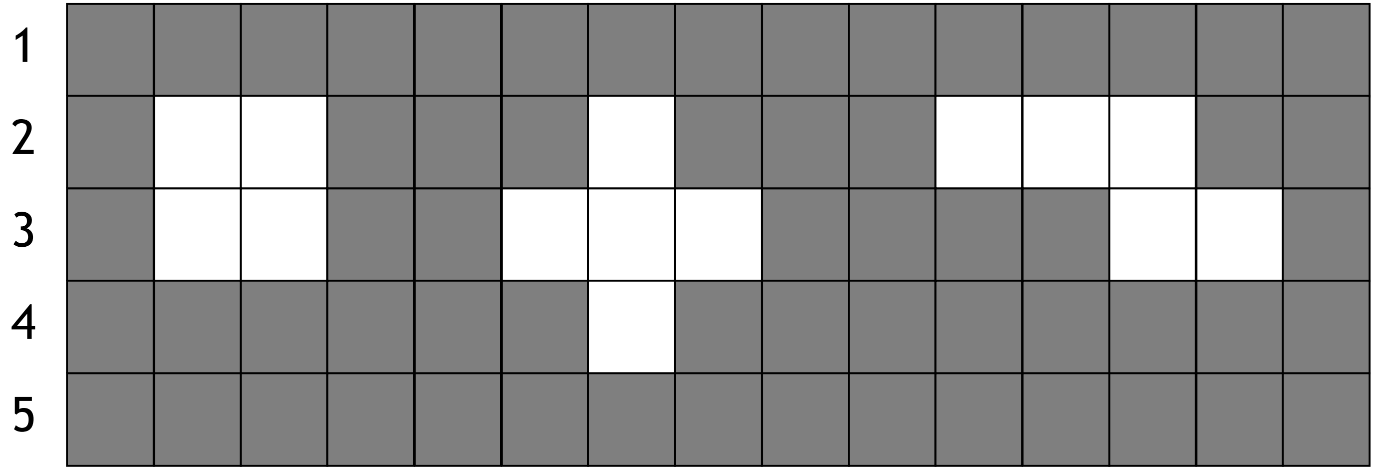
0	0		0	0		1	0		0	1			1	1		1	1		0	1		1	0
0	1		1	0		0	0		0	0			1	0		0	1		1	1		1	1

E: external corners

I: internal corners

Number of object = (number of 'E'- number of 'I')/4

Note: each object must be a 4-connected set of 1-pixels with no interior holes



	Line 1	Line 2	Line 3	Line 4	Sum
‘E’					
‘I’					

Number of object =

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

E: external corners

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

I: internal corners

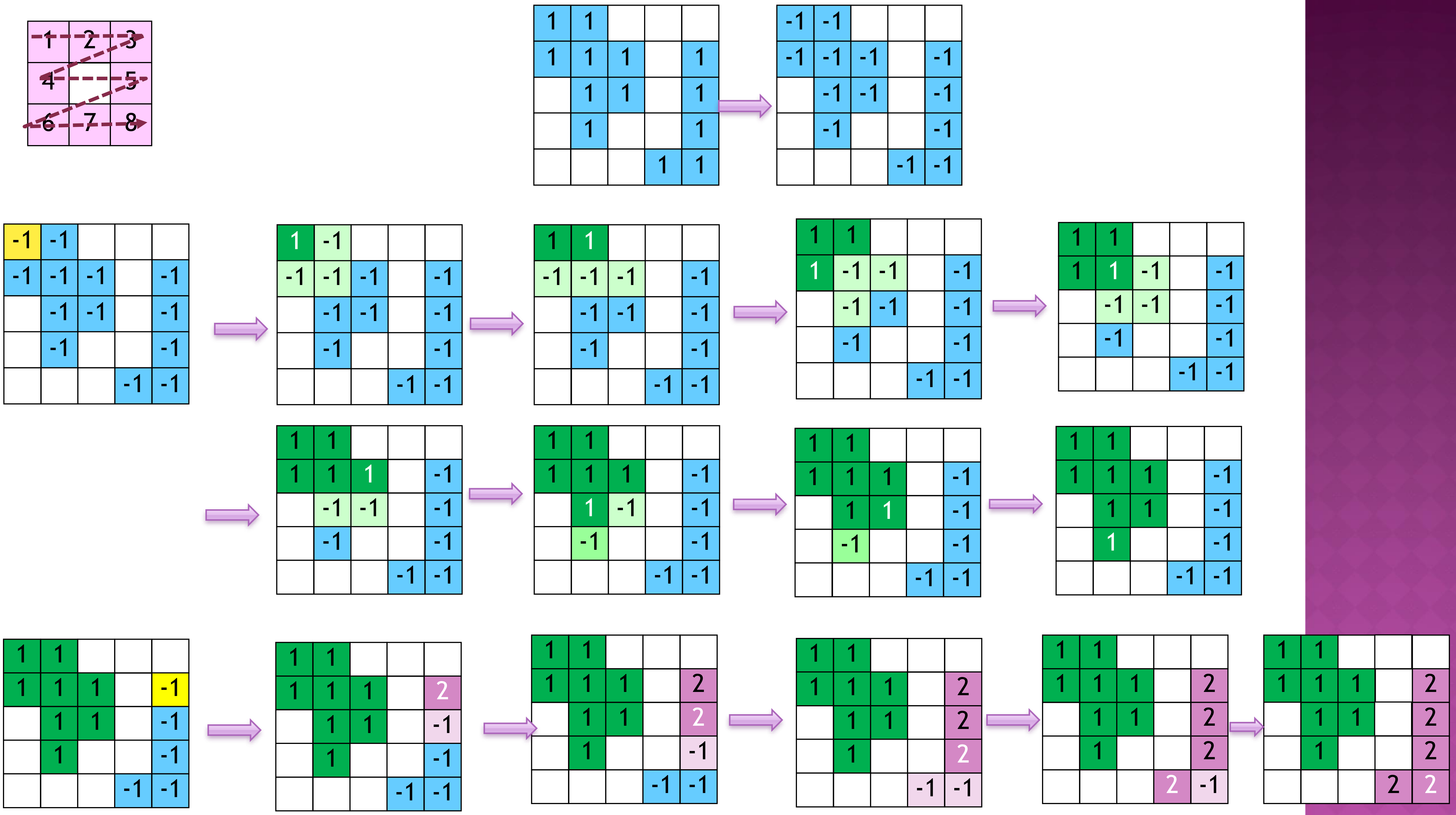
COMPONENT LABELING

- A recursive labeling algorithm
  1. Negate the binary image 1 become -1
  2. Finding a pixel whose value is -1
  3. Assigning it a new label
  4. Finding its neighbor which value is -1. If found go to 3 to recursive else go to 2 to find new object

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 2 3
4 5
6 7 8

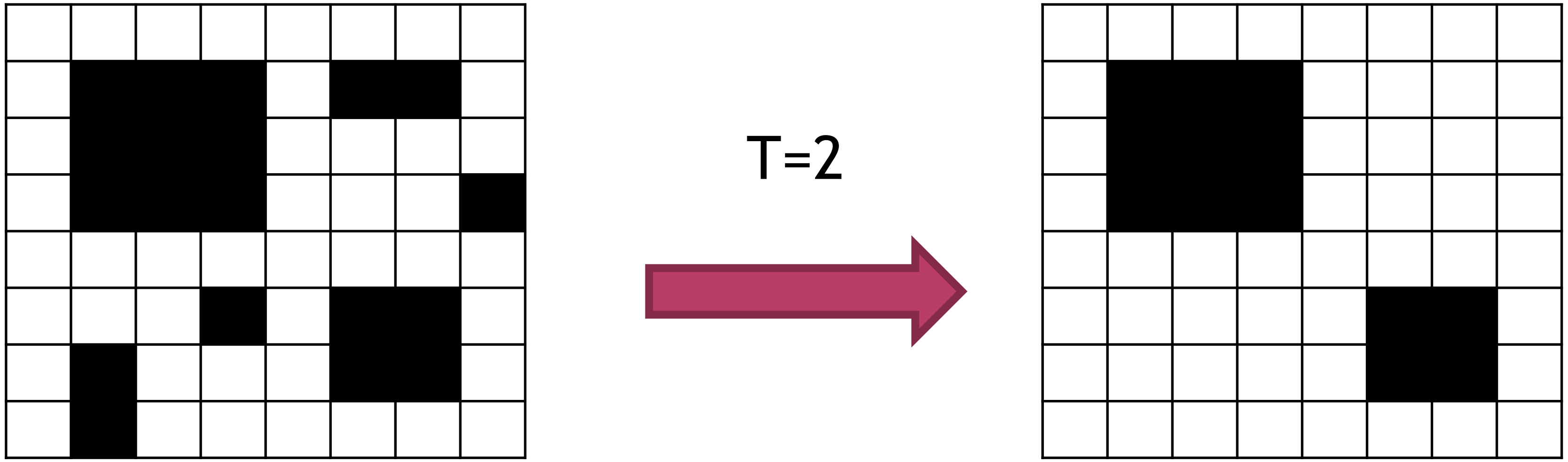
- Row by Row labeling algorithm
  - Row by Row labeling algorithm
    1. Scan image left to right, top to down
    2. If the pixel is 1 then assign a new label to pixel and enter label to tree structure with parent = 0 and go to 3
    3. If the pixel is label then
       - If right is 1 then assign its label to right
       - If right is other label assign right to lower label and change root of lower label to higher label
       - If lower is 1 then assign its label to lower
    4. Scan picture again, replace each label with root of label tree

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

label										
parent										

SIZE FILTER

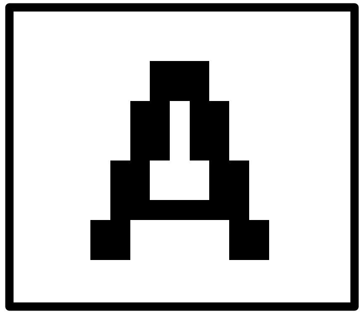
- There are some regions in an image that are due to noise. Usually, the noise regions are small.
- If the interesting objects are of size greater than T pixel
  - Size filter used to remove noise after component labeling
- All components which size below T are removed



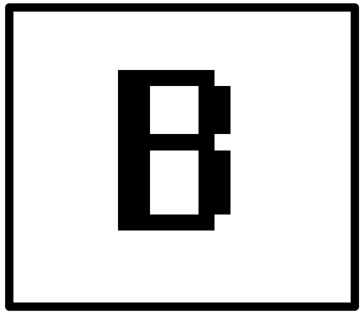
EULER NUMBER

- Euler Number = number of components – number of holes

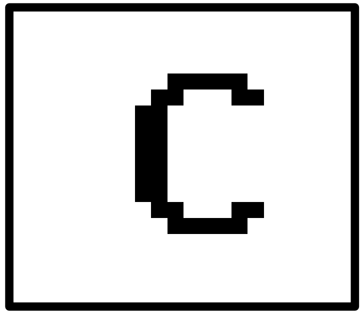
$$E = C - H$$



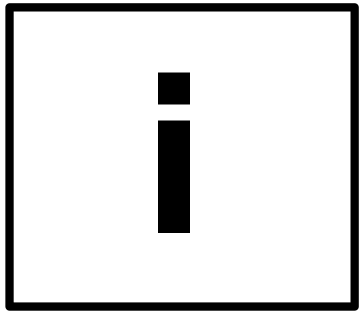
E = 0



E = -1



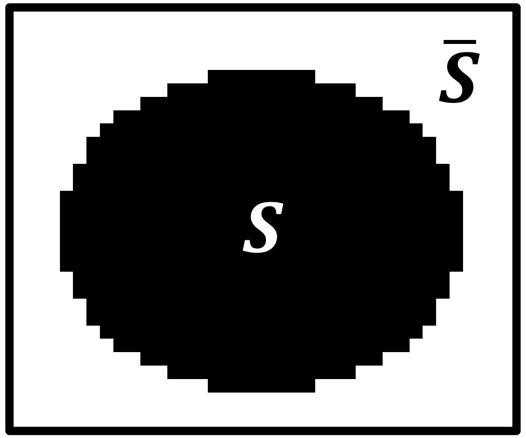
E = 1

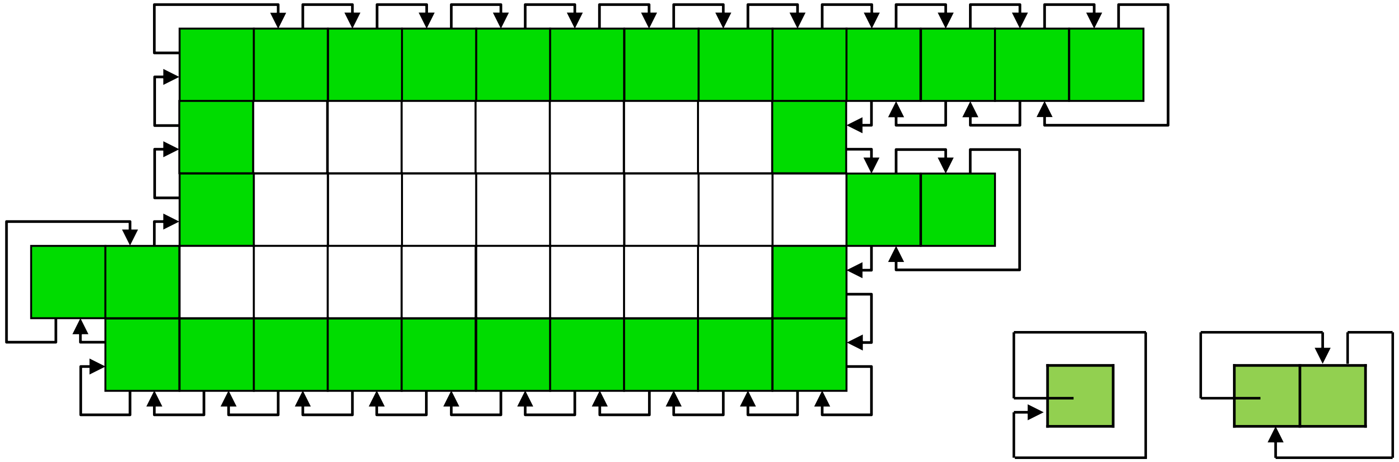


E = 2

REGION BOUNDARY

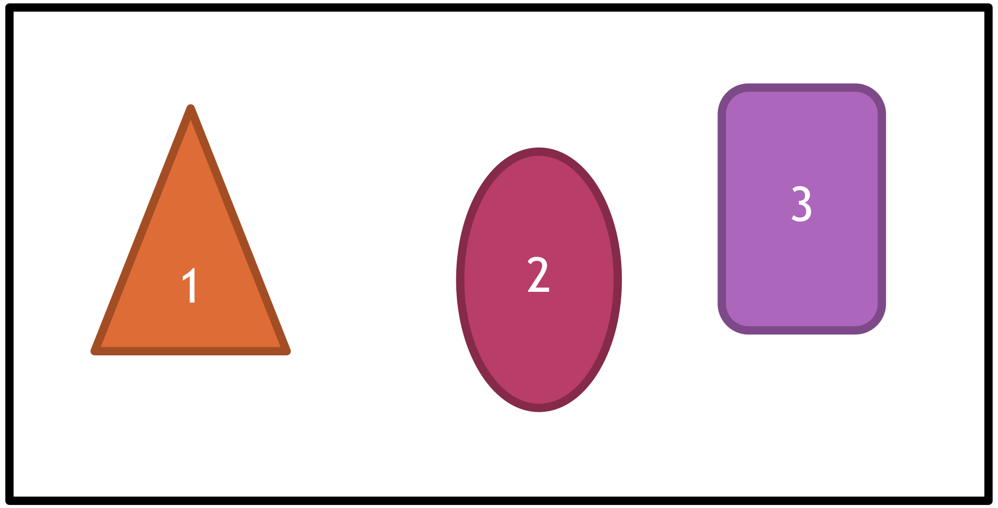
- Boundary of connected component S – set of pixels of S that are adjacent to $\bar{S}$
- Defined boundary by tracking pixels on the boundary in a particular order (ex clockwise sequence)
  - Boundary following algorithm - selects a starting pixel and tracks the boundary until it comes back to the starting pixel





AREA

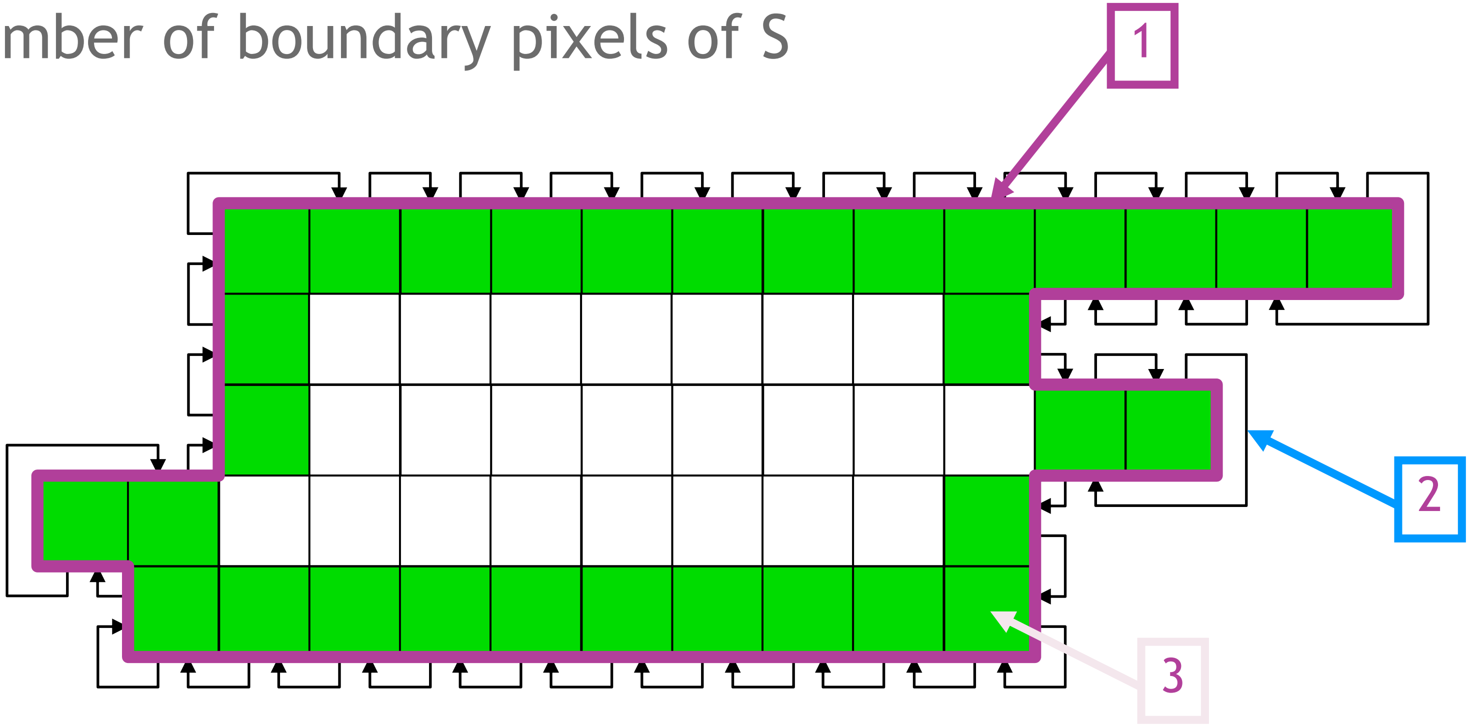
- Area - number of pixels in S
- For many components S1, S2, and S3 the area of each component is the number of pixels of each component



can be obtained in one scan by counting the pixels of each labeling component

PERIMETER

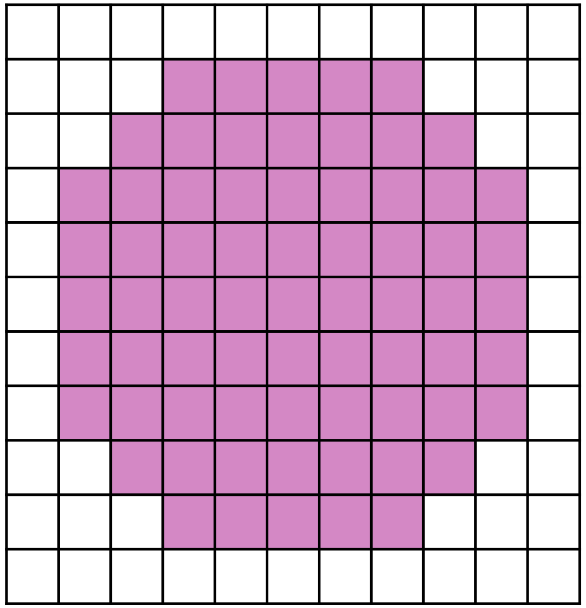
- The perimeter of a component may be defined in many different ways
  1. The sum of cracks separating pixels of S from S
  2. The number of steps taken by a boundary following algorithm
  3. The number of boundary pixels of S



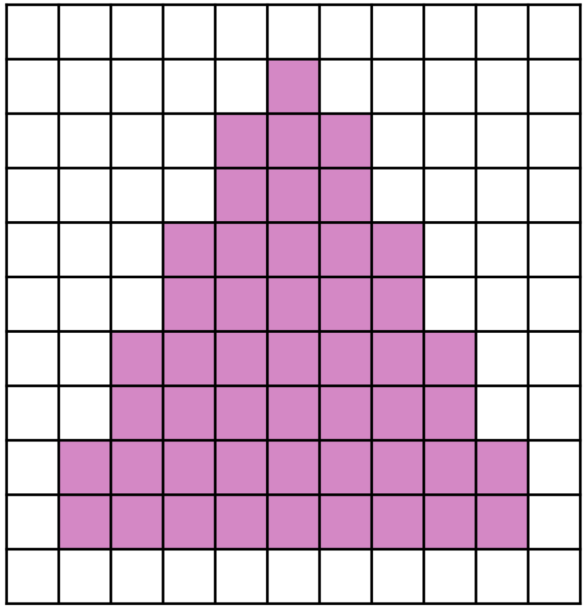
CIRCULARITY

- Measure the circularity of the component

$$C = \frac{|P|^2}{A}$$



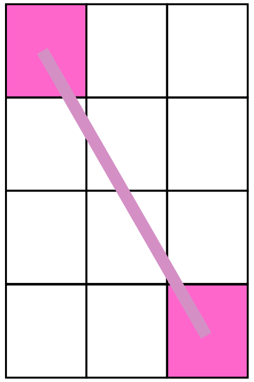
$C =$



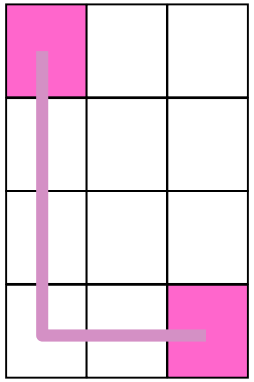
$C =$

DISTANCE MEASURE

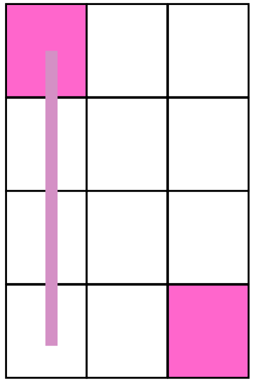
- Euclidean $$D_{Euclidean}([i_1, j_1],[i_2, j_2]) = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2}$$
- City-block $$D_{City}([i_1, j_1],[i_2, j_2]) = | i_1 - i_2 | + | j_1 - j_2 |$$
- Chessboard $$D_{Chess}([i_1, j_1],[i_2, j_2]) = \max(| i_1 - i_2 |, | j_1 - j_2 |)$$



$D_{Euclidean}$ =



$D_{city\text{-}block}$ =



$D_{chessboard}$ =

Euclidean

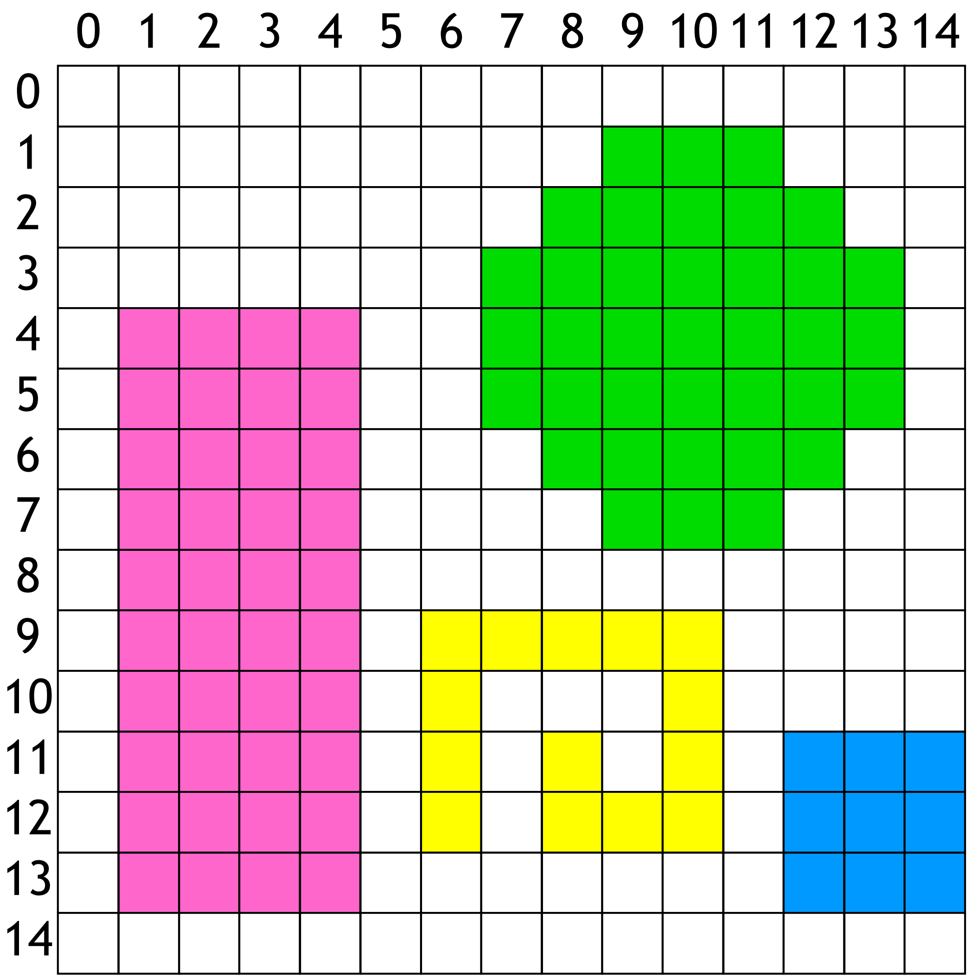
4.2	3.6	3.2	3.0	3.2	3.6	4.2
3.6	2.8	2.2	2.0	2.2	2.8	3.6
3.2	2.2	1.4	1.0	1.4	2.2	3.2
3.0	2.0	1.0	0	1.0	2.0	3.0
3.2	2.2	1.4	1.0	1.4	2.2	3.2
3.6	2.8	2.2	2.0	2.2	2.8	3.6
4.2	3.6	3.2	3.0	3.2	3.6	4.2

City-block

6	5	4	3	4	5	6
5	4	3	2	3	4	5
4	3	2	1	2	3	4
3	2	1	0	1	2	3
4	3	2	1	2	3	4
5	4	3	2	3	4	5
6	5	4	3	4	5	6

Chessboard

3	3	3	3	3	3	3
3	2	2	2	2	2	3
3	2	1	1	1	2	3
3	2	1	0	1	2	3
3	2	1	1	1	2	3
3	2	2	2	2	2	3
3	3	3	3	3	3	3



Region Boundary
Area
Row of Centroid
Column of Centroid
Perimeter Length
Circularity
Horizontal Projection
Vertical Projection

DISTANT TRANSFORM

- Show the minimum distance between a pixel of an object to the background
- The object region S and background $\bar{S}$
  - First iteration

$$f^0[i,j] = f[i,j]$$

  - Other iteration
    - m – iteration number for all pixels
    - [u,v] – the 4 neighbor of [i,j]

$$f^m[i,j] = f^0[i,j] + \min(f^{m-1}[u,v])$$

f^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	1	1	1	1			

f^1

	1	1	1	1	1	1			
		1	2	2	2	2	1		
			1	2	2	2	2	1	
		1	2	2	2	2	1		
	1	1	1	1	1	1			

f^2

	1	1	1	1	1	1			
		1	2	2	2	2	1		
			1	2	3	3	2	1	
		1	2	2	2	2	1		
	1	1	1	1	1	1			

Distant transform

MEDIAL AXIS

- Distance $d([i,j],\bar{S})$ is locally maximum if

$$d([i,j],\bar{S}) \geq d([u,v],\bar{S})$$

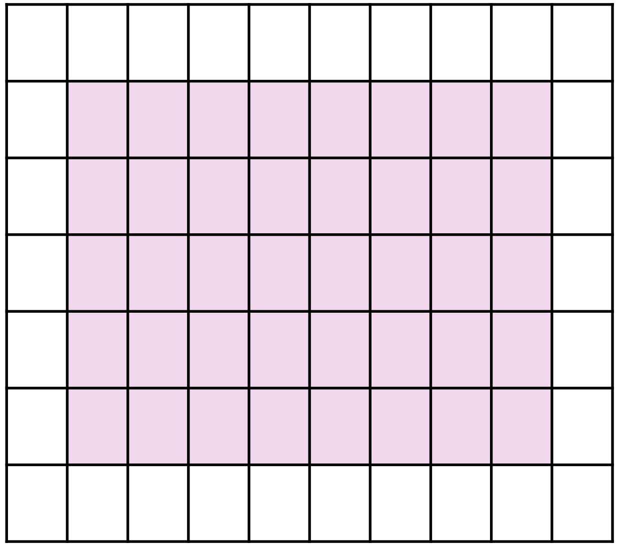
  - Note : [u,v] – the 4 neighbor of [i,j]

	1	1	1	1	1	1			
		1	2	2	2	2	1		
			1	2	3	3	2	1	
		1	2	2	2	2	1		
	1	1	1	1	1	1			

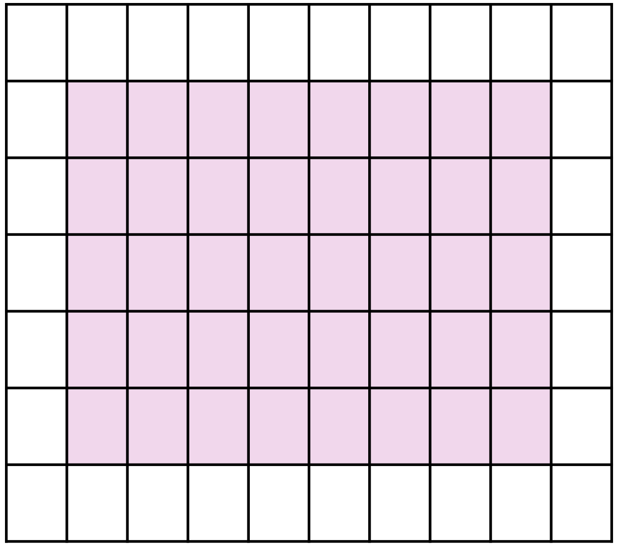
- Set of locally maximum pixels S* is medial axis / skeleton / symmetric axis

	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	

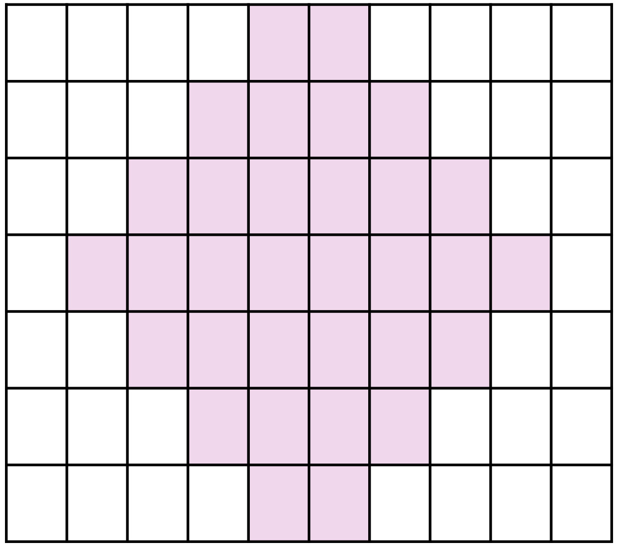
Distant transform

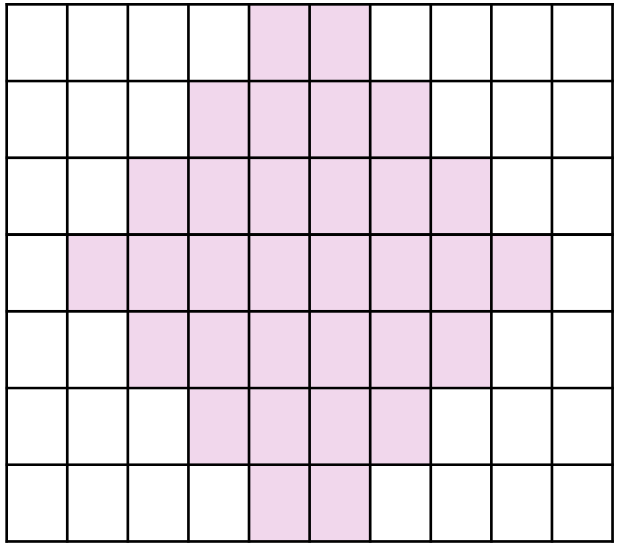


Medial axis



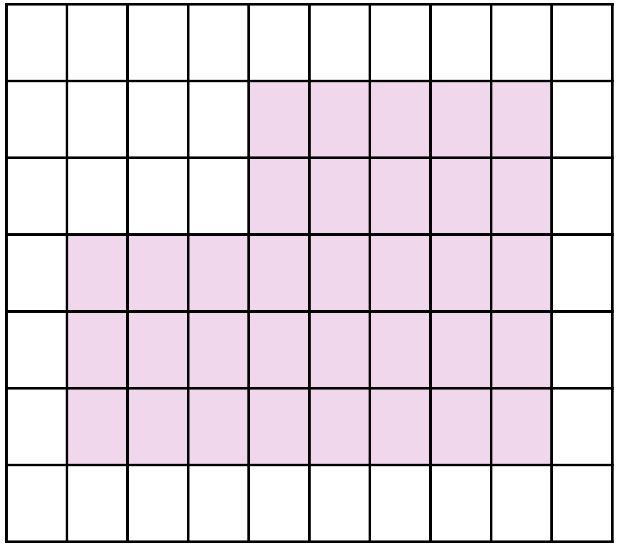
				1	1				
			1	1	1	1			
		1	1	1	1	1	1		
	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1		
			1	1	1	1			
				1	1				



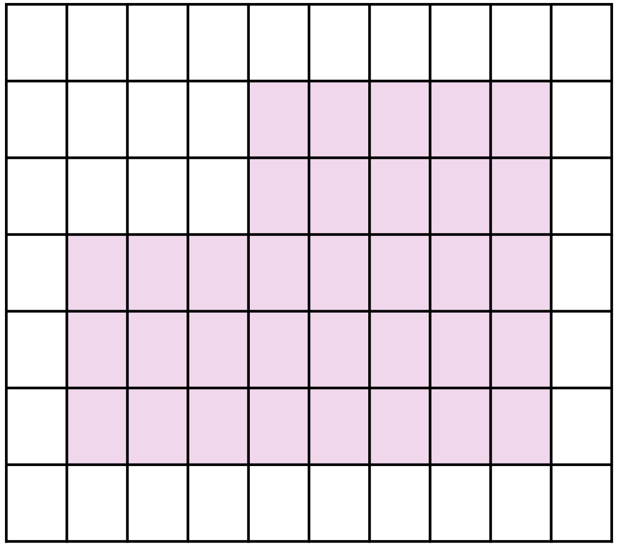


				1	1	1	1	1	
				1	1	1	1	1	
	1	1	1	1	1	1	1	1	
	1	1	1	1	1	1	1	1	
	1	1	1	1	1	1	1	1	

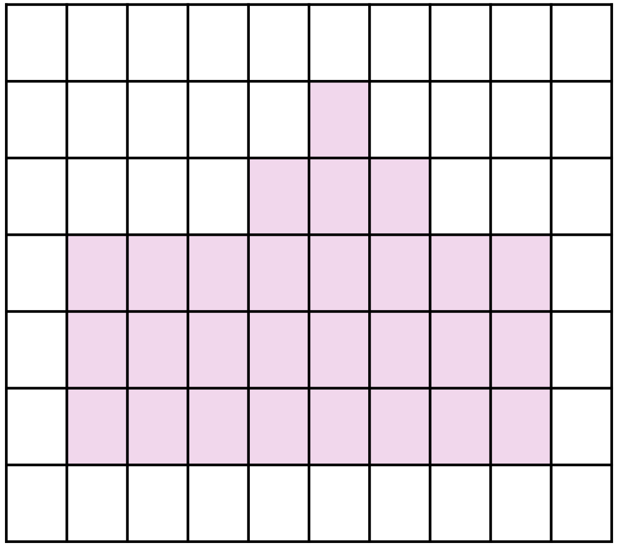
Distant transform

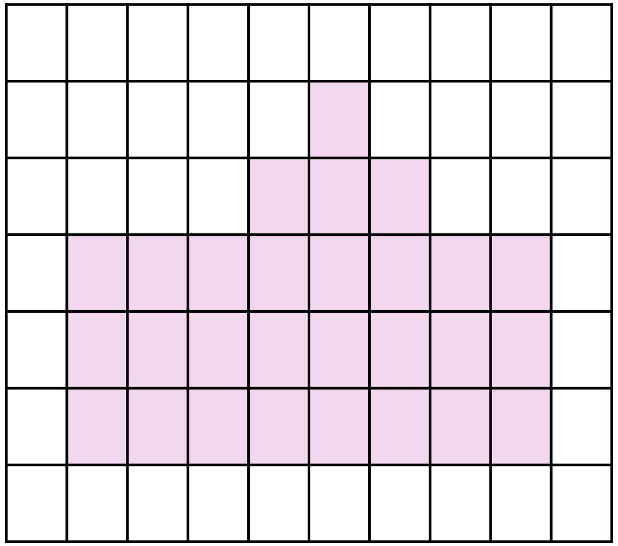


Medial axis



					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	

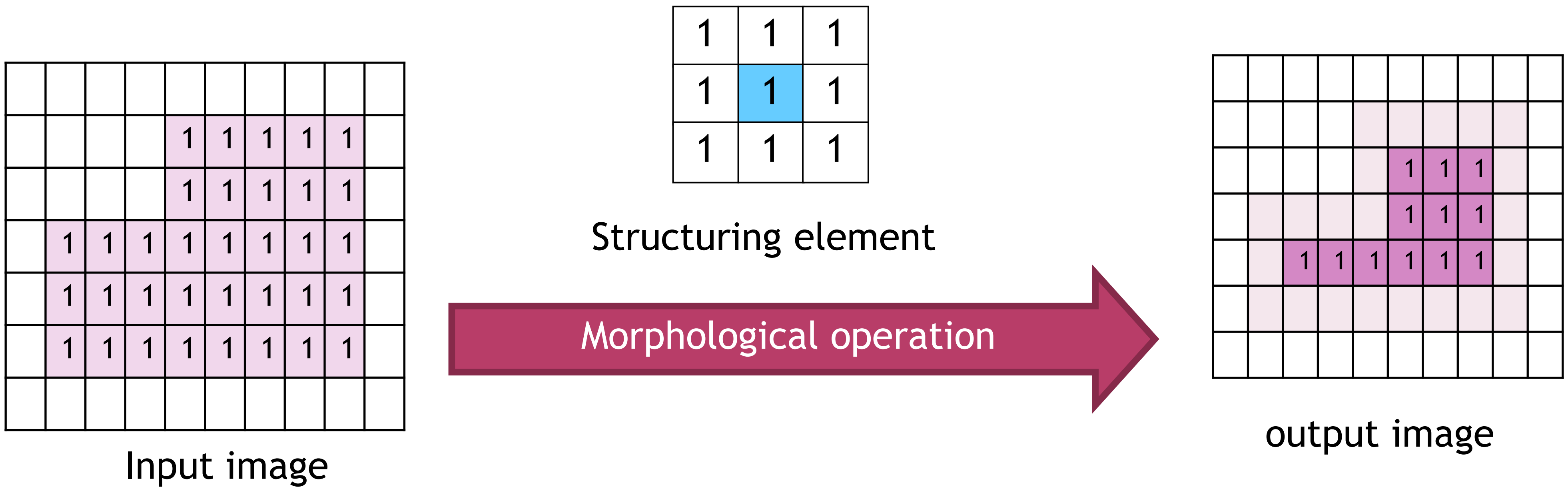




MORPHOLOGY

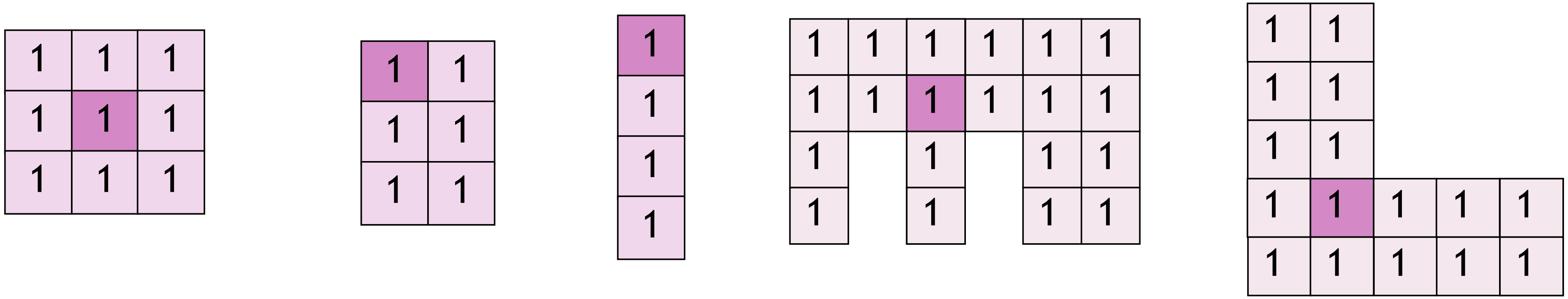
MORPHOLOGY

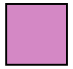
- image processing operations that process images based on shapes
- Morphological operations apply a structuring element to an input image, creating an output image of the same size



STRUCTURE ELEMENT

- matrix or a small-sized template that is used to traverse an image
- can be of any shape, usually smaller than input image



 Origin :reference point to place anywhere on image
:often the central pixel of symmetric element

There are a number of common structuring element such as

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

BOX(3,5)

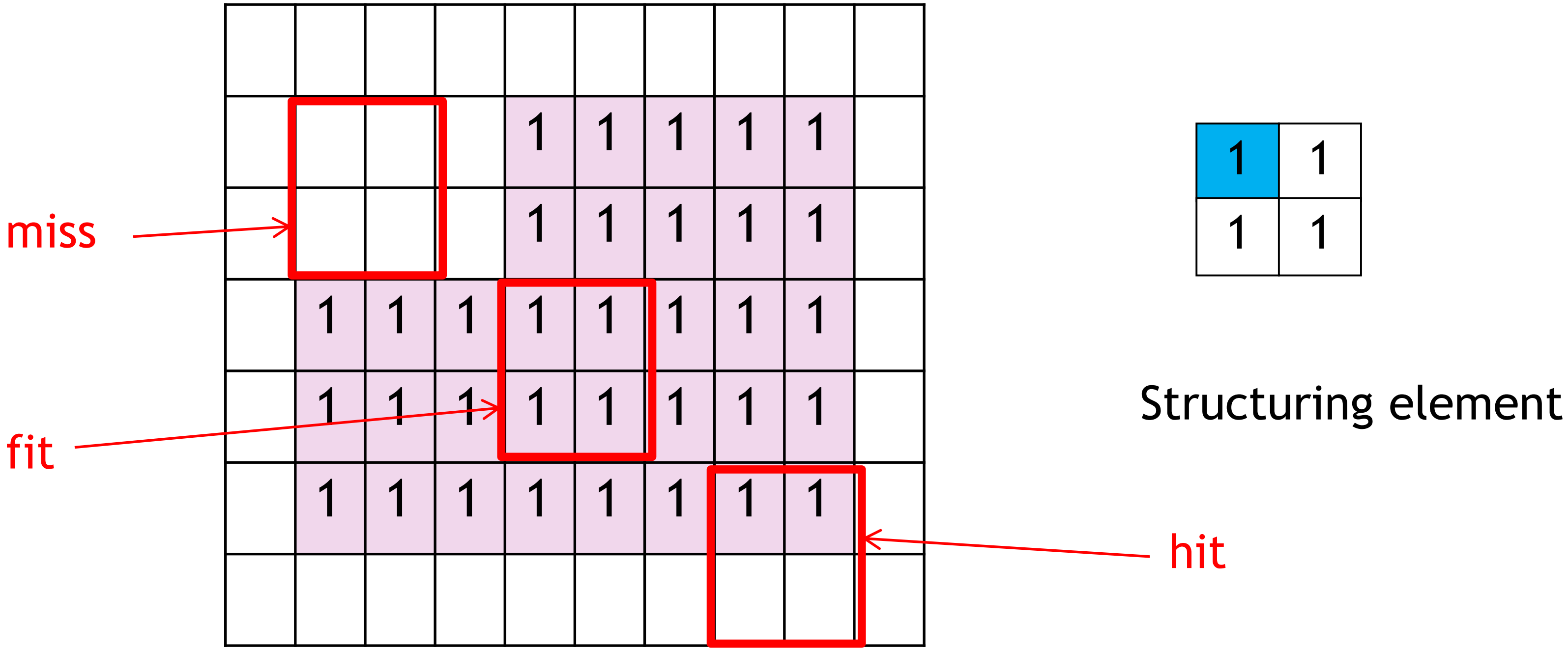
	1	1	1	
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
	1	1	1	

DISK(5)

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

RING(5)

- Fit
  - All the pixels in the structuring element cover the pixels of the object
- Hit
  - At least one of the pixels in the structuring element cover the pixels of the object
- Miss
  - No pixel in the structuring element cover the pixels of the object



MORPHOLOGICAL OPERATION

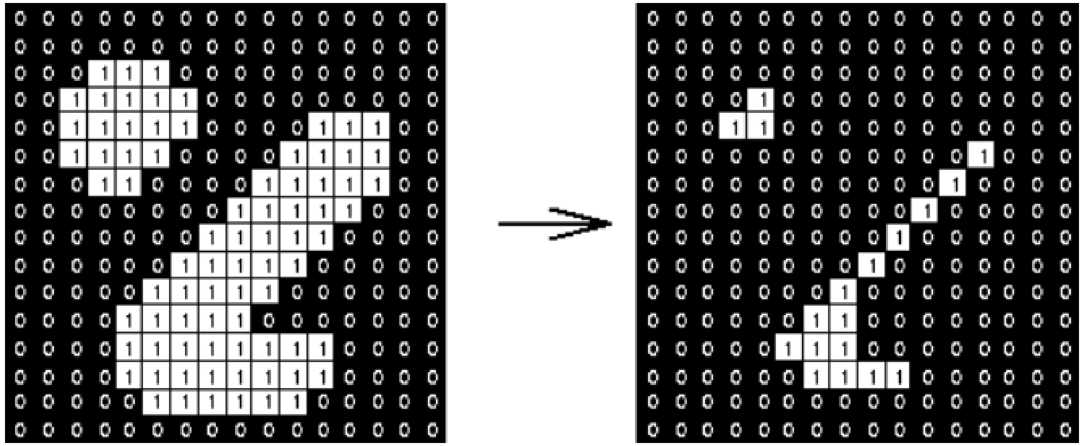
Dilation	Erosion
Opening	Closing

EROSION

- The erosion of a binary image B by a structuring element S denote by

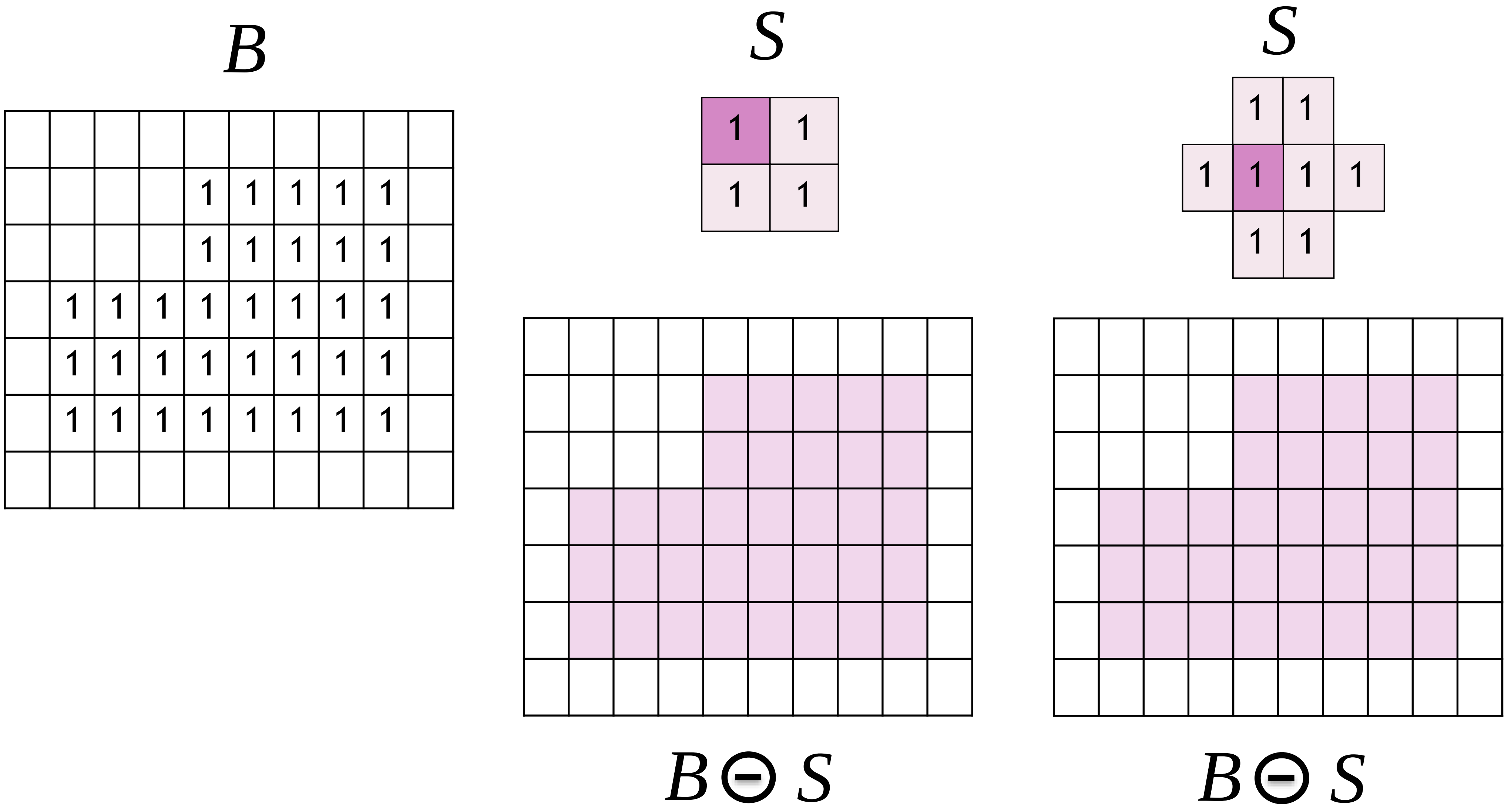
$$B \ominus S$$

- produces a new binary image G = B S with ones in all locations (x,y) if structuring element s fits the input image
  - i.e. G(x,y) = 1 is S fits B and 0 otherwise



Erosion: a 3×3 square structuring element
(www.cs.princeton.edu/~pshilane/class/mosaic/).

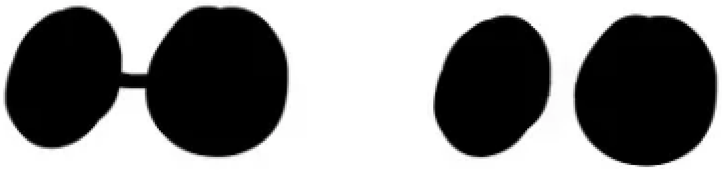
Erosion makes region smaller



B
S
S
$B \ominus S$
$B \ominus S$

- Morphological erosion reduces the size of regions by
  - stripping away the outer boundaries of regions.
  - The holes and gaps between different regions become larger,
  - Small details are eliminated

processed images after erosion using 3x3 and 5x5 structuring elements



It can split apart joint objects

It can split apart joint objects

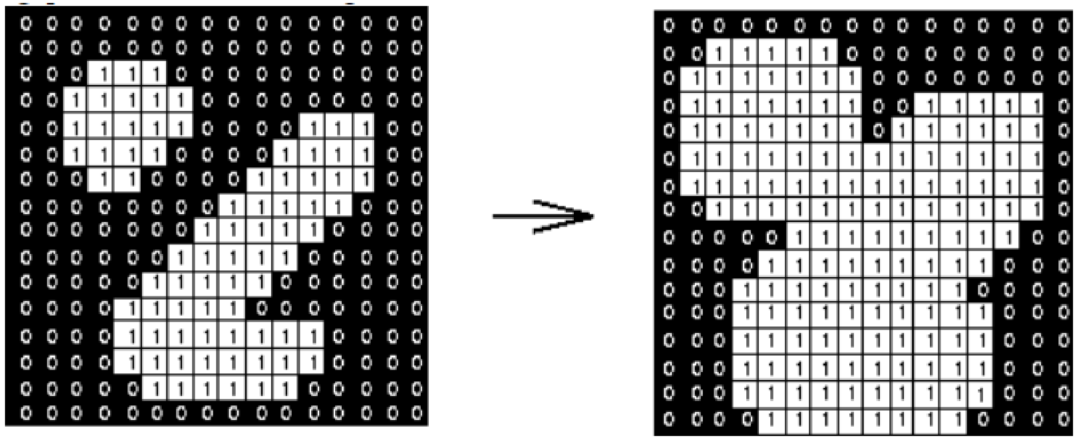
https://towardsdatascience.com/understanding-morphological-image-processing-and-its-operations-7bcf1ed11756

DILATION

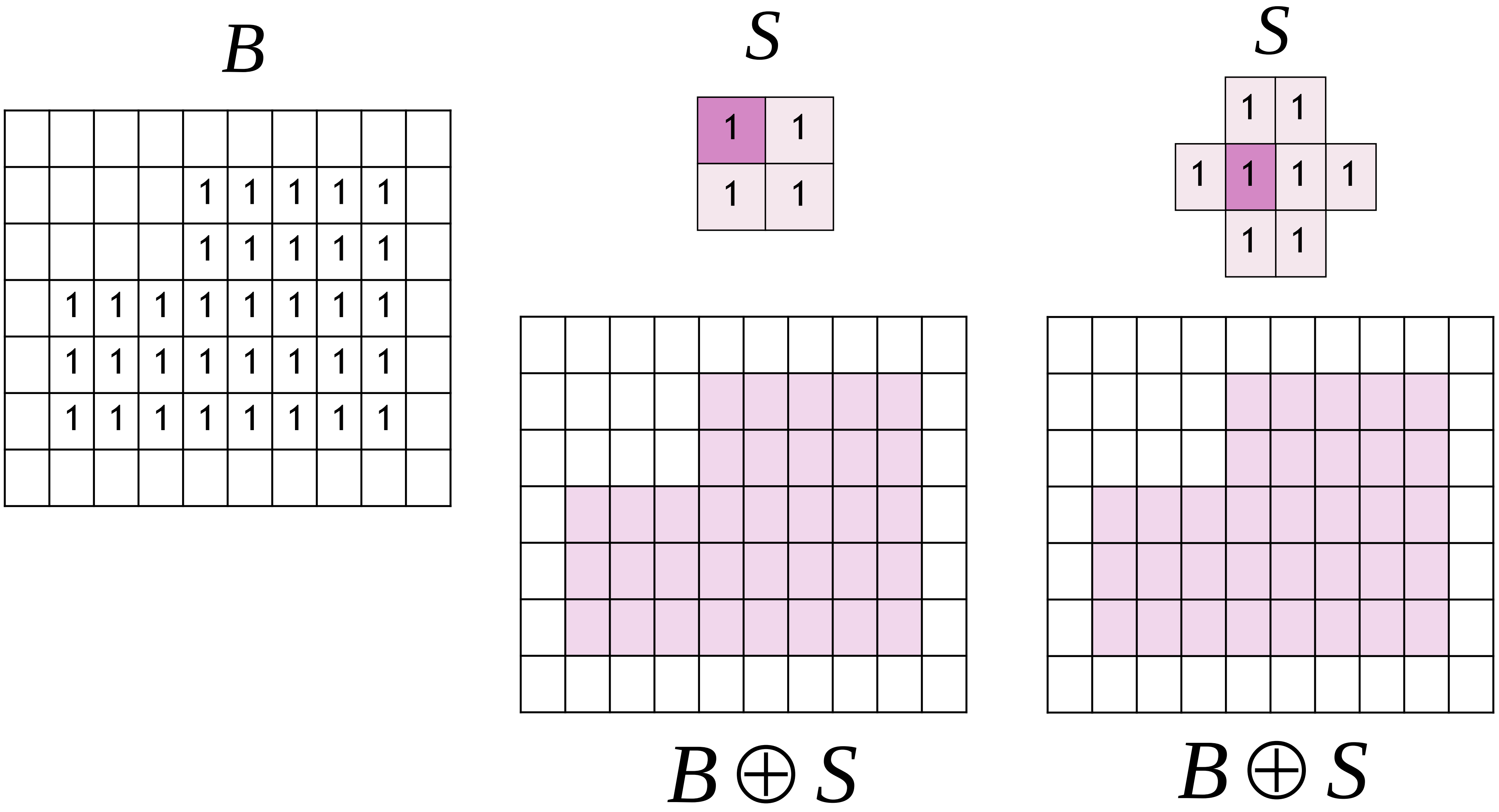
- The dilation of a binary image B by structuring element S denote by

$$B \oplus S$$

- produces a new binary image $G = B \oplus S$ with ones in all locations (x,y) of a structuring element's if structuring element's origin hits the input image,
  - i.e. G(x,y) = 1 is origin of S hits B and 0 otherwise

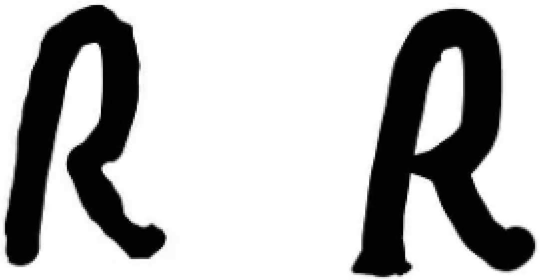


Dilation: a 3×3 square structuring element (www.cs.princeton.edu/~pshilane/class/mosaic/).

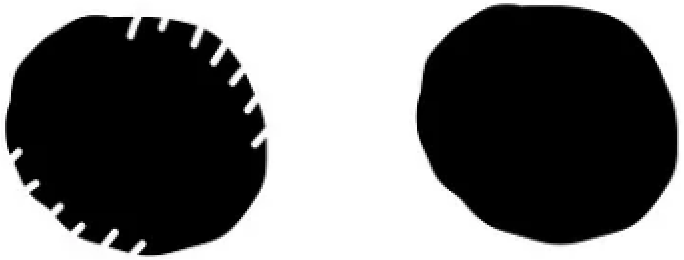

B
S
S
$B \oplus S$
$B \oplus S$

- Morphological dilation enlarge the size of regions by
  - increases object visibility
    - Shapes is more prominent, and lines appear thicker.
  - fills up small gaps and fills in small holes in objects

processed images after dilation using 3x3 and 5x5 structuring elements



It can repair breaks



It can repair intrusions

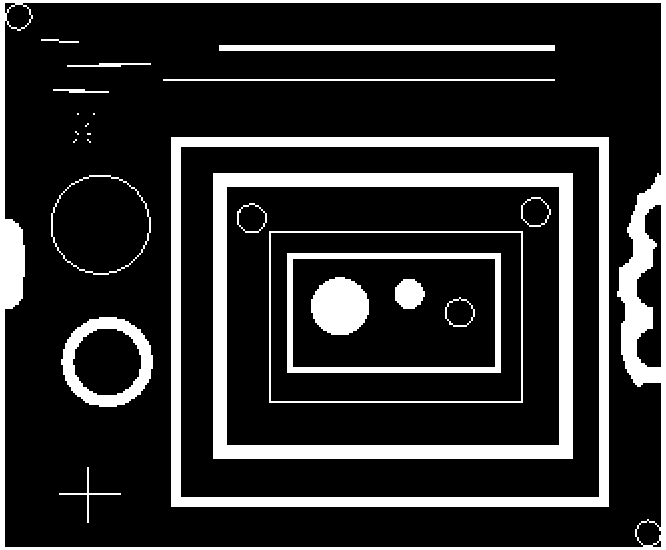
https://towardsdatascience.com/understanding-morphological-image-processing-and-its-operations-7bcf1ed11756

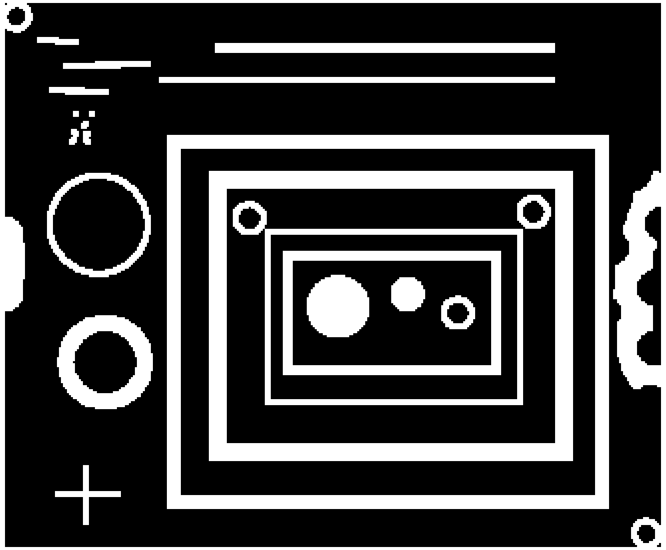
Dilation and erosion
are *dual* operations in that they
have opposite effects

Binary image

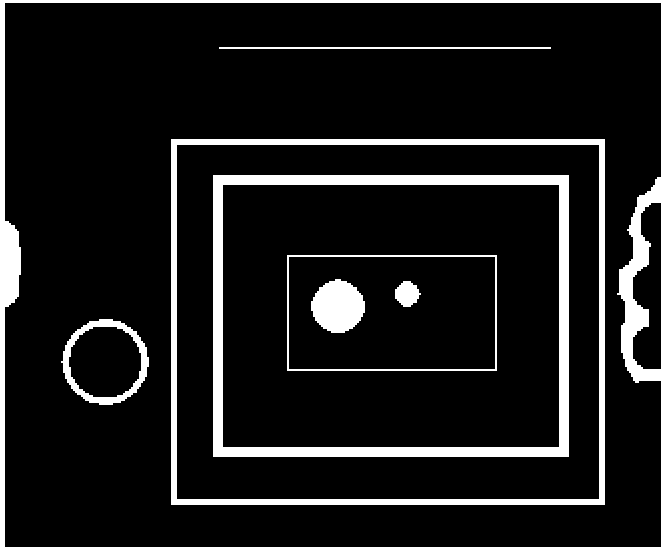
Dilation: a 2×2 square structuring element

Erosion: a 2×2 square structuring element





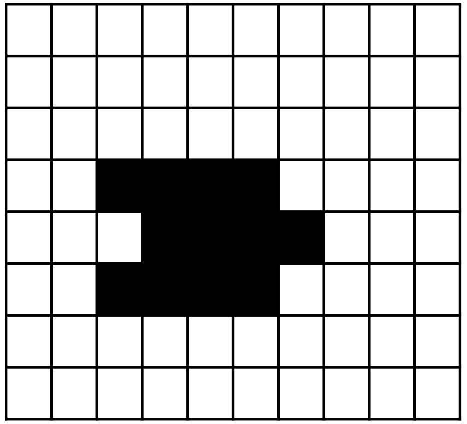
dilation



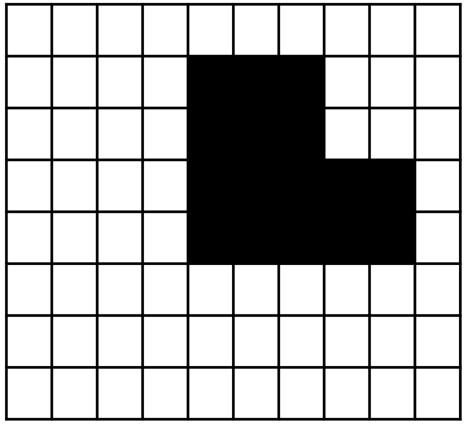
Erosion

https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html

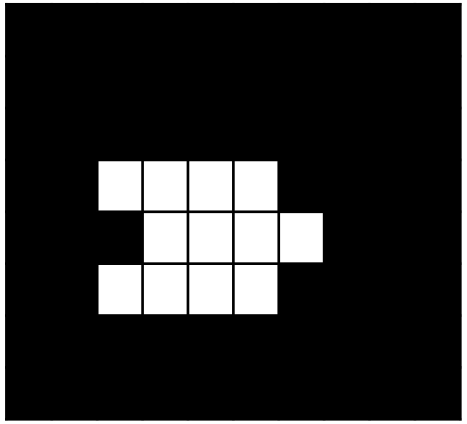
COMPLEMENT / UNION / INTERSECTION



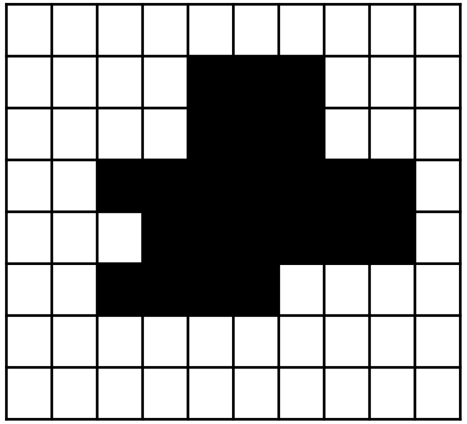
f



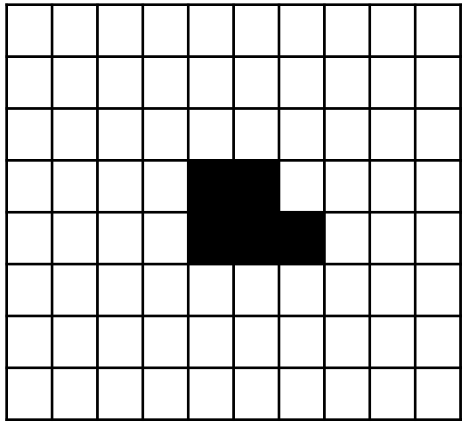
g



f^c



$f \cup g$



$f \cap g$

OPENING

- The **opening** of an image B by a structuring element S denoted by $B \circ S$ is an erosion followed by a dilation:

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

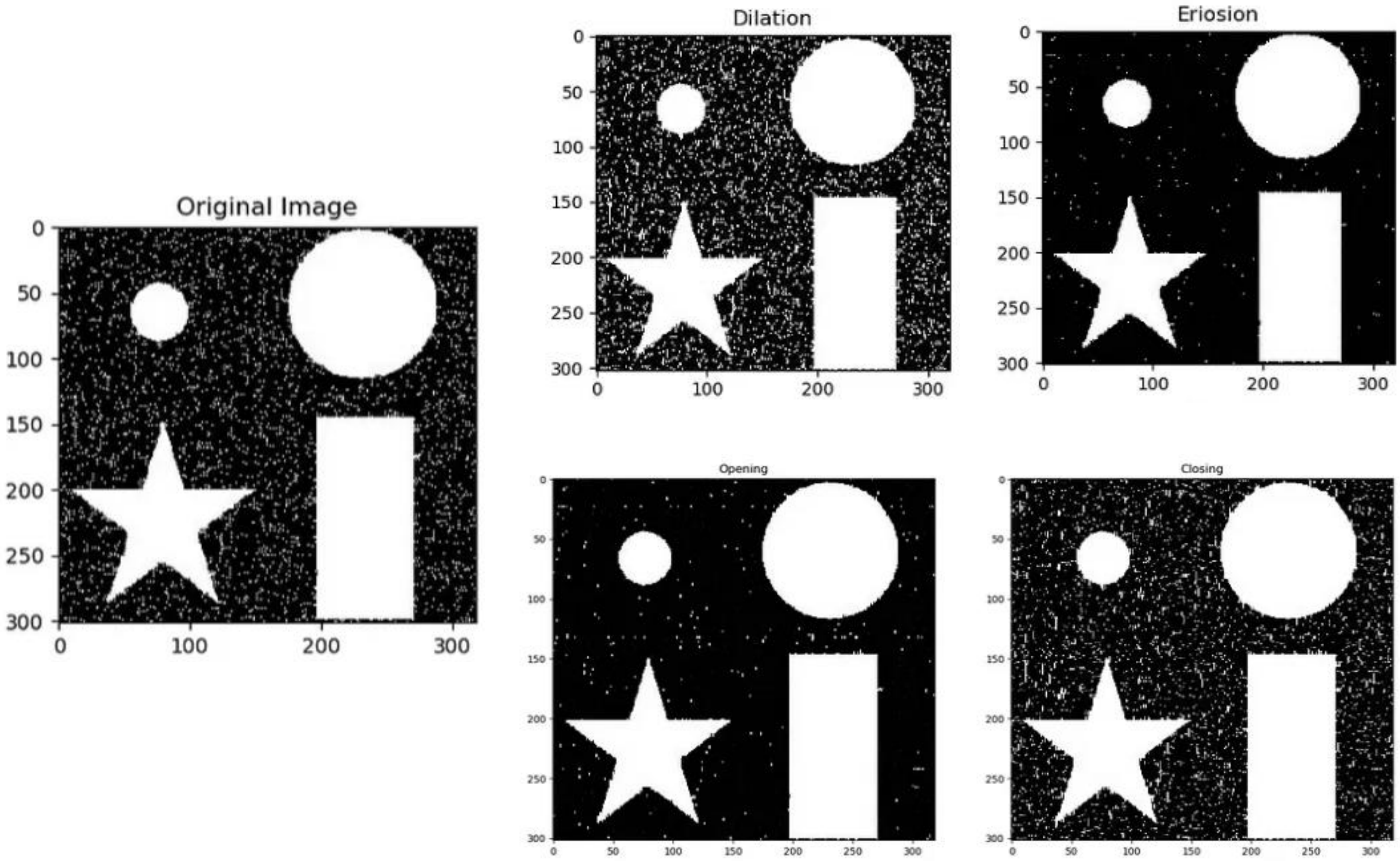
- Opening can open up a gap between objects connected by a thin bridge of pixels.
- Any regions that have survived the erosion are restored to their original size by the dilation
- Get rid of small portions of the region that jut out from the boundary into the background region.

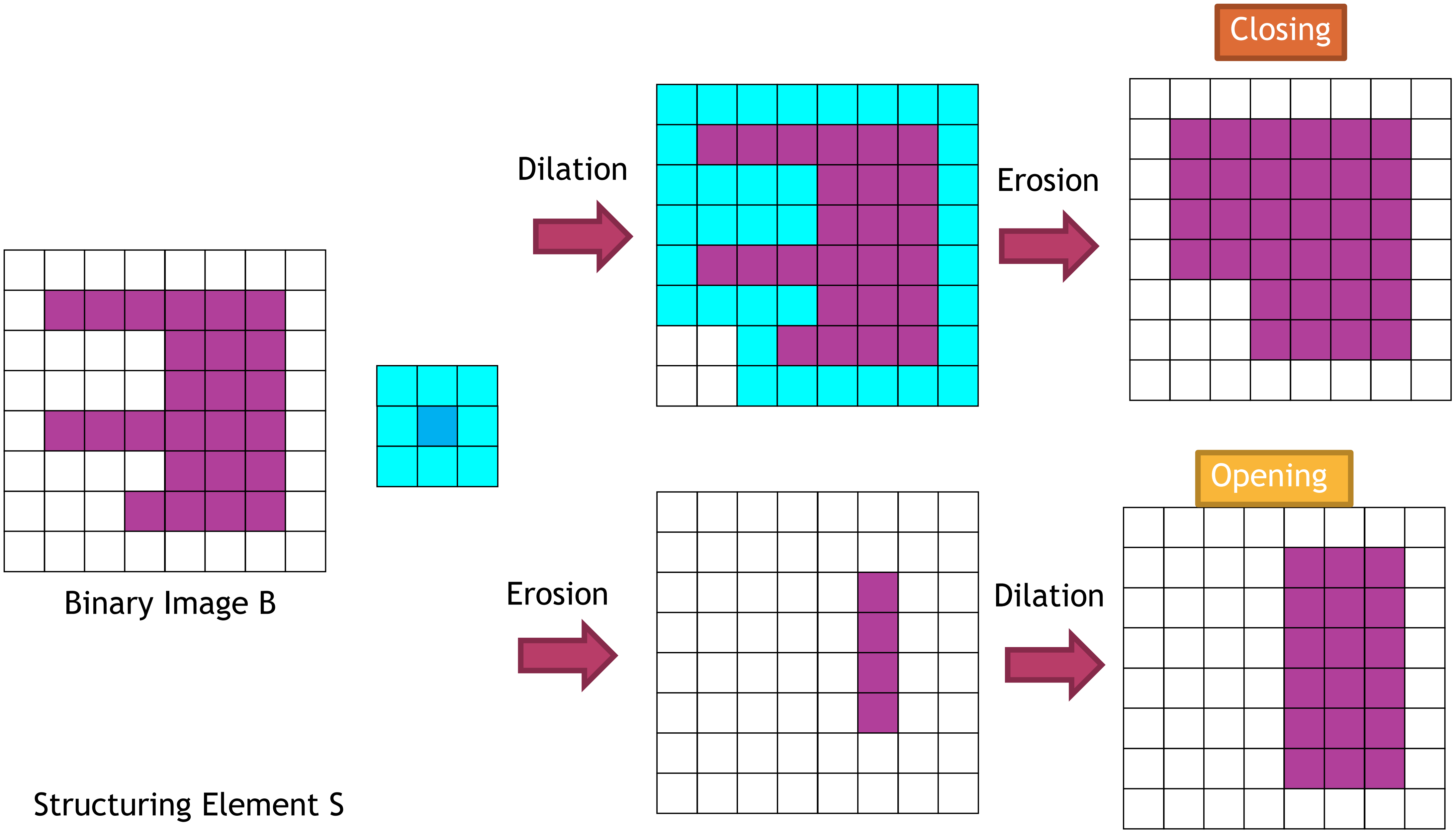
CLOSING

- The **closing** of an image B by a structuring element S denoted by $B \bullet S$ is a dilation followed by an erosion:

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

- Closing can fill holes in the regions while keeping the initial region sizes.
- Eliminate bays along boundary


Original Image
Dilation
Eriosion
Opening
Closing



Closing
Dilation
Erosion
Binary Image B
Opening
Erosion
Dilation
Structuring Element S

EDGE EXTRACTION

- Gain information and understand the features of an image.
- This process can help the researcher to acquire data from the image by following the below steps.
  - **Step 1.** Create an image (E) by erosion process; this will shrink the image slightly. The kernel size of the structuring element can be varied accordingly.
  - **Step 2.** Subtract image E from the original image. By performing this step, we get the boundary of our object.