GLA UNIVERSITY

DIGITAL IMAGE PROCESSING

By:
Dr. Ankush Agarwal

Outline

- Morphological Image Processing
  - Introduction, Logical Operations involving Binary Images,
  - Dilation and Erosion, Opening and Closing, The Hit-or-Miss Transformation,
  - Morphological Algorithms – Boundary Extraction, Region Filling, Extraction of Connected Components, Convex Hull, Thinning, Thickening
- Image Segmentation
  - Point, Line & Edge detection, Thresholding, Region-based Segmentation,
  - Region Extraction - Pixel Based Approach & Region Based Approach,
  - Edge and Line Detection - Basic Edge Detection, Canny Edge Detection,
  - Edge Linking - Hough Transform.
- Representation & Description
  - Representation - Boundary Following, Chain Codes,
  - Boundary Descriptors - Shape Numbers

MORPHOLOGY

Morphology

- Mathematical tool for processing shapes in image, including boundaries, skeletons, convex hulls, etc
- Morphological operations are typically applied to remove imperfections introduced during segmentation, and so typically operate on binary images

Morphology

- Mathematical tool for processing shapes in image, including boundaries, skeletons, convex hulls, etc
- Morphological operations are typically applied to remove imperfections introduced during segmentation, and so typically operate on binary images

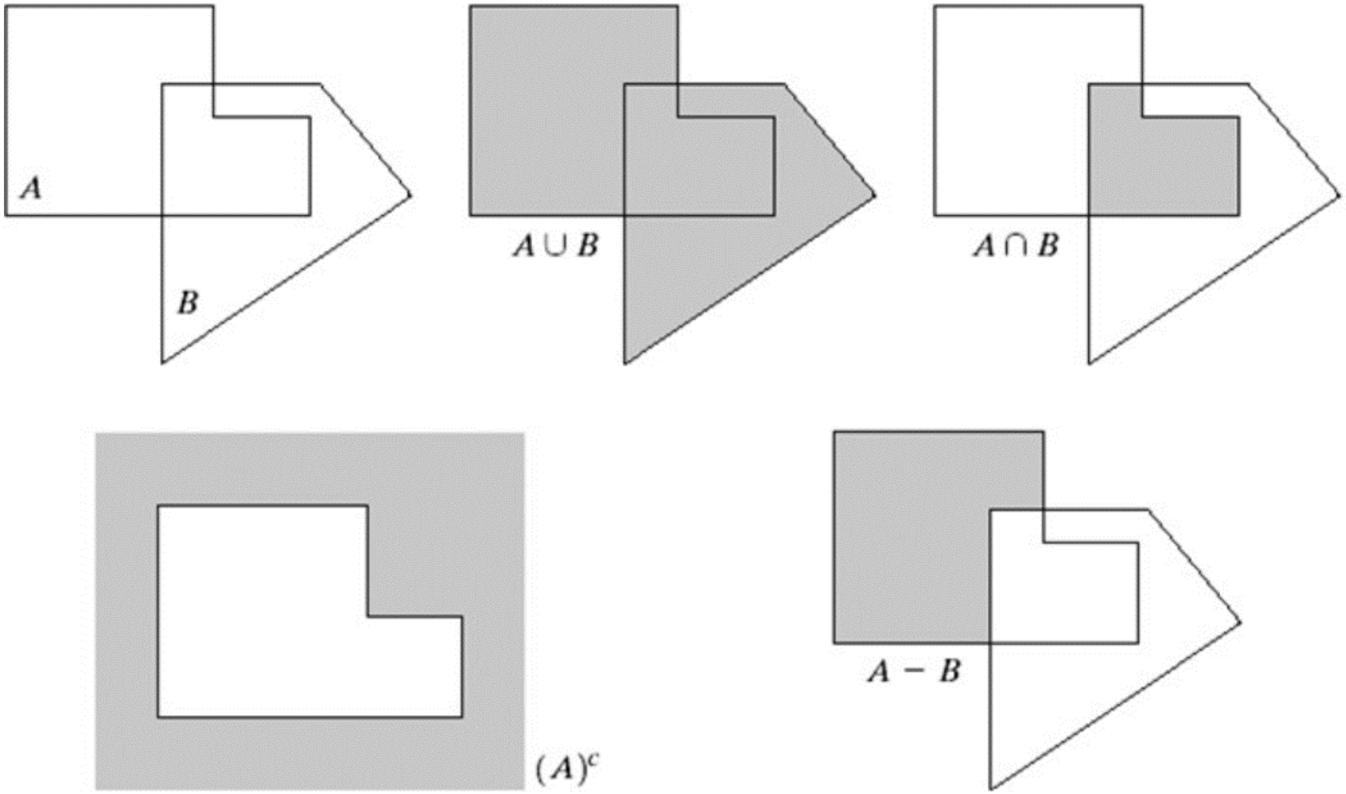
Set Theory

- Set (Ω): A collection of objects (elements)
- Membership ($\in$): If ω is an element of a set Ω, we can write $\omega \in \Omega$
- Subset ($\subset$): Let A, and B are two sets., If for every $a \in A$, we also have $a \in B$, then the set A is a subset of B, that is, $A \subset B$
  - If $A \subset B$ and $B \subset A$, then $A = B$
- Empty set ($\emptyset$)
- Complement: If $A \subset \Omega$, then its complement set $A^c = \{\omega | \omega \in \Omega, \text{ and } \omega \notin A\}$

Set Theory

- Union ($\cup$): $A \cup B = \{\omega | \ \omega \in A \text{ or } B\}$
- Intersection ($\cap$): $A \cap B = \{\omega | \ \omega \in A \text{ and } B\}$
- Set difference (-): $B \backslash A = B \cap A^c$
  - Note that $B\text{-}A \neq A\text{-}B$
- Disjoint sets: A and B are disjoint (mutually exclusive) if $A \cap B = \varnothing$

Example sets operations



Reflection and Translation

- Reflection
  - The reflection of a set B, denoted by $\hat{B}$, is defined as

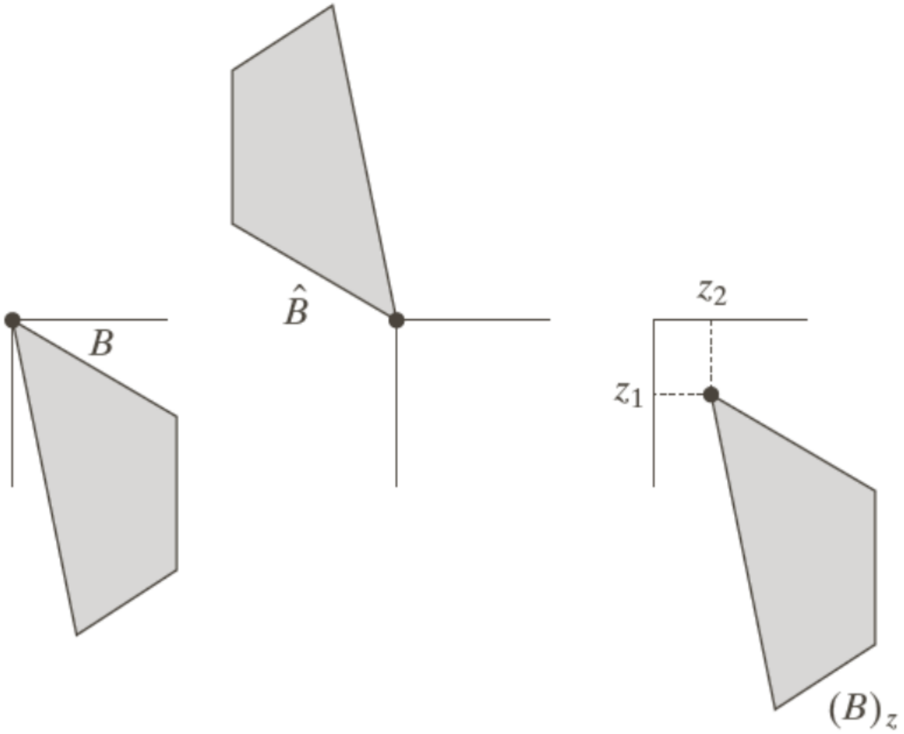
$$\hat{B} = \{w | w = -b, for\ b \in B\}$$

- Translation
  - The translation of a set B by point z = (z_1, z_2), denoted by $(B)_z$ is defined as

$$(B)_z = \{c | c = b + z, for\ b \in B\}$$

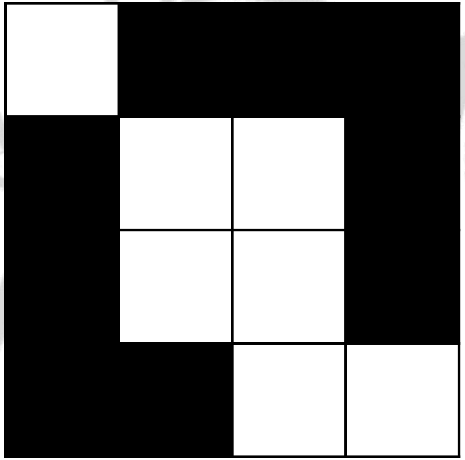
Reflection and Translation

- Eg:

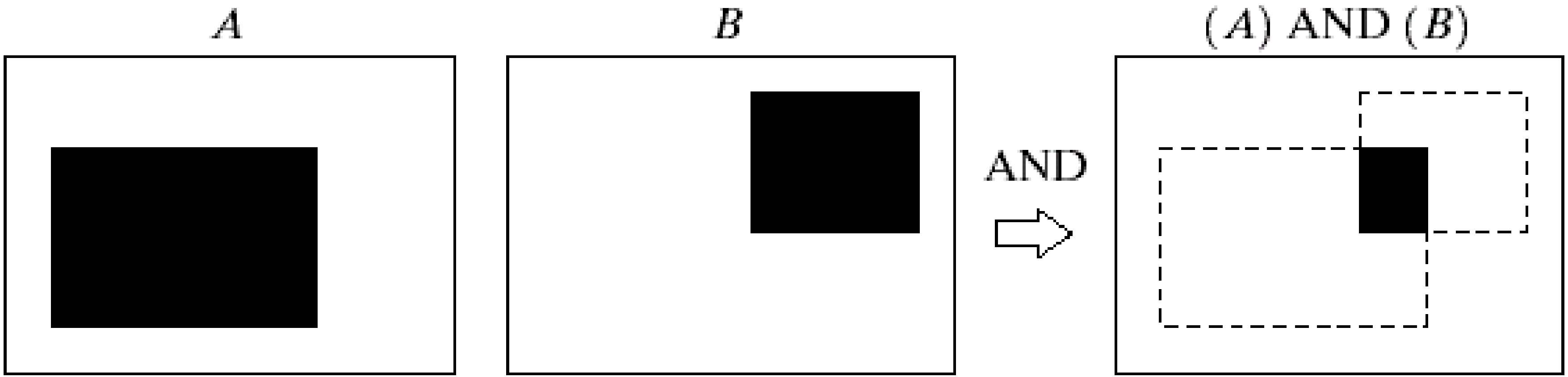


Binary Image

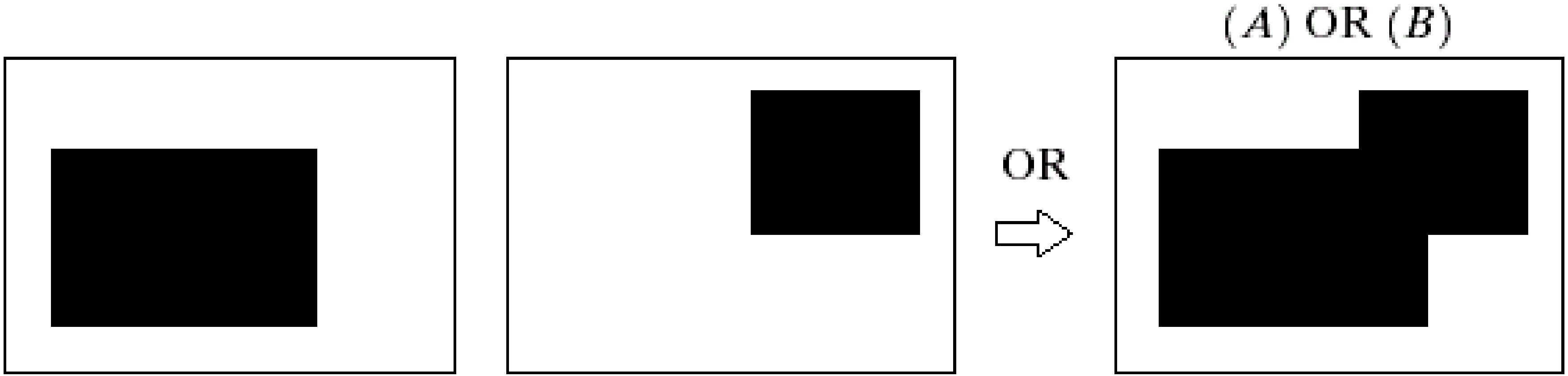
- Binary image
  - bi-valued function of x and y
- Morphological theory views
  - binary image as a set of its foreground (1-valued) pixels



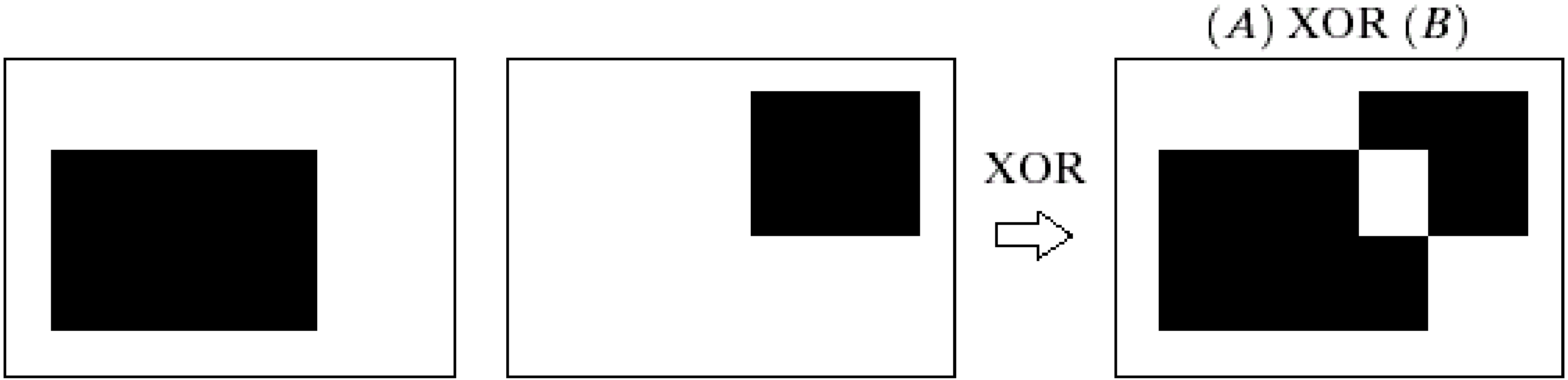
Logic operations between binary images



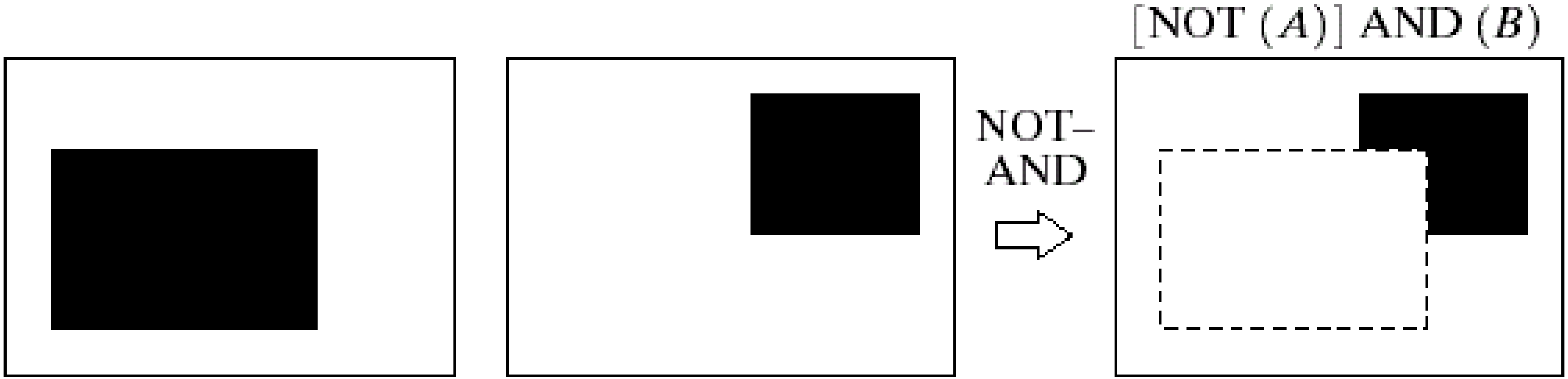
Logic operations between binary images



Logic operations between binary images

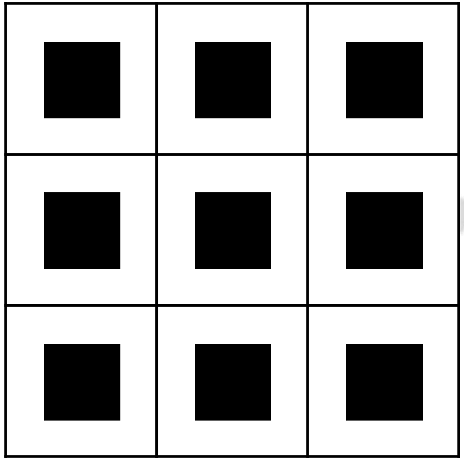


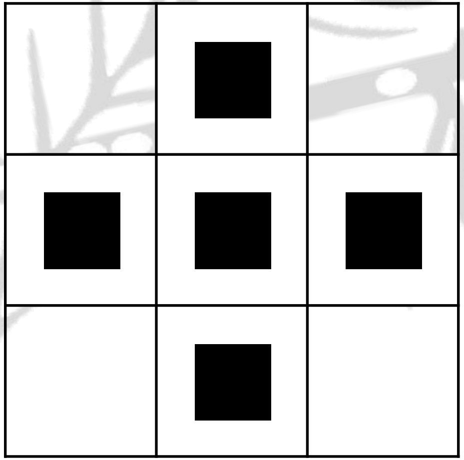
Logic operations between binary images

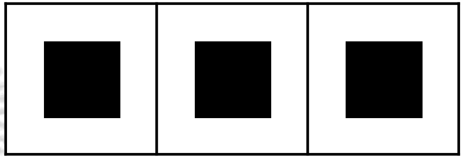


Basic components in Morphology

- Every operation has two elements
  - Input Image
  - Structuring element
- The results of the operation mainly depends upon the structuring element chosen

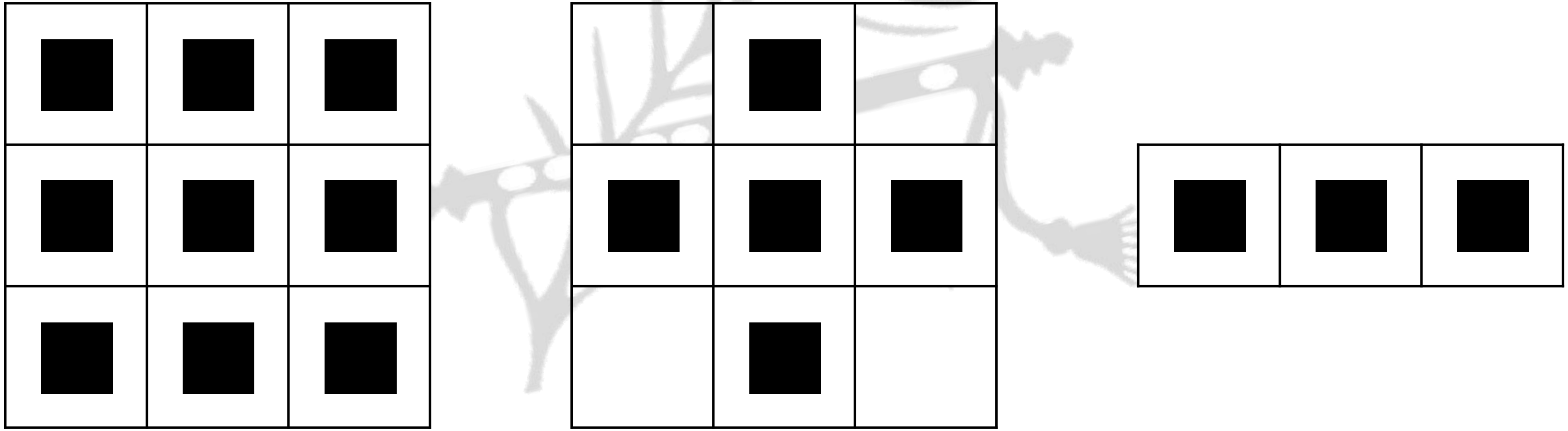






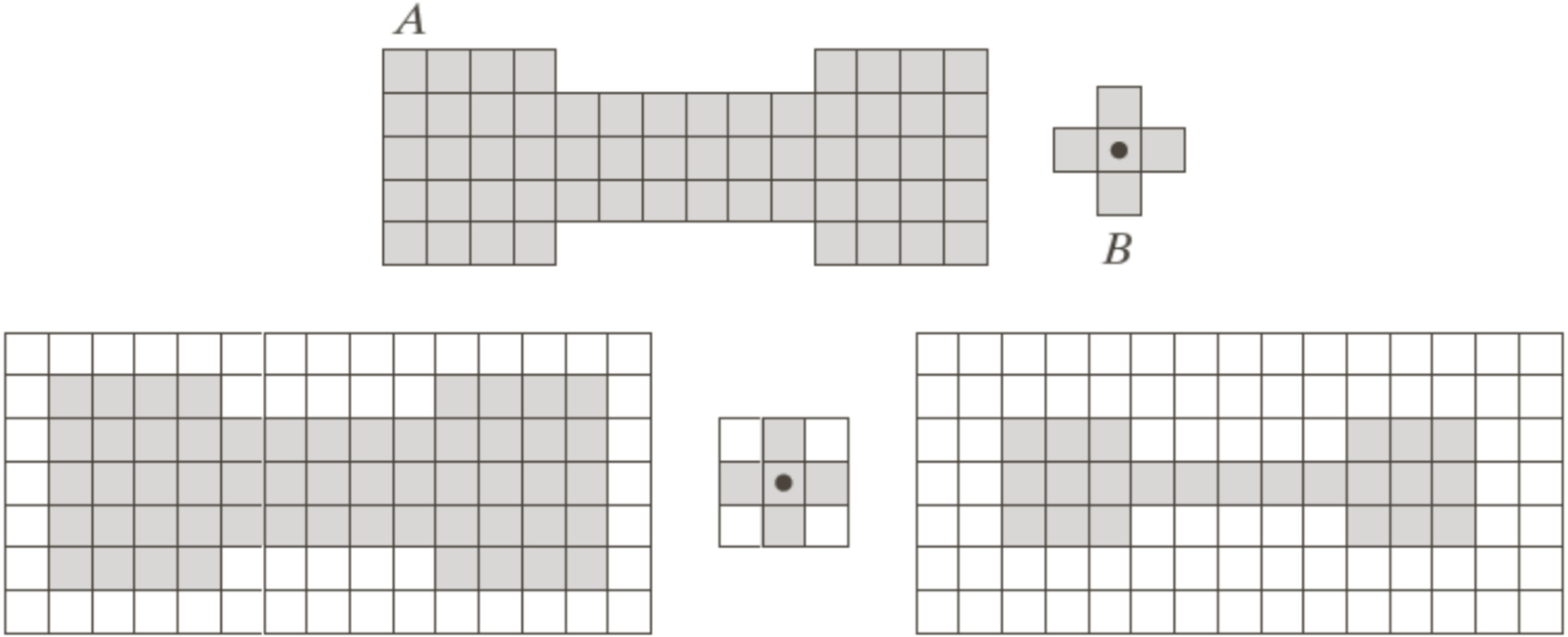
Structuring Elements

- Small sets or sub-images used to analyze an image for properties of interest
- Structuring elements can be any size and any shape



Structuring Elements

- Eg:



Fundamental Operations

- Fundamentally, morphological image processing is like spatial filtering
- The structuring element is moved across every pixel in the original image to give a new value of a pixel in processed image
- The value of this pixel depends on the operation performed
- There are two basic morphological operations
  - Dilation
  - Erosion

DILATION AND EROSION

Dilation

- Dilation is an operation that grows or thickens objects in a binary image
- The specific manner of this thickening is controlled by a shape referred to as a structuring element
- The structuring element is translated throughout the domain of the image to see where it overlaps with 1-valued pixels
- The output image is 1 at each location of the origin such that the structuring element overlaps at least one 1-valued pixel in the input image

Dilation

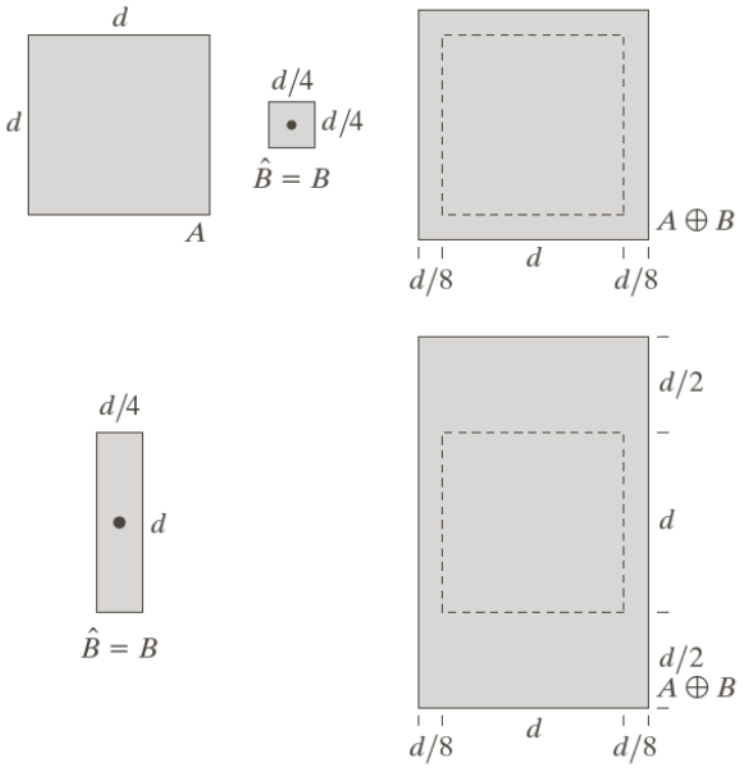
- The dilation of I and S is denoted by I $\oplus$ S

$$I \oplus S = \{z \mid (\hat{S})_z \cap I \neq \phi\}$$

- Theoretical way of generation:
  - Obtain the reflection of S about its origin
  - Shift this reflection by z
  - Dilation of I by S is the set of all structuring element origin locations where the reflected and translated S overlaps at least some portion of I

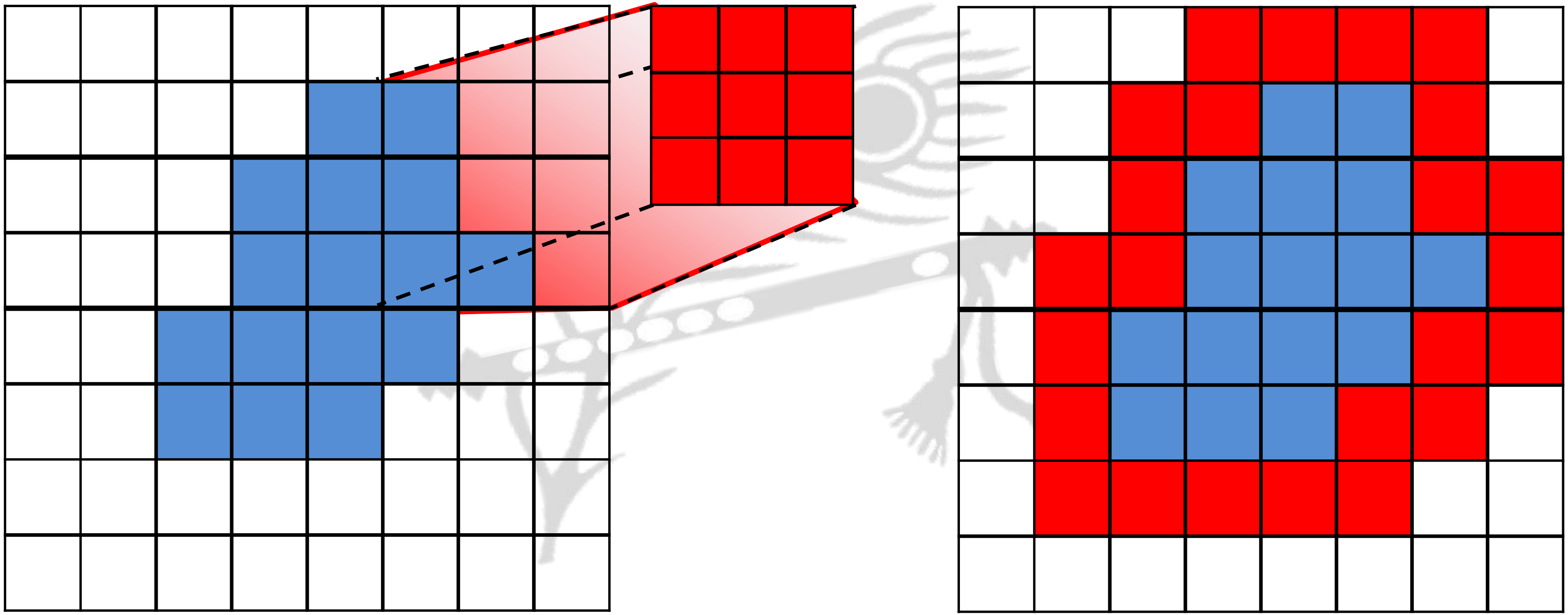
Dilation

- Eg:



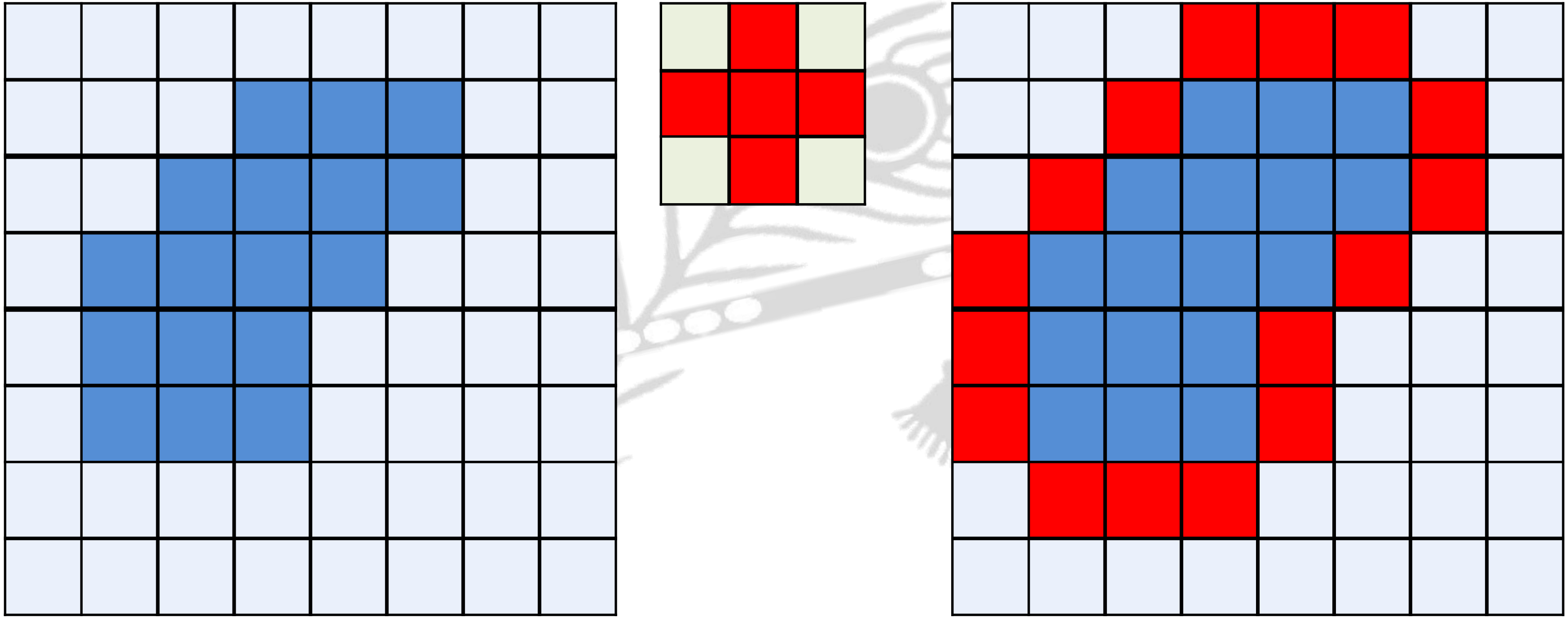
Dilation

- Eg:



Dilation

- Eg:



What is Dilation for…?

- Dilation can repair breaks

- Dilation can repair intrusions

Properties of Dilation

- Dilation is commutative

$$A \oplus B = B \oplus A$$

- Dilation is associative

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

- Dilation is invariant to translation

$$A_h \oplus B = (A \oplus B)_h$$

Erosion

- The erosion of I by S, denoted $I \ominus S$

$$I \ominus S = \{z \mid (S)_Z \subseteq I\}$$

  - The set of all points z such that, S translated by z, is contained by I

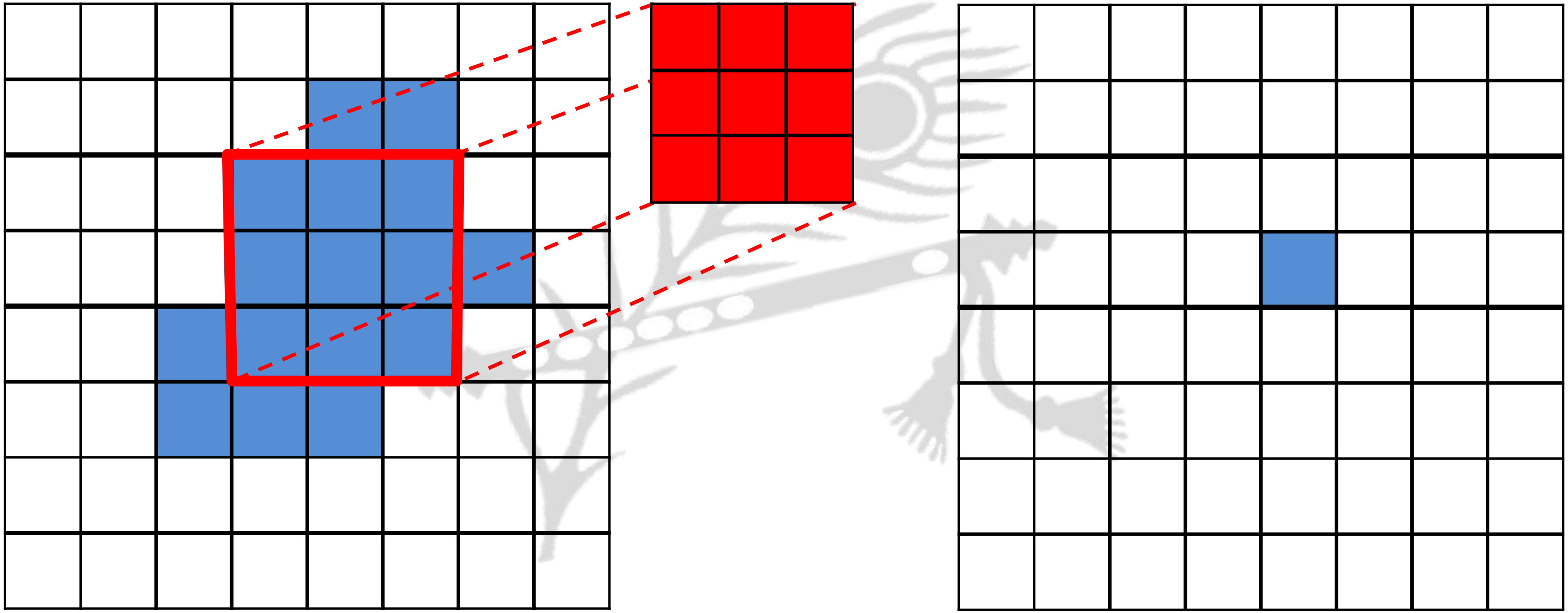
$$I \ominus S = \{z \mid (S)_Z \cap I^c = \emptyset\}$$

  - In other words, erosion of I by S is the set of all structuring element origin locations where the translated S has no overlap with the background of I

- Erosion “shrinks” or “thins” objects in a binary image

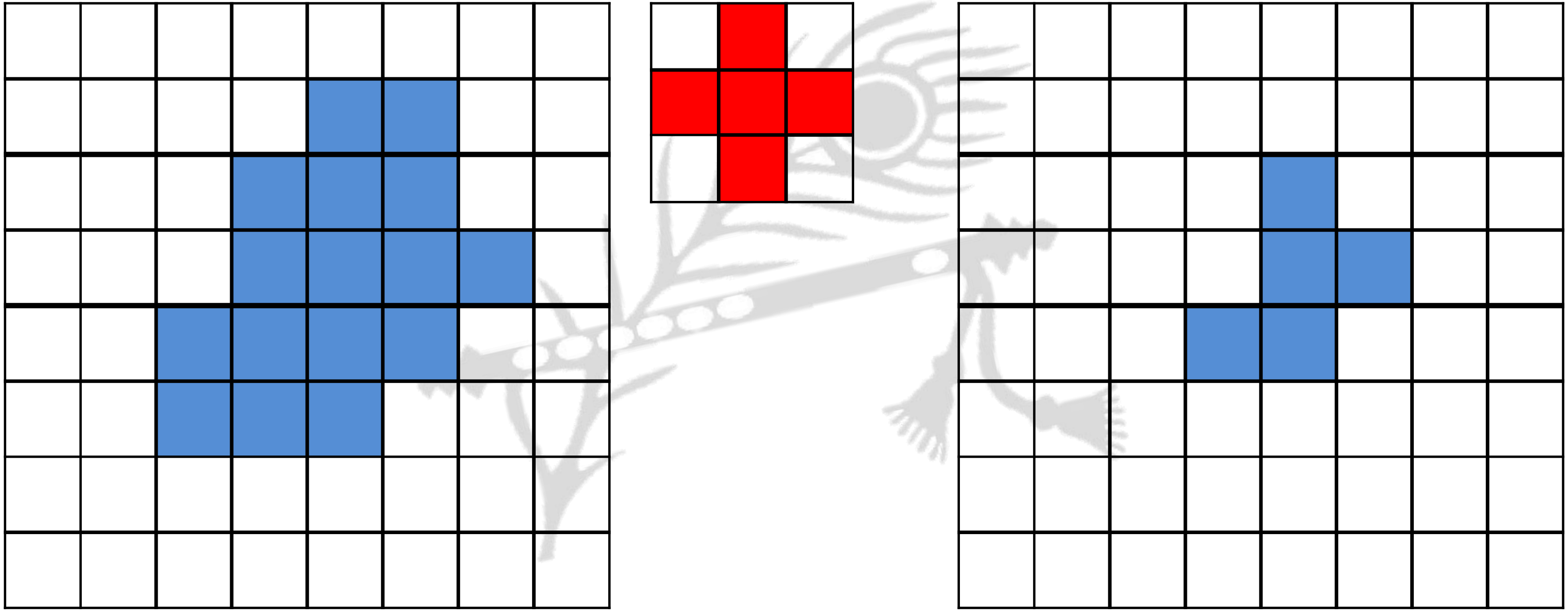
Erosion

- Eg:



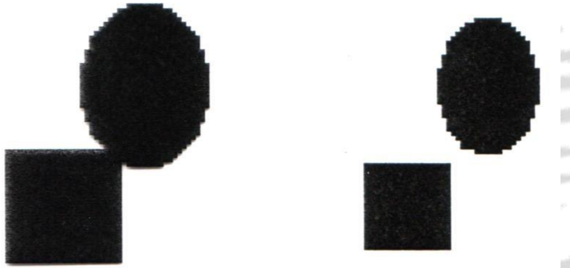
Erosion

- Eg:

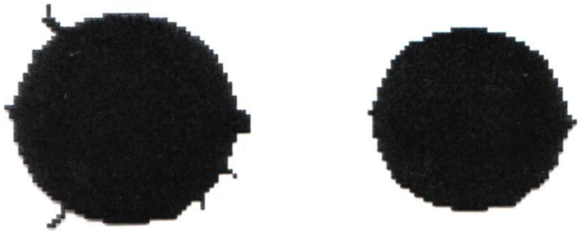


What is Erosion for…?

- Erosion can split apart joined objects



- Erosion shrinks objects and removes random outer edges



Dilation

- Binary image

	0	1	2	3	4	5	6	7
0	x							
1								
2								
3								
4								
5								
6								
7								

← Image (I)

Structure Element (S) →

	-1	0	1
-1			
0		X	
1			

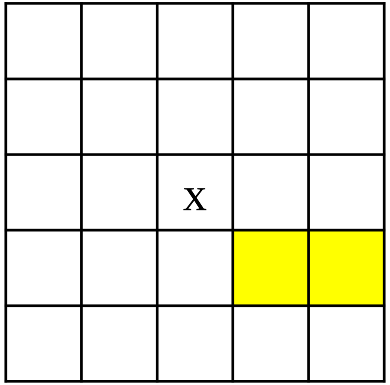
I = {(2,2), (3,2), (3,3), (4,3), (4,4), (5,4)}

U = {(0,0), ..., (7,7)}

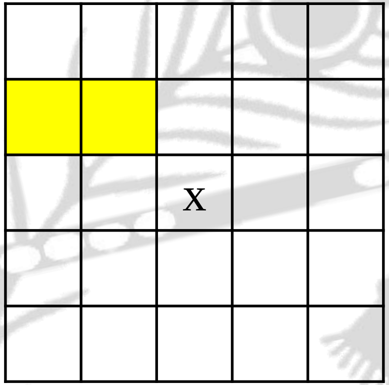
S = {(-1,-1), (0,-1)}

Dilation

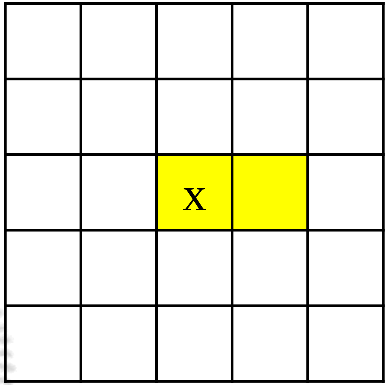
- Reflection and Translation operations



$I = \{(1,1), (1,2)\}$



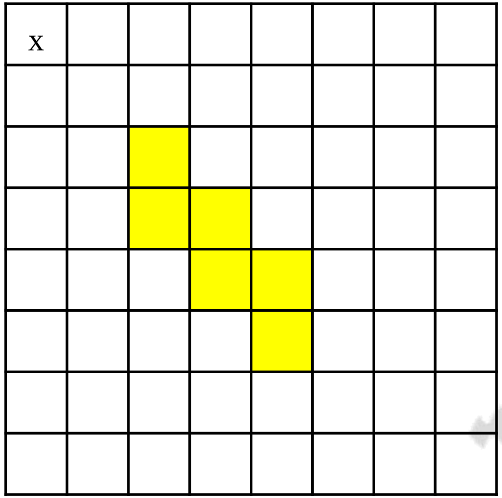
$\hat{I} = \{(-1,-1), (-1,-2)\}$

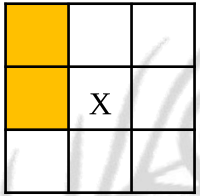


$I_{(-1,-1)} = \{(0,0),(0,1)\}$

Dilation (by coordinate system)

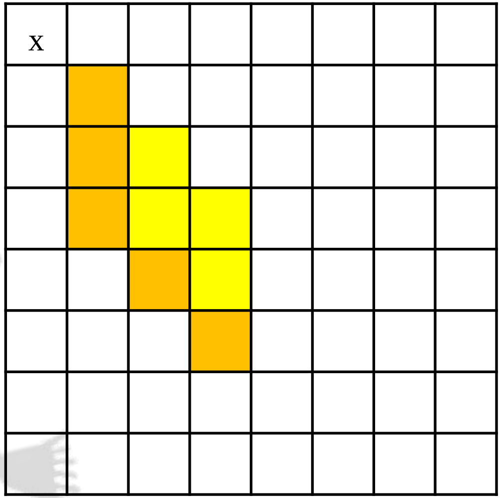
- Eg:





S = {(-1,-1), (0,-1)}

$$I \oplus S = \{p \mid p = i + s, i \in I, s \in S\}$$



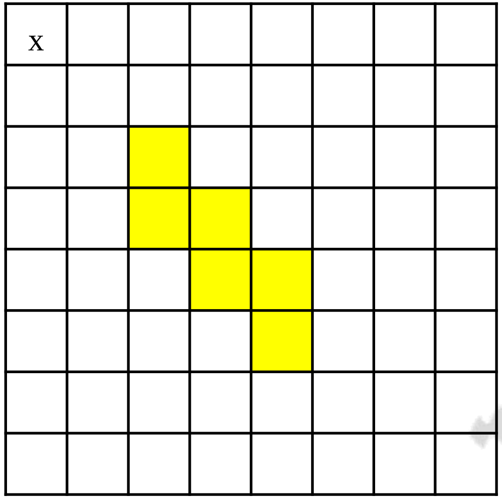
I = {(2,2), (3,2), (3,3), (4,3), (4,4), (5,4)}

$$I \oplus S = \left\{\begin{matrix}(1,1),(2,1),(2,2),(3,2),(3,3),(4,3) \\ (2,1),(3,1),(3,2),(4,2),(4,3),(5,3)\end{matrix}\right\}$$

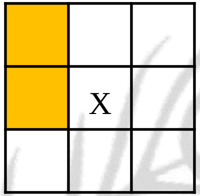
$$= \{(1,1),(2,1),(2,2),(3,1),(3,2),(3,3),(4,2),(4,3),(5,3)\}$$

Dilation (another definition)

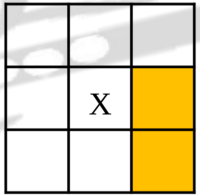
- Eg



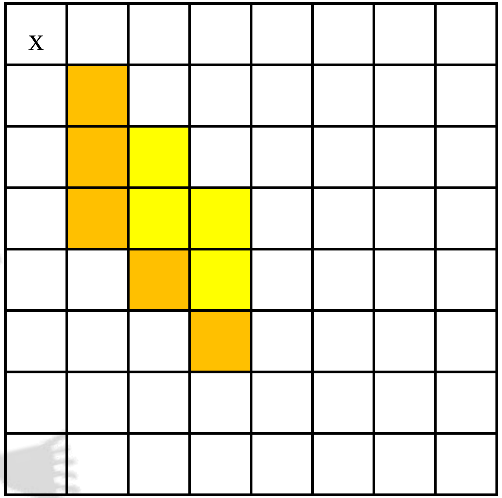
I



S = {(-1,-1), (0,-1)}



S = {(1,1), (0,1)}

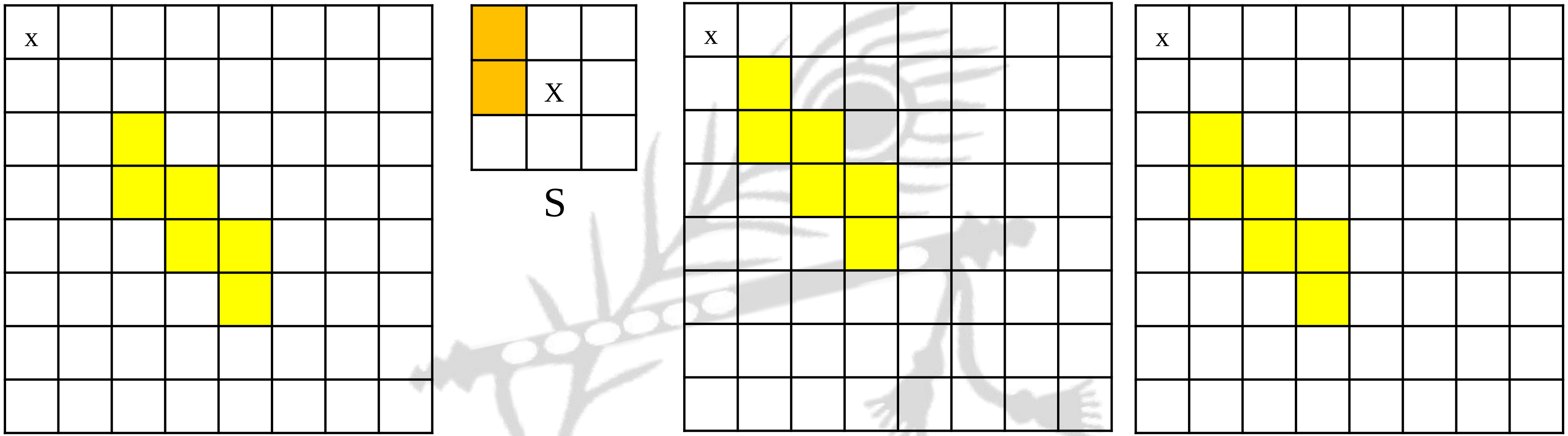


$$I \oplus S = \{p \mid [(\hat{S})_p \cap I] \neq \phi\}$$

$$= \{p \mid [(\hat{S})_p \cap I] \subseteq I\}$$

Dilation (as Union of object translation)

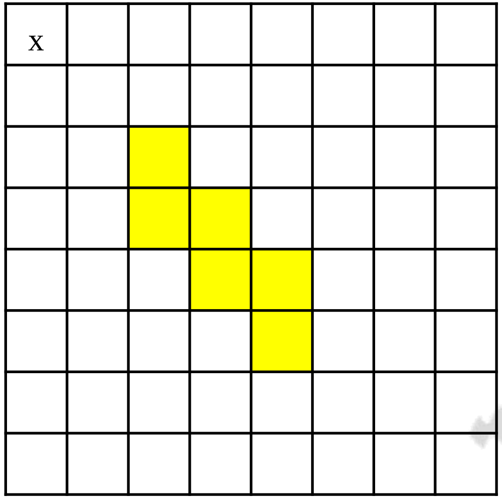
- Eg



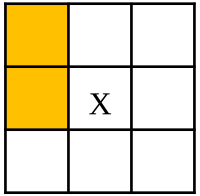
I   S $I_{(-1,-1)}$ $I_{(0,-1)}$

Dilation (as Union of object translation)

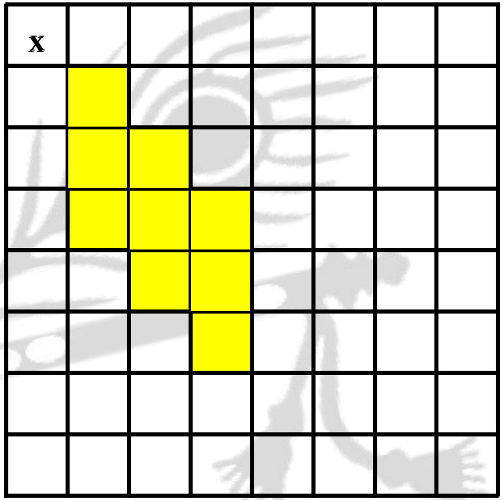
- Eg



I



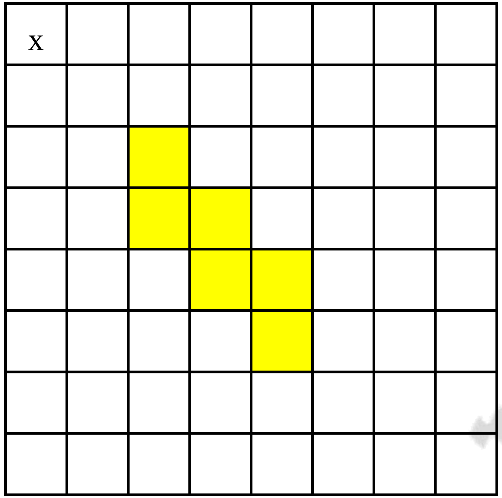
S



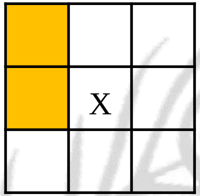
$$I \oplus S = \bigcup_{s \in S} I_s$$

Erosion (by coordinate system)

- Eg:

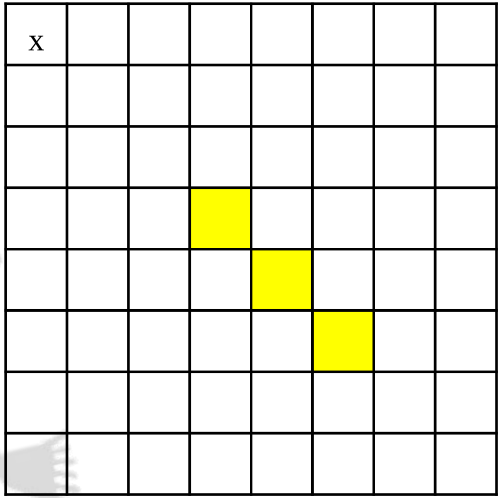


I = {(2,2), (3,2), (3,3), (4,3), (4,4), (5,4)}



S = {(-1,-1), (0,-1)}

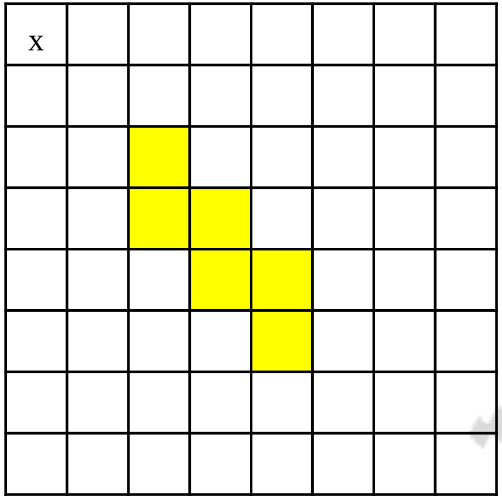
$$I \Theta S = \{p \mid p+s \in I, s \in S\}$$



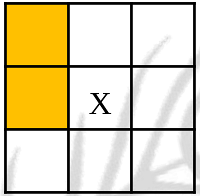
$$(3,3)+\{(-1,-1),(0,-1)\}=\{(2,2),(3,2)\}\in \mathrm{I}$$
$$(4,4)+\{(-1,-1),(0,-1)\}=\{(3,3),(4,3)\}\in \mathrm{I}$$
$$(5,5)+\{(-1,-1),(0,-1)\}=\{(4,4),(5,4)\}\in \mathrm{I}$$

Erosion (another definition)

- Eg



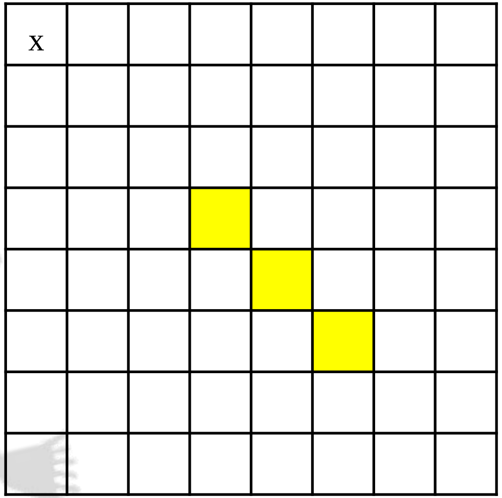
I



S = {(-1,-1), (0,-1)}

$$I \Theta S = \{\mathrm{p} \mid (\mathrm{S})_p \cap \mathrm{I}^c = \phi\}$$

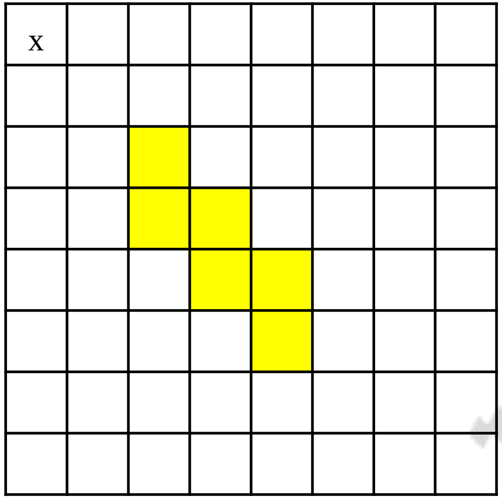
$$= \{\mathrm{p} \mid [(\mathrm{S})_p] \subseteq \mathrm{I}\}$$



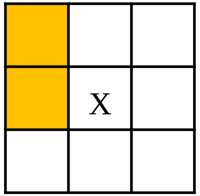
$$(3,3),(4,4),(5,5)$$

Erosion (as Intersection of object translation)

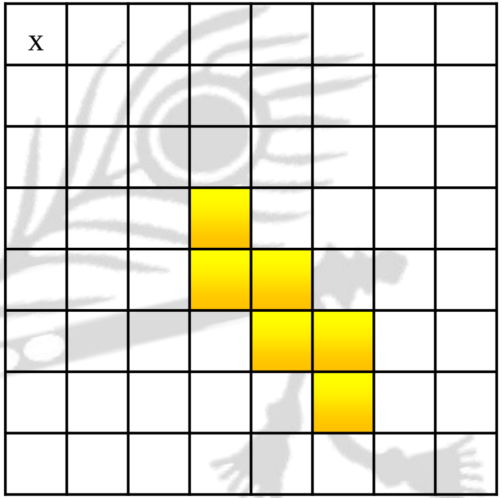
- Eg



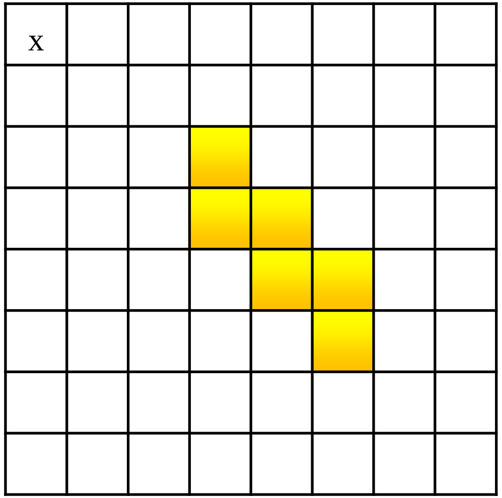
I



S



$I_{(1,1)}$

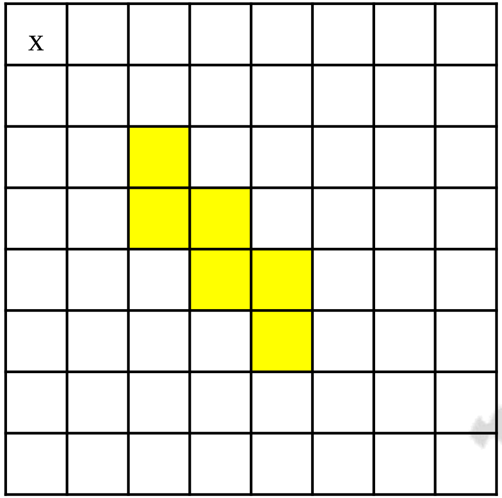


$I_{(0,1)}$

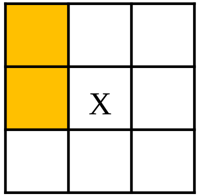
$$I \Theta S = \bigcap_{s \in S} I_{-s}$$

Erosion (as Intersection of object translation)

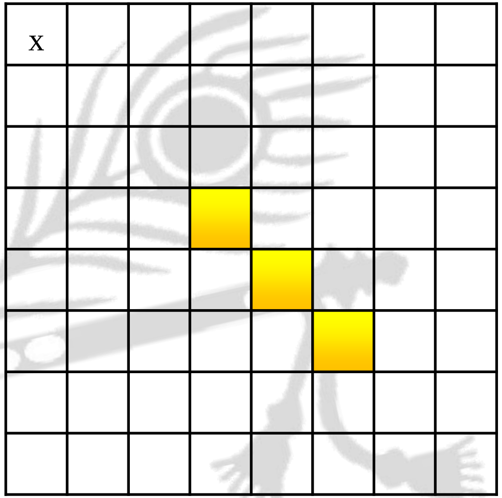
- Eg



I



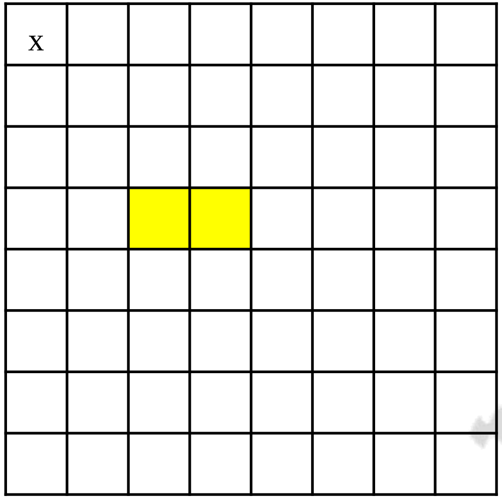
S



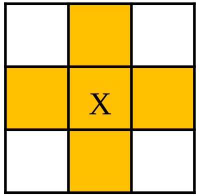
$$I \Theta S = \bigcap_{s \in S} I_{-s}$$

Eg:

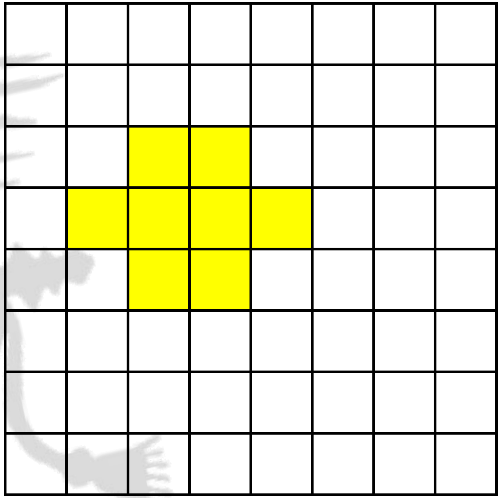
- Find $I \oplus S$



I

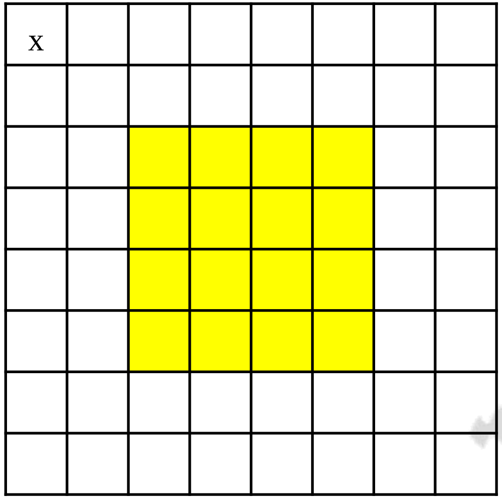


S

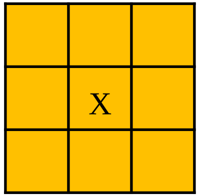


Eg:

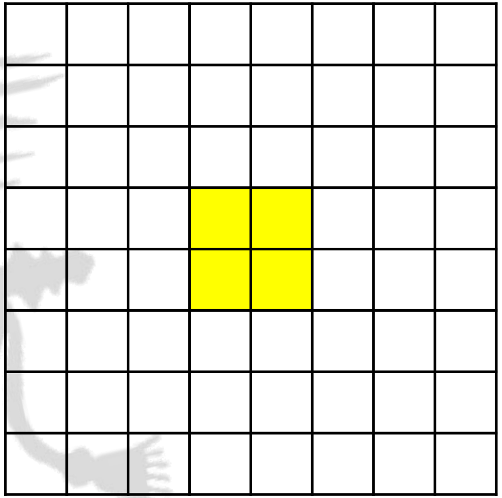
- Find $I \Theta S$



I

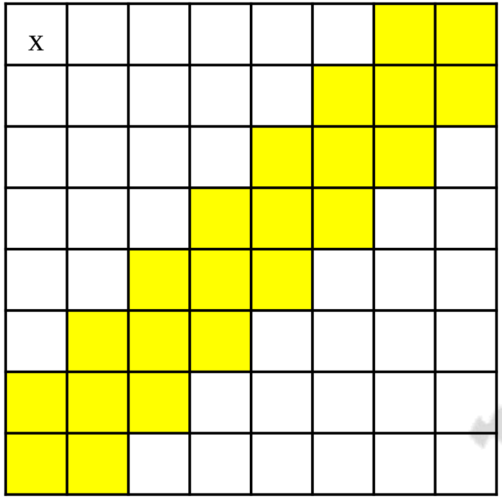


S

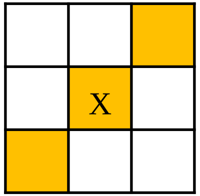


Eg:

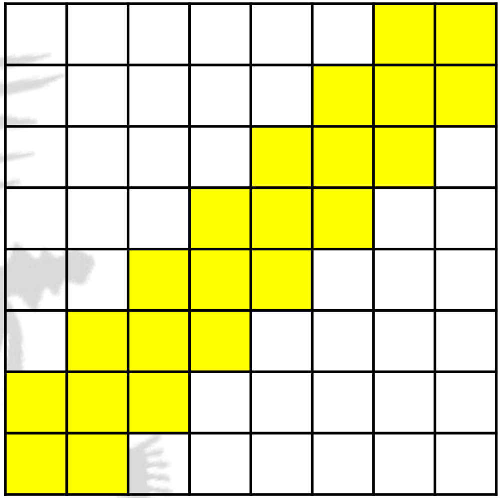
- Find $I \oplus S$



I

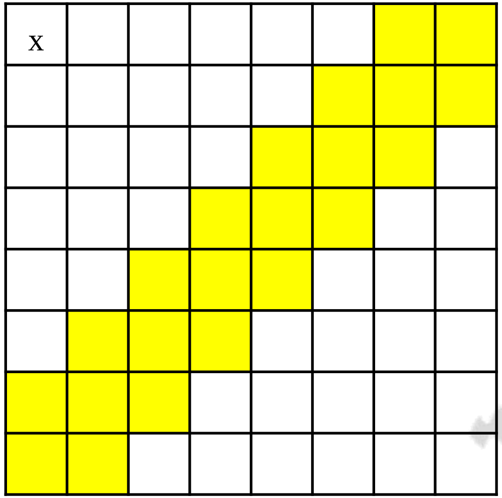


S

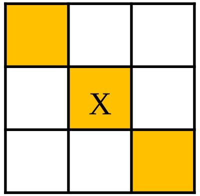


Eg:

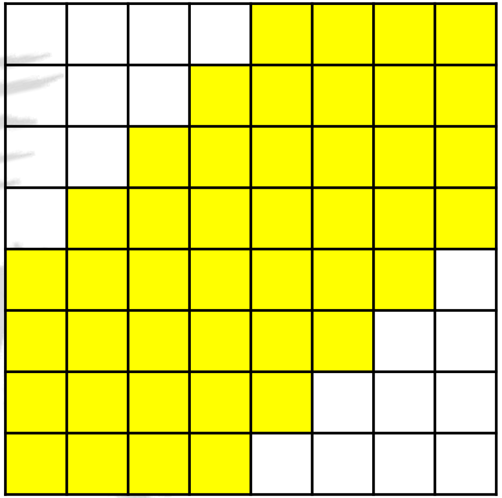
- Find $I \oplus S$



I

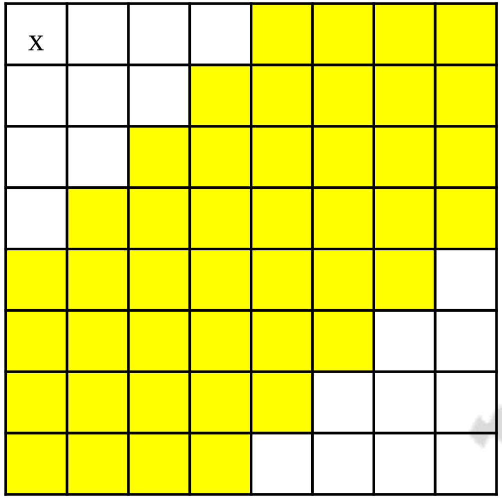


S

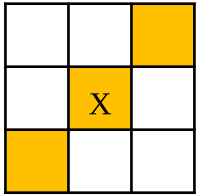


Eg:

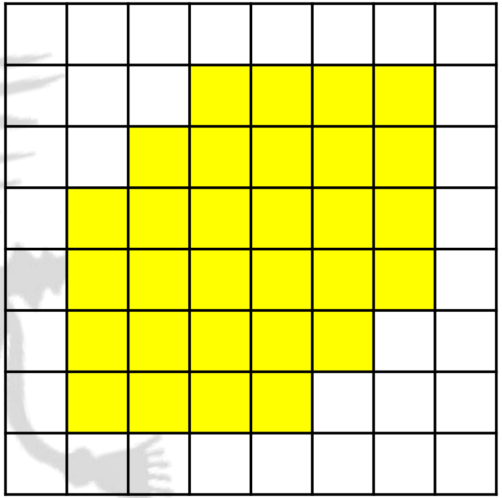
- Find $I \Theta S$



I

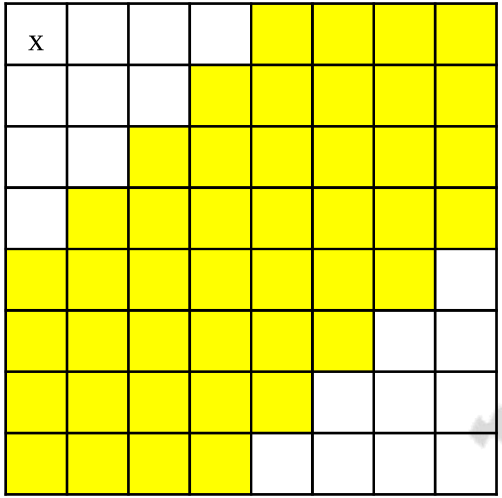


S

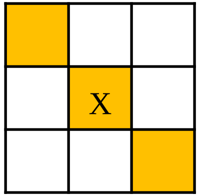


Eg:

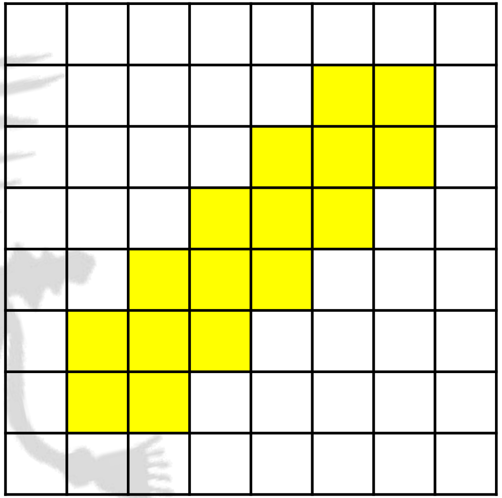
- Find $I \Theta S$



I



S



Duality

- Erosion and dilation are duals of each other with respect to set complementation and reflection

$$
\begin{aligned}
(A \ominus B)^c &= \{z \mid (B)_z \subseteq A\}^c \\
&= \{z \mid (B)_z \cap A^c = \varnothing\}^c \\
&= \{z \mid (B)_z \cap A^c \neq \varnothing\} \\
&= A^c \oplus B
\end{aligned}
$$

Combining Dilation and Erosion

- Dilation and Erosion are not inverse transformations
- If an image is eroded & then dilated (or vice-versa), the original image can not be obtained
- In practical applications, dilation and erosion are used most often in various combinations
- Three of the most common combinations of dilation and erosion are
  - Opening
  - Closing
  - Hit or miss transformation

OPENING AND CLOSING

Opening and Closing

- Opening is erosion followed by dilation
- The opening is given as

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

- Closing is dilation followed by erosion
- The closing is given as

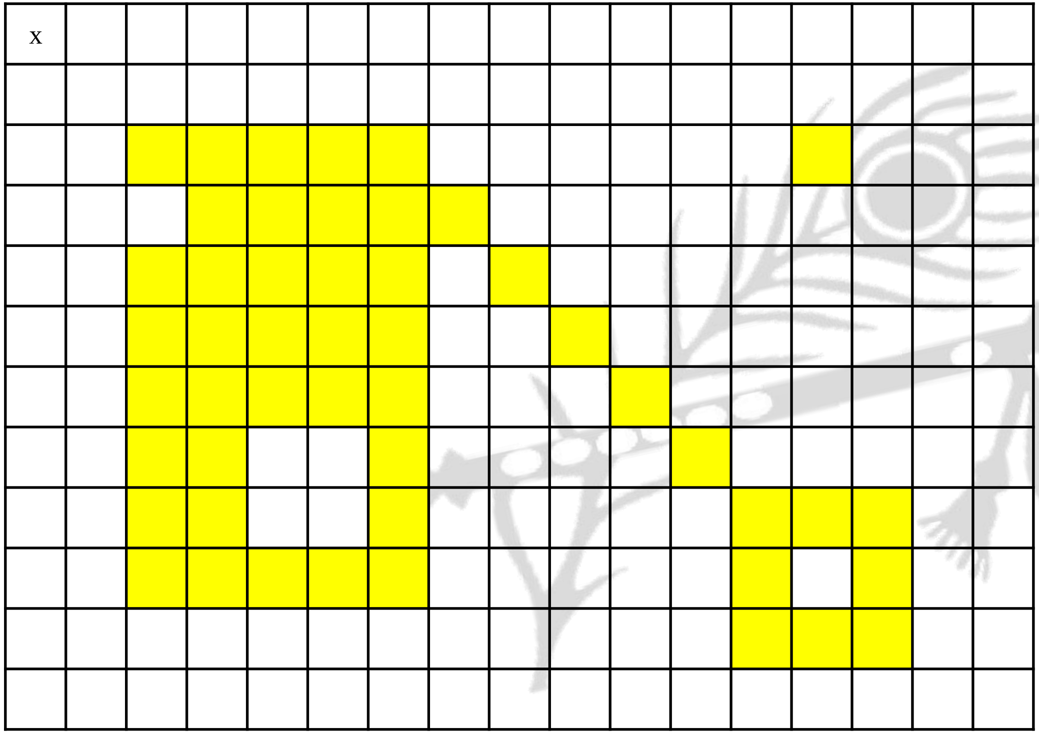
$$A \bullet B = (A \oplus B) \ominus B$$

Opening and Closing

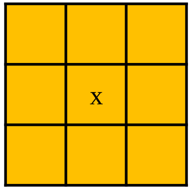
- Opening
  - smooth the contours of an object
  - breaks narrow strips
  - eliminates thin edges
  - it is less destructive than the Erosion

- Closing
  - smooth sections of the contours
  - fuses narrow breaks & long thin gulfs
  - eliminates small holes & fills gaps in the contour

Closing

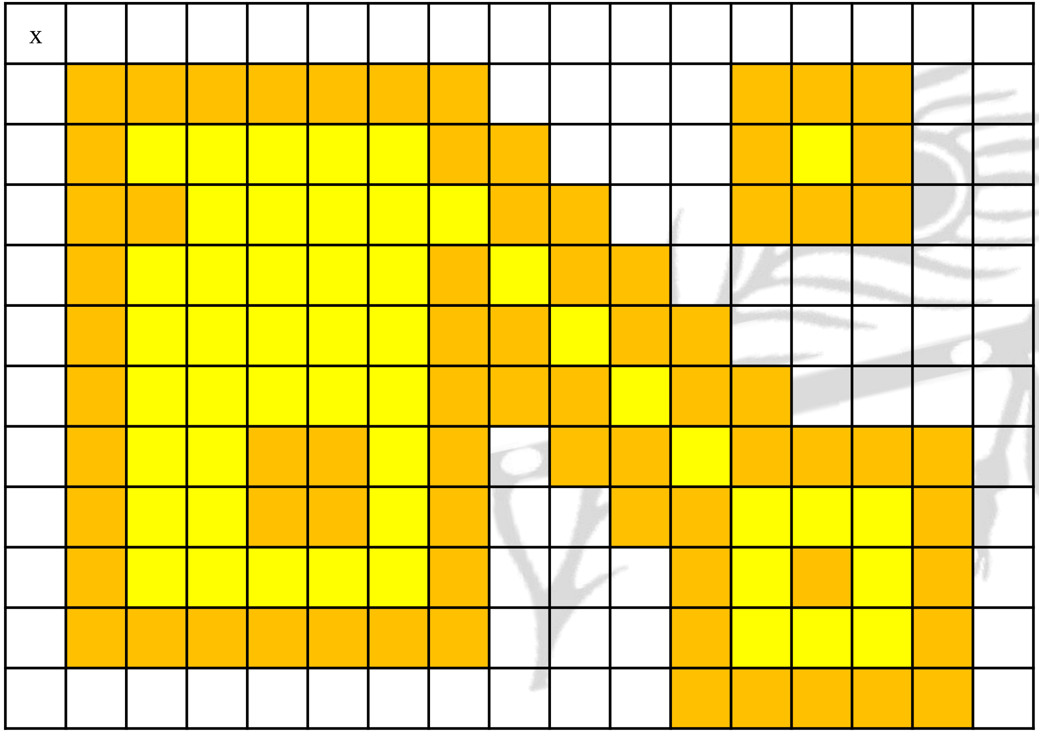


I

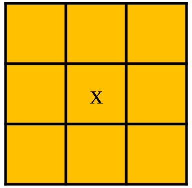


S

Closing

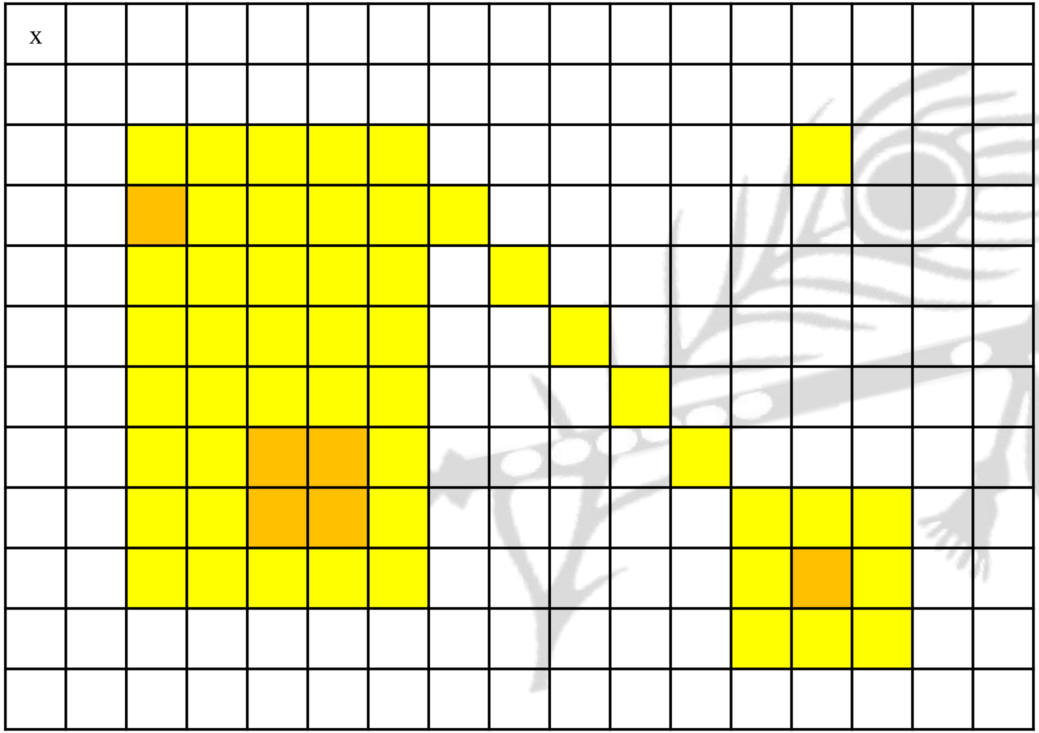


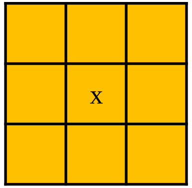
$I \oplus S$



S

Closing

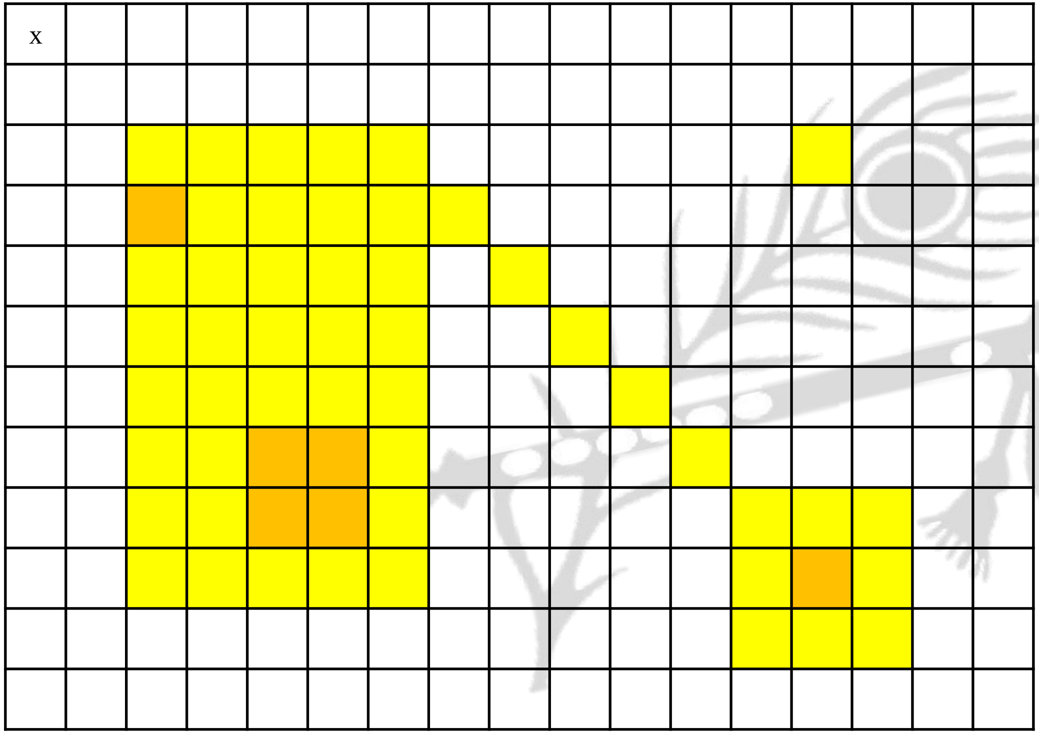




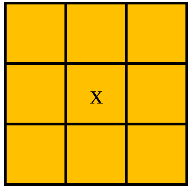
S

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

Opening

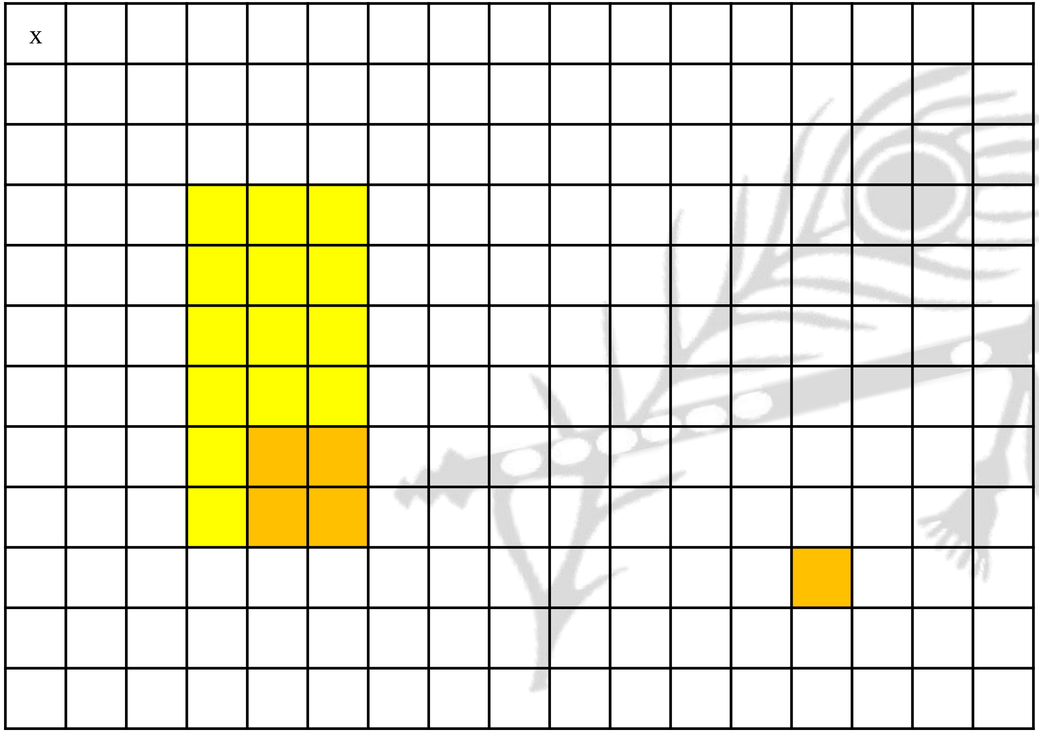


I

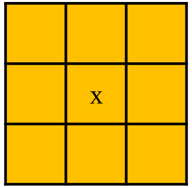


S

Opening

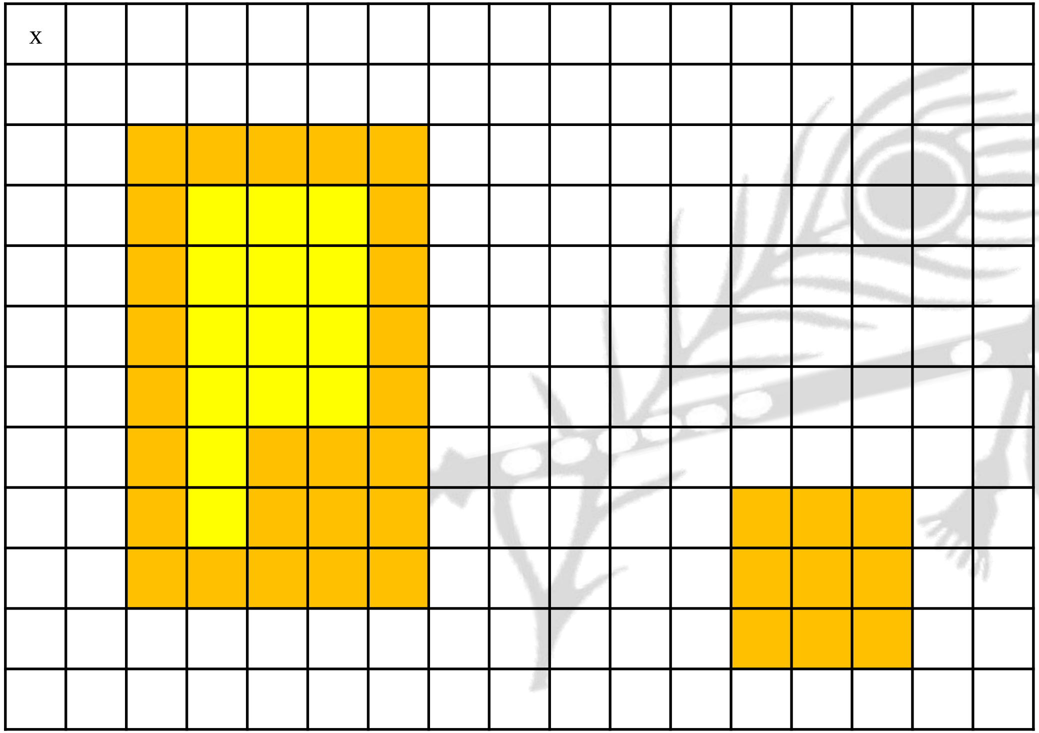


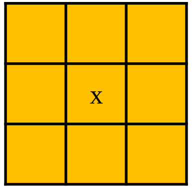
$I \Theta S$



S

Opening





S

$$I \circ S = (I \Theta S) \oplus S$$

HIT OR MISS TRANSFORM

Hit or Miss Transform

- A basic tool for shape detection
- It is a morphological operator for finding local patterns of pixels
- The hit-or-miss transform is a general binary morphological operation that can be used to look for particular patterns of foreground and background pixels in an image

- Concept:
  - Hit object
  - Miss background

Hit or Miss Transform

- It is given as

$$I \circledast S = (I \Theta S) \bigcap (I^c \Theta (W - S))$$

- It can be written as

$$I \circledast S = (I \Theta S_1) \bigcap (I^c \Theta S_2)$$

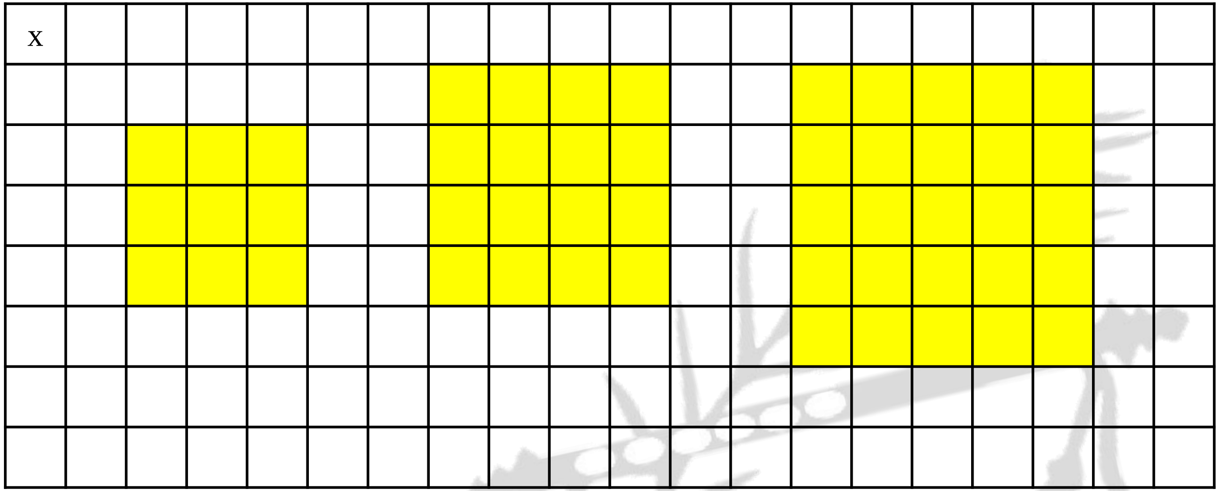
  – where,
    - S_1 is the set formed from elements of S associated with an object (S in this case)
    - S_2 is the set of elements of S associated with the corresponding background (W – S)
- The set contains all the points at which, S_1 found a match (hit) in I and S_2 found a match in I^c

Hit or Miss Transform

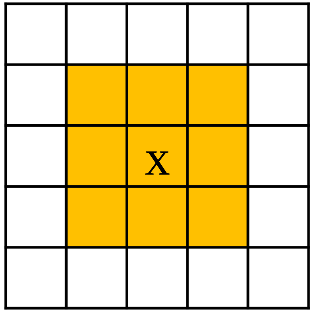
- Using the definition of set difference & the dual relationship between erosion & dilation, the equation can be rewritten as

$$I \circledast S = (I \ominus S_1) - (I \oplus \hat{S}_2)$$

Hit or Miss Transform

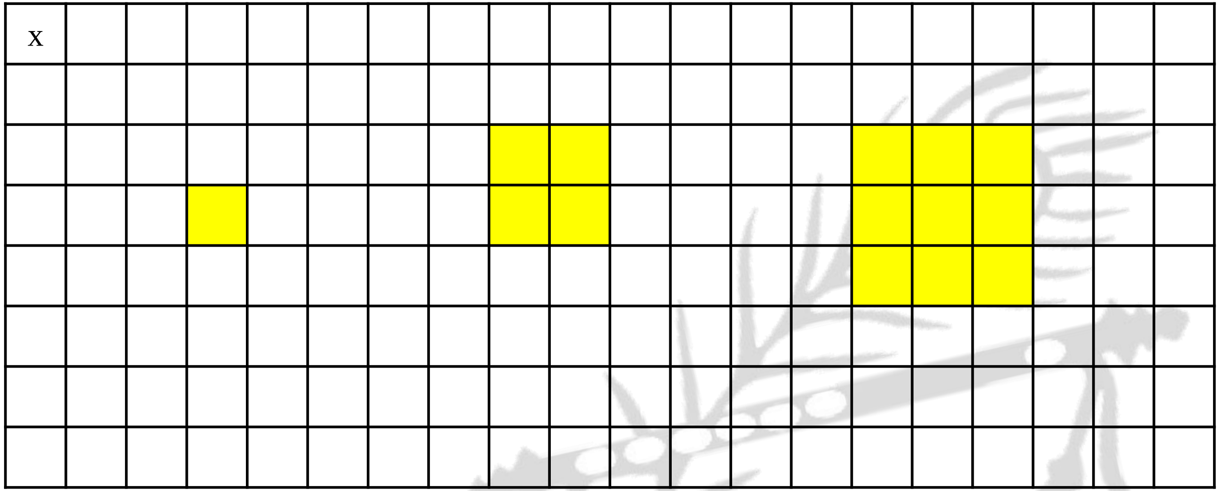


I

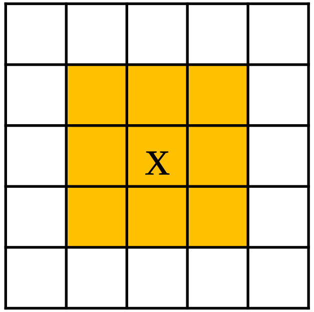


S

Hit or Miss Transform

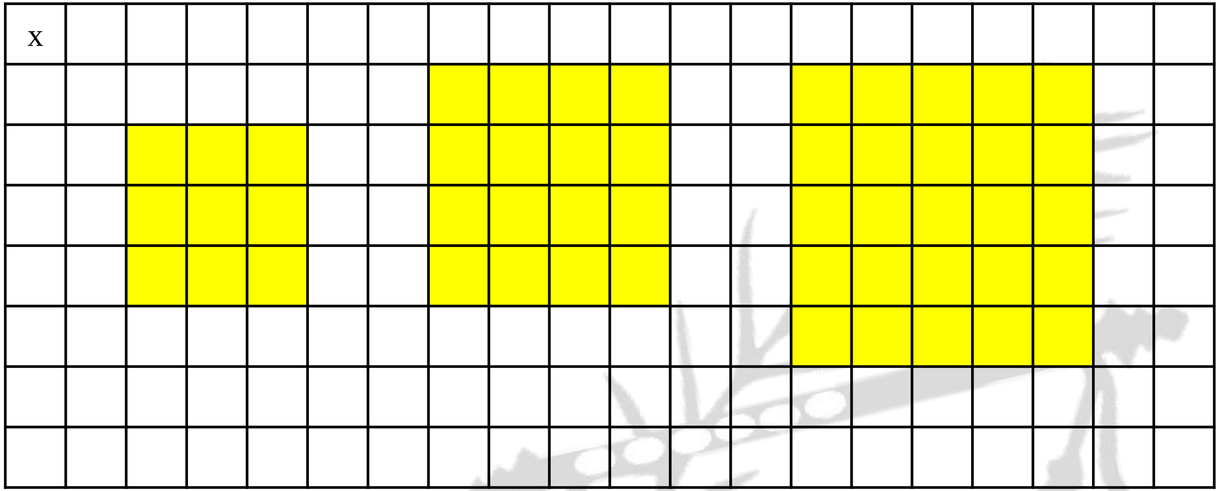


$I \Theta S$

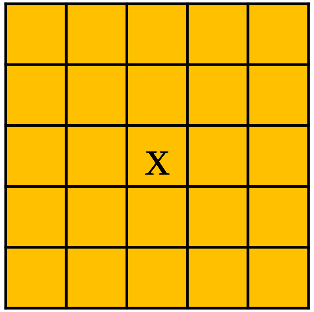


S

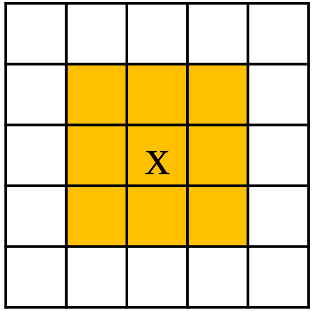
Hit or Miss Transform



I

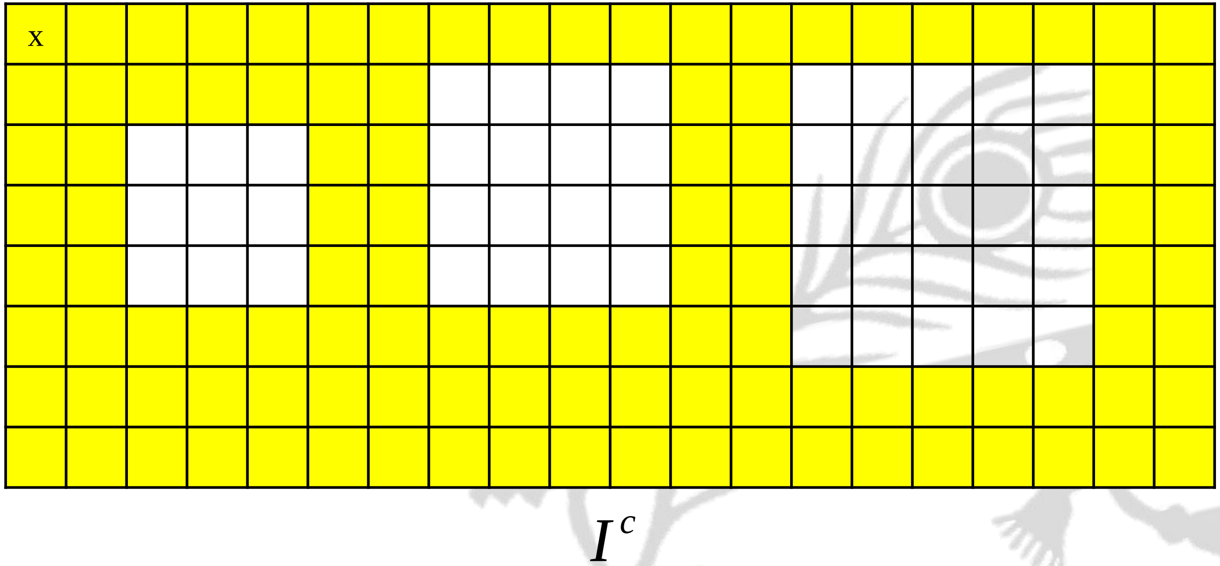


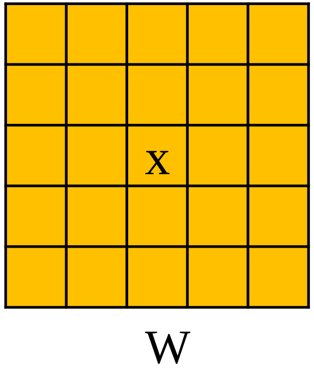
W

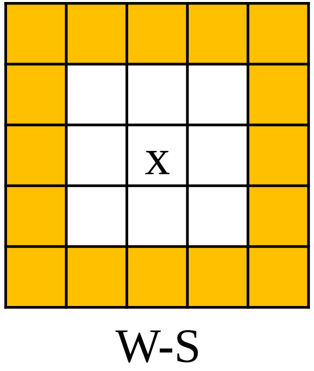


S

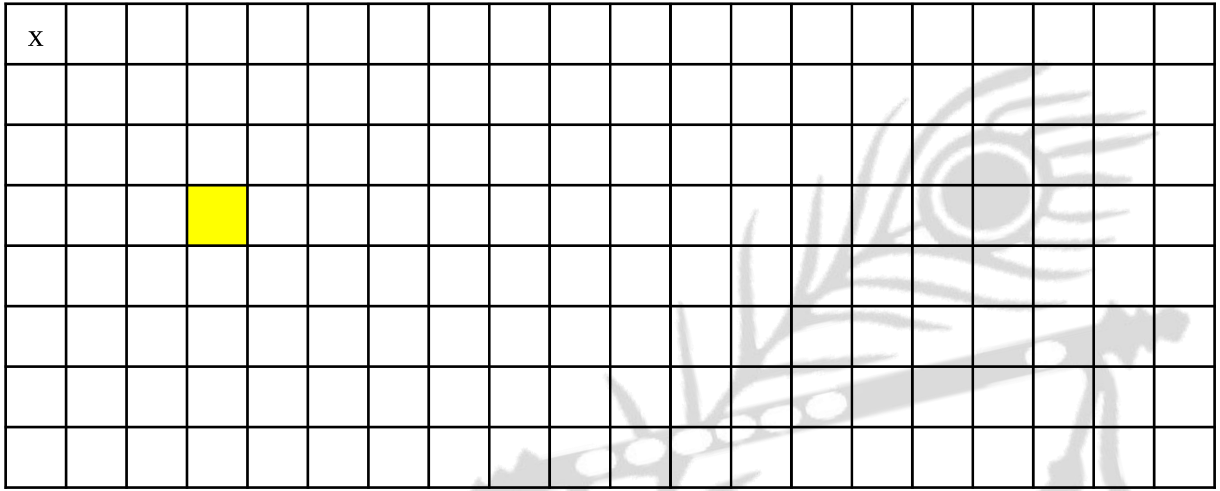
Hit or Miss Transform



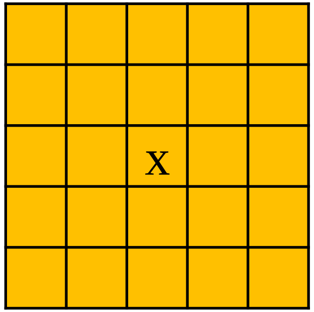




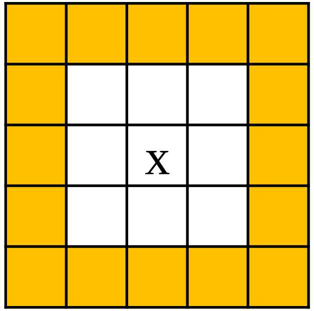
Hit or Miss Transform



$$I^c \Theta (W - S)$$

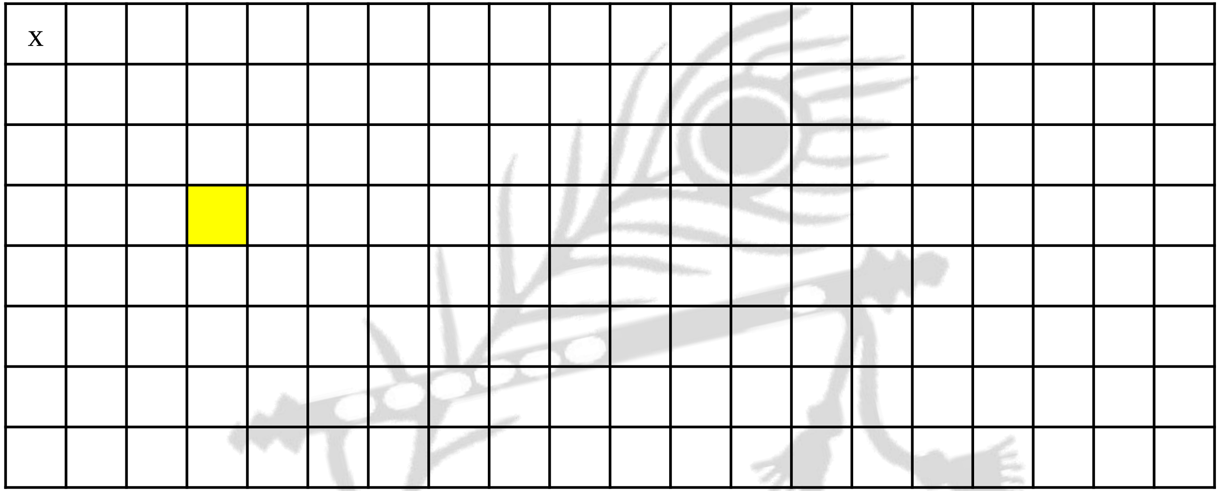


W



W-S

Hit or Miss Transform



$$I \circledast S = (I \Theta S) \bigcap (I^c \Theta (W - S))$$

Eg

- Find I⊛S

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

←I

S→

x	1	x
0	<u>1</u>	1
0	0	x

Eg:

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

$I \Theta S$

x	1	x
0	$\underline{1}$	1
0	0	x

S

Eg:

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

I^c

x	1	x
0	<u>1</u>	1
0	0	x

S

Eg:

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

I^c

x	1	x
0	<u>1</u>	1
0	0	x

S

x	0	x
1	<u>0</u>	0
1	1	x

W-S

Eg:

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

$$I^c \Theta (W - S)$$

x	1	x
0	<u>1</u>	1
0	0	x

S

x	0	x
1	<u>0</u>	0
1	1	x

W-S

Eg:

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

Eg

- Find $I \circledast S$

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

←I

S→

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

Eg

- Find I⊛S

0	0	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	1	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

$I \Theta S$

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

$I^c \Theta (W - S)$

Boundary Extraction

Boundary Extraction

- The boundary of a region R is the set of pixels in the region that have one or more neighbours that are not in R
- The boundary of a set I can be obtained by first eroding I by S and then performing the set difference between S and its erosion
- It is given by

$$\beta(I) = I - (I \Theta S)$$

Boundary Extraction

- Eg:
  - find $\beta(I)$

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

←I

S→

1	1	1
1	1	1
1	1	1

Boundary Extraction

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

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

$I \Theta S$

Boundary Extraction

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

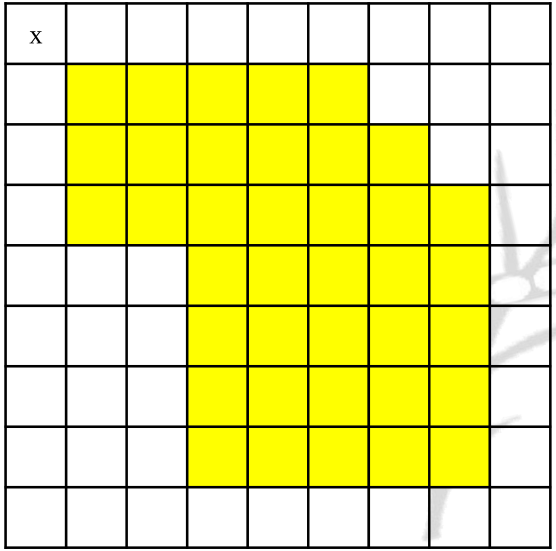
$I - (I \Theta S)$

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

$I \Theta S$

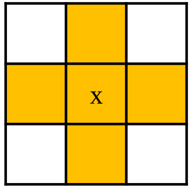
Boundary Extraction

- Eg:
  - find $\beta(I)$

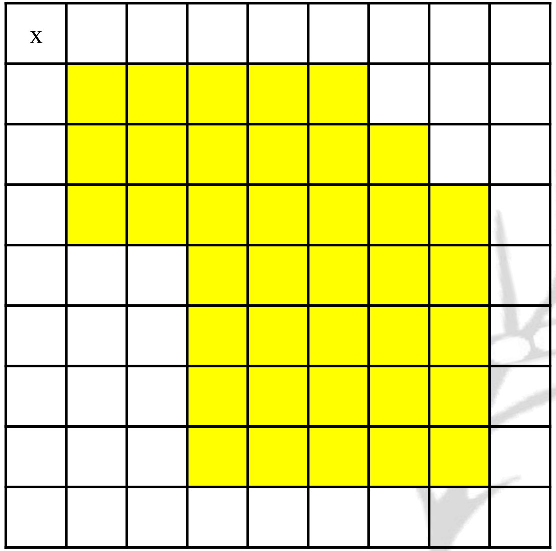


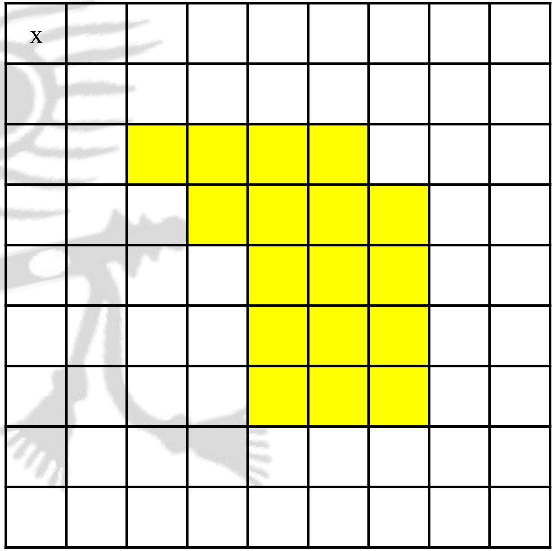
←I

S→



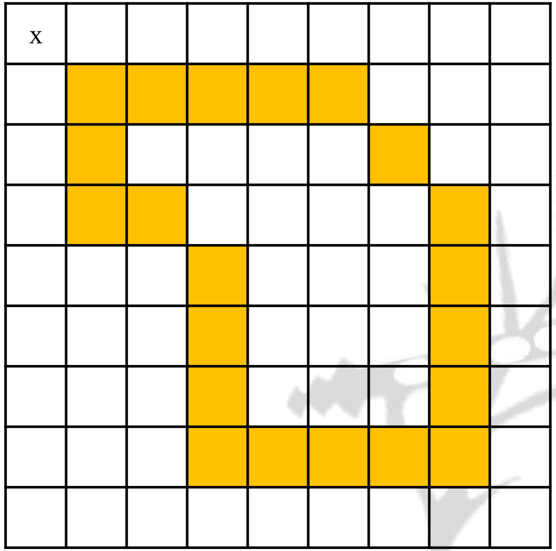
Boundary Extraction





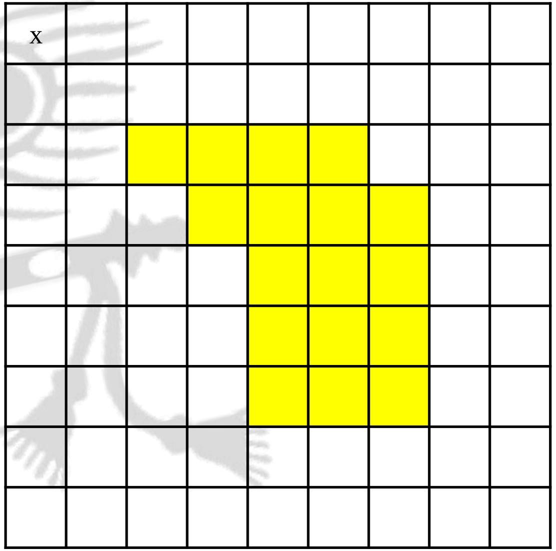
$I \Theta S$

Boundary Extraction



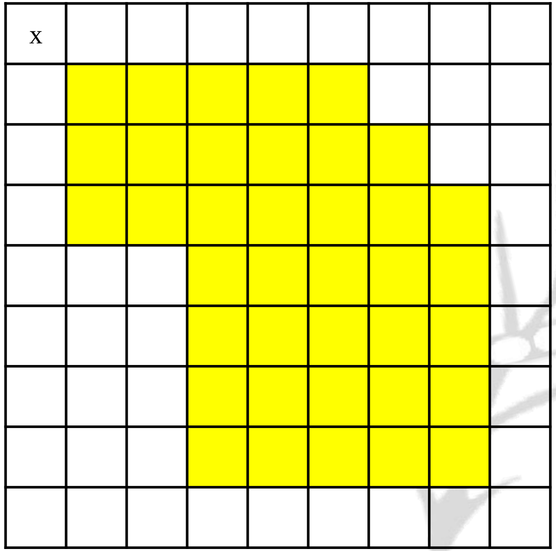
$I - (I \Theta S)$

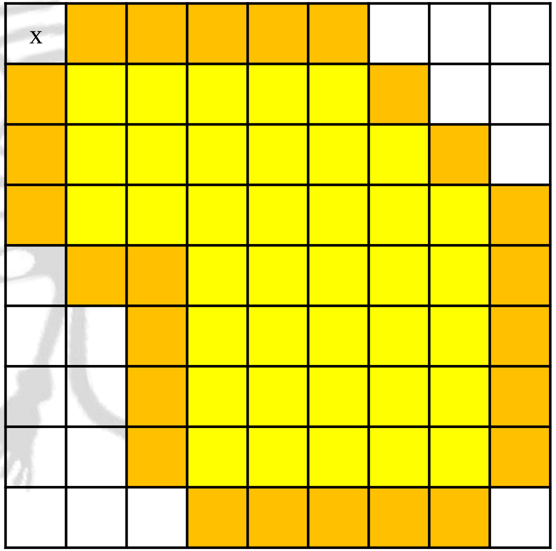
Internal Boundary



$I \Theta S$

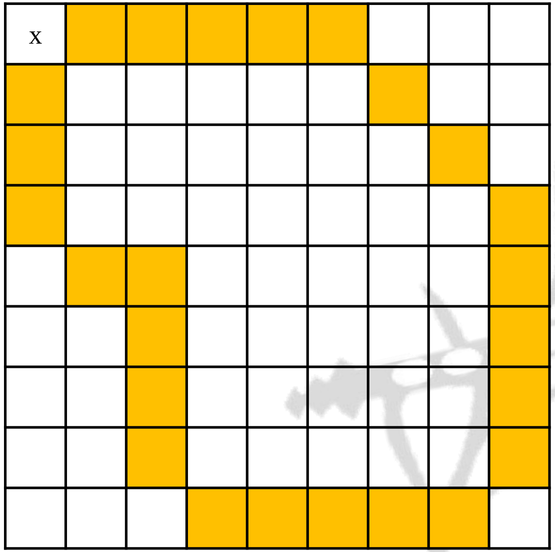
Boundary Extraction





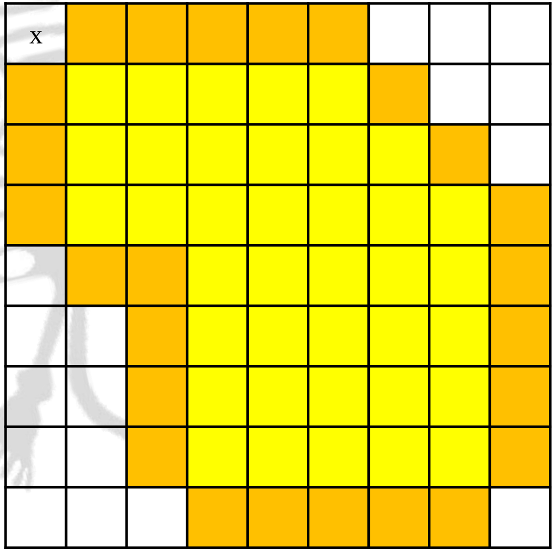
$I \oplus S$

Boundary Extraction



$(I \oplus S) - \mathrm{I}$

External Boundary



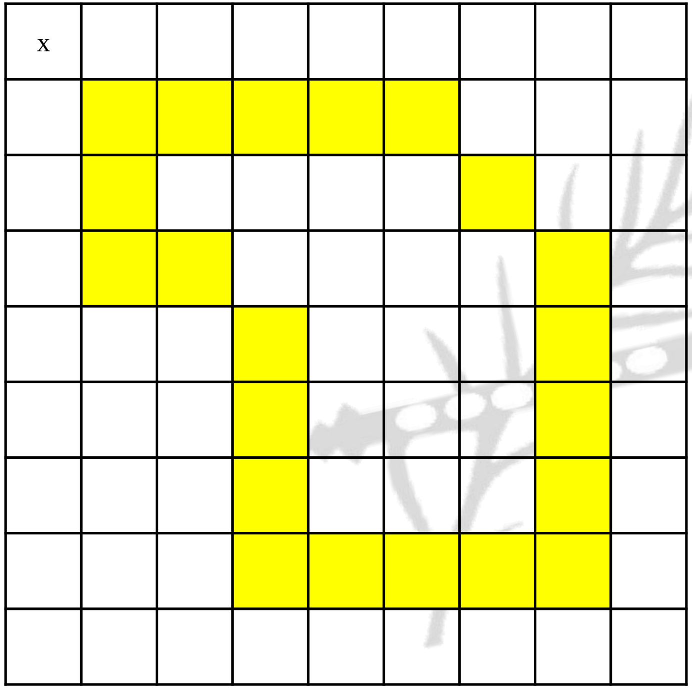
$I \oplus S$

Region Filling

Region Filling

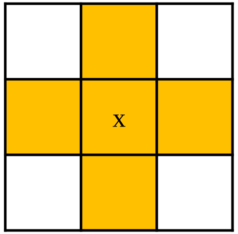
- Region filling is used to fill the selected region of the object
- Steps includes
  - Choose a seed point X_0
  - Iterate $X_k = (X_{k-1} \oplus S) \cap I^c$ until convergence

Region Filling

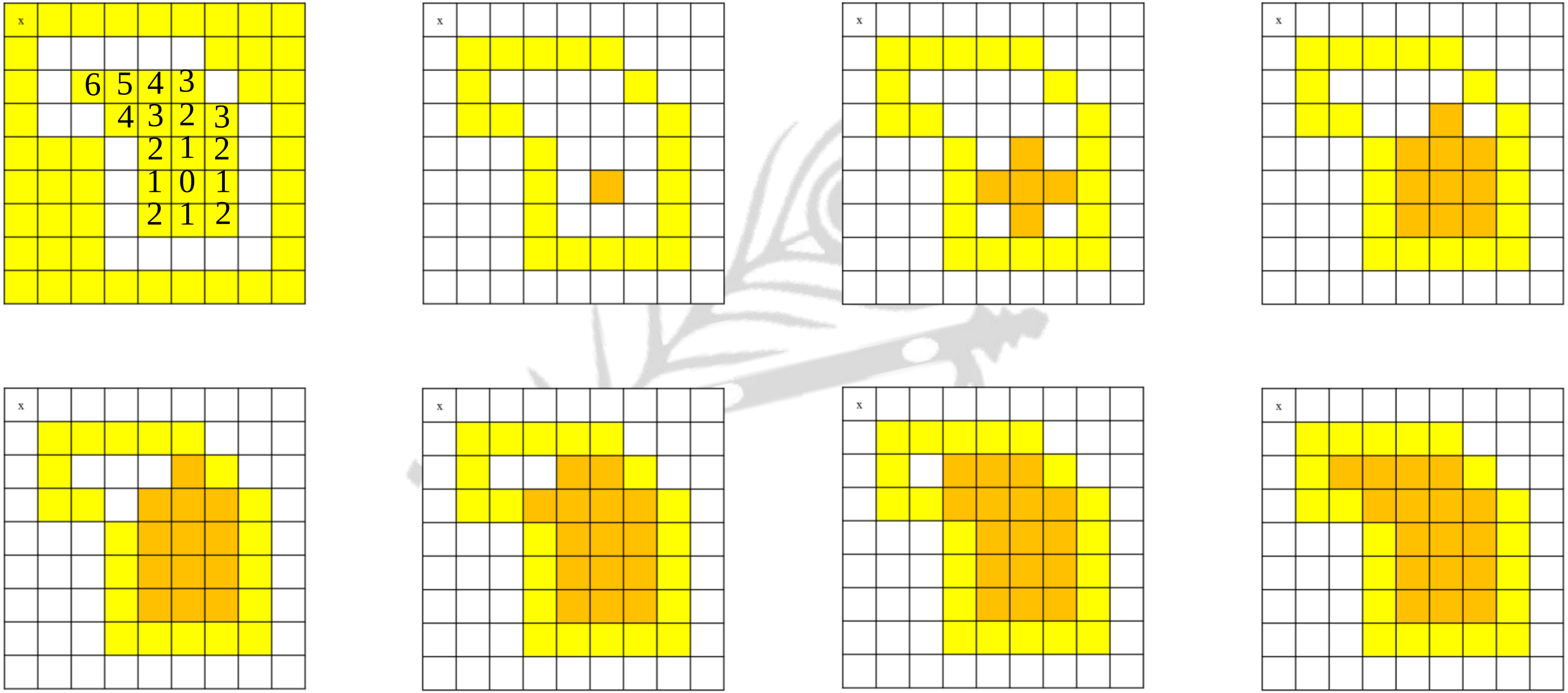


←I

S→

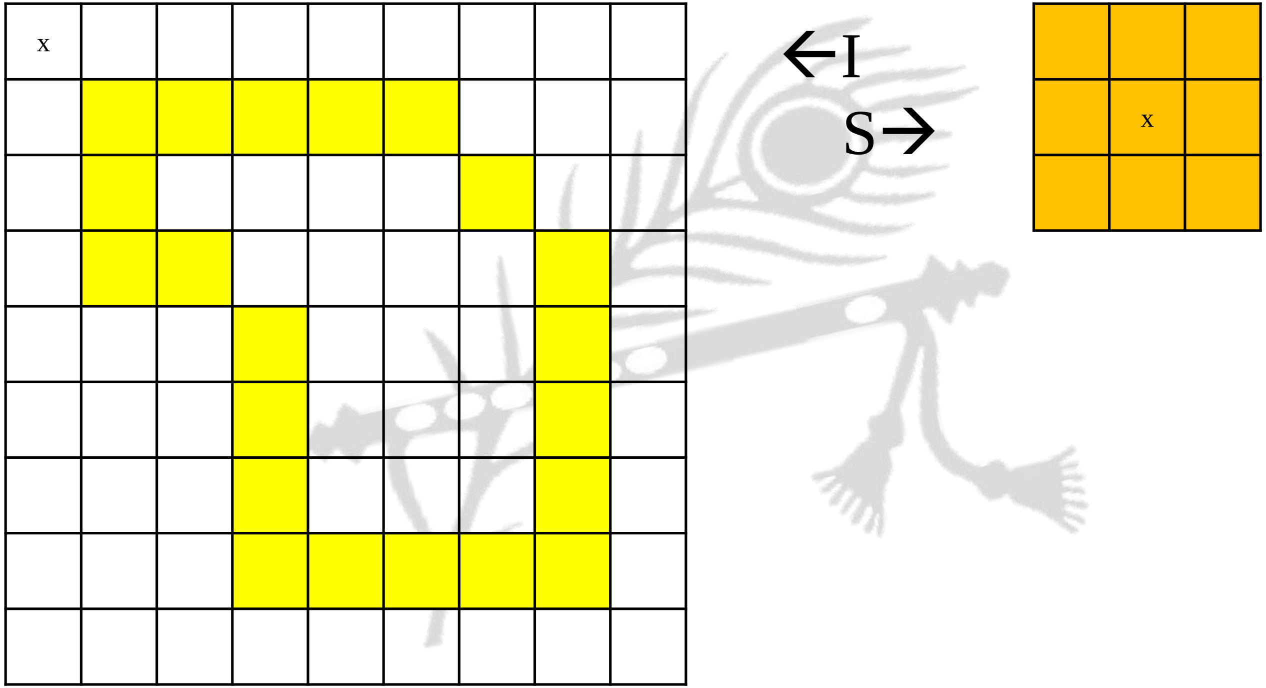


Region Filling



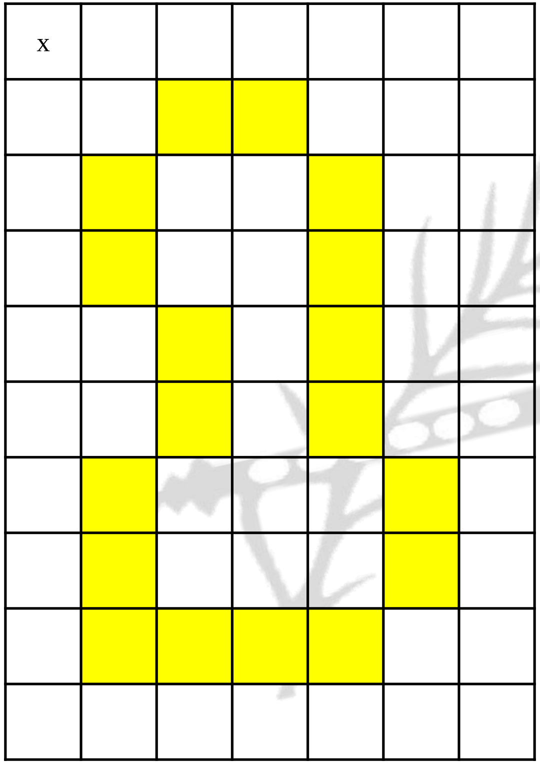
Region Filling

- Eg:



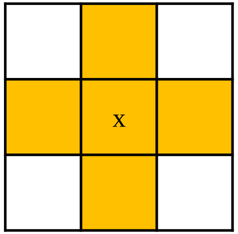
Region Filling

- Eg:



←I

S→

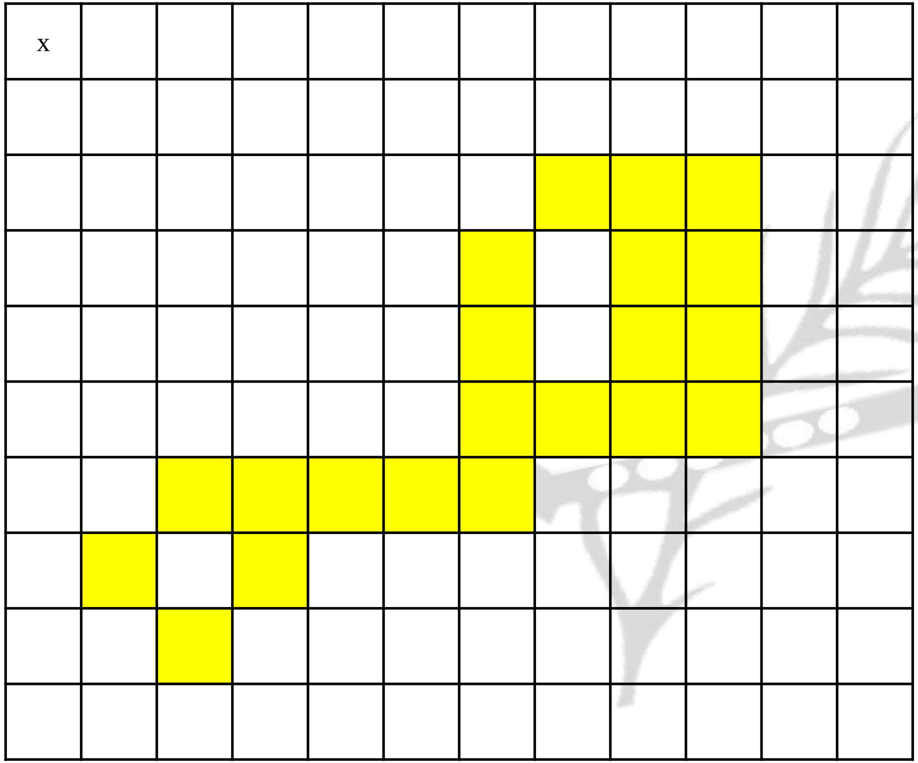


EXTRACTION OF CONNECTED COMPONENTS

Extraction of Connected Components

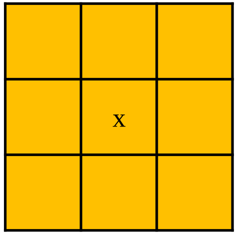
- Connected component labeling is used in computer vision to detect connected regions in the images
- It groups the pixels into components based on the pixel connectivity
- Steps includes
  - Choose a seed point X_0
  - Iterate $X_k = (X_{k-1} \oplus S) \cap I$ until convergence

Extraction of Connected Components

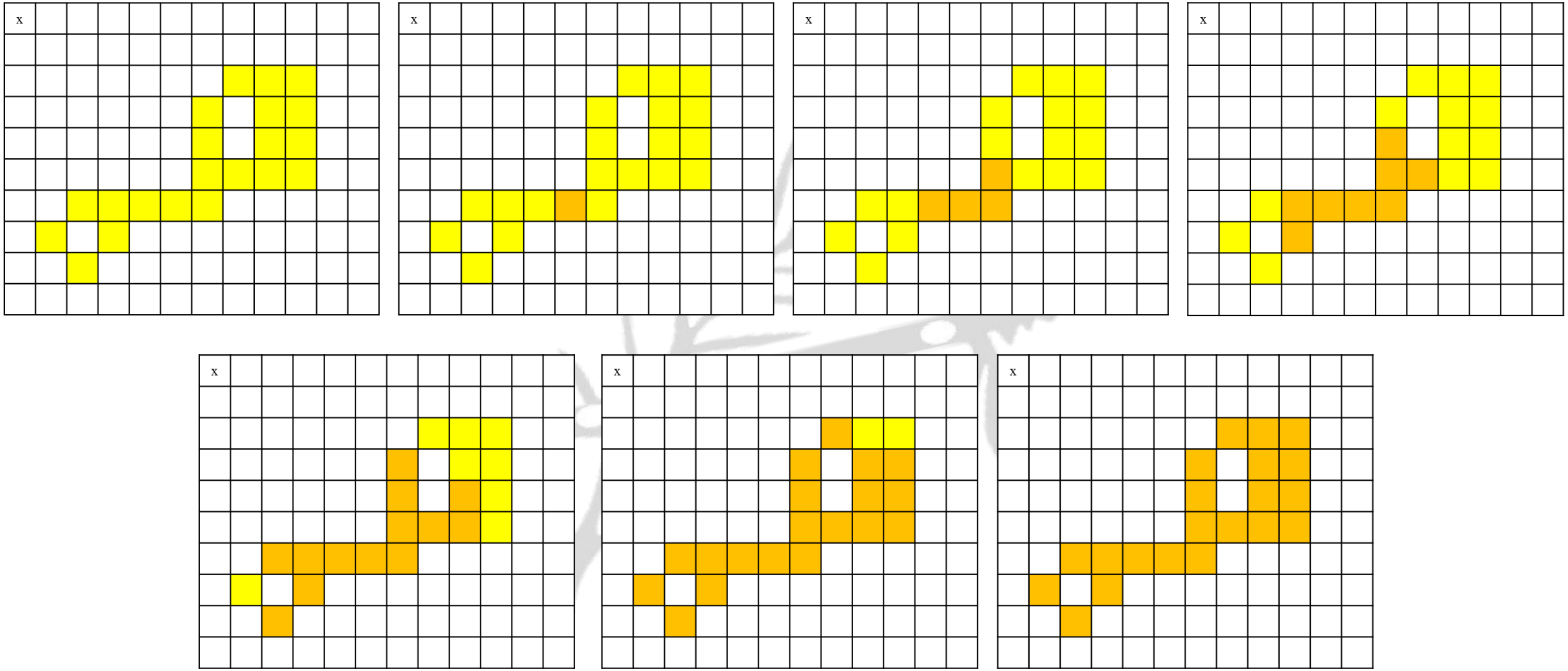


←I

S→

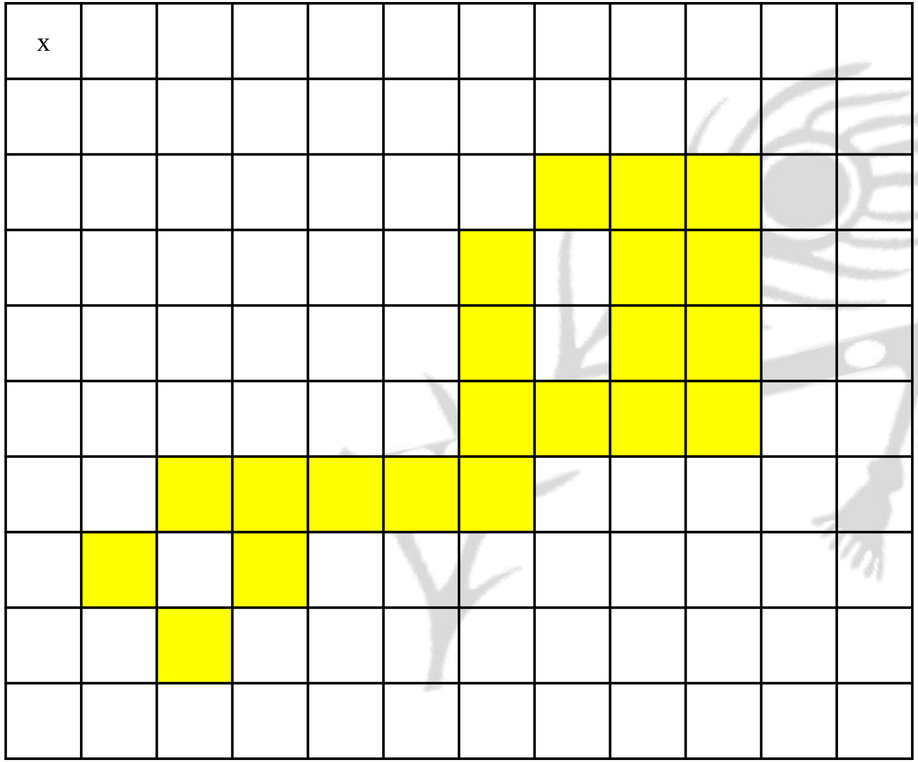


Extraction of Connected Components



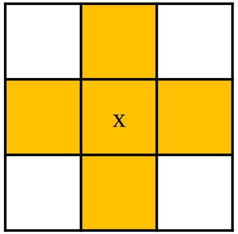
Extraction of Connected Components

- Eg:



←I

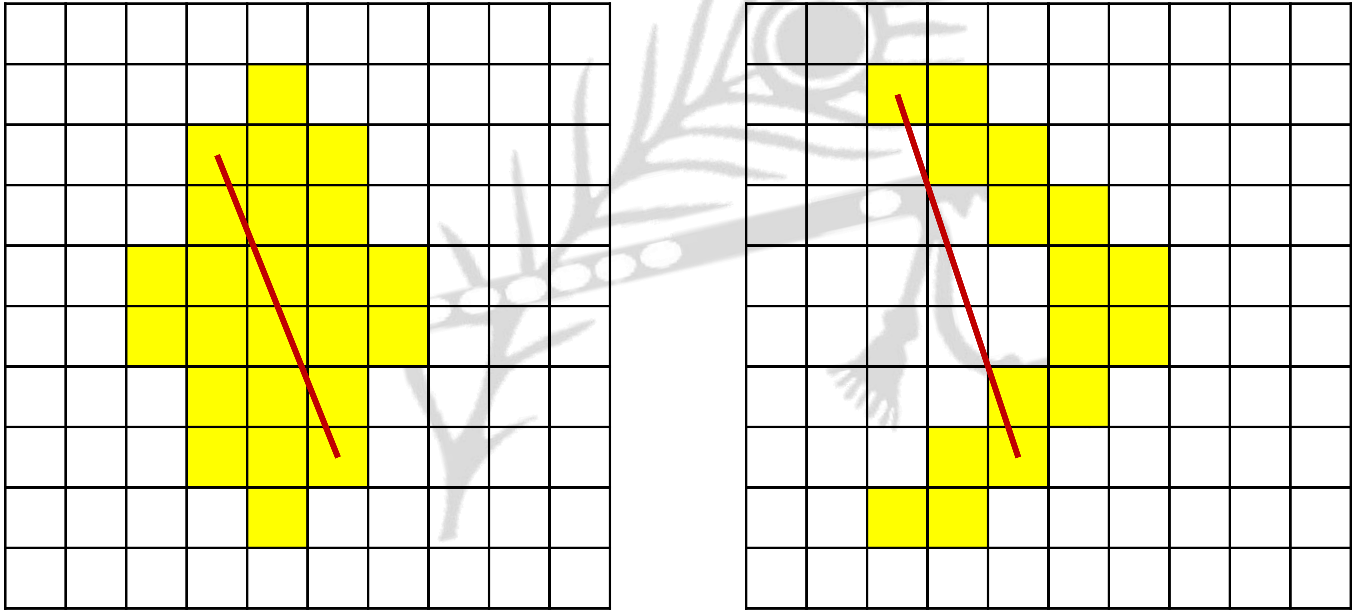
S→



CONVEX HULL

Convex Hull

- A set I is said to be convex if the straight line segment joining any two points in I lies entirely within I

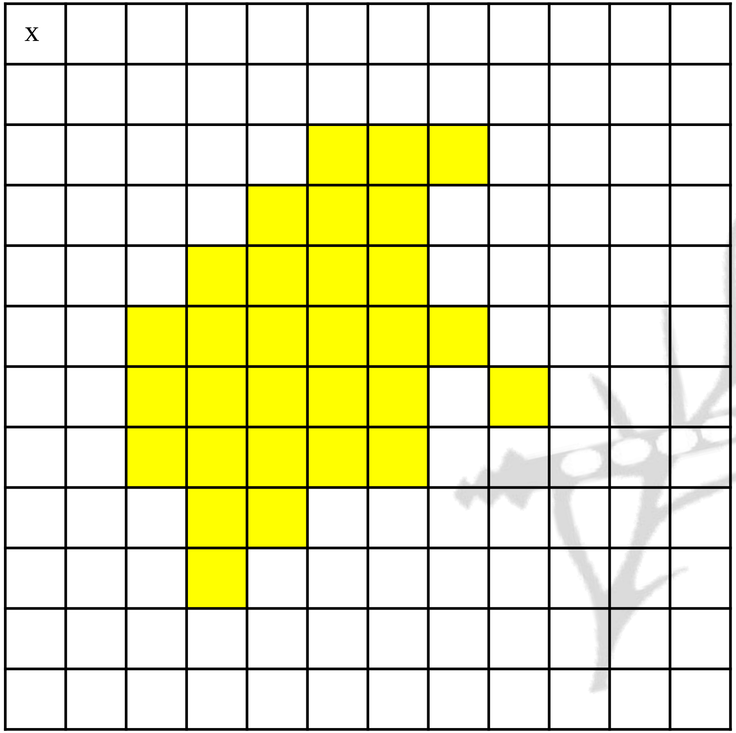


Convex Hull

- Convex Hull (H) = Minimum convex set containing set I
- Hull Deficiency (D) = H – I

- Steps include
  - Choose a seed point X_0
  - do i = 0 to 3
    - Iterate $X_k = (X_{k-1} \circledast S_i) \bigcup I$ until convergence
  - Minimize convex set using bounding box of I

Convex Hull



←I

S→

1	x	x
1	0	x
1	x	x

S_0

1	1	1
x	0	x
x	x	x

S_1

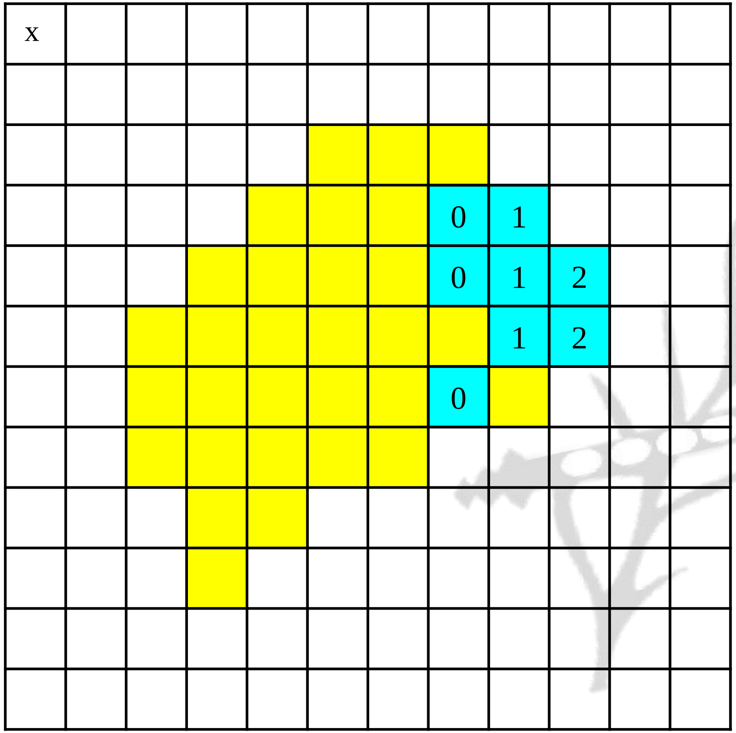
x	x	1
x	0	1
x	x	1

S_2

x	x	x
x	0	x
1	1	1

S_3

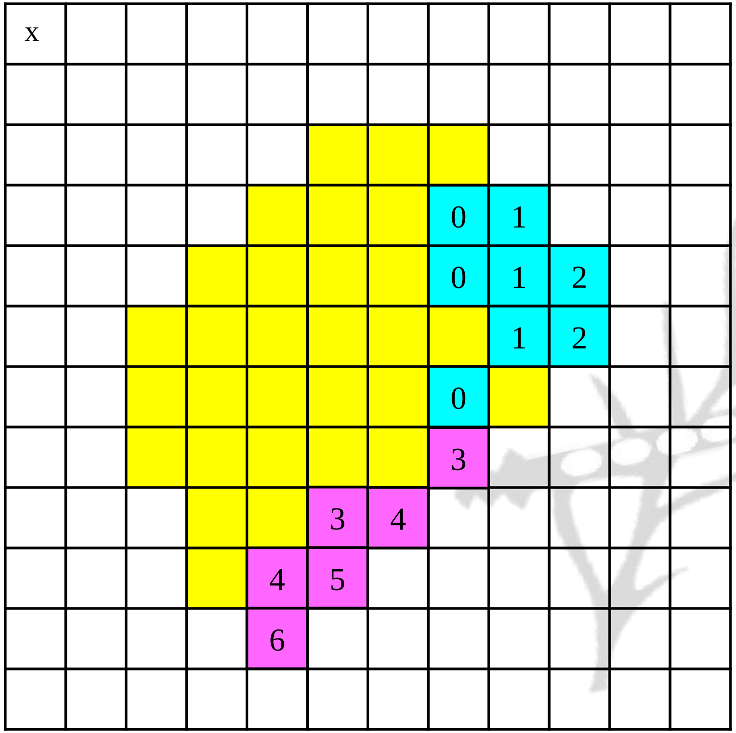
Convex Hull

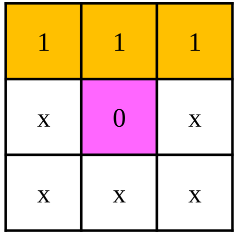


1	x	x
1	0	x
1	x	x

S_0

Convex Hull





S_1

Convex Hull

x											
							0	1			
							0	1	2		
								1	2		
	7						0				
	8						3				
		7			3	4					
				4	5						
				6							

x	x	1
x	0	1
x	x	1

S_2

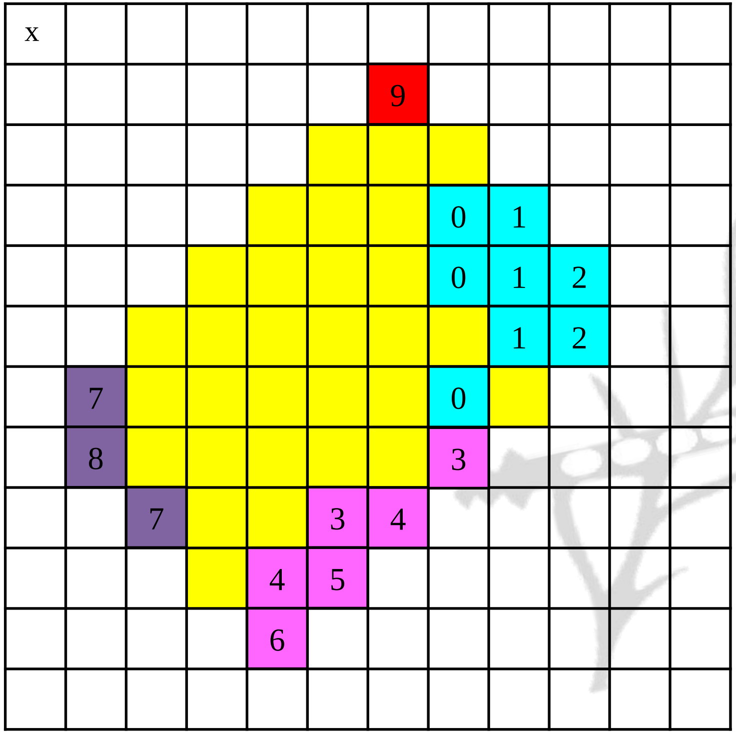
Convex Hull

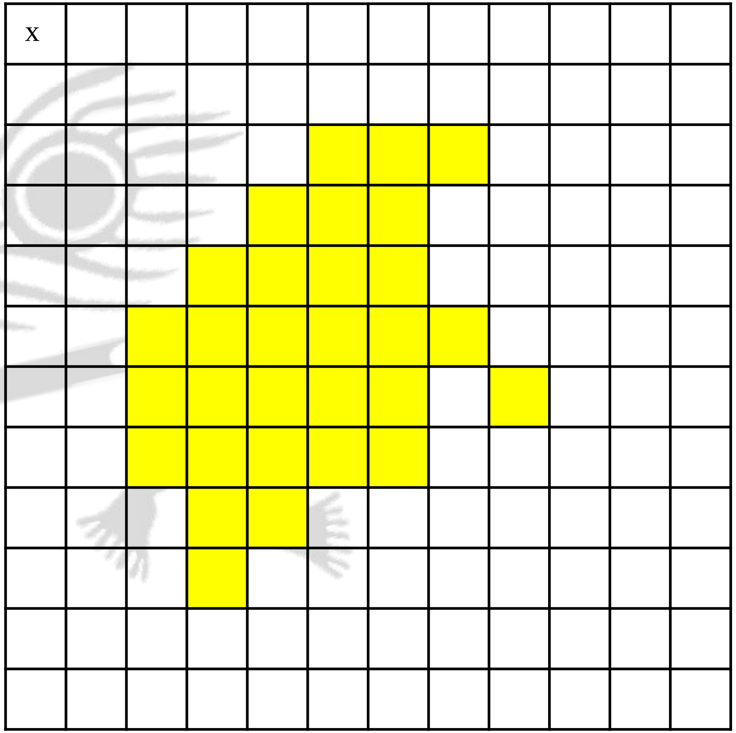
x											
						9					
							0	1			
							0	1	2		
								1	2		
	7						0				
	8						3				
		7			3	4					
				4	5						
				6							

x	x	x
x	0	x
1	1	1

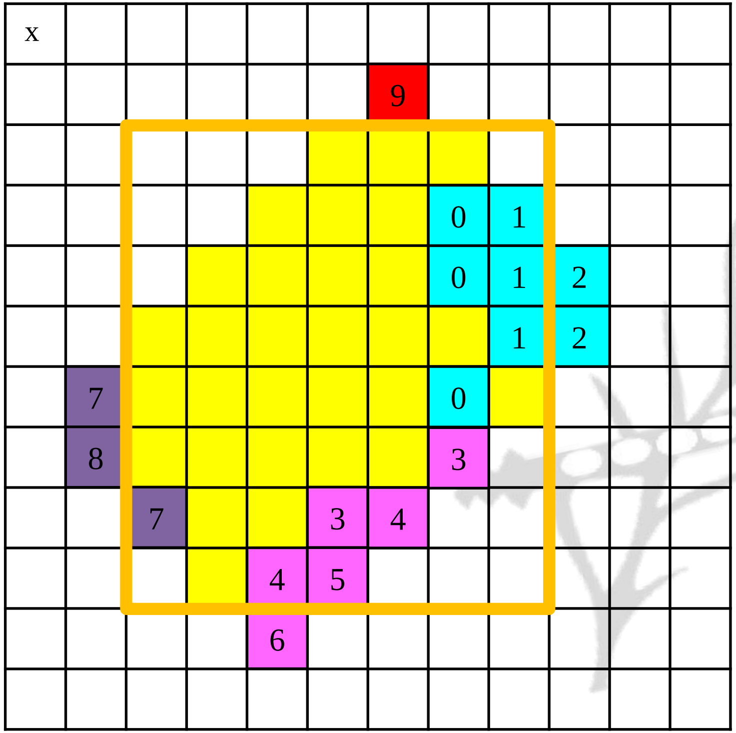
S_3

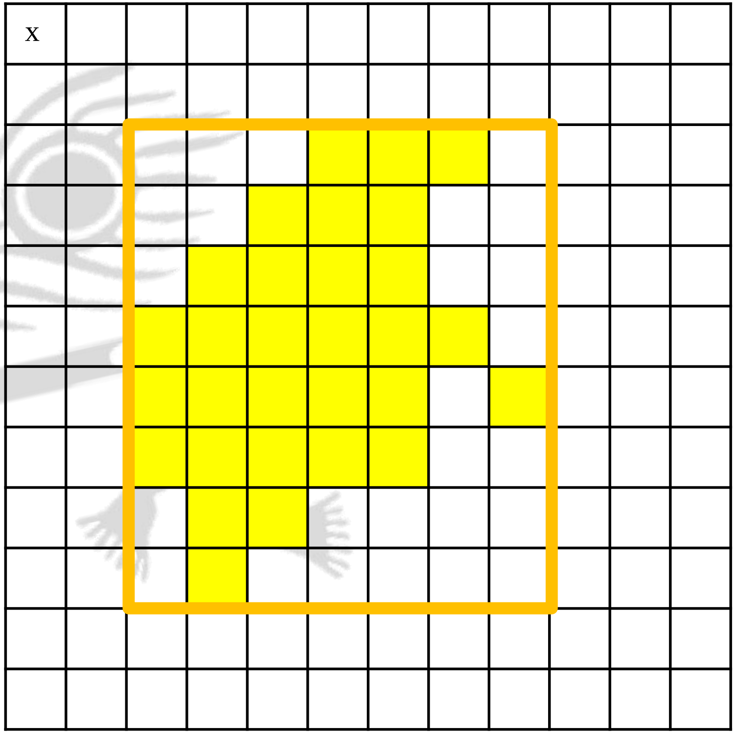
Convex Hull



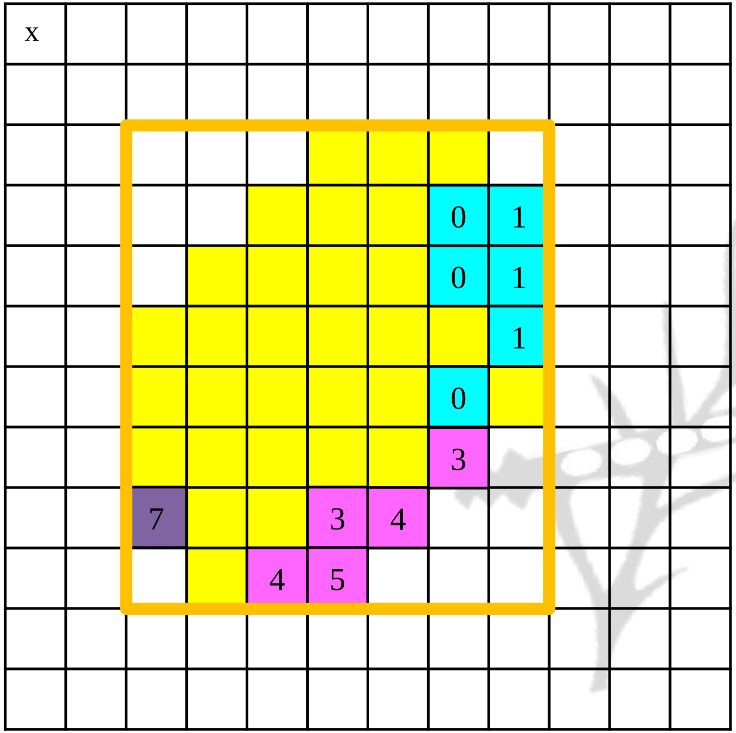


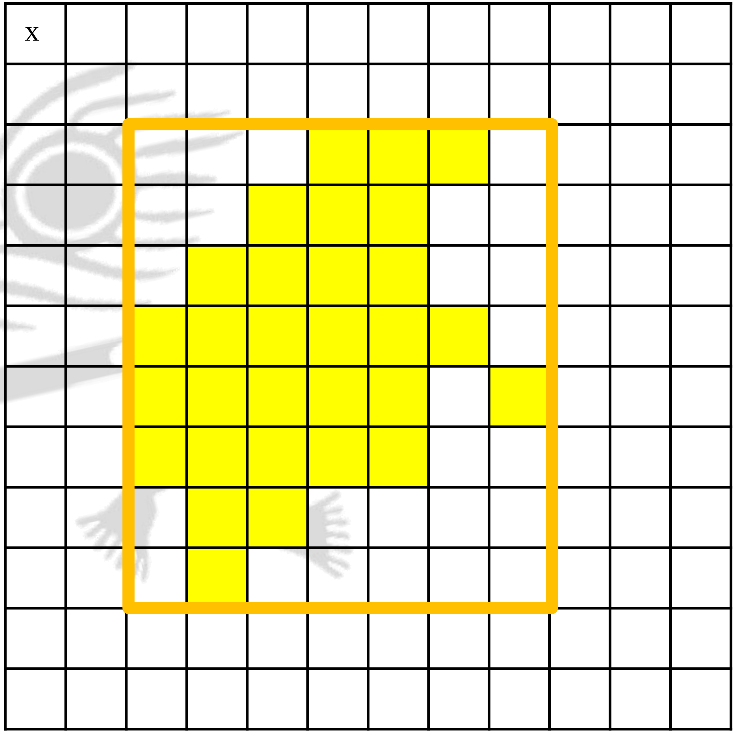
Convex Hull



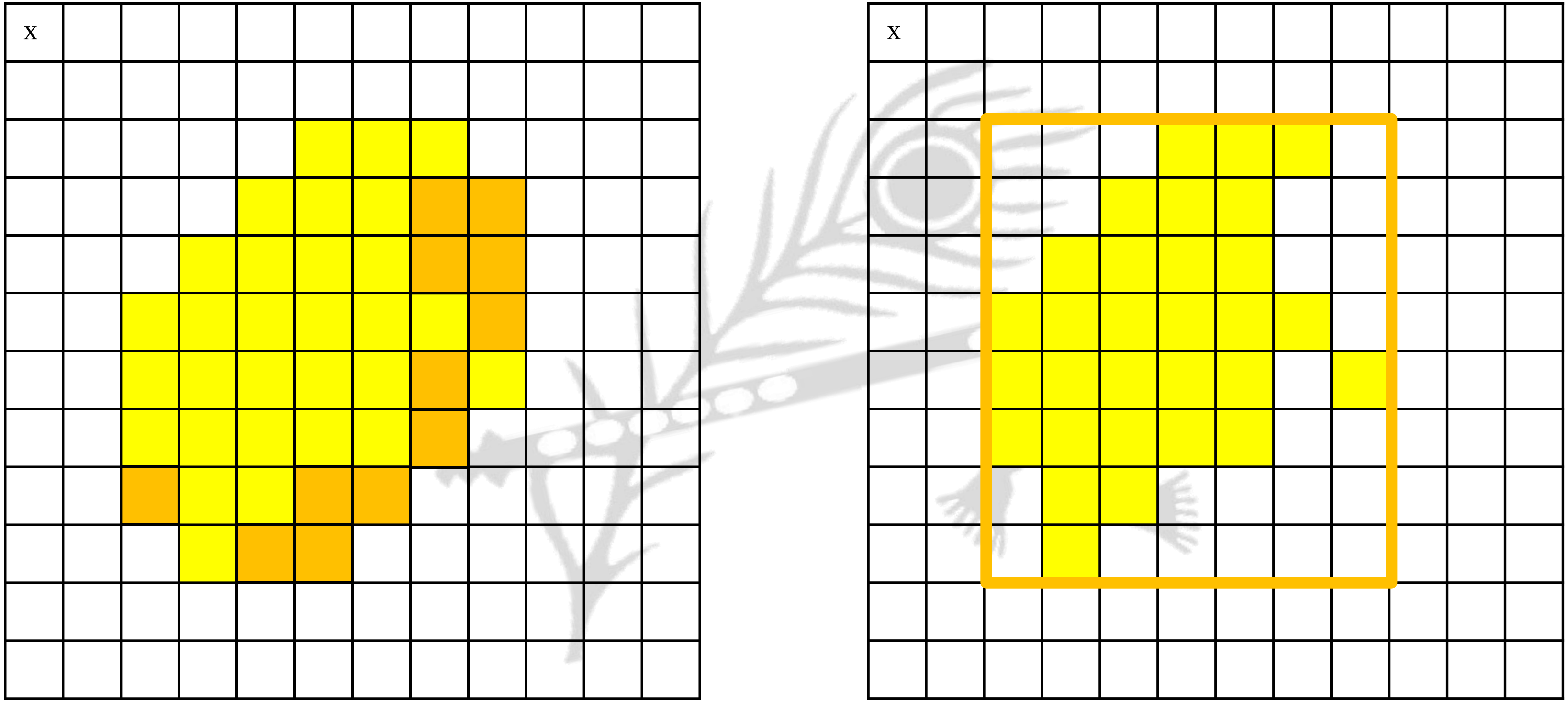


Convex Hull

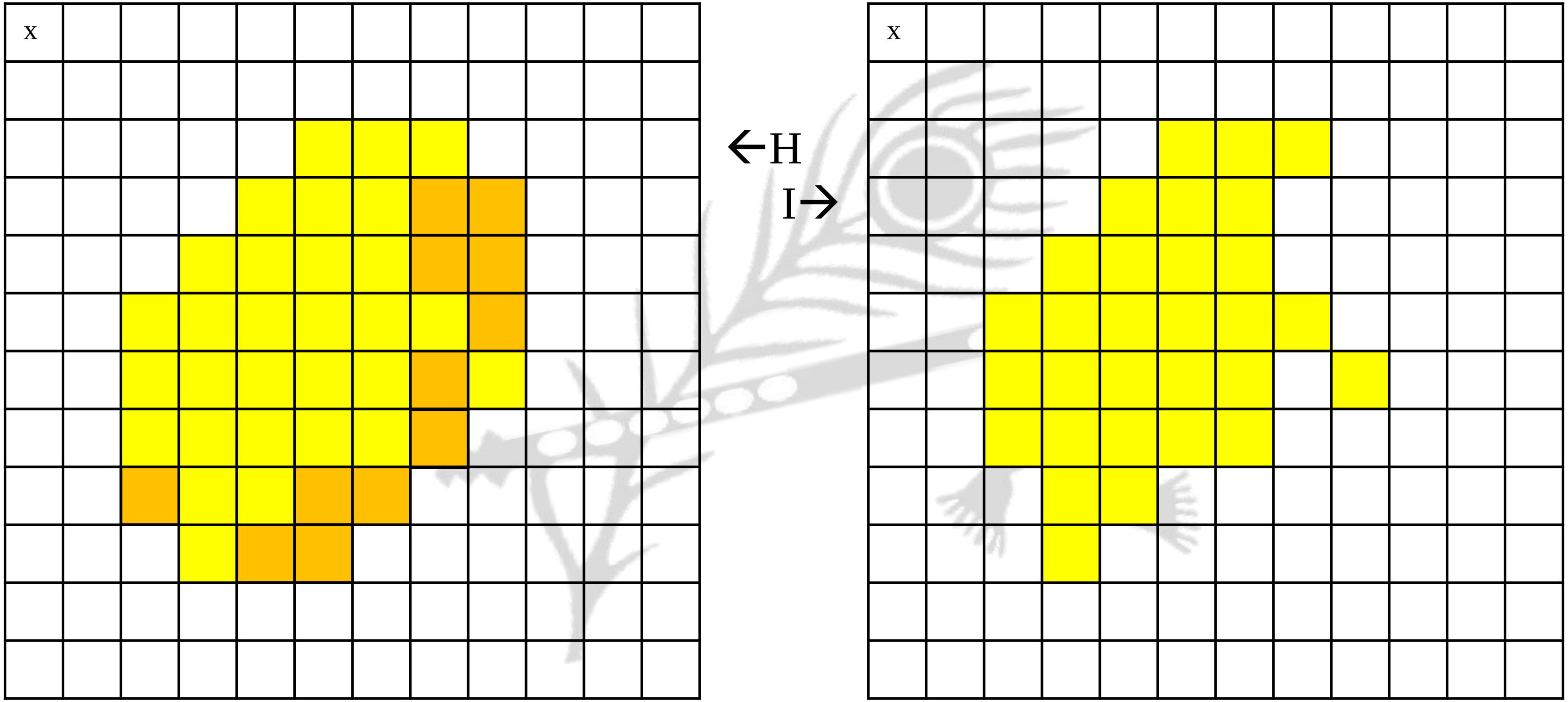




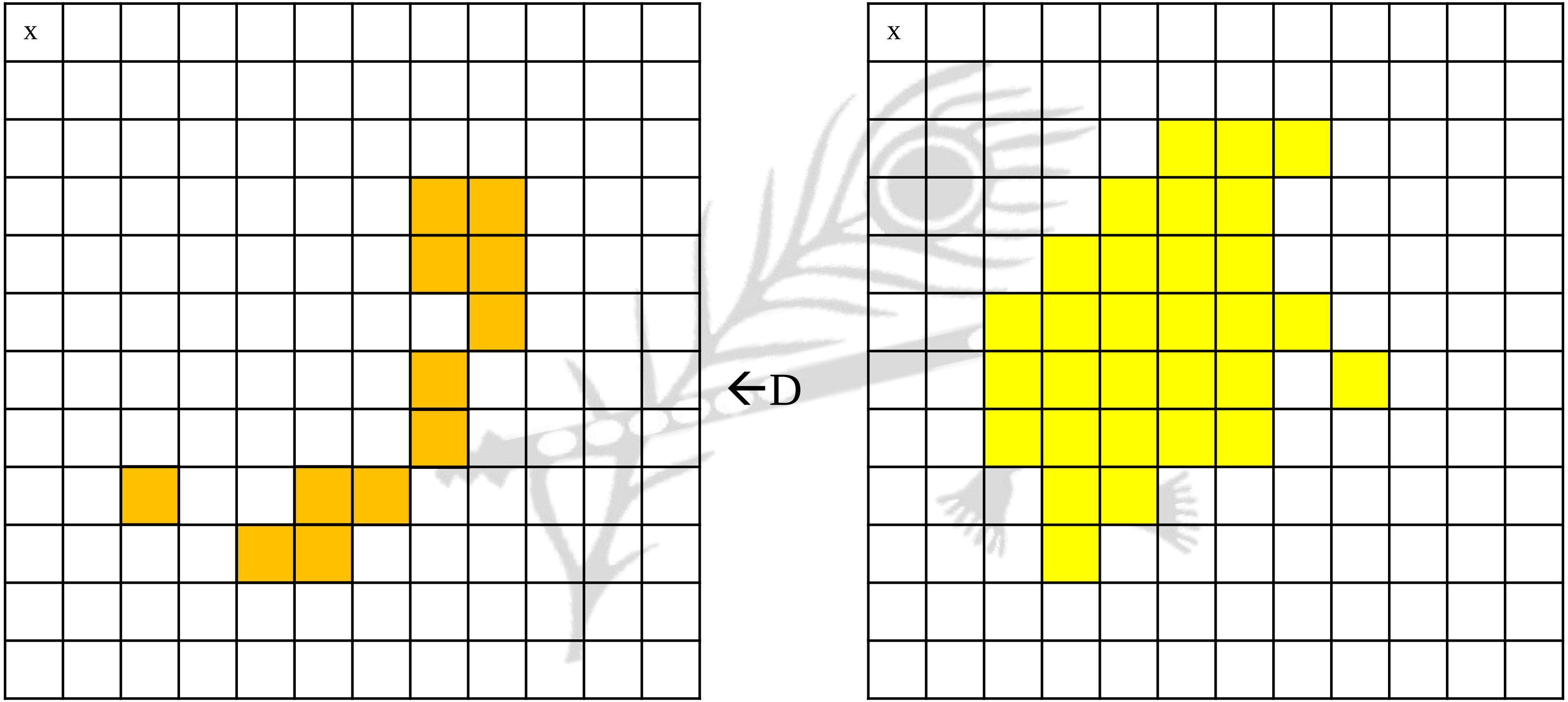
Convex Hull



Convex Hull



Convex Hull



MORPHOLOGICAL THINNING AND THICKENING

Thinning

- It is an operation that is used to remove selected foreground pixels from binary images
- It is the process of reducing an object in a digital image to the minimum size
- It is given by

$$I \otimes S = I - (I \circledast S_i)$$

$$I \otimes S = I \bigcap (I \circledast S_i)$$

$$I \otimes \{S\} = ((((I \circledast S_0) \circledast S_1) \circledast S_2) \ldots \circledast S_7)$$

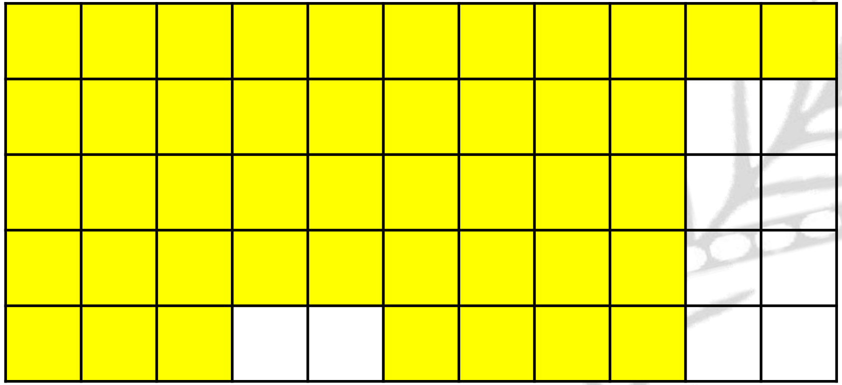
Thinning

- Thinning is basically a search & delete process
- It removes only those boundary pixels from the image whose deletion
  - Does not connectivity of their neighbours locally
  - Does not reduce the length of the already thinned curve

- Critical pixel
  - Its deletion changes the connectivity of its neighbourhood locally

- End pixel
  - Its deletion reduces the length of an already thinned curve

Thinning



I

0	0	0
X	1	X
1	1	1

S_0

X	0	0
1	1	0
1	1	X

S_1

1	X	0
1	1	0
1	X	0

S_2

1	1	X
1	1	0
X	0	0

S_3

1	1	1
X	1	X
0	0	0

S_4

X	1	1
0	1	1
0	0	X

S_5

0	X	1
0	1	1
0	X	1

S_6

0	0	X
0	1	1
X	1	1

S_7

Thinning

	0	0	0	0	0	0	0			

0	0	0
X	1	X
1	1	1

S_0

Thinning

X	0	0
1	1	0
1	1	X

S_1

	0	0	0	0	0	0	0			

Thinning

	0	0	0	0	0	0	0			
								0		
								0		

1	X	0
1	1	0
1	X	0

S_2

Thinning

	0	0	0	0	0	0	0			
								0		
								0		
		0								

1	1	X
1	1	0
X	0	0

S_3

Thinning

	0	0	0	0	0	0	0			
								0		
			0					0		
	0	0			0	0				

1	1	1
X	1	X
0	0	0

S_4

Thinning

	0	0	0	0	0	0	0			
								0		
			0	0				0		
	0	0			0	0				

X	1	1
0	1	1
0	0	X

S_5

Thinning

	0	0	0	0	0	0	0			
0								0		
			0	0				0		
	0	0			0	0				

0	X	1
0	1	1
0	X	1

S_6

Thinning

	0	0	0	0	0	0	0			
0								0		
			0	0				0		
	0	0			0	0				

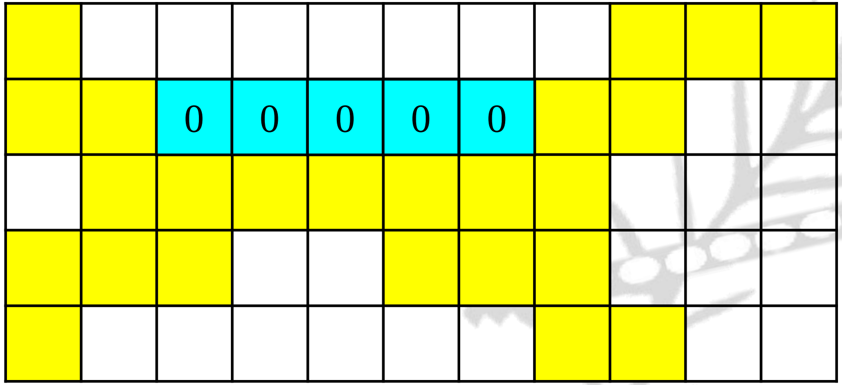
0	0	X
0	1	1
X	1	1

S_7

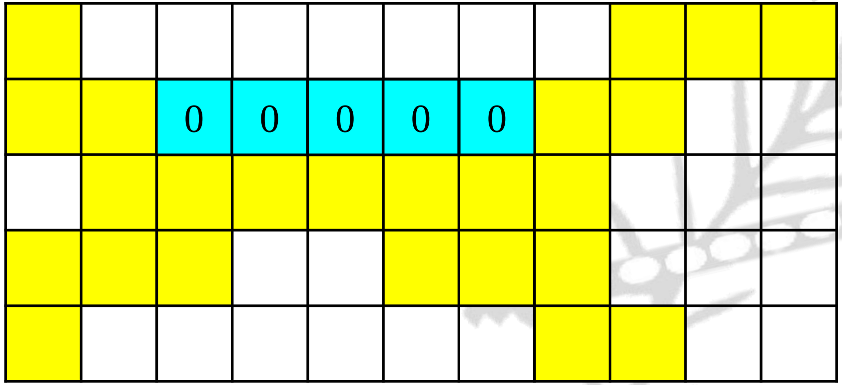
Thinning

0	0	0
X	1	X
1	1	1

S_0



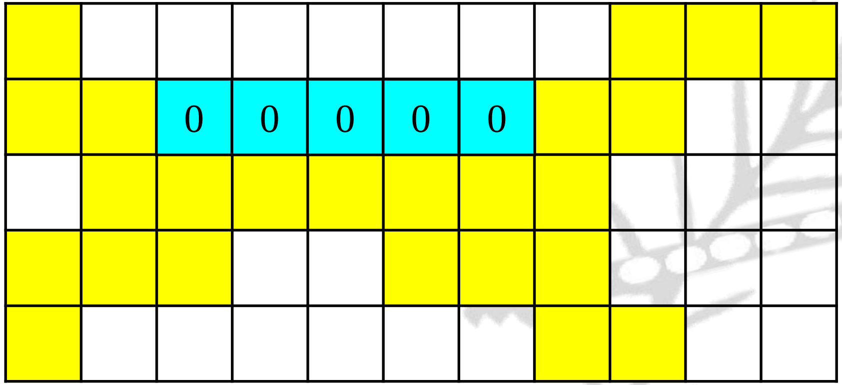
Thinning



X	0	0
1	1	0
1	1	X

S_1

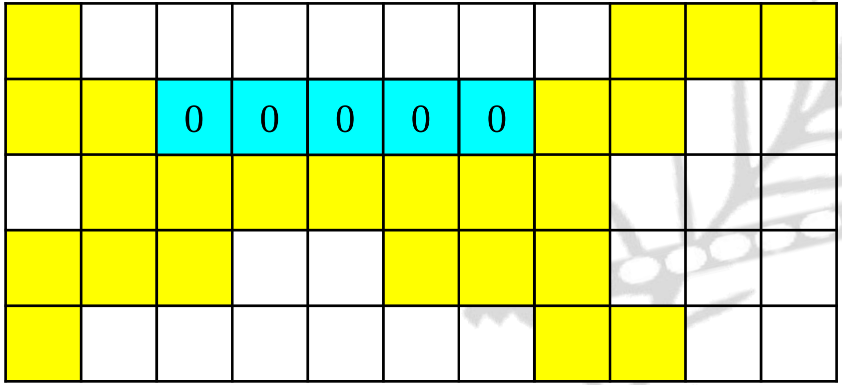
Thinning



1	X	0
1	1	0
1	X	0

S_2

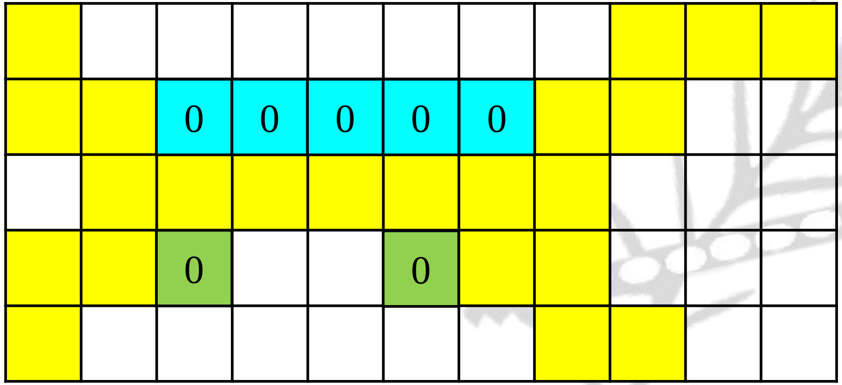
Thinning



1	1	X
1	1	0
X	0	0

S_3

Thinning



1	1	1
X	1	X
0	0	0

S_4

Thinning

		0	0	0	0	0				
		0			0	0				

X	1	1
0	1	1
0	0	X

S_5

Thinning

		0	0	0	0	0				
		0			0	0				

0	X	1
0	1	1
0	X	1

S_6

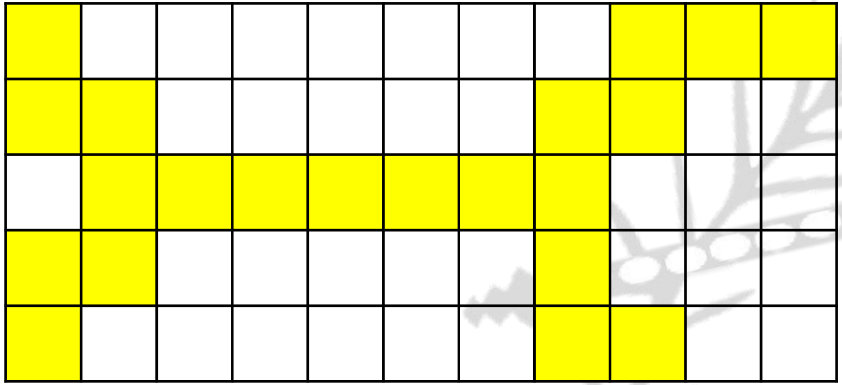
Thinning

		0	0	0	0	0				
		0			0	0				

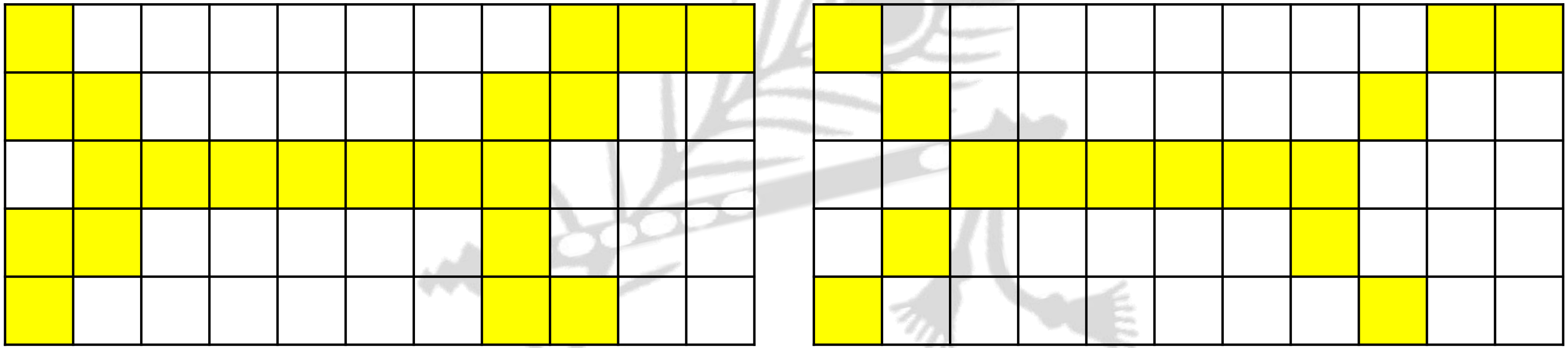
0	0	X
0	1	1
X	1	1

S_7

Thinning

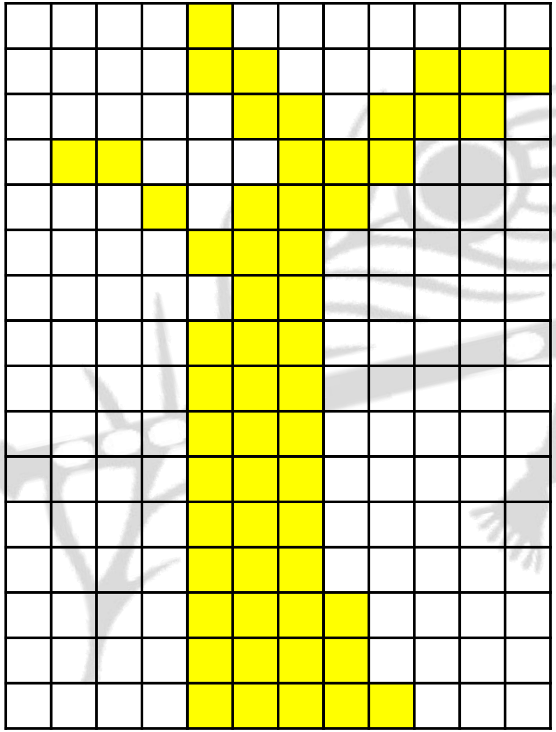


Thinning



Thinning

- Eg:



Thickening

- Thickening is the morphological dual of thinning
- It is defined as

$$I \odot S = I \bigcup (I \circledast S_i)$$

$$I \odot \{S\} = ((((I \circledast S_0) \circledast S_1) \circledast S_2) \ldots \circledast S_7)$$

- Approach
  - Take I^c
  - res = Apply thinning on I^c
  - Take res^c

$$I \odot \{S\} = |(I^c \otimes S)\,\text{followed by isolated pixel removal}\,|^c$$

RECAP

- Dilation $$I \oplus S = \{z \mid (\hat{S})_z \cap I \neq \phi\}$$
- Erosion $$I \Theta S = \{z \mid (S)_z \cap I^c = \phi\}$$
- Opening $$I \circ S = (I \Theta S) \oplus S$$
- Closing $$I \bullet S = (I \oplus S) \Theta S$$
- Hit or Miss $$I \circledast S = (I \Theta S) \cap (I^c \Theta (W - S))$$
- Boundary (Internal) $$\beta(I) = I - (I \Theta S)$$
- Boundary (External) $$\beta(I) = (I \oplus S) - \mathrm{I}$$

RECAP

- Region Filling $$X_k = (X_{k-1} \oplus S) \bigcap I^c$$
- Connected Components $$X_k = (X_{k-1} \oplus S) \bigcap I$$
- Convex Hull $$X_k = (X_{k-1} \circledast S_i) \bigcup I$$
- Thinning $$I \otimes \{S\} = ((((I \circledast S_0) \circledast S_1) \circledast S_2) \ldots \circledast S_7)$$
- Thickening $$I \odot \{S\} = |(I^c \otimes S) \,\text{isolated pixel removal}\,|^c$$

SEGMENTATION

Segmentation

- Image segmentation is a technique to extract the attributes of the image
- Segmentation attempts to partition the pixels of an image into groups that strongly correlate with the objects in an image
- Mostly used in any automated computer vision application

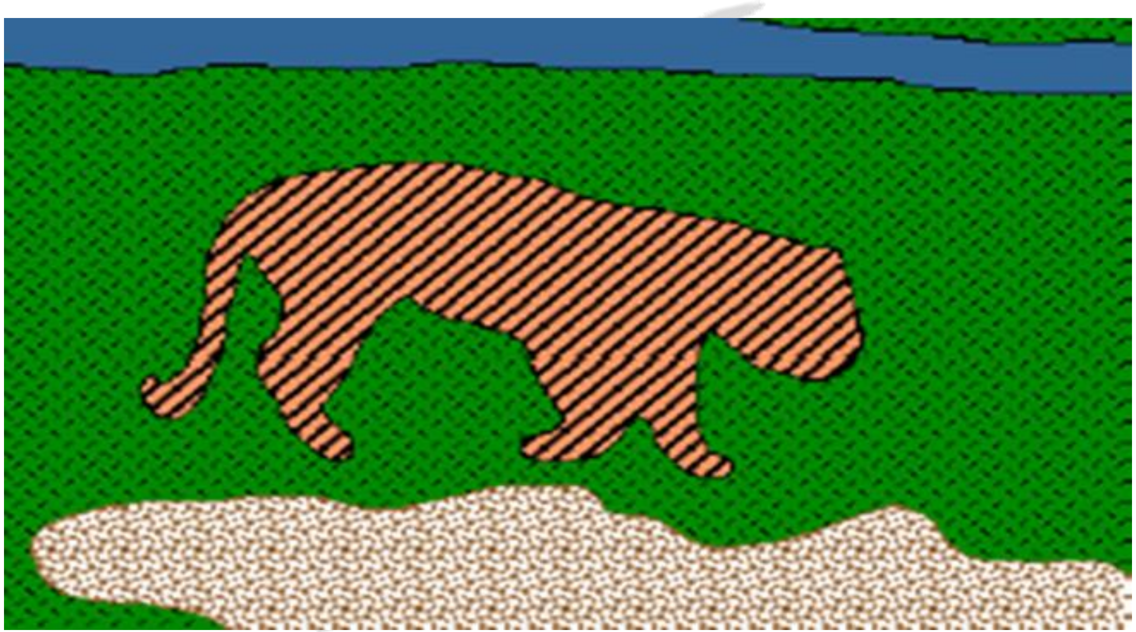
Image → Segmentation → Attributes

Segmentation

- Separate image into coherent objects

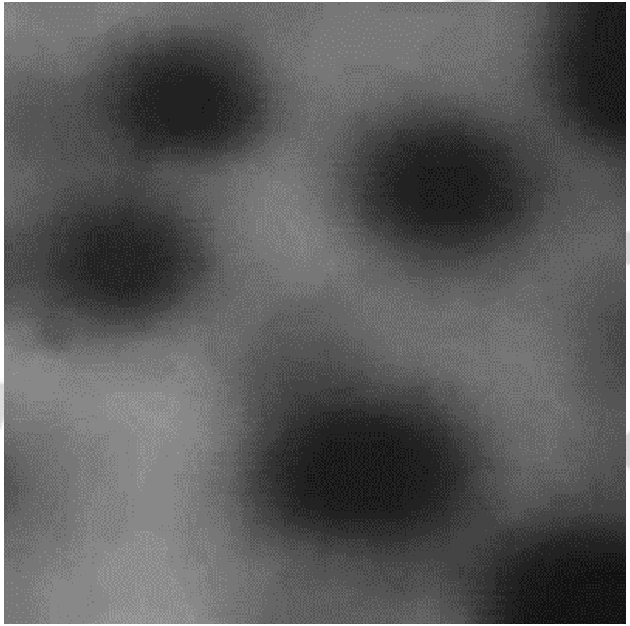
Segmentation

- Separate image into coherent objects



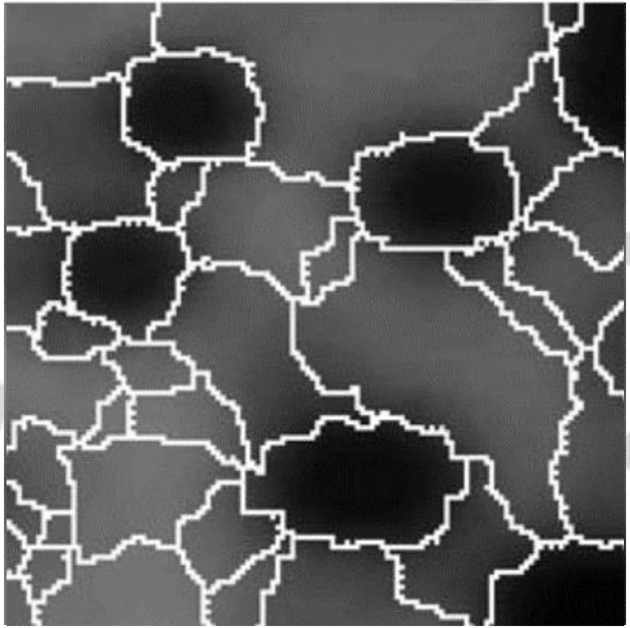
Segmentation

- Separate image into coherent objects



Segmentation

- Separate image into coherent objects

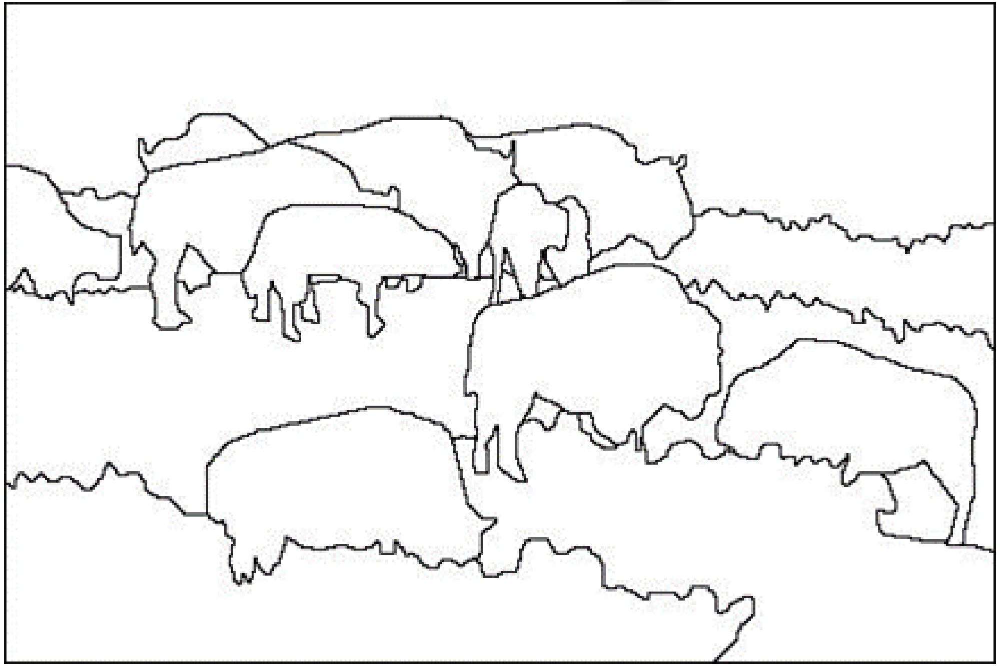


Segmentation

- Separate image into coherent objects

Segmentation

- Separate image into coherent objects



Segmentation

- The purpose of image segmentation is to partition an image into meaningful regions with respect to a particular application
- The segmentation is based on measurements taken from the image which includes intensity, color, texture, depth, etc

Segmentation (Approach)

- Segmentation algorithms generally are based on one of 2 basis properties of intensity values

- Discontinuity (abrupt changes)
  - to partition an image based on abrupt changes in intensity (such as edges)

- Similarity (homogeneity)
  - to partition an image into regions that are similar according to a set of predefined criteria
  - Two types: pixel based and region based

Segmentation (Fundamental)

- Let v be the spatial domain on which the image is defined
- Image segmentation divides v into n regions R_i (i = 1 to n), such that

$$\bigcup_{i=1}^{n} R_i = v$$

$$R_i \bigcap R_j = \phi$$

Segmentation (Basic Idea)

- All the image segmentation methods assume that
  - the intensity values are different in different regions
  - within each region of the corresponding object, the intensity values are similar

- For this, we need to apply threshold

Thresholding

- Gray level thresholding is the simplest segmentation process
- Correct thresholding leads to better segmentation
- Thresholding is computationally inexpensive and fast which can easily be applied in real time
- Thresholding as a transformation function

$$T = T[x, y, p(x, y), f(x, y)]$$

  - where
  - $f(x, y)$ is the gray level of the point (x, y)
  - $p(x, y)$ denotes some local property of the point (e.g. average gray level of a pixels centred at (x, y))

Thresholding (Types)

- Global Thresholding
  - If T is only a function of f(x, y), then the threshold is called global threshold
  - Apply the same threshold to the whole image

- Local Thresholding
  - If T is a function of both f(x, y) & p(x, y), then the threshold is called local
  - The threshold depends on local property

- Dynamic (Adaptive) Thresholding
  - The threshold depends on spatial coordinates (x, y)

Thresholding (Types)

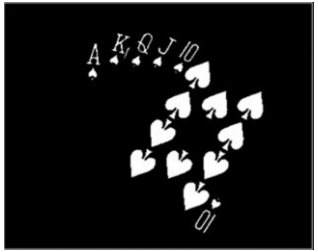
- Thresholding is the transformation of an input image f to an output (segmented) binary image g

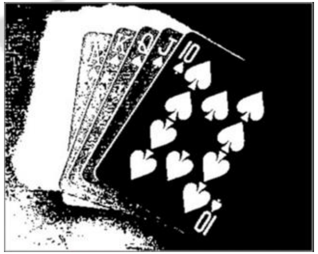
$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{otherwise} \end{cases}$$

  - where, T is threshold

- From a grayscale image, thresholding can be used to create binary images

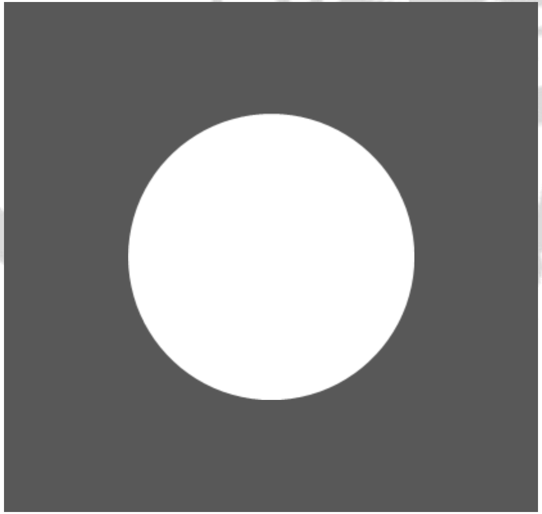
Thresholding





Thresholding (Eg)

- Disk = 255 and background = 127
- Segment the image into two regions – disk and background
- What will be the threshold values?



Global Thresholding

- Based on the histogram of an image
- Partition the image histogram using a single global threshold
- The success of this technique strongly depends on how well the histogram can be partitioned

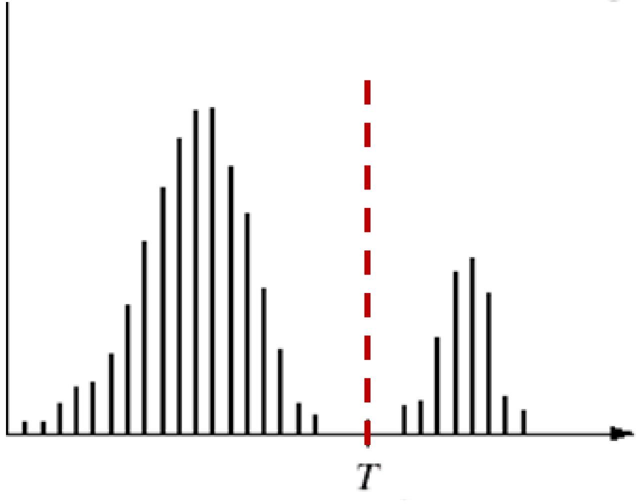
Global Thresholding (Algorithm)

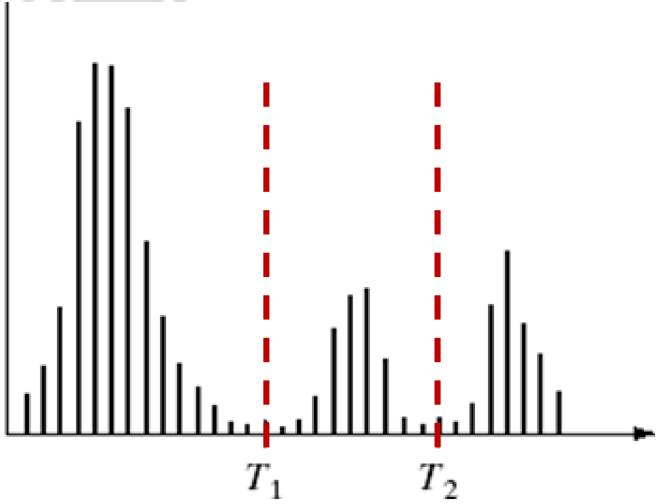
- Select an initial estimate for T (typically the average grey level in the image)
- Segment the image using T to produce two groups of pixels:
  - G1 consisting of pixels with grey levels >T
  - G2 consisting pixels with grey levels ≤ T
- Compute the average grey levels of pixels in G1 to give μ1 and G2 to give μ2
- Compute a new threshold value T = (μ1+ μ2)/2

- Repeat steps 2 – 4 until the difference in T in successive iterations is less than a predefined limit

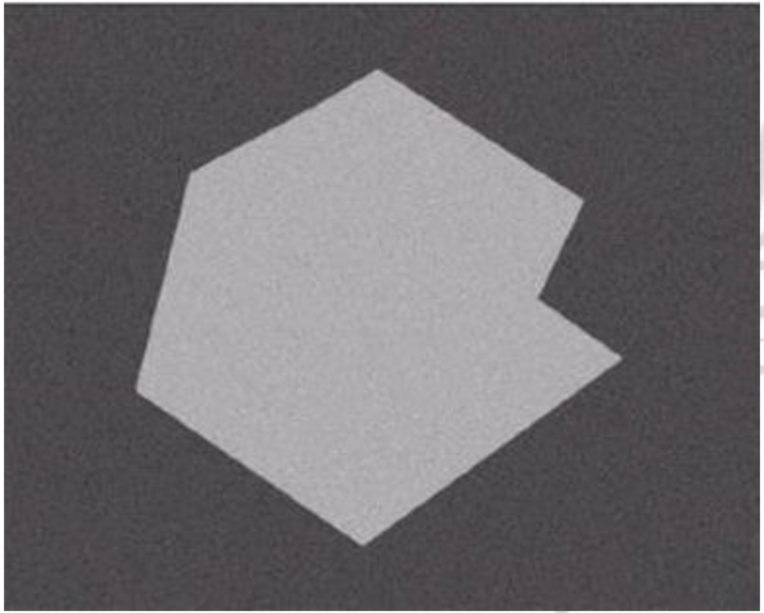
Global Thresholding (Problem)

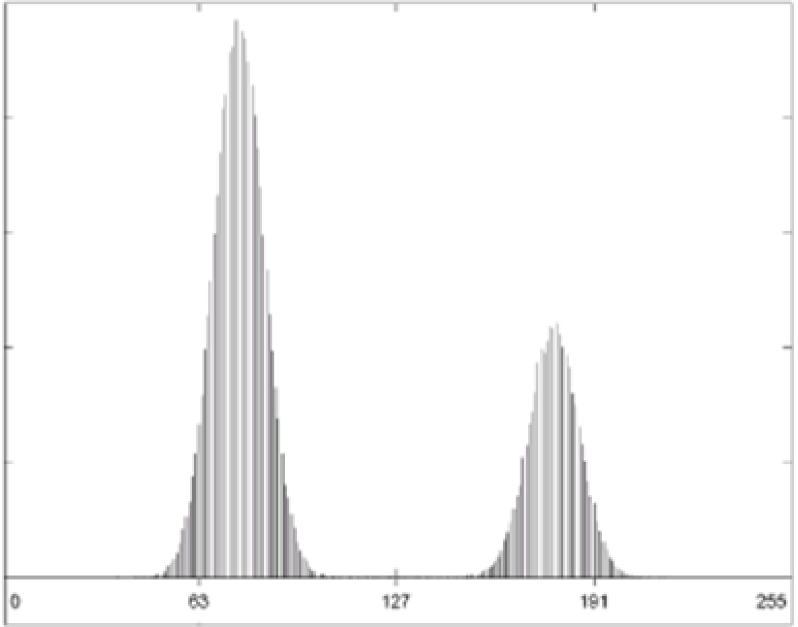
- Single value thresholding only works for bimodal histograms (two peaks)
- Images with other kinds of histograms need more than a one threshold



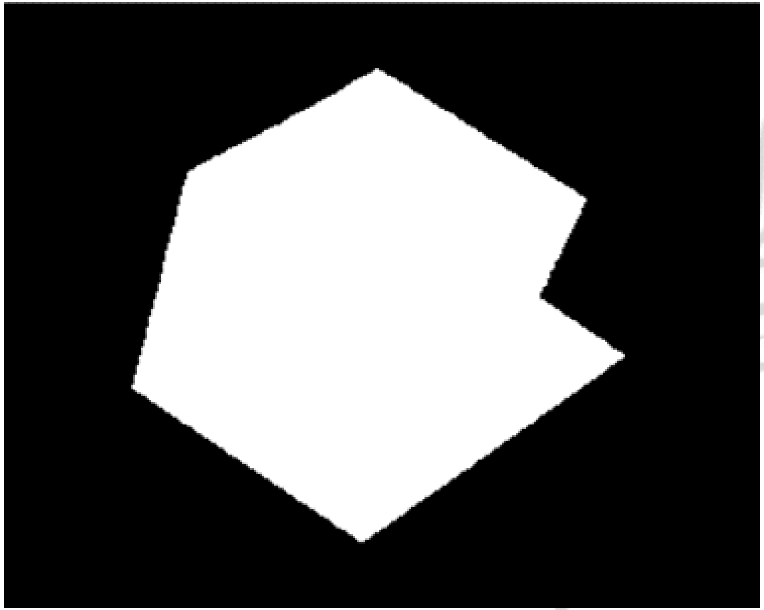


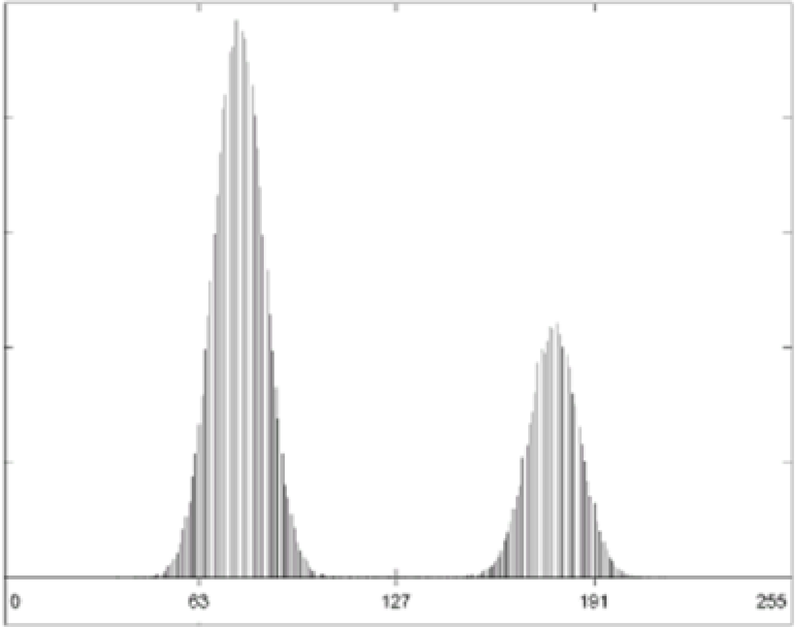
Global Thresholding (Problem)



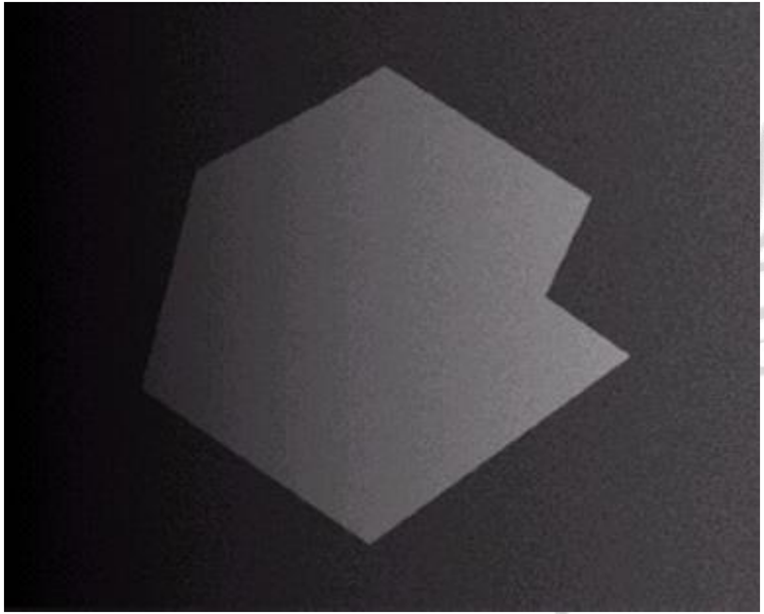


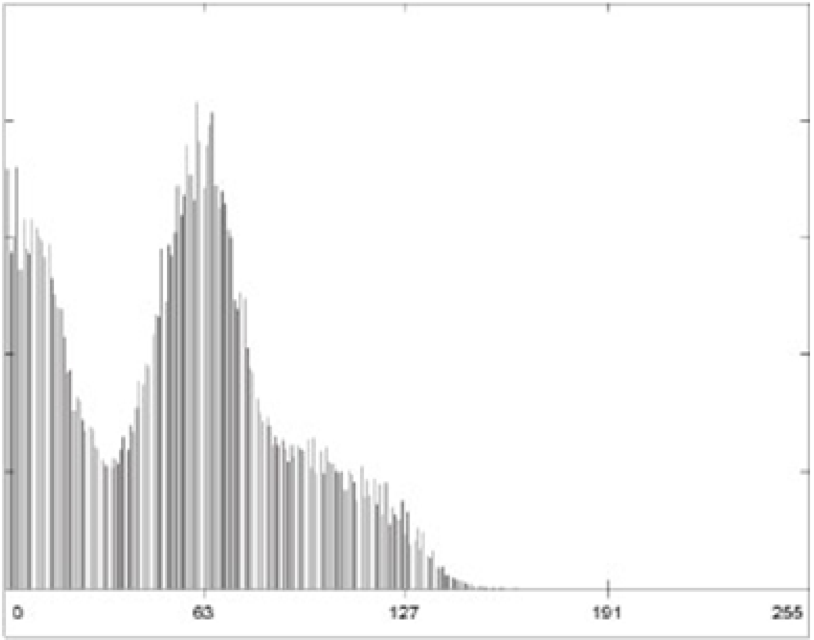
Global Thresholding (Problem)



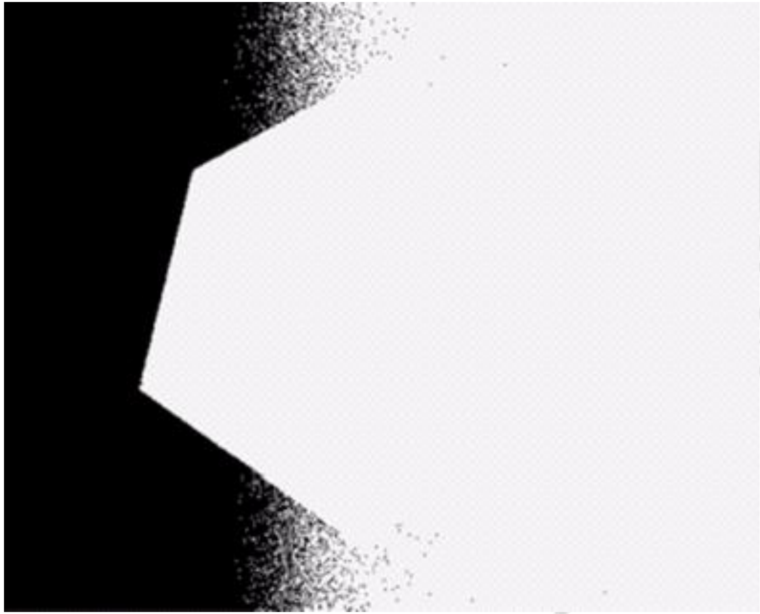


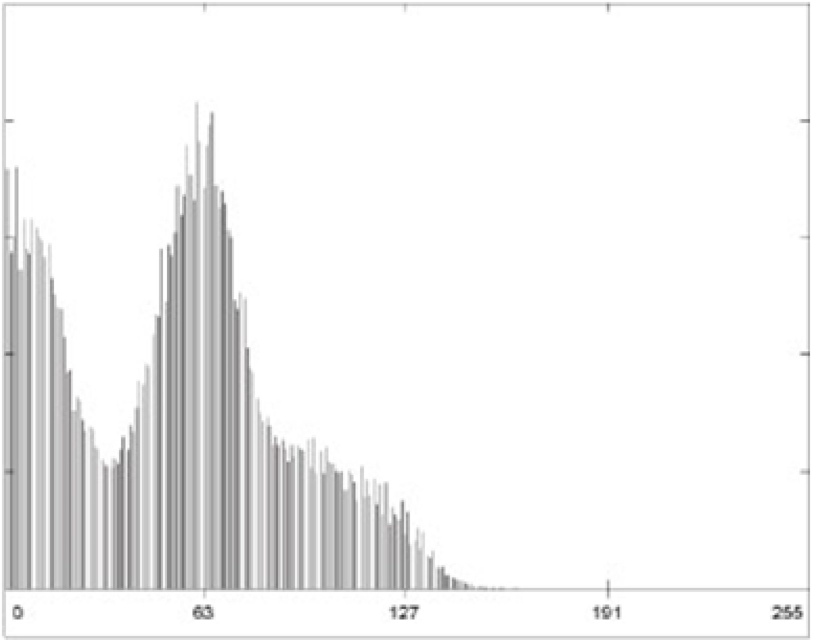
Global Thresholding (Problem)





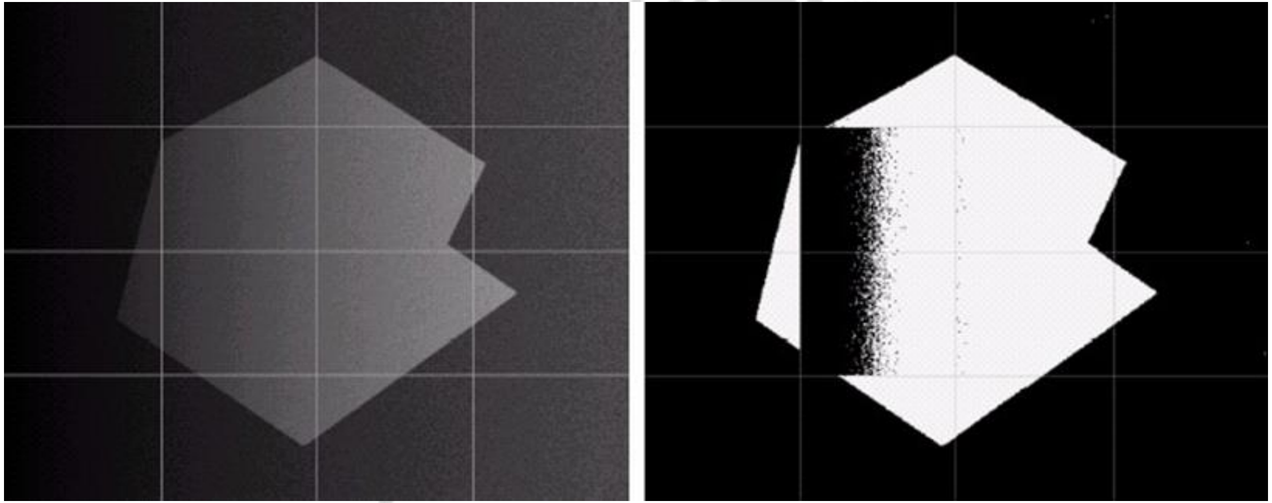
Global Thresholding (Problem)





Adaptive Thresholding

- The threshold for each pixel depends on its location within an image, this technique is said to adaptive



Region

- A group of connected pixels with similar properties
- For correct interpretation, image must be partitioned into regions that correspond to objects or parts of an object

Region based approach

- Let R represent the entire image region
- Segmentation R into n sub regions, R_1, R_2, ..., R_n, such that

$$\bigcup_{i=1}^{n} R_i = R$$

$$R_i \text{ } is\, a\, connected\ region, i = 1, 2, \ldots, n$$

$$R_i \cap R_j = \phi,\ for\ all\ i\ and\ j, i \neq j$$

Region based approach

- The fundamental drawback of histogram based region detection is that histograms provide no spatial information
- Region growing approaches exploit the important fact that pixels which are close together have similar gray values

Region growing

- Choose the seed pixels (1 for every segment)
- Check the neighboring pixels and add them to the region if they are similar to the seed
- Repeat previous step for each of the newly added pixels, stop if no more pixels can be added

1	1	9	9	9
1	1	9	9	9
5	1	9	9	9
5	5	5	3	9
3	3	3	3	3

Region growing

- Choose the seed pixels (1 for every segment)
- Check the neighboring pixels and add them to the region if they are similar to the seed
- Repeat previous step for each of the newly added pixels, stop if no more pixels can be added

1	1	9	9	9
1	1	9	9	9
5	1	9	9	9
5	5	5	3	9
3	3	3	3	3

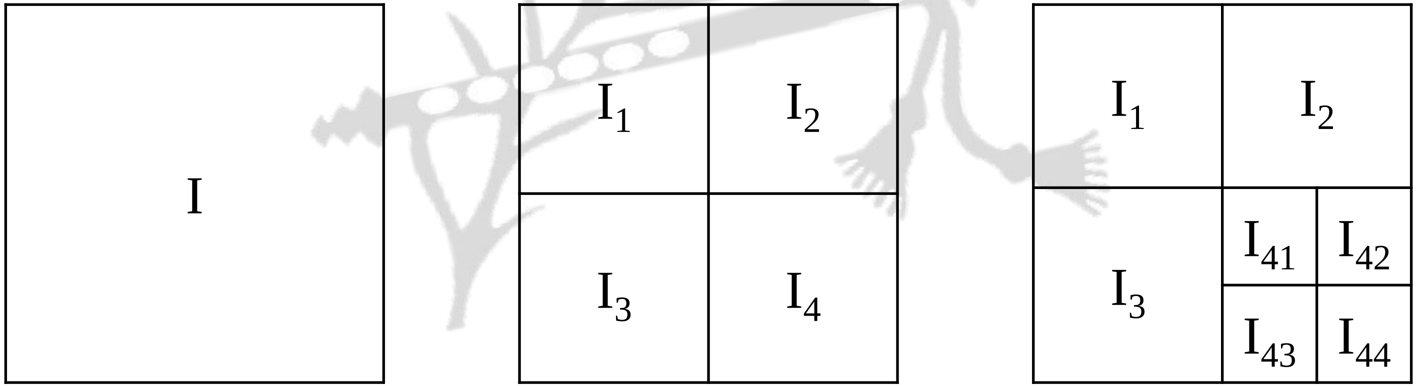
Region growing

- Choose the seed pixels (1 for every segment)
- Check the neighboring pixels and add them to the region if they are similar to the seed
- Repeat previous step for each of the newly added pixels, stop if no more pixels can be added

1	1	9	9	9
1	1	9	9	9
5	1	9	9	9
5	5	5	3	9
3	3	3	3	3

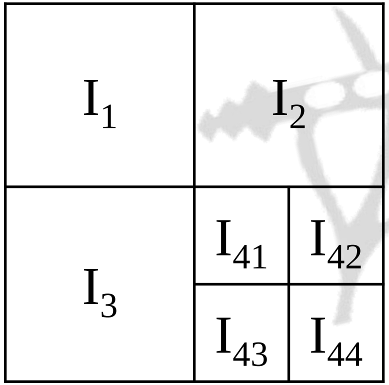
Region splitting

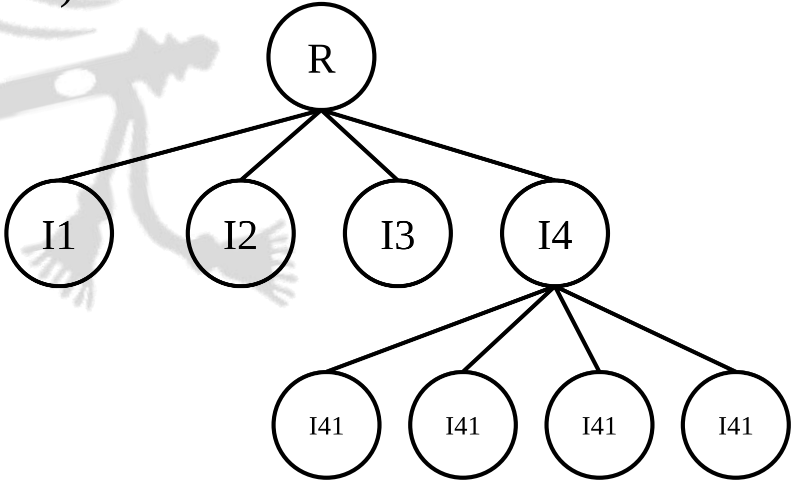
- Split starts from the assumption that the entire image is homogeneous
- If this is not true (by the homogeneity criterion), then split image into four sub images
- This splitting procedure is repeated recursively until we split the image into homogeneous regions



Region splitting

- If an image of dimensions N x N, that is in the powers of 2 ($N = 2^n$):
  - All regions produced by the splitting algorithm are squares having dimensions M x M, where M is a power of 2 as well (M = 2m, M<= n)
  - Since the procedure is recursive, it produces an image that can be described by a tree, in which each nodes have four child (except leaf node)
  - Such a tree is called a Quadtree





Region splitting

- Advantage
  - Created regions are adjacent and homogenous
- Disadvantage
  - Over splitting, since no merge is performed

- Improvement
  - Split and Merge

Region splitting and Merging

- After splitting
- Merging phase
  - If 2 adjacent regions are homogenous, they are merged
- Repeat merging step, until no further merging is possible

Segmentation (Discontinuity)

- There are basic three types of grey level discontinuities:
  - Points
  - Lines
  - Edges

- We typically find discontinuities using masks and correlation
- Discrete form of derivative is used

Segmentation (Discontinuity)

- The filter is expected to be isotropic
  - response of the filter is independent of the direction of discontinuities in an image
- The digital implementation of the 2-Dimensional Laplacian is obtained by summing 2 components

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

0	1	0
1	-4	1
0	1	0

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

Segmentation (Discontinuity)

0	1	0
1	-4	1
0	1	0

0	-1	0
-1	4	-1
0	-1	0

← 90° isotropic

45° isotropic →

1	1	1
1	-8	1
1	1	1

-1	-1	-1
-1	8	-1
-1	-1	-1

Segmentation (Discontinuity→Point)

- A point has been detected at the location on which the mask center if

$$|R| >= T$$

  - where
  - T is a nonnegative threshold
  - R is the sum of products of the coefficients with the gray levels contained in the region encompassed by the mark

Segmentation (Discontinuity → Point)

7	7	7	7	7	7	7
7	10	7	7	7	7	7
7	7	7	7	7	7	7
7	7	7	7	7	7	7
7	7	7	7	4	7	7
7	7	7	7	7	7	7
7	7	7	7	7	7	7

I

-1	-1	-1
-1	8	-1
-1	-1	-1

P

	24	-3	0	0	0	
	-3	-3	0	0	0	
	0	0	3	3	3	
	0	0	3	-24	3	
	0	0	3	3	3	

I º P

Segmentation (Discontinuity → Point)

7	7	7	7	7	7	7
7	10	7	7	7	7	7
7	7	7	7	7	7	7
7	7	7	7	7	7	7
7	7	7	7	4	7	7
7	7	7	7	7	7	7
7	7	7	7	7	7	7

I

-1	-1	-1
-1	8	-1
-1	-1	-1

P

	24	3	0	0	0	
	3	3	0	0	0	
	0	0	3	3	3	
	0	0	3	24	3	
	0	0	3	3	3	

$R_p = |I \circ P|$

Segmentation (Discontinuity → Line)

- The next level of complexity is to try to detect lines
- The masks below will extract lines that are one pixel thick and running in a particular direction

-1	-1	-1
2	2	2
-1	-1	-1

-1	-1	2
-1	2	-1
2	-1	-1

-1	2	-1
-1	2	-1
-1	2	-1

2	-1	-1
-1	2	-1
-1	-1	2

Segmentation (Discontinuity → Line)

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	3
1	1	1	1	1	1	3	1
1	1	1	1	1	3	1	1
3	3	3	3	3	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

I

-1	-1	-1
2	2	2
-1	-1	-1

H

	0	0	0	0	2	2	
	0	0	0	2	2	0	
	6	6	6	0	0	2	
	12	12	12	6	2	2	
	6	6	6	4	2	0	
	0	0	0	0	0	0	

|I º H|

Segmentation (Discontinuity → Line)

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	3
1	1	1	1	1	1	3	1
1	1	1	1	1	3	1	1
3	3	3	3	3	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

I

-1	-1	2
-1	2	-1
2	-1	-1

D_{45^o}

	0	0	0	0	2	4	
	0	0	0	2	4	12	
	0	0	0	0	12	4	
	0	0	0	6	4	2	
	0	0	0	4	2	0	
	0	0	0	0	0	0	

$|I \circ D_{45^o}|$

Segmentation (Discontinuity → Line)

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	3
1	1	1	1	1	1	3	1
1	1	1	1	1	3	1	1
3	3	3	3	3	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

I

-1	2	-1
-1	2	-1
-1	2	-1

V

	0	0	0	0	2	2	
	0	0	0	2	2	0	
	0	0	0	0	0	2	
	0	0	0	0	2	2	
	0	0	0	2	2	0	
	0	0	0	0	0	0	

$|I \circ V|$

Segmentation (Discontinuity → Line)

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	3
1	1	1	1	1	1	3	1
1	1	1	1	1	3	1	1
3	3	3	3	3	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

I

2	-1	-1
-1	2	-1
-1	-1	2

D_{-45^o}

	0	0	0	0	4	4	
	0	0	0	4	4	0	
	0	0	0	6	0	4	
	0	0	0	0	4	4	
	0	0	0	2	4	0	
	0	0	0	0	0	0	

$|I \circ D_{-45^o}|$

Segmentation (Discontinuity → Line)

	0	0	0	0	4	4	
	0	0	0	4	4	12	
	6	6	6	6	12	4	
	12	12	12	6	4	4	
	6	6	6	4	4	0	
	0	0	0	0	0	0	

$$L=\max\{|I^\circ H|,|I^\circ V|,|I^\circ D_{45^\circ}|,|I^\circ D_{-45^\circ}|\}$$

	0	0	0	0	2	2	
	0	0	0	2	2	0	
	6	6	6	0	0	2	
	12	12	12	6	2	2	
	6	6	6	4	2	0	
	0	0	0	0	0	0	

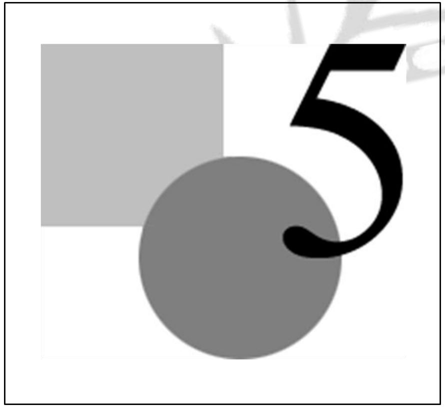
	0	0	0	0	2	4	
	0	0	0	2	4	12	
	0	0	0	0	12	4	
	0	0	0	6	4	2	
	0	0	0	4	2	0	
	0	0	0	0	0	0	

	0	0	0	0	2	2	
	0	0	0	2	2	0	
	0	0	0	0	0	2	
	0	0	0	0	2	2	
	0	0	0	2	2	0	
	0	0	0	0	0	0	

	0	0	0	0	4	4	
	0	0	0	4	4	0	
	0	0	0	6	0	4	
	0	0	0	0	4	4	
	0	0	0	2	4	0	
	0	0	0	0	0	0	

Segmentation (Discontinuity → Edge)

- Edges characterize boundaries
- Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next
- Edge detecting in an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image



Segmentation (Discontinuity → Edge)

- Difference between an edge and a line
  - An edge, for instance be the border between a block of blue color and a block of yellow

  - In contrast, a line can be a small number of pixels of different color
  - For a line, there may usually be one edge on each side of the line

Segmentation (Discontinuity → Edge)

- Edges are the most common approach for detecting meaningful discontinuities in gray level
- The process of edge detection can be broadly classified into two categories
  - Derivative approach
    - Edge pixels (or edgels) are detected by taking derivative followed by thresholding
    - Incorporate noise cleaning scheme
  - Pattern fitting approach
    - A series of edge approximating functions in the form of edge templates over neighbourhood are analysed
    - Parameters along with their properties corresponding to the best fitting function are determined
    - Based on this information, it is decided whether an edge is present or not
    - These are edge filters

Segmentation (Discontinuity → Edge)

- Edge linking
  - process takes an unordered set of edge pixels produced by an edge detector as i/p to form an ordered list of edgels

- Edge following
  - process takes the entire edge strength or gradient image as i/p & produces geometric primitives such as lines or curves

Segmentation (Discontinuity → Edge → Derivative)

- First Order Derivative/Gradient Methods
  - Robert operator
  - Sobel operator
  - Prewitt operator
- Second Order Derivative
  - Laplacian
  - Laplacian of Gaussian
  - Difference of Gaussian
- Optimal Edge Detection
  - Canny Edge Detection

Segmentation (Discontinuity → Edge → Derivative)

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

I

$$R=|I^\circ G_{45^\circ}|+|I^\circ G_{-45^\circ}|$$

Robert

1	0
0	-1

G_{45^o}

0	1
-1	0

G_{-45^o}

-1	-2	-1
0	0	0
1	2	1

S_H

-1	0	1
-2	0	2
-1	0	1

S_V

Sobel

$$S=|I^\circ S_H|+|I^\circ S_V|$$

-1	-1	-1
0	0	0
1	1	1

P_H

-1	0	1
-1	0	1
-1	0	1

P_V

Prewitt

$$P=|I^\circ P_H|+|I^\circ P_V|$$

Segmentation (Discontinuity → Edge → Derivative)

- Motivation
  - Detect sudden changes in image intensity
  - Gradient: sensitive to intensity changes
- Gradient

$$\nabla f = \left[\frac{\partial f}{\partial x} \frac{\partial f}{\partial y} \right]^T$$

Segmentation (Discontinuity → Edge → Derivative)

- The gradient of the image I at location (x, y) is the vector

$$\nabla I = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial I(\mathrm{x},\mathrm{y})}{\partial x} \\ \dfrac{\partial I(\mathrm{x},\mathrm{y})}{\partial y} \end{bmatrix}$$

- Magnitude

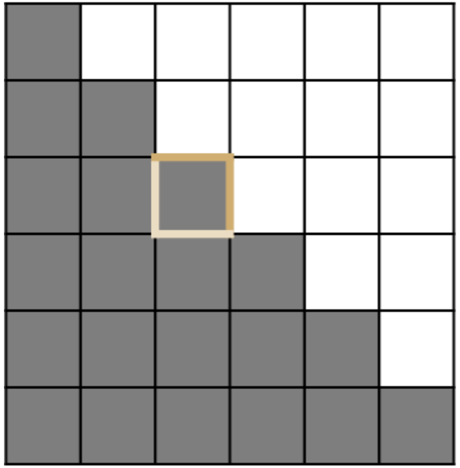
$$|\nabla I| = \sqrt{G_x^2 + G_y^2}$$

- Gradient

$$\theta(\mathrm{x},\mathrm{y}) = \tan^{-1}\left(\frac{\mathrm{G}_y}{\mathrm{G}_x}\right)$$

Segmentation (Discontinuity → Edge → Derivative)

- Eg:
  - Find the strength & the direction of the edge at the highlighted pixel
  - Pixels in gray are 0 and white are 1



Solution

- Derivative is computed by using a 3x3 neighbourhood
  - subtract the pixels in the top row from bottom row (x direction)
  - similarly obtain the derivative in the y direction

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

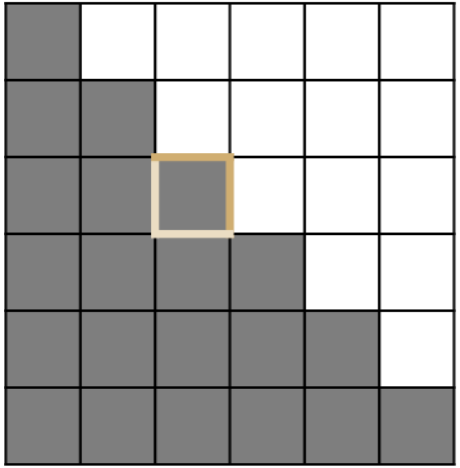
Segmentation (Discontinuity → Edge → Derivative)

- Eg:
  - Find the strength & the direction of the edge at the highlighted pixel
  - Pixels in gray are 0 and white are 1

$$\nabla I = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial I(\mathrm{x},\mathrm{y})}{\partial x} \\ \dfrac{\partial I(\mathrm{x},\mathrm{y})}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

$$|\nabla I| = \sqrt{G_x^2 + G_y^2} = 2\sqrt{2}$$

$$\theta(\mathrm{x},\mathrm{y}) = \tan^{-1}\left(\frac{\mathrm{G}_y}{\mathrm{G}_x}\right) = -1 = -45^\circ$$



Simple edge detection using gradient

- Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters
- Compute gradient magnitude at each pixel
- If magnitude at a pixel exceeds a threshold, report a possible edge point

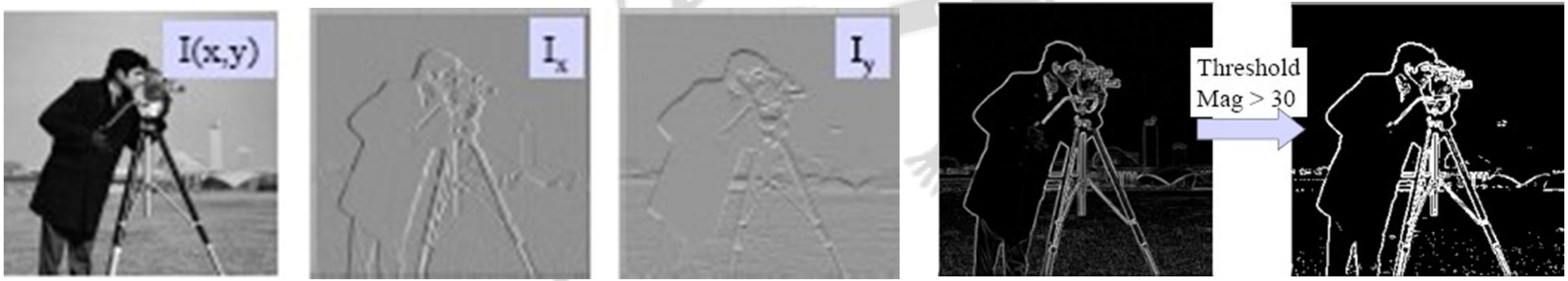


IMAGE REPRESENTATION AND DESCRIPTION

Image Representation and Description

- To represent and describe information embedded in an image in other forms that are more suitable to visualize and understand

- Benefits
  – Easier to understand
  – Require fewer memory, faster to be processed

- What kind of information we can use?
  – Boundary, shape
  – Region
  – Texture
  – Relation between regions

Image Representation and Description

- Segmentation techniques yield raw data in the form of pixels along a boundary or pixels contained in a region
- After segmentation, the image needs to be described and interpreted
  - Representation
    - An object may be represented by its external characteristics such as boundary
    - Its internal characteristics such as texture
  - Description
    - The object boundary may be described by its length, orientation, etc
    - The features that represent the image are used as descriptors
    - Descriptors should not be sensitive to variations like size change, translation, rotation, etc

Border

- An ordered list of points representing the boundary of an object
- Boundary as a sequence of connected points

Border (Algo)

- Step 1: Start scanning row-wise from top left corner of the image matrix
- Step 2: Mark the first object pixel obtained as the start pixel
- Step 3: Based on the previous pixel (p) and the current pixel (c), find the next pixel (n)
  - n is found by searching in the neighbourhood of c, clockwise starting from previous pixel p
- Step 4: Mark the current pixel as previous pixel & the next pixel as the current pixel
- Step 5: Go to step 3

Border (Eg)

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

p	c

Border (Eg)

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

p	c

Border (Eg)

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

p	c

Border (Eg)

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

p	c

Border (Eg)

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

p	c

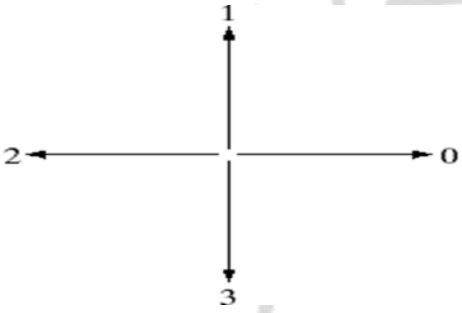
Border (Eg)

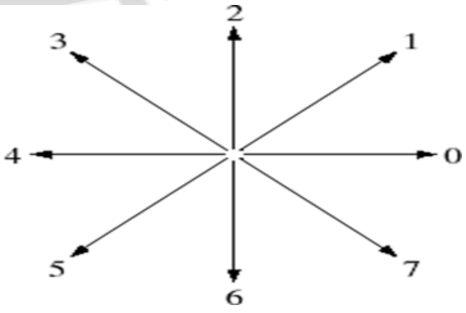
		1	1	1		
	1			1		
		1		1		
	1			1		
	1	1	1	1		

p	c

Chain Code

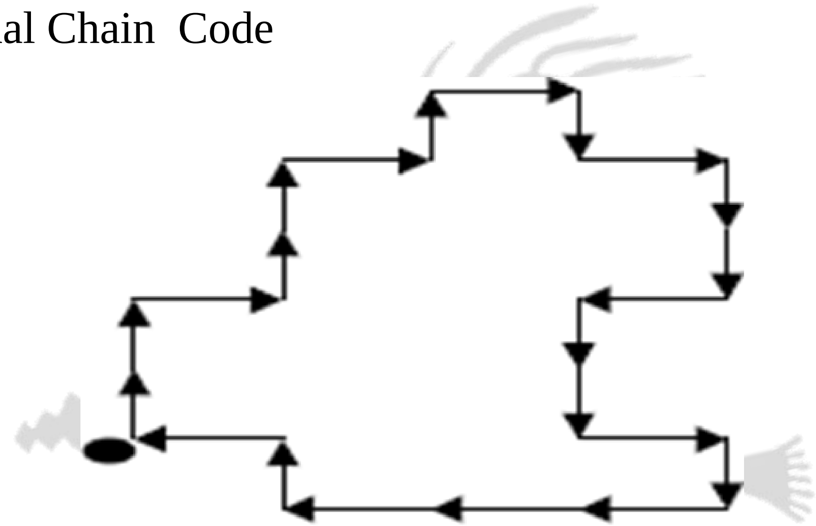
- The boundary is a good representation of an object shape and it requires less memory
- It represents an object boundary by a connected sequence of straight line of specified length and direction
- The direction of each segment is coded by using a numbering scheme as shown below





Chain Code

- Eg:
  - Find the 4 directional Chain Code

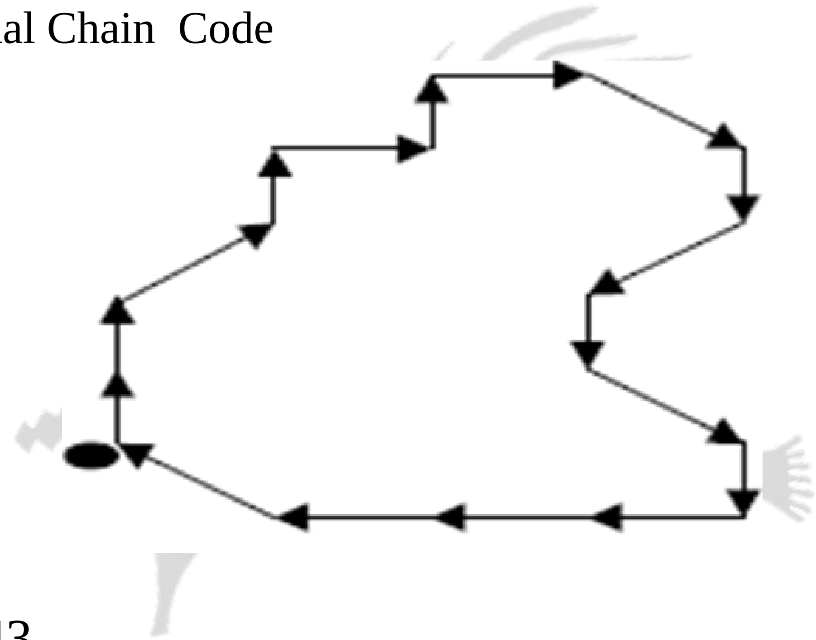


Solution

- 1101101030332330322212

Chain Code

- Eg:
  - Find the 8 directional Chain Code



Solution

- 2212020765676444 3

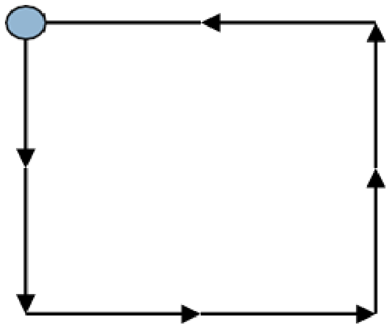
Chain Code (Problem)

- The resulting chain codes are quite long
- Any small change along the boundary due to noise causes changes in the code
- Dependent on the starting point
- Dependent on the orientation

Chain Code

- To use boundary representation in object recognition, we need to achieve invariance to starting point and orientation
- For this, we use
  - Normalized codes
    - To overcome the problem of starting point
  - Differential codes
    - To make rotational invariant

Chain Code → Normalization



Sort

33001122	00112233
30011223	01122330
00112233	11223300
01122330	12233001
11223300	22330011
12233001	23300112
22330011	33001122
23300112	30011223

First row gives the normalized chain code: 00112233

Chain Code → Differential

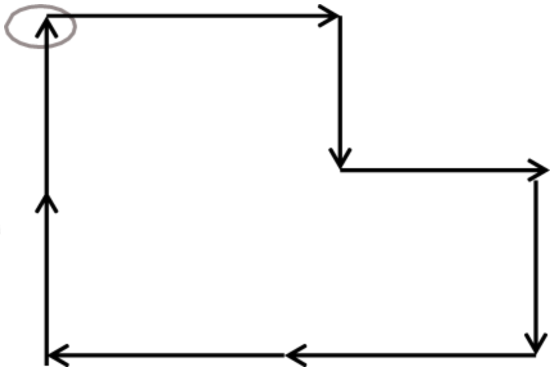
- The first difference of a chain code
  - counting the number of direction change (in counter-clockwise) between 2 adjacent elements of the code
- Eg
  - a chain code: 10103322
  - The first difference: 3133030
  - Treating a chain code as a circular sequence, we get the first difference: 31330303

Chain Code → Differential

- The shape number of a boundary obtained from a chain code is defined as the smallest magnitude of the circular first difference
- The order of the shape number is defined as the number of digits in its representation

Chain Code → Differential

- Eg:
  - Find the shape number & order of the given boundary



Solution

- 4-direction chain code: 0 3 0 3 2 2 1 1
- First difference: 3 1 3 3 0 3 0
- Circular first difference: 3 1 3 3 0 3 0 3
- Shape number: 0 3 0 3 3 1 3 3
- Order: 8