Machine Vision

Lecture Set – 03

Binary Image Processing

Huei-Yung Lin

Robot Vision Lab

Binary Image Processing

- An image contains a continuum of intensity values before it is quantized to obtain a digital image
  - Commonly used quantization levels: 256 (8 bits)
  - Also used: 32, 64, 128, 512, and 4096 (12 bits, usually for medical images)
  - More quantization levels: better representation, more storage
  - Binary images: 2 gray level quantization (1 bit)
- Why binary images?
  - Memory and computing power are limited in early days
  - Algorithms are well understood
  - Require less memory and fast execution time
  - Object and background separation with mask
  - Easy to analyze

Binary Image

- Binary images are particularly usefully for
  - Identifying objects with distinctive silhouettes
    - e. g. components on a conveyor in a manufacturing plant
  - Recognizing text and symbols
    - e.g. document processing or interpreting road signs
  - Determining the orientation of objects
- Disadvantages:
  - Need proper illumination control to obtain good contrast
  - Not possible to recover information using only two intensity levels for some applications

Binary Image Processing

- Representation of binary image
  - An image with size of $m \times n$ pixels
  - 1 (white) for object pixel and 0 (black) for background pixel

- Topics on binary image processing
  - Formation of binary images
  - Geometric properties
  - Topological properties
  - Object recognition in binary images

Image Segmentation

- Image segmentation
  - Partition an image into regions
  - Identify the subimage that represents objects
  - One of the most important problem in vision systems
  - It can be defined as a method to partition an image, $F[i, j]$, into subimages, called regions, $P_1, \ldots, P_k$, such that each subimage is an object candidate
- Region
  - A subset of an image

Binary Image Segmentation

- Segmentation is grouping pixels into regions such that
  - $\cup^{k}_{i=1} P_i$ = Entire image ($\{P_i\}$ is an exhaustive partitioning)
  - $P_i \cap P_j = \varnothing$, $i \neq j$ ($\{P_i\}$ is an exclusive partitioning)
  - Each region P_i satisfies a predicate; i.e., all points of the partition have *some* common property
  - Pixels belonging to adjacent regions, when taken jointly, do not satisfy the predicate
- The predicate can be "having uniform intensity", etc.
- A binary image is obtained using an appropriate segmentation of a gray image

Thresholding & Segmentation

- If the intensity values of an object are in some interval and that of background are outside that interval
  - Thresholding can be used to set one interval to 1 and the other interval to 0 to segment the object and background regions

- For binary vision, segmentation and thresholding are synonymous

Thresholding for Binary Images

- Thresholding is a method to convert a gray scale image into a binary image so that objects of interest are separated from the background
- Sufficient contrast on objects and background is necessary to do the thresholding (why?)

Thresholding for Binary Images

- If we use a threshold T for the original image $F[i, j]$ to obtain a binary image $B[i, j] = F_T[i, j]$, then

$$F_T[i,j] = \begin{cases} 1 & \text{if } F[i,j] \le T \\ 0 & \text{otherwise} \end{cases}$$

- If the object intensity values are in the range $[T_1, T_2]$, then we can use the following equation for thresholding

$$F_T[i,j] = \begin{cases} 1 & \text{if } \mathrm{T}_1 \le F[i,j] \le T_2 \\ 0 & \text{otherwise} \end{cases}$$

Thresholding for Binary Images

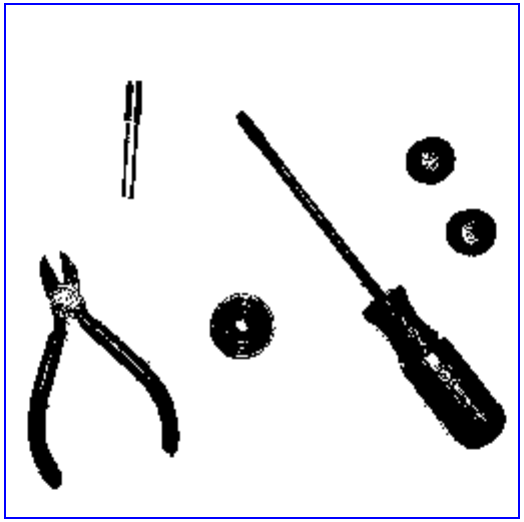
- A general thresholding scheme in which the intensity levels for an object may come from several disjoint intervals are represented as

$$F_T[i,j] = \begin{cases} 1 & \text{if } F[i,j] \in Z \\ 0 & \text{otherwise} \end{cases}$$

  where Z is a set of intensity values for object components
- If there are two, or more threshold values, the threshold is usually selected on the basis of experience with the application domain
- Automatic thresholding of images is often the first step in the analysis of images in machine vision systems

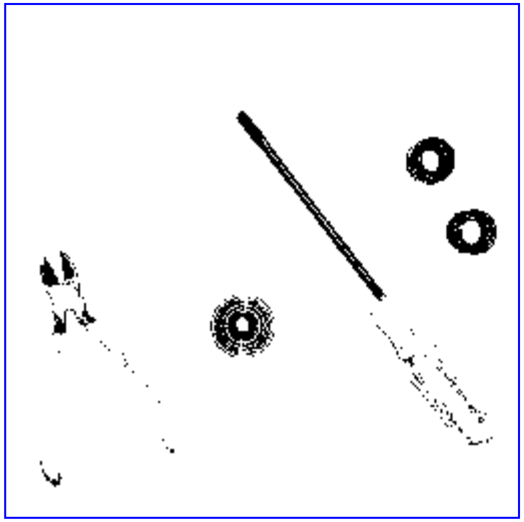
Example

Original image



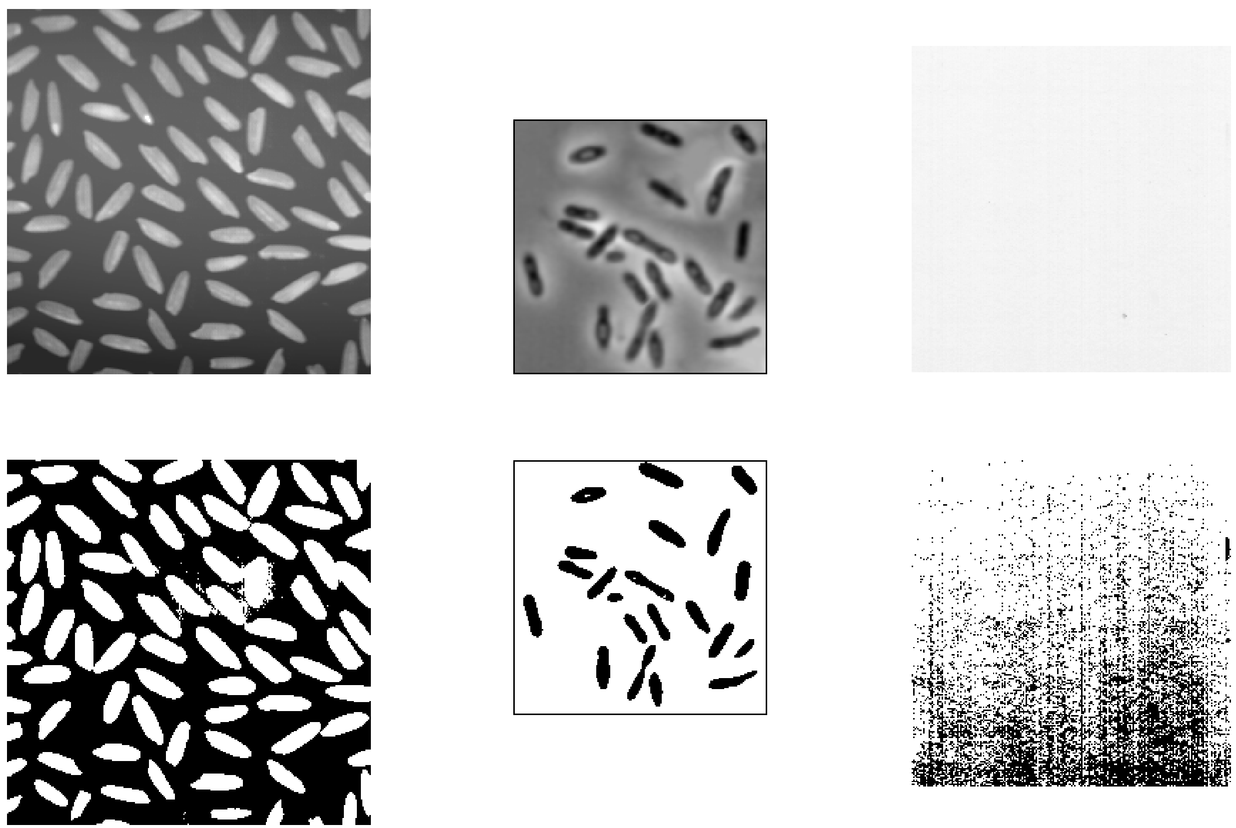
Threshold segmentation

Threshold too low



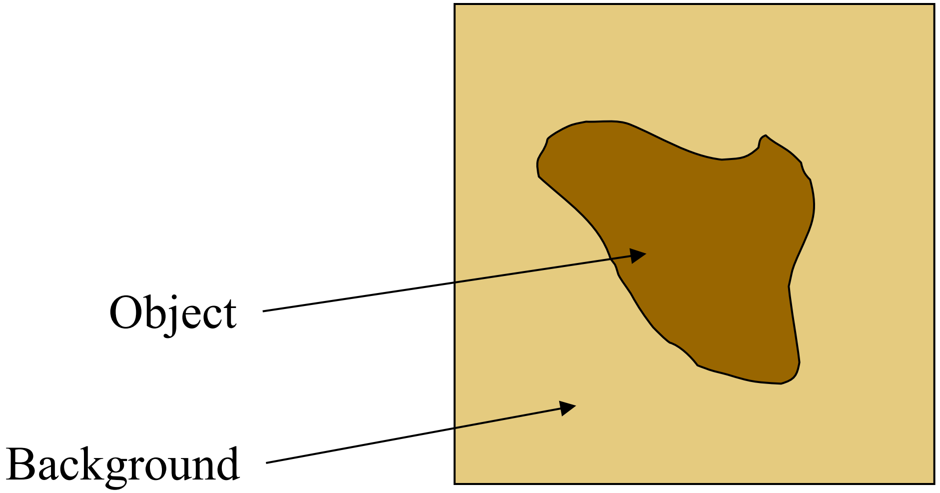
Threshold too high

Example – Applications



Geometric Properties

- The geometric properties of an object in a binary image include
  - Size
  - Position
  - Perimeter
  - Orientation



- They will be defined through moments of the object

Moments of Objects

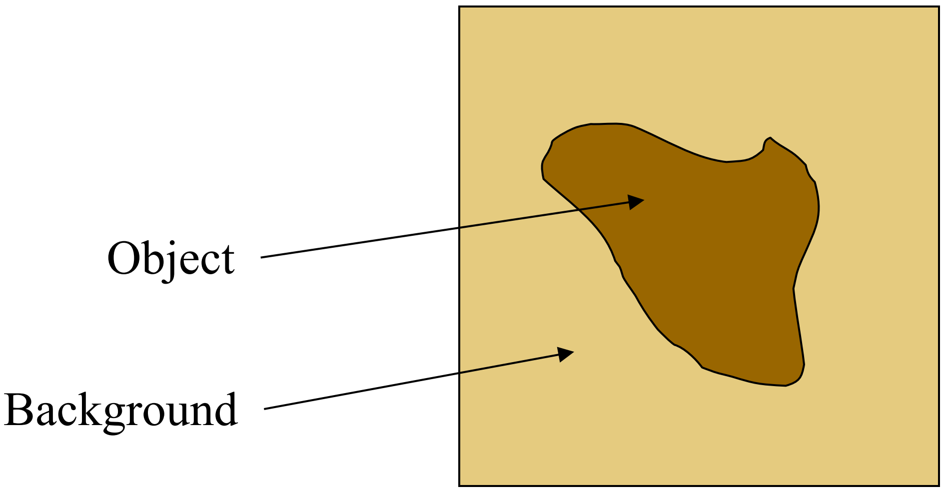
- Lots of useful information about a binary object can be gained from the moments of the object
- Define the binary image B of an object to be

$$B(i,j) = \begin{cases} 1 & \text{for points on the object} \\ 0 & \text{else} \end{cases}$$

Size of an Object

- The size of an object (area) is given by the 0^{th} *moment*

$$A = \sum_{i=1}^{n} \sum_{j=1}^{m} B(i, j)$$

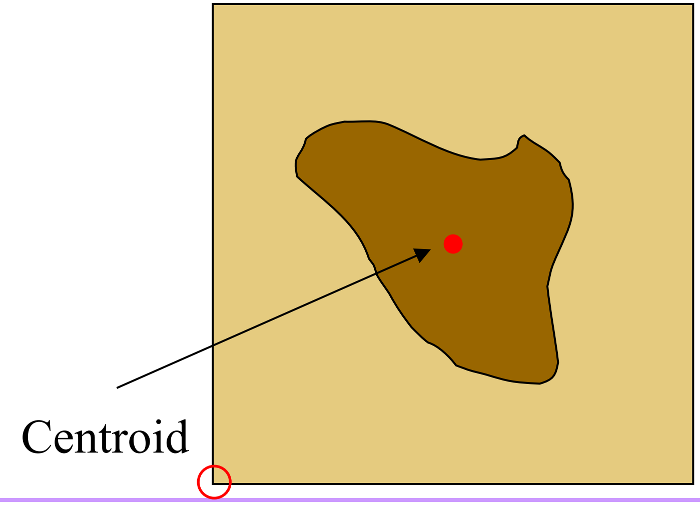


Position of an Object

- The position of an object can be represented by
  - Enclosing rectangle – bounding box
  - Center of area – relatively insensitive to noise
- For a binary image, the center of area is the same as the center of mass (intensity value)
- The center of mass is given by the 1st *moments*

$$\bar{x} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} jB(i,j)}{\sum_{i=1}^{n}\sum_{j=1}^{m} B(i,j)}, \quad \bar{y} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} iB(i,j)}{\sum_{i=1}^{n}\sum_{j=1}^{m} B(i,j)}$$

$$\int xf(x)dx = \bar{x}\int f(x)dx$$



Orientation of an Object

- The orientation of an object is not necessary unique (such as circle)
- The orientation of an object is defined as the axis of minimum inertia
- This is the least 2nd moment, the orientation of which is

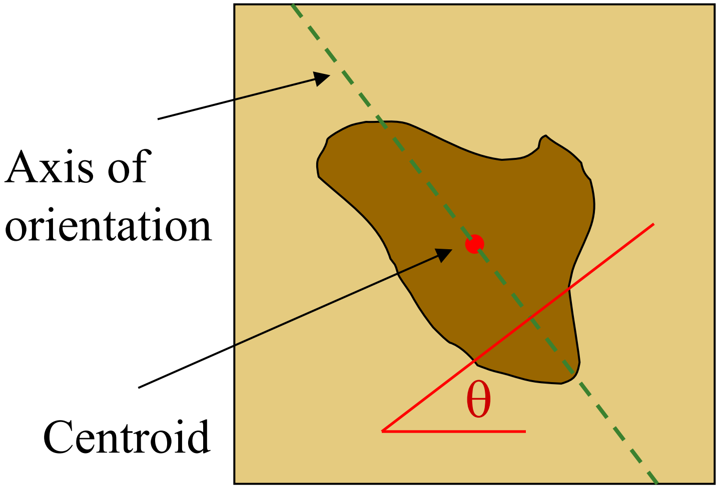
$$\theta = \frac{1}{2}\tan^{-1}\frac{M_{xy}}{M_{xx} - M_{yy}}$$

where the 2nd moments are

$$M_{xx} = \sum_x \sum_y (x-\bar{x})^2 B(x,y)$$

$$M_{xy} = \sum_x \sum_y 2(x-\bar{x})(y-\bar{y}) B(x,y)$$

$$M_{yy} = \sum_x \sum_y (y-\bar{y})^2 B(x,y)$$



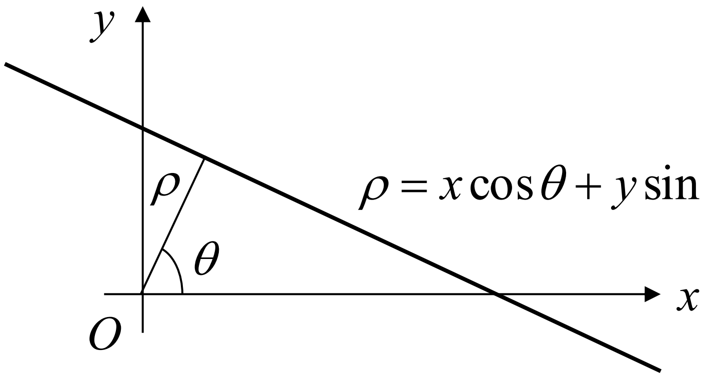
Derivation of Orientation

- The axis of second moment
  - The line for which the sum of the squared distances between object points and the line is minimum
  - Compute the least-squares fit of a line to the object points
  - Let r_{ij} be the distances of all object points from the line, then

$$\chi^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij}^2 B[i, j]$$

- Using polar coordinate system, let $\rho = x \cos \theta + y \sin \theta$, then for each objects to the line

$$r_{ij}^2 = (x_{ij} \cos\theta + y_{ij} \sin\theta - \rho)^2$$



Derivation of Orientation

- Thus, $\chi^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij}\cos\theta + y_{ij}\sin\theta - \rho)^2 B[i,j]$
- Take the derivative w.r.t. ρ, set to zero, and solve for ρ:*

$$\frac{\partial \chi^2}{\partial \rho} = -2\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij}\cos\theta + y_{ij}\sin\theta - \rho)B_{ij}$$

$$\Rightarrow \quad \rho = \bar{x}\cos\theta + \bar{y}\sin\theta$$

- Let $x' = x - \bar{x}$ and $y' = y - \bar{y}$, then $\chi^2 = a\cos^2\theta + b\sin\theta\cos\theta + c\sin^2\theta$ with*

$$a = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij}')^2 B[i,j],\ b = 2\sum_{i=1}^{n}\sum_{j=1}^{m}x_{ij}' y_{ij}' B[i,j],\ c = \sum_{i=1}^{n}\sum_{j=1}^{m}(y_{ij}')^2 B[i,j]$$

- Thus, $\chi^2 = \frac{1}{2}(a+c) + \frac{1}{2}(a-c)\cos 2\theta + \frac{1}{2}b\sin 2\theta$
- Take the derivate w.r.t. θ, set to zero, and solve for θ:

$$\tan 2\theta = \frac{b}{a-c}$$

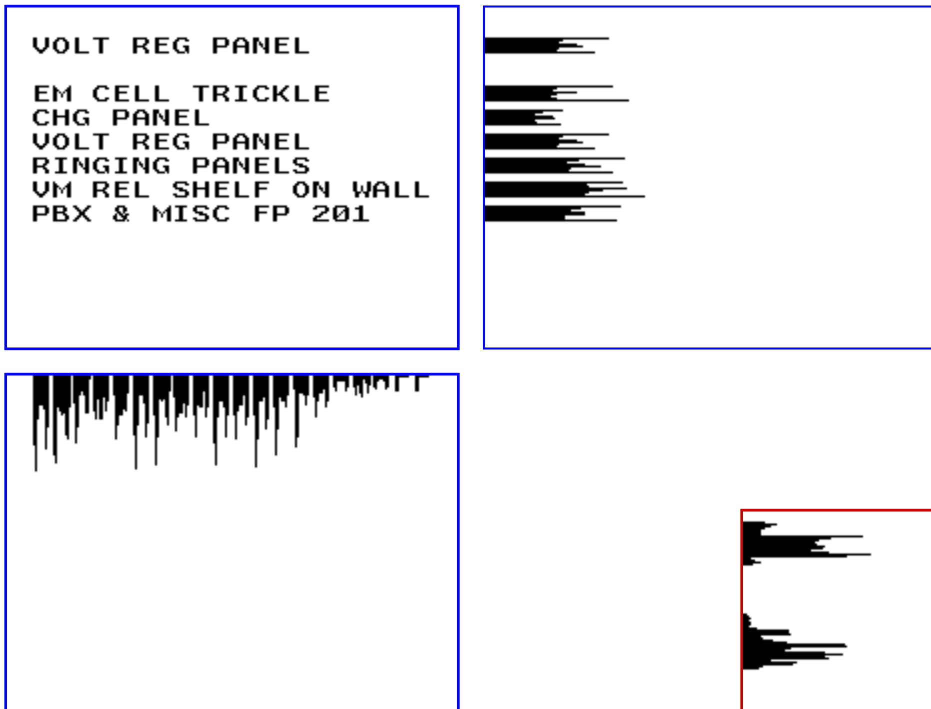
Projections

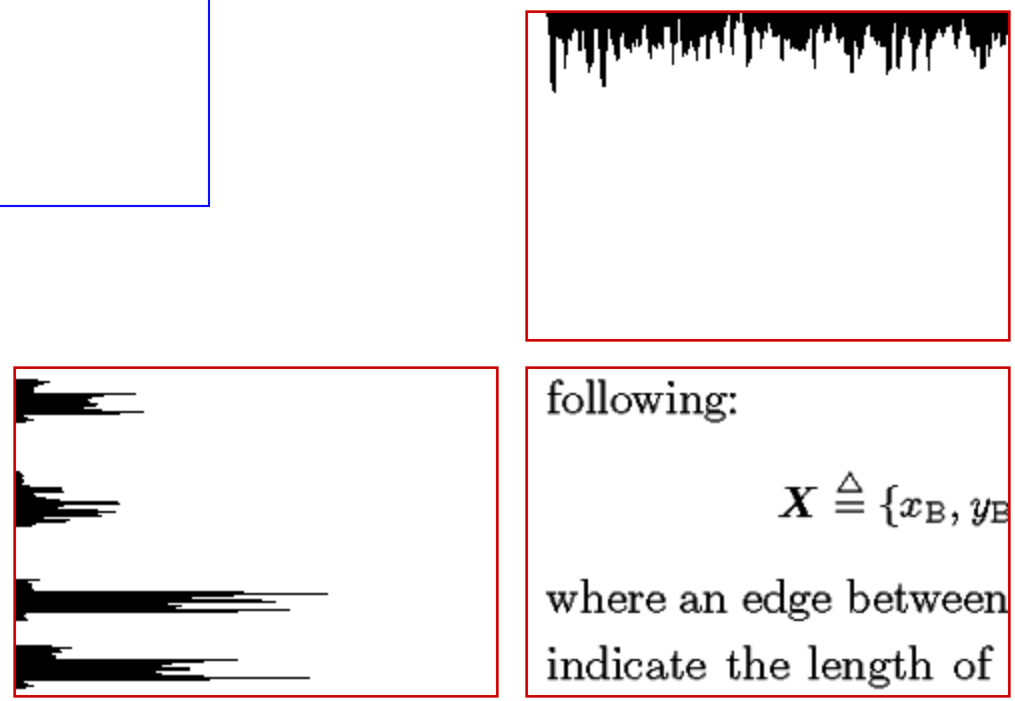
- Projection is a compact representation of binary images
  - Projections are not unique
  - More than one image may have the same projection
- Projection of a binary image onto a line
  - Partition the line into bins and finding the number of 1 pixels that are on lines perpendicular to each bin
- The projection $H[i]$ along the rows and the projection $V[j]$ along the columns of a binary image are given by

$$H[i] = \sum_{j=1}^{m} I(i, j), \qquad V[j] = \sum_{i=1}^{n} I(i, j)$$

Projections

- Pictures of projections along both directions (Fig. 2.4, 2.5, 2.6)





Projections and Position

- A general projection onto any line may be defined
- The first moments of an image equal to the first moments of its projection (Why?)
- Calculation of the position of an object requires only the first moment
- The position can be computed from the horizontal and vertical projections

$$A = \sum_{j=1}^{m} V[j] = \sum_{i=1}^{n} H[i], \quad \bar{x} = \frac{\sum_{j=1}^{m} jV[j]}{A}, \quad \bar{y} = \frac{\sum_{i=1}^{m} iH[j]}{A}$$

Run-Length Encoding

- Used for image transmission
- Use numbers indicating the lengths of the runs of 1 pixels in the image
- Two common approaches:
  - The start position and lengths of runs of 1s for each row are used
  - Use only the length of runs, starting with the length of the 1 run

Run-Length Encoding

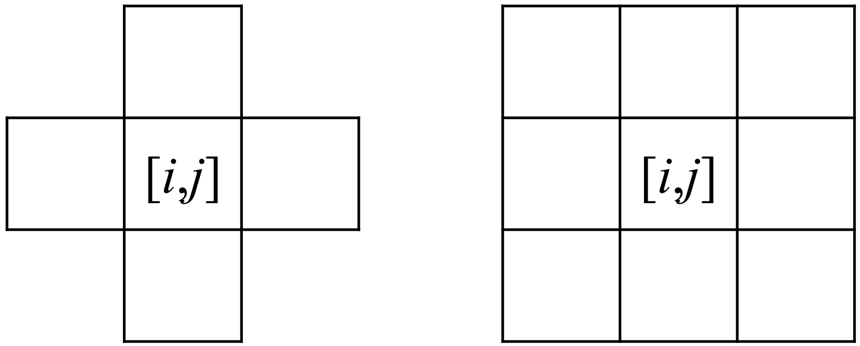
- Binary image:

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

- Start and length of 1 runs:
  - (1,3) (7,2) (12,4) (17,2) (20,3)
  - (5,13) (19,4)
  - (1,3) (17,6)
- Length of 1 and 0 runs:
  - 3,3,2,3,4,1,2,1,3
  - 0,4,13,1,4
  - 3,13,6

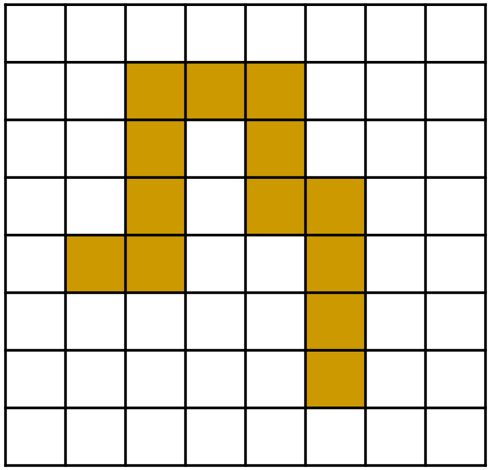
Neighbors

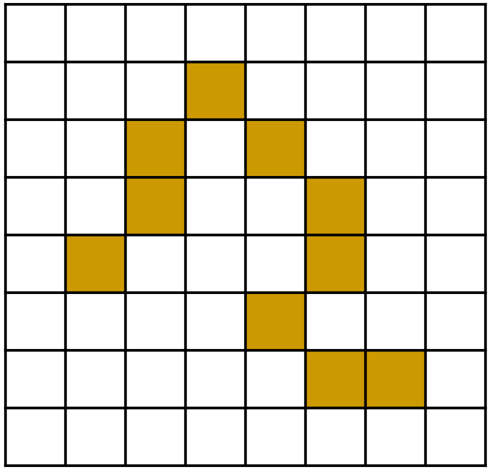
- In a digital image, a pixel has a common boundary with four pixels and shares a corner with four additional pixels
  - Two pixels are *4-neighbors* if they share a common boundary
  - Two pixels are *8-neighbors* if they share at least one corner
  - The pixel at location $[i, j]$ has 4-neighbors $[i+1, j]$, $[i-1, j]$, $[i, j+1]$, $[i, j-1]$
  - The 8-neighbors of the pixel include the 4-neighbors plus $[i+1, j+1]$, $[i+1, j-1]$, $[i-1, j+1]$, $[i+1, j-1]$
- A pixel is said to be *4-connected* to its 4-neighbors and *8-connected* to its 8-neighbors



Path

- A *path* from the pixel at $[i_0, j_0]$ to the pixel at $[i_n, j_n]$ is a sequence of pixel indices $[i_0, j_0], [i_1, j_1], \ldots, [i_n, j_n]$ such that the pixel at $[i_k, j_k]$ is a neighbor of the pixel at $[i_{k+1}, j_{k+1}]$ for all k with $0 \leq k \leq n - 1$
- If the neighbor relation uses 4-connection, the path is a *4-path*; If the neighbor relation uses 8-connection, the path is a *8-path*





Perimeter

- The length of the perimeter P of a region is a global property
- A definition of the perimeter of a region without holes is the set of its interior border pixels
- Perimeter:

$$P_4 = \{(r,c) \in R \mid N_8\,(r,c) - R \neq \emptyset\}$$
$$P_8 = \{(r,c) \in R \mid N_4\,(r,c) - R \neq \emptyset\}$$

(Check with a simple image.)

- To compute length $|P|$ of perimeter P, the pixels in P must be ordered in a sequence $P = \langle (r_0,c_0),\ldots,(r_{k-1},c_{k-1})\rangle$
- Perimeter length:

$$\begin{aligned}|P| = \; & |\{k \mid (r_{k+1},c_{k+1}) \in N_4(r_k,c_k)\}| \\ & + 2^{1/2}\, |\{k \mid (r_{k+1},c_{k+1}) \in (N_8(r_k,c_k) - N_4(r_k,c_k))\}|\end{aligned}$$

Circularity

- Circularity (or compactness) can be defined as the length of the perimeter squared divided by the area

$$C_1 = |P|^2 / A$$

- In this definition, it has the smallest value for digital octagons or diamonds depending on whether 4- or 8-neighbor used
- *Smaller is better!*

Circularity

- Circularity can also defined as $C_2 = \mu_R/\sigma_R$ where μ_R and σ_R are the mean and standard deviation of the distance from the centroid of the shape
  - Mean radial distance:

$$\mu_R = \frac{1}{K}\sum_{k=0}^{K-1} \| (r_k, c_k) - (\bar{r}, \bar{c}) \|$$

  - Standard deviation of radial distance:

$$\sigma_R = \left(\frac{1}{K}\sum_{k=0}^{K-1} [\| (r_k, c_k) - (\bar{r}, \bar{c}) \| - \mu_R]^2 \right)^{\frac{1}{2}}$$

  - *Larger is better!*

Example

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

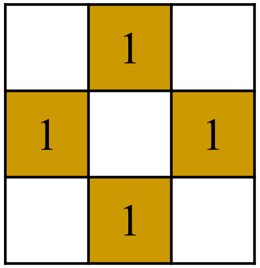
region number	region area	row of center	column of center	perimeter length	circularity 1	circularity 2	radius mean	radius variance
1	44	6	11.5	21.2	10.2	15.4	3.33	.05
2	48	9	1.5	28	16.3	2.5	3.80	2.28
3	9	13	7	8	7.1	5.8	1.2	0.04

Connectivity

- A pixel $p \in S$ is said to be *connected* to $q \in S$ if there is a path from p to q consisting entirely of pixels of S
- Connectivity is an *equivalence relation*
- For any three pixel p, q and r in S, we have the following properties:
  - *Reflectivity*: pixel p is connected to p
  - *Commutativity*: if p is connected to q, then q is connected to p
  - *Transitivity*: if p is connected to q and q is connected to r, then p is connected to r
- A set of pixels in which each pixel is connected to all other pixels is called a *connected component*

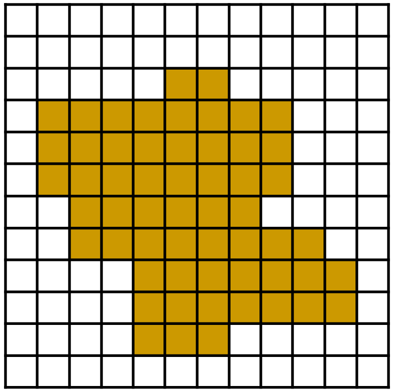
Foreground, Background

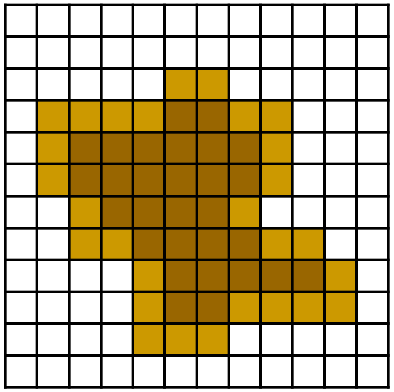
- Foreground
  - The set of all 1 pixels in an image is called the foreground and is denoted by S
- Background
  - The set of all connected components of $\hat{S}$ (the complement of S) *that have points on the border of an image* is called the background
- All other components of $\hat{S}$ are called holes
- Different connectedness should be used for object and background
  - If 8-connectedness is used for S, the 4-connectedness should be used for $\hat{S}$
  - (Why? Check page 43 in the textbook.)



Boundary, Interior, Surrounds

- Boundary
  - The *boundary* of S is the set of pixel of S that have 4-neighbors in $\hat{S}$
  - The boundary is usually denoted by S'
- Interior
  - The *interior* is the set of pixels of S that are not in its boundary
  - The interior of S is $(S - S')$
- Surrounds
  - Region T *surrounds* region S (or S is inside T), if any 4-path from any point of S to the border of the picture must intersect T



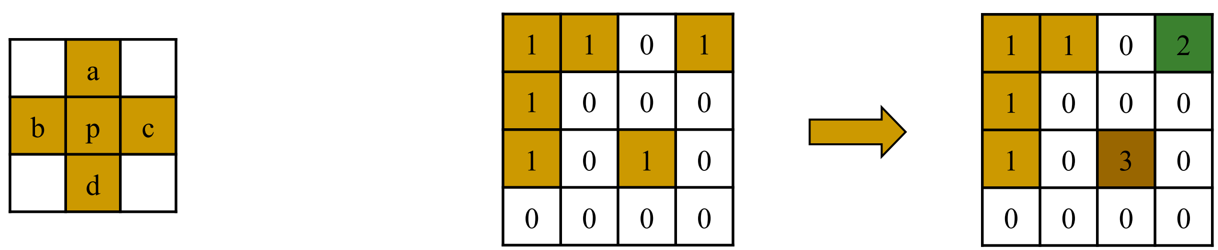


Component Labeling

- Uniquely label each cluster of positive connected components
- Zero-elements are considered part of the background and remain zero
- Two algorithms:
  - Recursive algorithm (very inefficient, used only on parallel machines)
  - Sequential algorithm
- Recursive Algorithm:
  - Scan the image to find an unlabeled 1 pixel and assign it a new label L
  - Recursively assign a label L to all its 1 neighbors
  - Stop if there are no more unlabeled 1 pixels
  - Go to step 1

Sequential Algorithm (Labeling)

- Labeling 4-connected components via a raster scan (row by row starting top left)
  - if **p** = 0 ignore
  - else if **a** and **b** not labeled, increment label and label **p**
  - else if only one of **a** or **b** labeled then copy label to **p**
  - else if **a** and **b** labeled
    - if **a** and **b** labeled the same then copy label to **p**
    - else copy either label to **p** and record the equivalence of the labels



Counting Objects in an Image

- For counting foreground objects
  - The *external corner patterns* (E) are 2×2 masks that have three 0's and one 1-pixel
  - The *internal corner patterns* (I) are 2×2 masks that have three 1's and one 0-pixel

0	0
0	1

0	0
1	0

0	1
0	0

1	0
0	0

0	1
1	1

1	1
0	1

1	0
1	1

1	1
1	0

- Algorithm count_objects:
  - E = 0, I = 0
  - For L = 0 to row_number
    - For P = 0 to column_number
      - If external_match(L,P) then E = E + 1
      - If internal_match(L,P) then I = I + 1
  - Return (E-I)/4

Counting Foreground Objects

	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	1

0	0
1	0

0	1
0	0

1	0
0	0

0	1
1	1

1	1
0	1

1	0
1	1

1	1
1	0

e			e					e		e		e		e	
								e	i						
e			e	e		e				i	e	e	i		
			e	i		i	e				i		i		
			e	i		i	e		e					e	
				e		e									

Number of e's : 21
Number of i's : 9
Number of objects = (21-9) / 4 = 3
(Why?)

Example

- Component labeling

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

=

4-connected

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

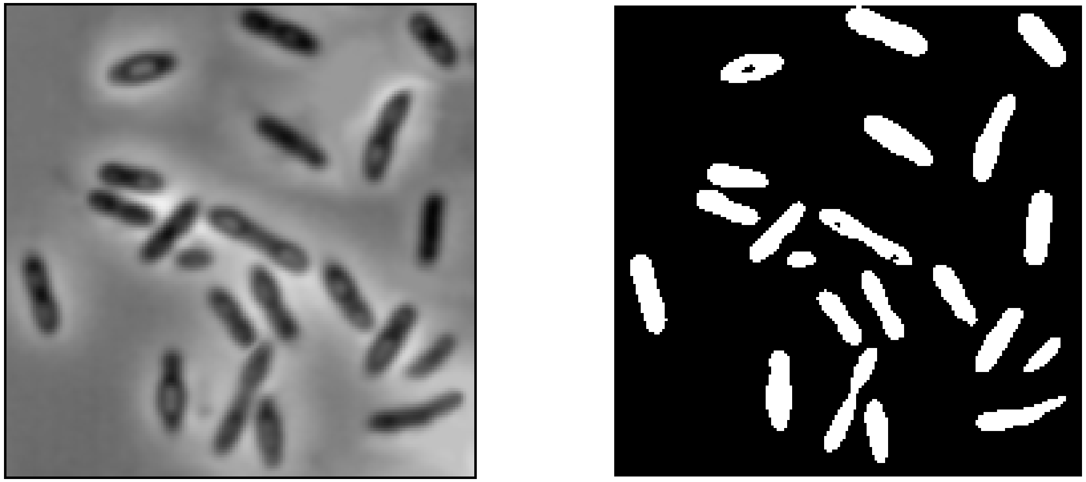
=

8-connected

0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	1	0	0	0	0	0	0
0	1	0	0	2	2	2	2
0	1	0	0	2	2	2	2
0	0	0	2	2	0	0	0

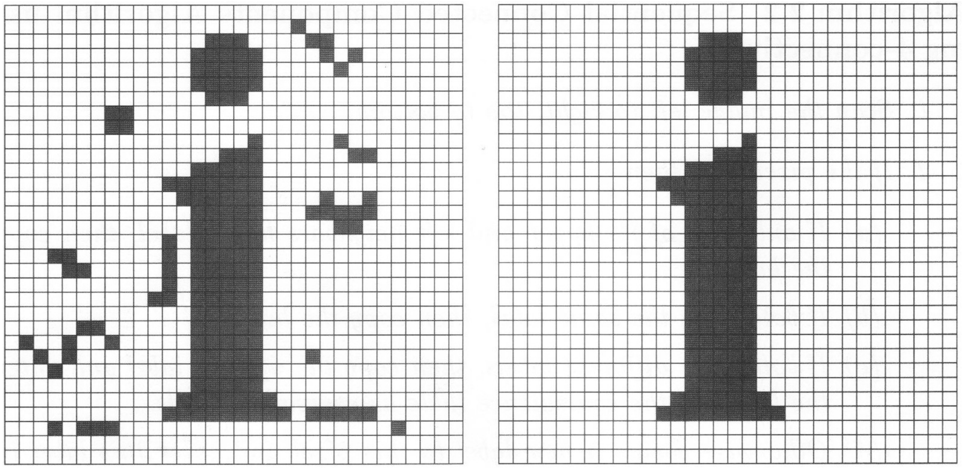
Example

- Object counting

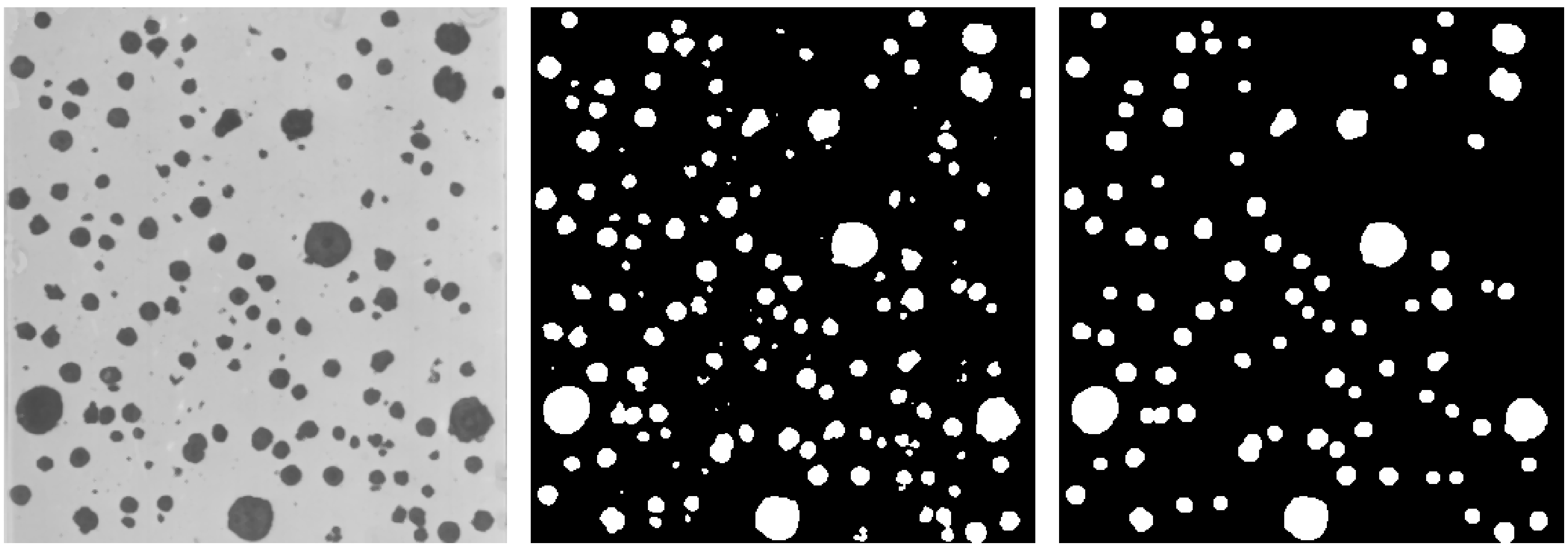


Size Filtering

- In binary images, usually the noise regions are small
- If the size of object is greater than T_0 pixels, a size filter can be used to remove noise after component labeling
- All components below T_0 in size are removed by changing the corresponding pixels to 0

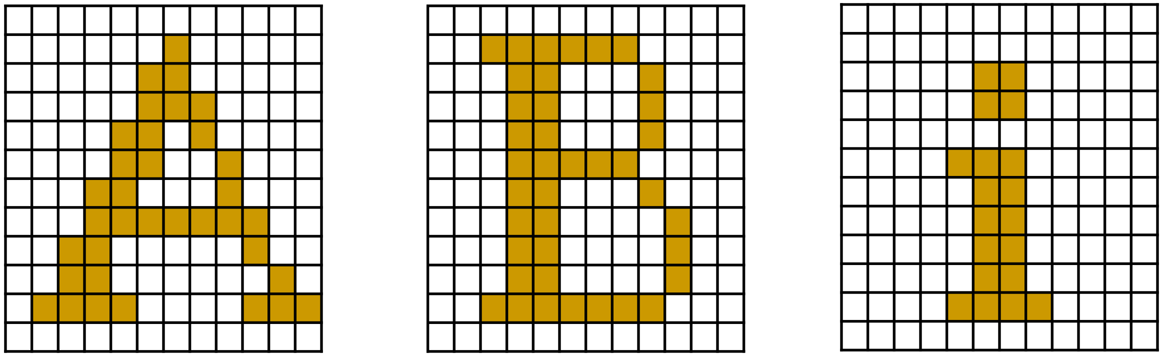


Example



Euler Number

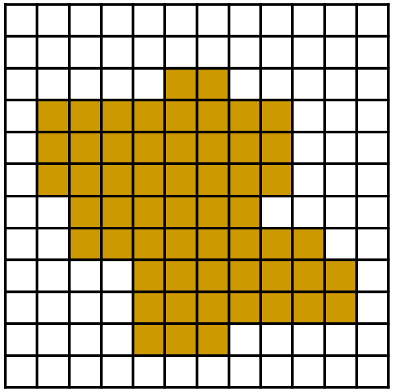
- Genus or Euler number can be used as a feature of an object
- Genus is defined as the number of components minus the number of holes: $E = C - H$
- Genus provides a simple topological feature that is *invariant to translation, rotation and scaling*

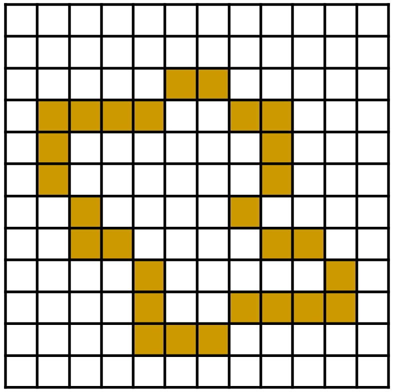


$E = 0$ $E = -1$ $E = 2$

Region Boundary

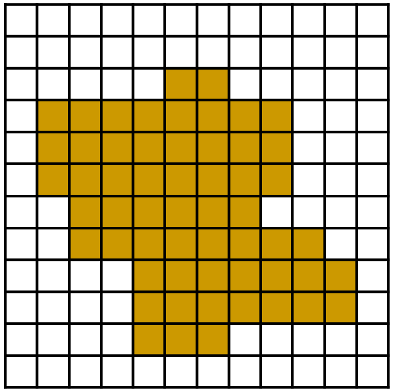
- The boundary of a connected component S is the set of pixels of S that are adjacent to $\bar{S}$
- In most application, we want to track pixels on the boundary in a particular order
- The boundary-following algorithm select a starting pixel $s \in S$ and track the boundary until it comes back to the starting pixel

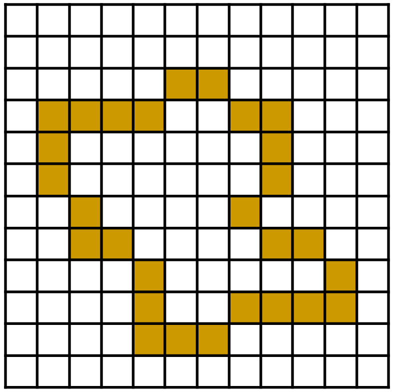




Region Boundary

- Boundary-Following Algorithm
  - Find a starting pixel $s \in S$ for the region via raster scan
  - Let the current pixel in boundary tracking be denoted by c, set $c = s$ and let the 4-neighbor to the west of s be $b \in \hat{S}$
  - Let the eight 8-neighbors of c starting with b in clockwise order be $n_1, n_2, \ldots, n_8$. Find n_i, for the first i that is in S
  - Set $c = n_i$ and $b = n_{i-1}$
  - Repeat above



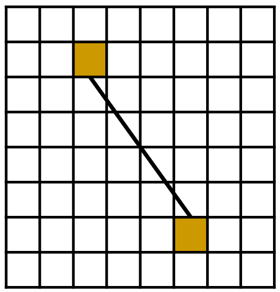


Distance Measure

- To find the distance between two pixels or two components of an image
- For all pixels p, q and r, any distance metric must satisfy all of the following properties:
  - $d(p,q) \geq 0$ and $d(p,q) = 0$ if and only if $p = q$
  - $d(p,q) = d(q,p)$
  - $d(p,r) \leq d(p,q) + d(q,r)$
- Common distance functions:
  - Euclidean: $d_{\text{Euclidean}}([i_1, j_1],[i_2, j_2]) = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2}$
  - City-block: $d_{\text{city}}([i_1, j_1],[i_2, j_2]) = |i_1 - i_2| + |j_1 - j_2|$
  - Chessboard: $d_{\text{chess}}([i_1, j_1],[i_2, j_2]) = \max(|i_1 - i_2|,|j_1 - j_2|)$

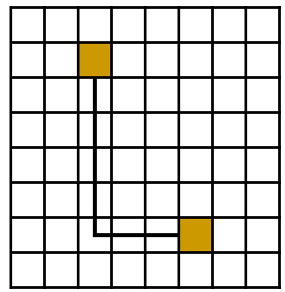
Different Distance Measures

Euclidean distance



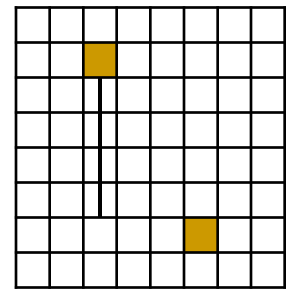
			3			
	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$	
	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$	
3	2	1	0	1	2	3
	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$	
	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$	
			3			

City-block distance



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

Chessboard distance



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

Distance Transform

- In some application, the *minimum distance* between *a pixel* of an object component and the *background* is needed
- Distance transform is to compute the distance to the background region $\hat{S}$, for all pixels in S
  - $f^0[i,j] = f[i,j]$
  - $f^m[i,j] = f^0[i,j] + \min(f^{m-1}[u,v])$

where (u,v) is in the 4-neighbor of (i,j)

					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					

0th pass

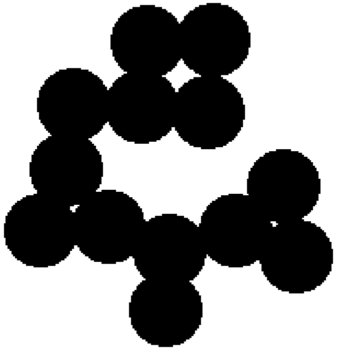
					1	1					
	1	1	1	1	2	2	1	1			
	1	2	2	2	2	2	2	1			
	1	2	2	2	2	2	2	1			
		1	2	2	2	2	1				
		1	1	2	2	2	2	1	1		
				1	2	2	2	2	2	1	
				1	2	2	1	1	1	1	
				1	1	1					

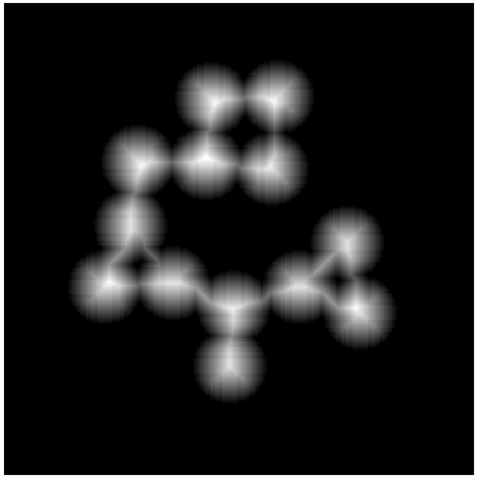
1st pass

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

2nd pass

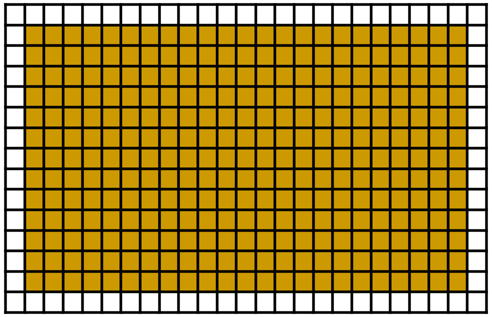
Example

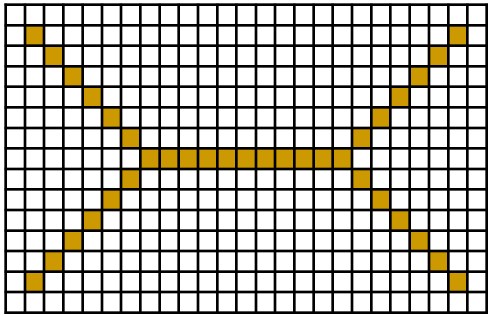




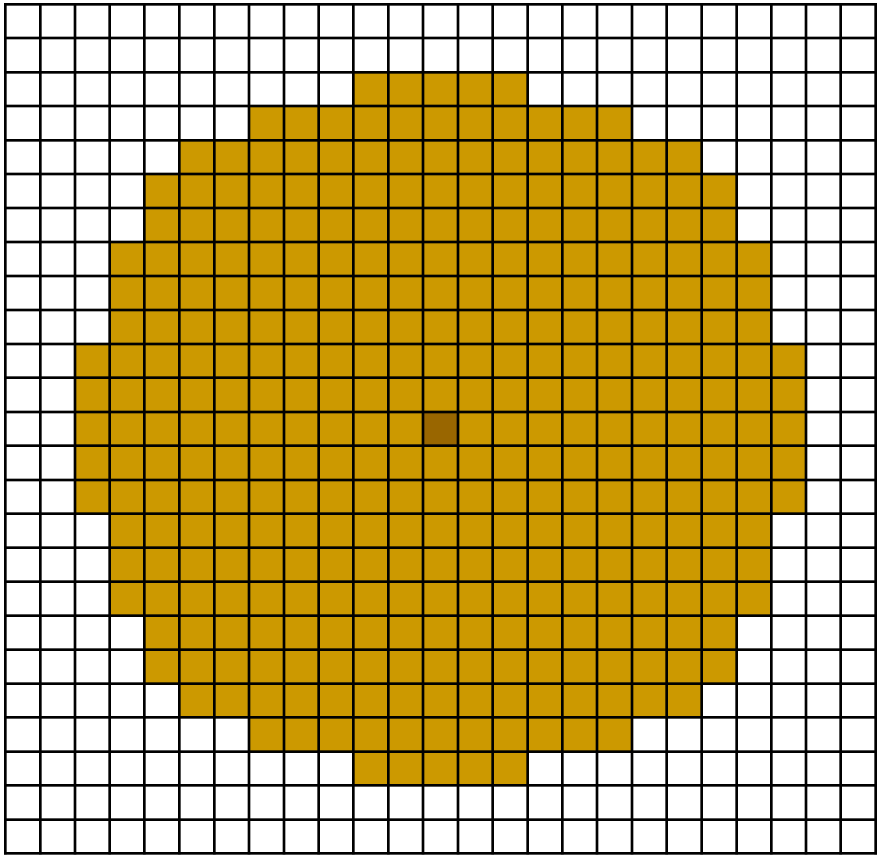
Medial Axis

- The distance $d([i,j], \hat{S})$ from the pixel $[i,j]$ in S to $\hat{S}$ is locally maximum if $d([i,j], \hat{S}) \geq d([u,v], \hat{S})$ for all pixels $[u,v]$ in the neighborhood of $[i,j]$
- The set of pixels in S with distances from $\hat{S}$ that are locally maximum is called the skeleton, symmetric axis, or medial axis of S, and denoted by S^*

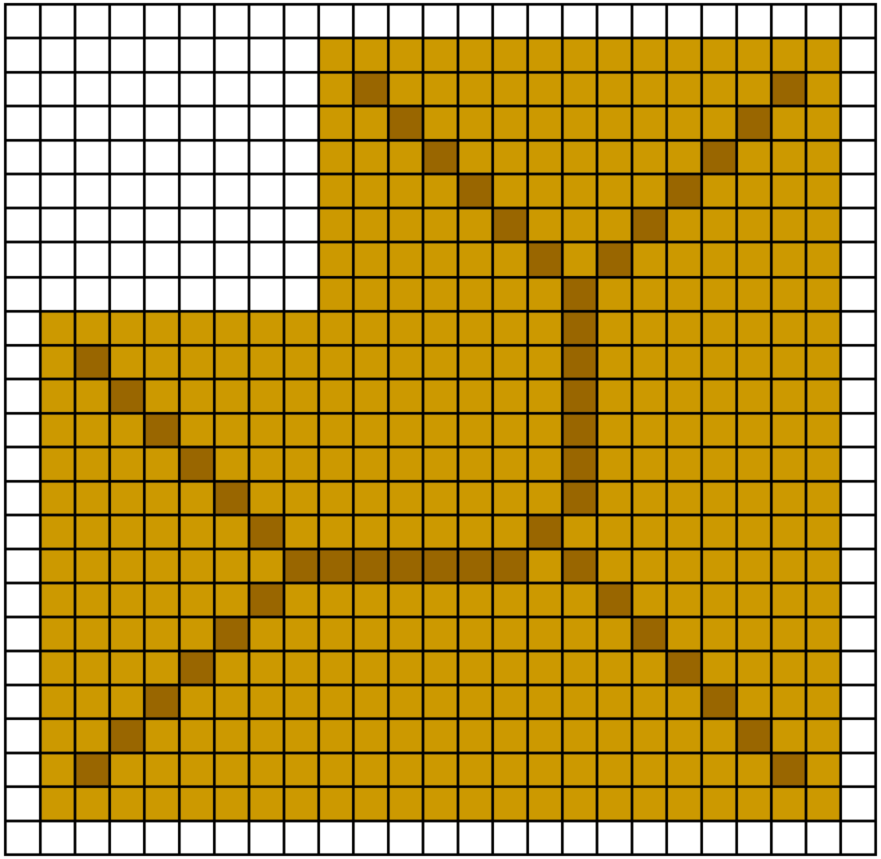




Example



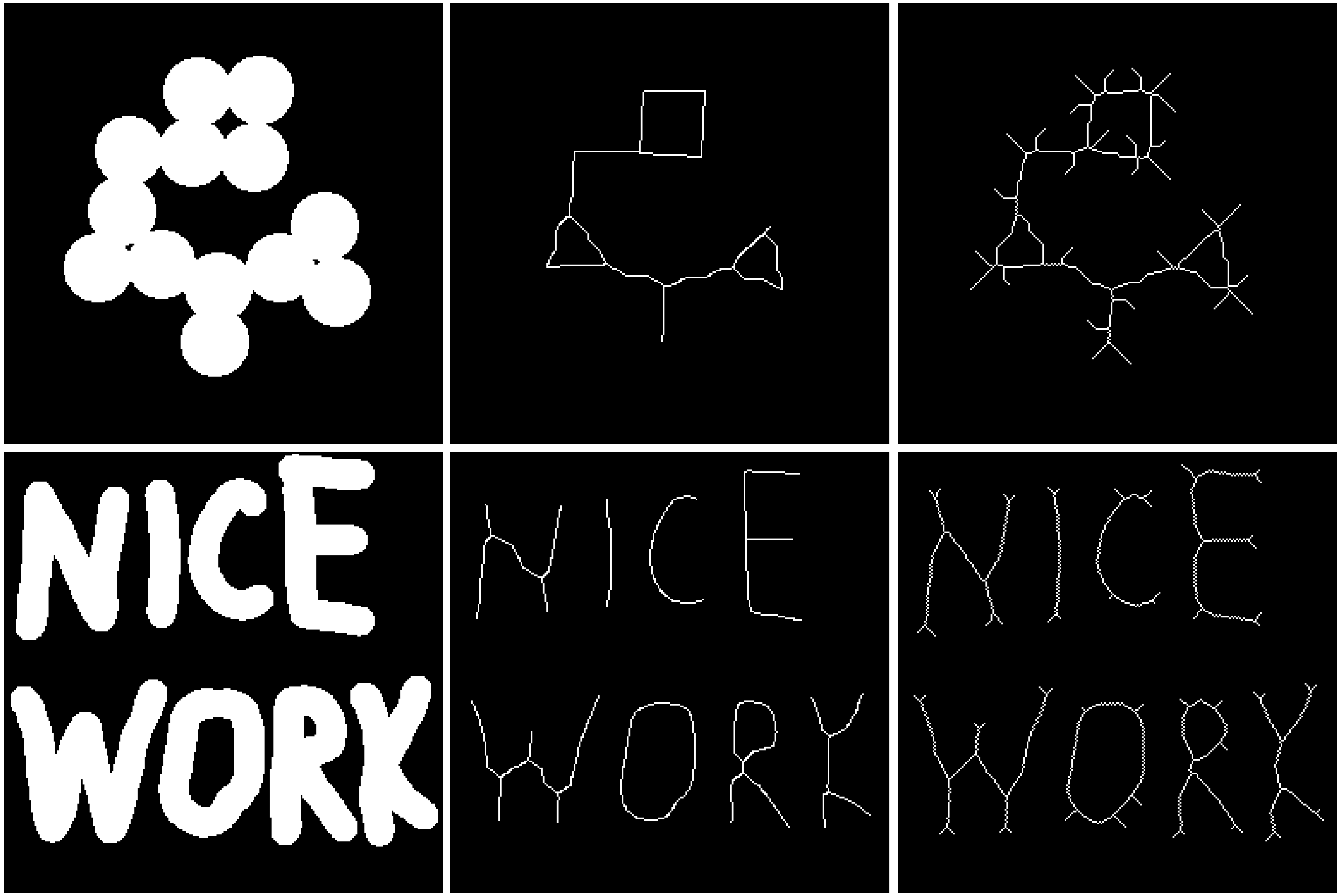
Example



Medial Axis

- The original set S can be reconstructed from S^* and the distance of each pixel of S^* from $\hat{S}$
- S^* is a compact representation of S
- S^* is used to represent the shape of a region
- By deleting pixels of S^* whose distances from $\hat{S}$ are small, we can create a simplified version of S^*
- Two representations – boundary and medial axis:
  - For arbitrary objects, a boundary is a more compact representation of a region
  - To find whether a given pixel is in the region or not, medial axis is a better representation

Example



Thinning

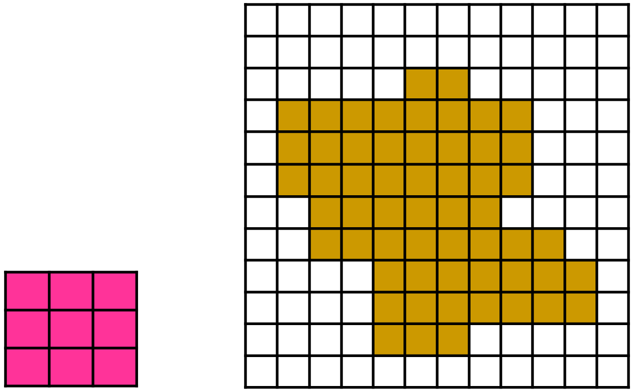
- Thinning:
  - Binary image *regions* are reduced to *lines* that approximate their center lines, also called skeleton or core-line
  - To reduce the image components to their essential information so that further analysis and recognition are facilitated
- Thinning requirement:
  - *Connected image regions* must thin to *connected line structures*
  - The thinned result should be minimally 8-connected
  - *Approximate end-line locations should be maintained*
  - *The thinning results should approximate the medial lines*
  - Extraneous spurs (short branches) caused by thinning should be minimized

Thinning

- Examine each pixel in the image within the context of its neighborhood region of at least 3×3 pixels and to "peel" the region boundaries, one pixel at a time, until the regions have been reduced to thin lines
- The process is performed iteratively

Expanding and Shrinking

- *Expanding*
  - A component is allowed to change such that some background pixels are converted to 1
  - Change a pixel from 0 to 1 if <u>any</u> neighbors of the pixel are 1
- *Shrinking*
  - Object pixels are systematically deleted or converted to 0
  - Change a pixel from 1 to 0 if <u>any</u> neighbors of the pixel are 0



Expanding and Shrinking

- Let $S^{(k)}$: S expanded k times and $S^{(-k)}$: S shrunk k times, then
  - $(S^{m})^{-n} \neq (S^{-n})^{m} \neq S^{(m-n)}$
  - $S \subset (S^{k})^{-k}$
  - $S \supset (S^{-k})^{k}$
- Expanding and shrinking can be used to determine isolated components and clusters (Fig. 2.23 and 2.24)
  - Expanding followed by shrinking can be used for *filling undesired holes*
  - Shrinking followed by expanding can be used for *removing isolated noise pixels*

Binary Image Morphology

- The word *morphology* refers to form and structure
- In computer vision, it can be used to refer to the shape of a region
- The operations of mathematical morphology were originally defined as *set operations*
- *Morphological operators* can:
  - Thin,
  - Thicken,
  - Find boundaries,
  - Find skeletons (medical axis),
  - Convex hull,
  - And more

Structuring Elements

- The operations of binary morphology input a binary image B and a structuring element S
- The structuring element S is usually another smaller binary image
  - It represents a shape
  - It can be any size and have arbitrary structure that can be represented by a binary image
  - Examples:

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

Point Sets and Notation

- Binary objects are considered as *point sets*
- For point sets A and B denote the:
  - Translation of A by x as $A_x = \{a_i + x \mid a_i \in A\}$
  - Reflection of B as $B^r = \{-b_i \mid b_i \in B\}$
  - Complement of A as $A^c = \{a_i \mid a_i \notin A\}$
  - Difference of A and B as $A - B = \{c_i \mid (c_i \in A) \text{ XOR } (c_i \in B)\}$

Basic Operations

- The basic operations of binary morphology are *dilation*, *erosion*, *closing*, and *opening*
  - Dilation enlarges a region
  - Erosion makes a region smaller
  - A closing operation can close up internal holes in a region and eliminate bays along the boundary
  - An opening operation can get rid of small portions of the region that just out from the boundary into the background region

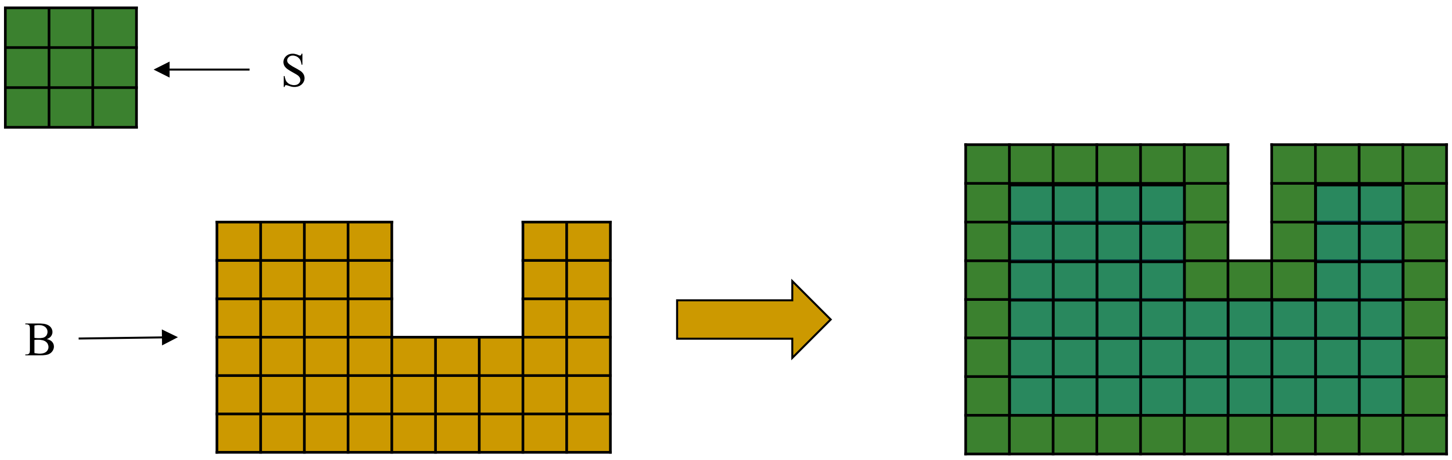
Some Applications

- Binary morphology can be used to extract primitive features of an object that can be used to recognize the object
- A shape matching system can use morphological feature detection to rapidly detect primitives that are used in object recognition

Dilation

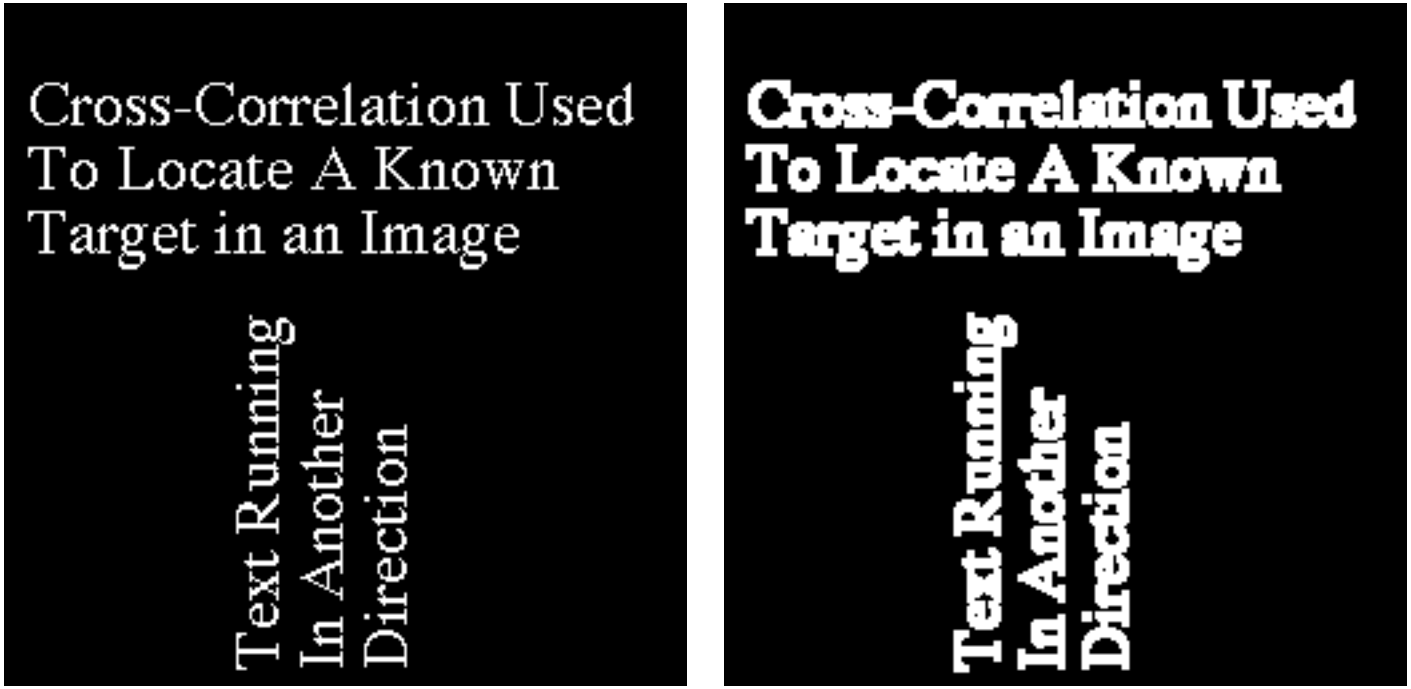
- Dilation of B by structuring element S:

$$B \oplus S = \{x \mid (S^r{}_x \cap B) \neq \varnothing\} = \cup_{b \in B} S_b$$



- Example: dilating A with a 3×3 structuring element B centered at the origin

Example

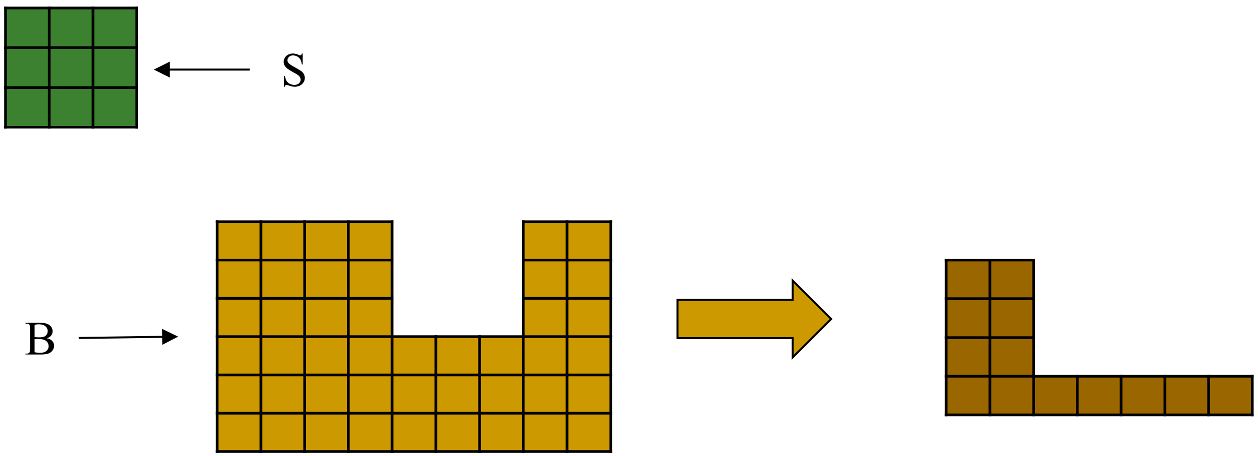


Cross-Correlation Used
To Locate A Known
Target in an Image
Text Running
In Another
Direction
Cross-Correlation Used
To Locate A Known
Target in an Image
Text Running
In Another
Direction

Erosion

- Erosion of B by structuring element S:

$$B \ominus S = \{x \mid S_x \subseteq B\} = \{b \mid b + s \in B, \forall s \in S\}$$



- Example: eroding A with a 3×3 structuring element B centered at the origin

Example

Combining Dilation and Erosion

- Combining dilation and erosion for
  - *Opening*
  - *Closing*
  - Thickening
  - Thinning
  - Skeleton

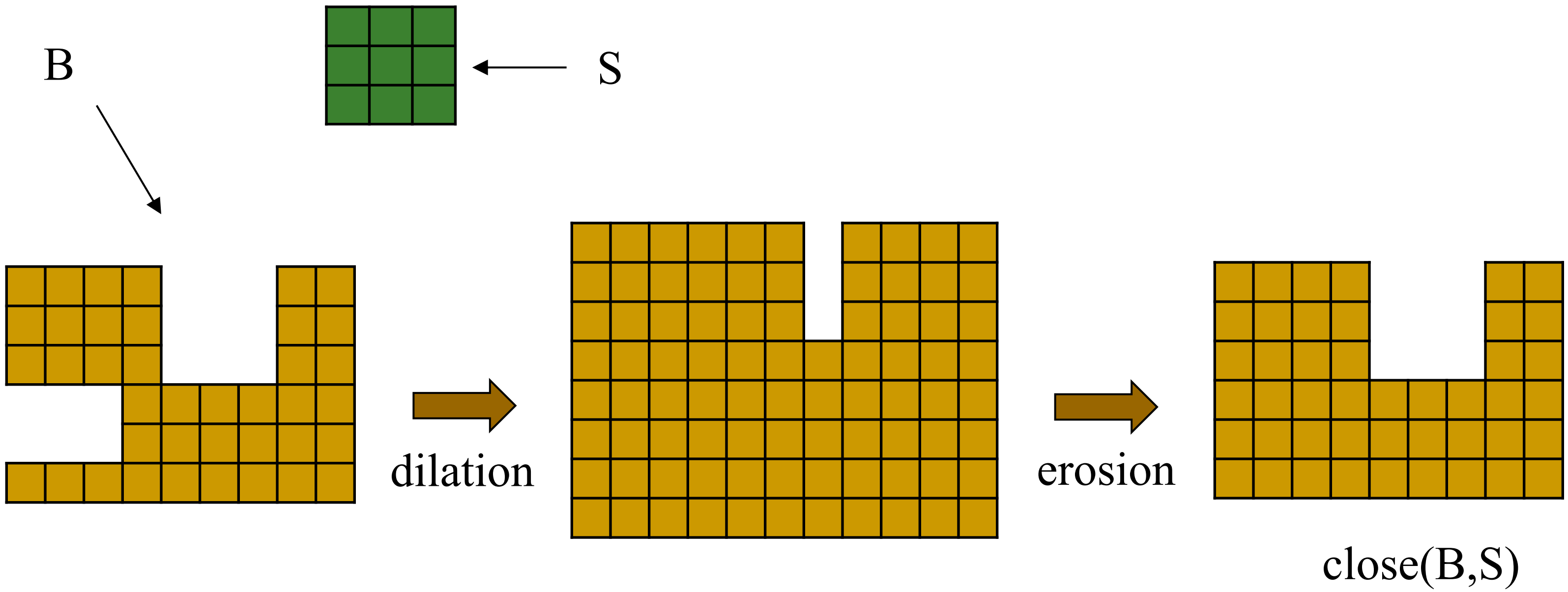
Intuitive Interpretation

- Dilation expands an object
- Erosion contracts an object
- Opening
  - Smooths contours
  - Enlarges narrow gaps
  - Eliminates thin protrusions
- Closing
  - Fills narrow gaps, holes and small breaks

Closing

- Closing: Like "smoothing from the outside"

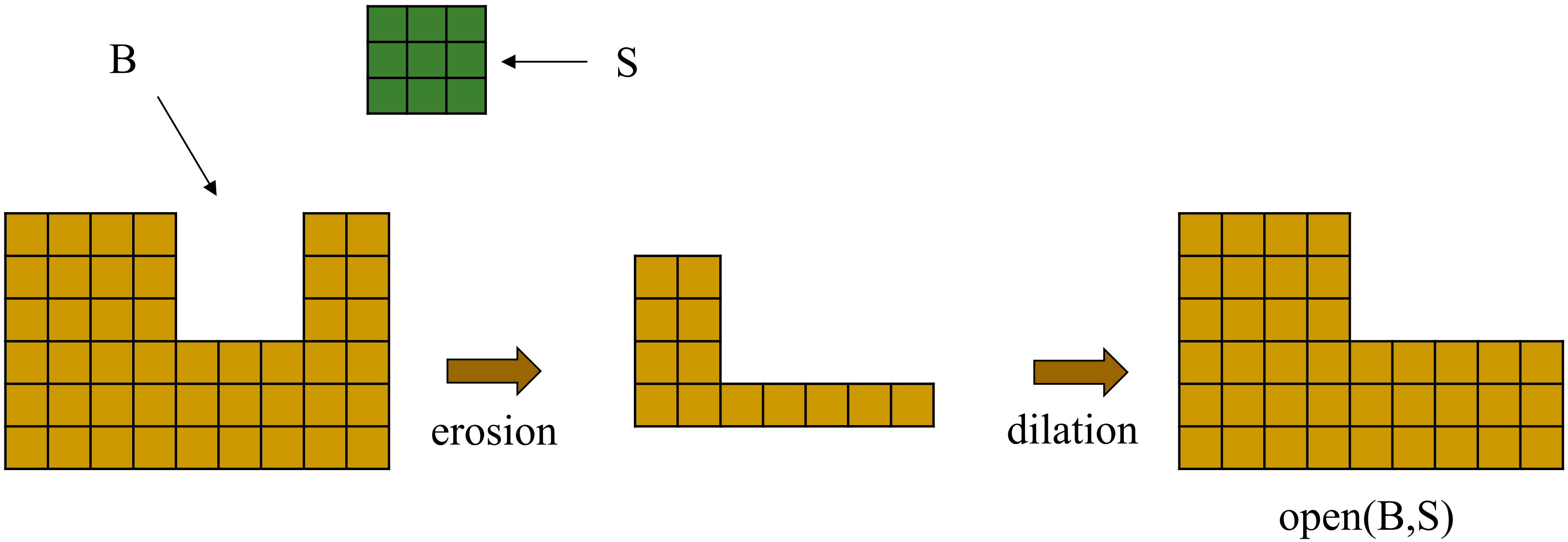
$$B \bullet S = (B \oplus S) \ominus S$$



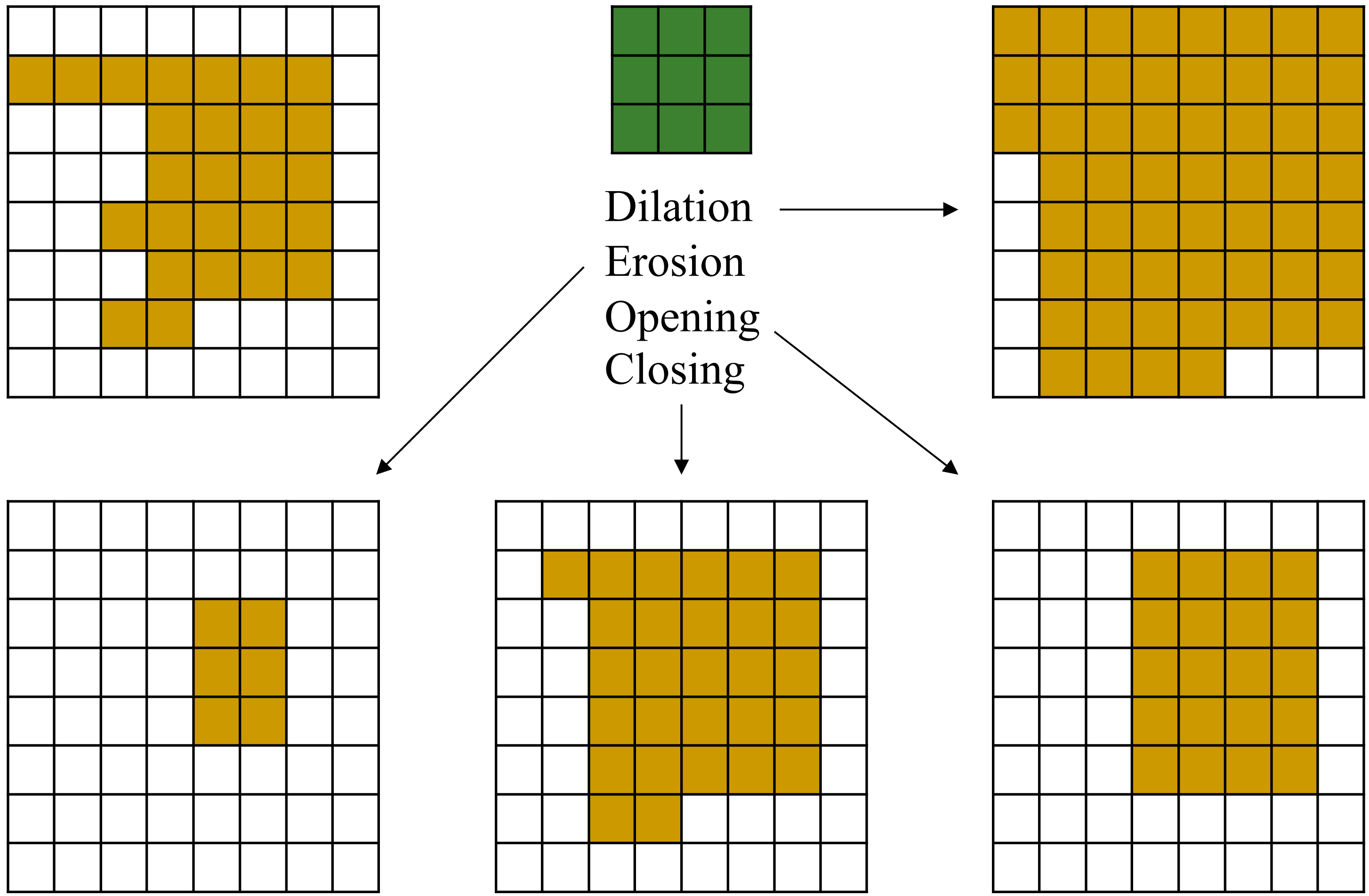
Opening

- Opening: Like "smoothing from the inside"

$$B \circ S = (B \ominus S) \oplus S = \cup \{S_x \mid S_x \subseteq B\}$$



More Example



Idempotency

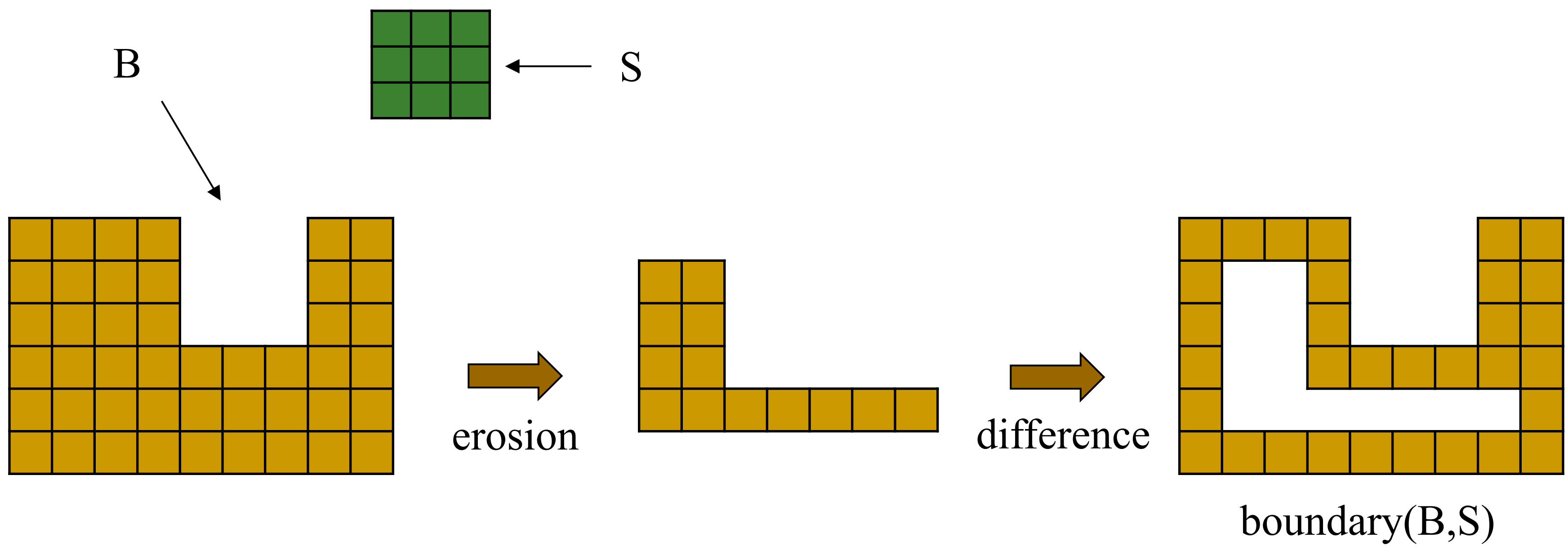
- Applying opening or closing more than once has no further effect
  - open(open(A,B),B) = open(A,B)
  - close(close(A,B),B) = close(A,B)

Additional Structuring Operations

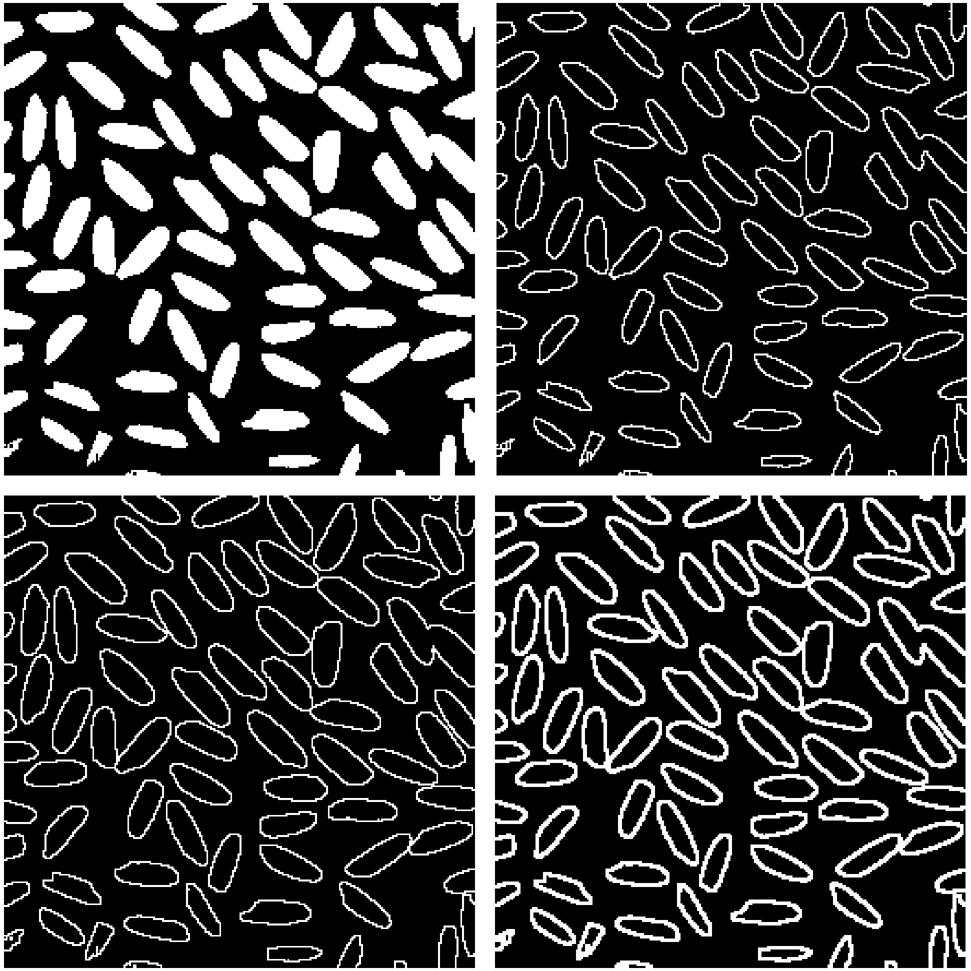
- Find boundary of an object
- Region filling
- Skeleton

Find Boundary

$$\text{boundary}(B,S) = B - B \ominus S$$



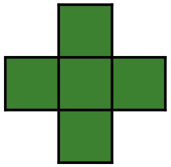
Example

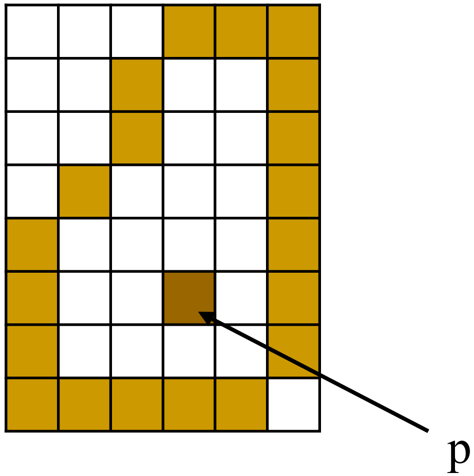


Region Filling

- Problem:
  - Fill 8-connected boundary A with 1's given a point inside the boundary p

S





B

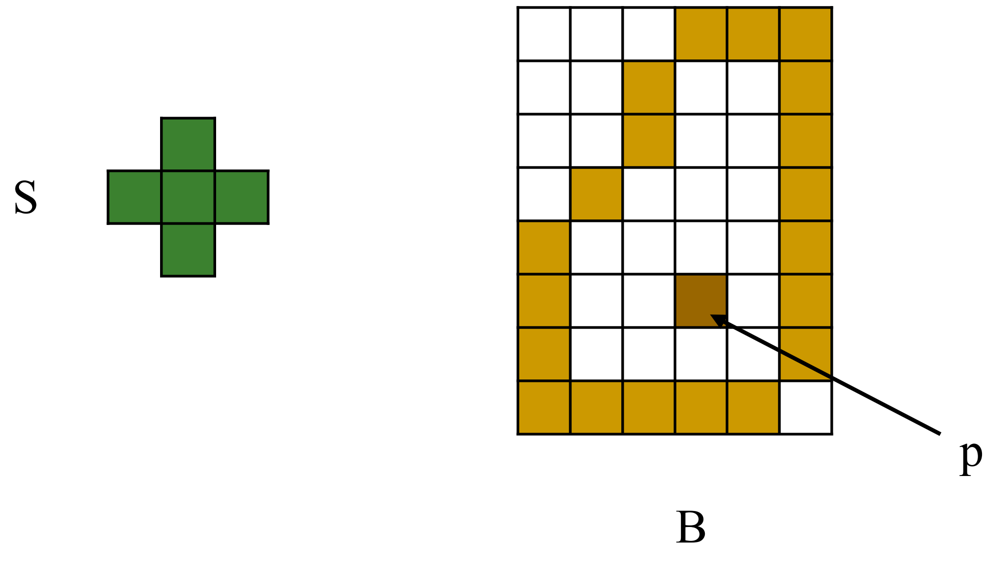
- Use structuring element S, and
  - Iterative dilations
  - Intersection
  - Complement

Region Filling

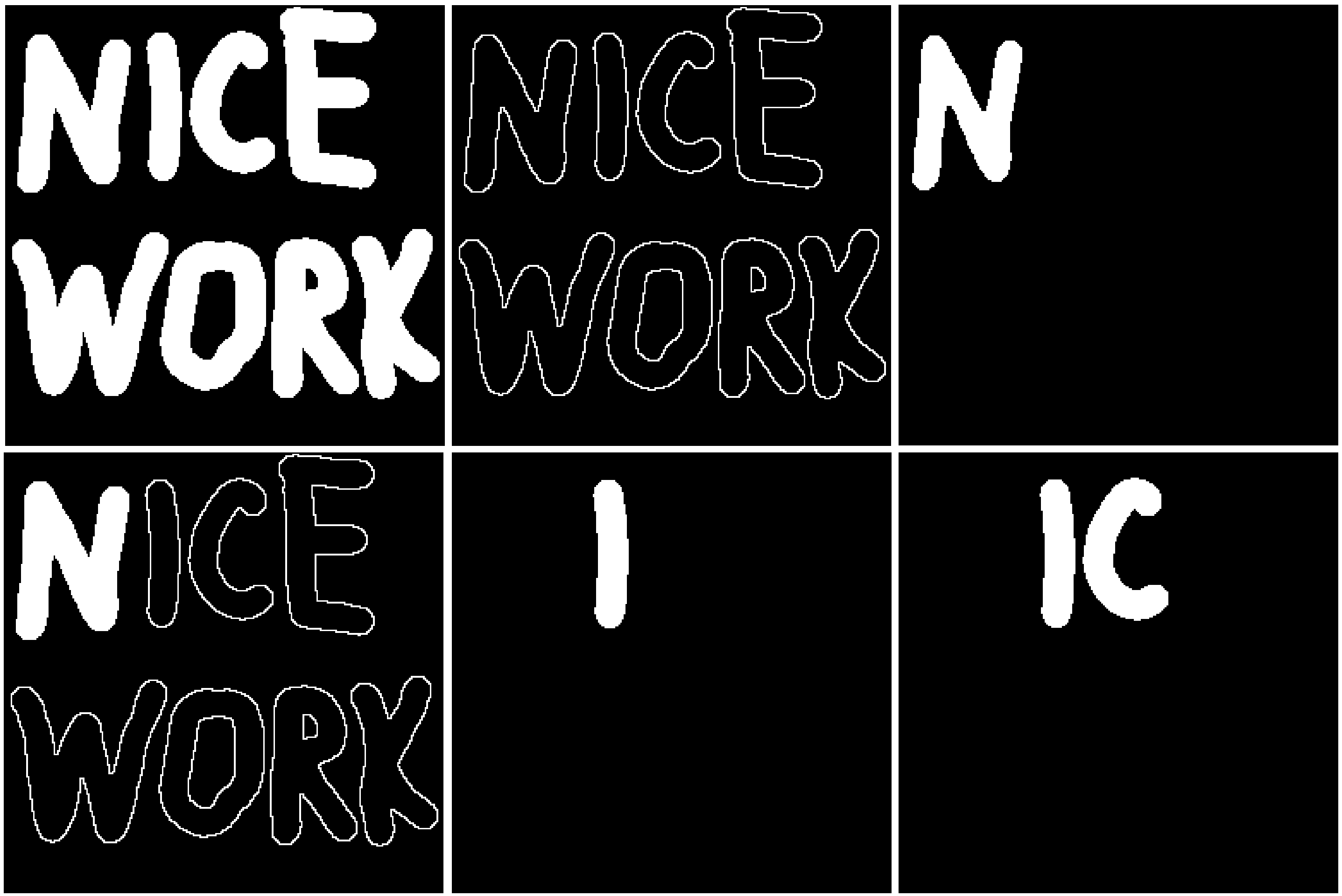
- Let $C_0 = p$
- Calculate

$$C_k = (C_{k-1} \oplus S) \cap B^c, \text{ for } k = 1,2, \ldots$$

- Stop when $C_k = C_{k-1}$
- C_k is the interior of B



Example



Reading

- Chapter 2 of Jain's book