Digital Image: Geometry in a discrete grid

Jayanta Mukhopadhyay
Dept. of CSE,
IIT Kharagpur

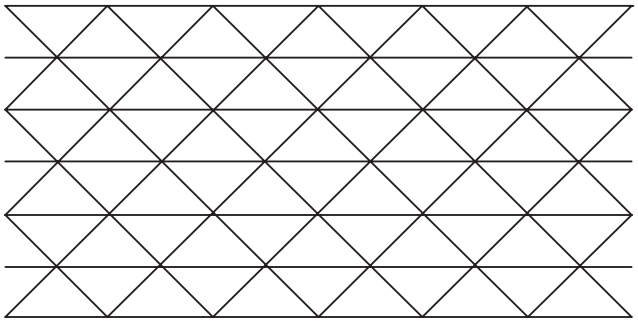
Discrete Space and Tiling

- Discrete Space
  - Integral coordinate space: Z^n
- Digital grid
  - A finite subset of integral coordinate space: $G^n \subset Z^n$
    - represented by a n-D rectangular array
- Tessellation or Tiling
  - 2D: A bounded region of a specific shape that fills the plane with no overlaps and no gaps
    - Center of the region lies at a grid point
  - Regular tiling: All the space filling units are similar and of the shape of a regular polygon
    - Generalization to n-D regular hyper-polyhedron

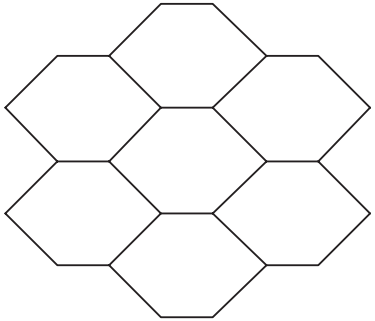
J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Regular Archimedean Tiling

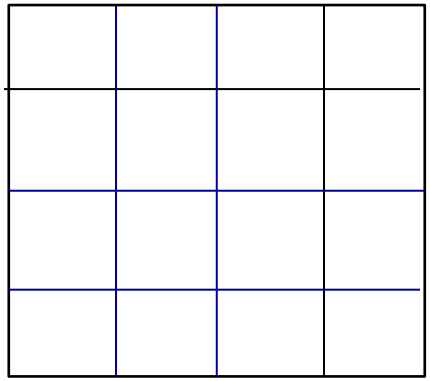
- In 2-D: 3



Triangular



Hexagonal



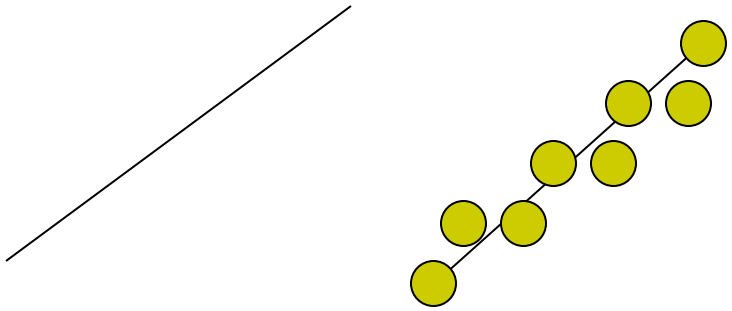
Rectangular

- In 3-D: 1 (Rectangular)
- In 4-D: 3
- In n-D, n>4: 1 (Rectangular)

Our focus: Digital geometry in rectangular grid !
2-D or 3-D

Why digital geometry?

- Digital Distance Function (different domain and range)
  d: Z^n x Z^n $\longrightarrow$ P (or R)
- Finite neighbourhood.
- Things are different (due to digitization.)



Motivation

- **Exact analysis:** As they are in the digital space.
- **Analysis of approximation:** Error bounds from Euclidean measures, better understanding in correlating corresponding objects and shapes.
- **Efficient computation:** Finite neighbourhood and integer arithmetic.
- **Empirical Techniques:** Finite enumeration.

Distance Function

- Distance Function:

  $d: R^n \times R^n \longrightarrow R$

- Digital Distance Function:

  $d: Z^n \times Z^n \longrightarrow P$ (or R)

- A function of two points in a space measuring their separation or dissimilarity.

Examples:

$$\underline{u} \equiv (u(1), u(2), \ldots\ldots, u(n))$$

$$\underline{v} \equiv (v(1), v(2), \ldots\ldots, v(n))$$

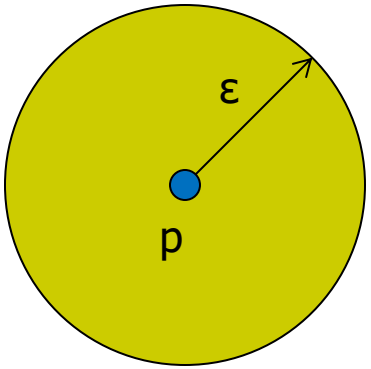
$$L_p(\underline{u}, \underline{v}) = \left(\sum_{i=1}^{n} | u(i) - v(i) |^p \right)^{\frac{1}{p}}$$

$$L_1(\underline{u}, \underline{v}) = \sum_{i=1}^{n} | u(i) - v(i) |$$

$$L_2(\underline{u}, \underline{v}) = E_n(u, v) = \sqrt{\sum_{i=1}^{n} | u(i) - v(i) |^2}$$

$$L_\infty(\underline{u}, \underline{v}) = \max_{1 \le i \le n} \{ | u(i) - v(i) | \}$$

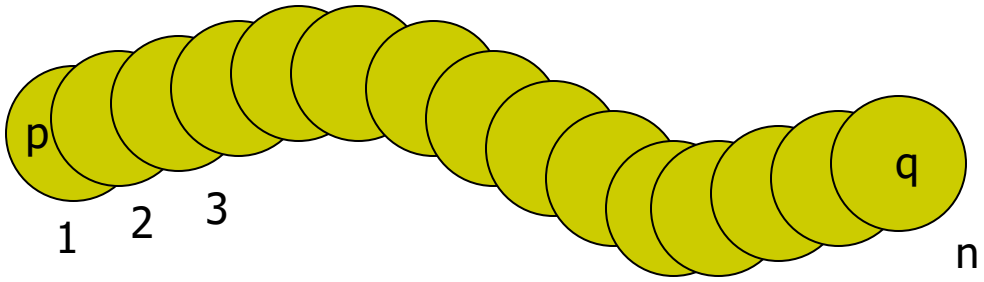
Neighbourhood



ε-Neighbour:
a point q is in N(p),
iff d(p,q) ≤ ε.

Desirable: Compact and bounded.

Path



Length of the path ≈n. ε

Desirable feature:

Distance = Length of the shortest Path.

Metric & Conditions:

i. $d(\underline{u}, \underline{v}) = 0$ iff $\underline{u} = \underline{v}$

ii. $d(\underline{u}, \underline{v}) = d(\underline{v}, \underline{u})$

iii. $d(\underline{u}, \underline{v}) + d(\underline{u}, \underline{w}) \geq d(\underline{u}, \underline{w})$,

$\forall\ \underline{u}, \underline{v}, \underline{w} \in R^n (Z^n)$

d is defined for every pair of points in the space.

N($\underline{x}$;d) is a Norm of d if

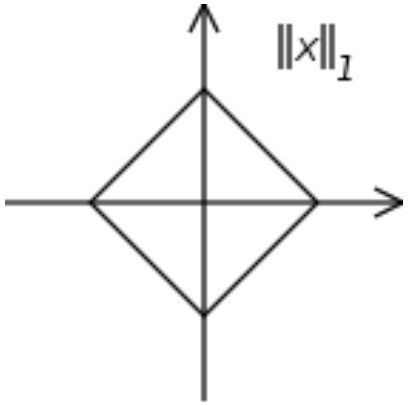
$N(\underline{x};d) = d(\underline{x}, \underline{0})$ where $\underline{x} = \underline{u} - \underline{v}$

Why a metric?

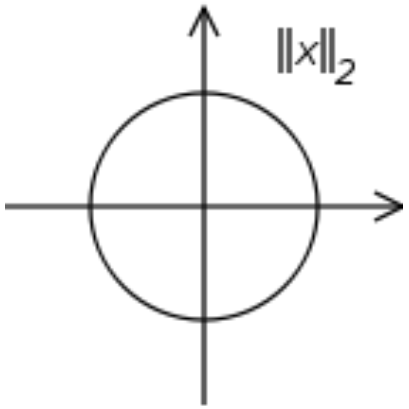
- A bounded and compact neighbourhood around a point in the space.
- There exists an optimal path between two points, such that the length of this path is given by the distance function.

Neighbourhood structure: Examples

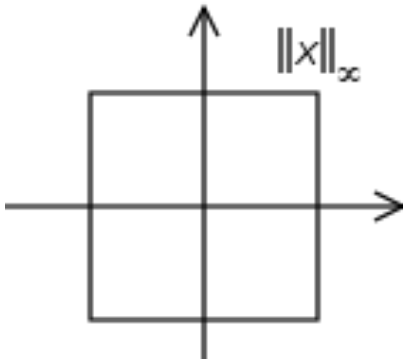
L_1 norm



L_2 norm



L_∞ norm



Desirable features:

Isotropy and symmetry:
Isotropic in all directions.

Uniformity:
Identical at all points of the space.

Convexity:
In the sense of Euclidean geometry.

Self-similar:
Similar structure at varying resolution.

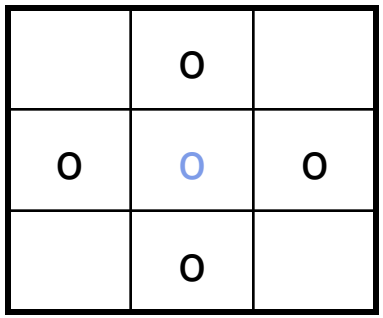
Acknowledgement: Figure taken from http://en.wikipedia.org/wiki/File:Vector_norms.svg

Geometry built on distances

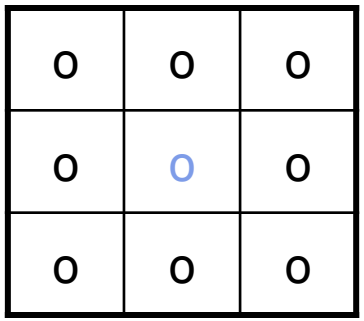
- Shortest Paths in Euclidean Space
  - Straight Lines
- Geodesics on Earth
- Parallel Lines
  - Equidistant Ever
- Circle
  - Trajectory of a point equidistant from Center
  - Least Perimeter with Largest Area
- Conics are distance defined.

Neighbors of Pixels in 2-D

- P = (x,y)
  - 4-Neighbours
    - $N_4(P) = \{(x-1,y), (x+1,y) , (x,y-1), (x,y+1)\}$
  - Diagonal-Neighbours
    - $N_D(P) = \{(x-1,y-1), (x-1,y+1) , (x+1,y-1), (x+1,y+1)\}$
  - 8-Neighbours
    - $N_8(P) = N_4(P) \cup N_D(P)$



4-neighbors



8-neighbors

For a digital distance:
A neighbor is at distance 1.

Digital Distances in 2D

For $\underline{u}, \underline{v} \in Z^2$ and $\underline{x} = |\underline{u} - \underline{v}| = (x_1, x_2)$,

Cityblock:

$$d_4(\underline{u}, \underline{v}) = x_1 + x_2 \ ,$$

Chessboard:

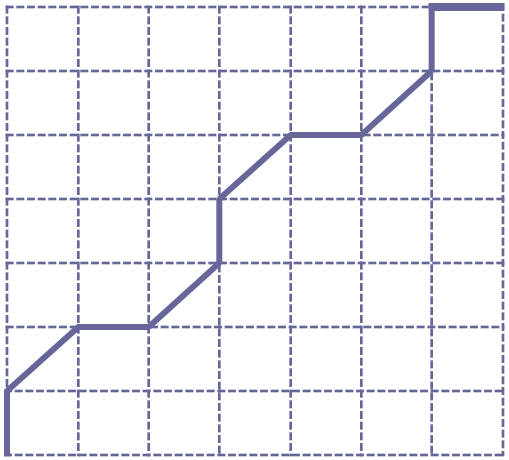
$$d_8(\underline{u}, \underline{v}) = \max(x_1, x_2),$$

Octagonal (uses non-uniform alternating neighborhoods):

$$d_{oct}(u, v) = \max(x_1, x_2, \lceil 2(x_1 + x_2)/3 \rceil) \ .$$

Rosenfeld and Pfaltz '68

Octagonal Distance



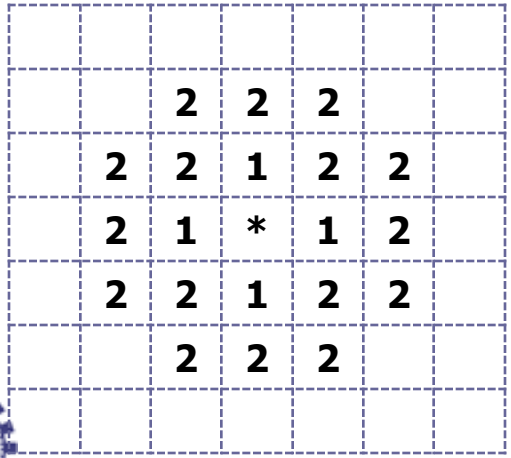
b

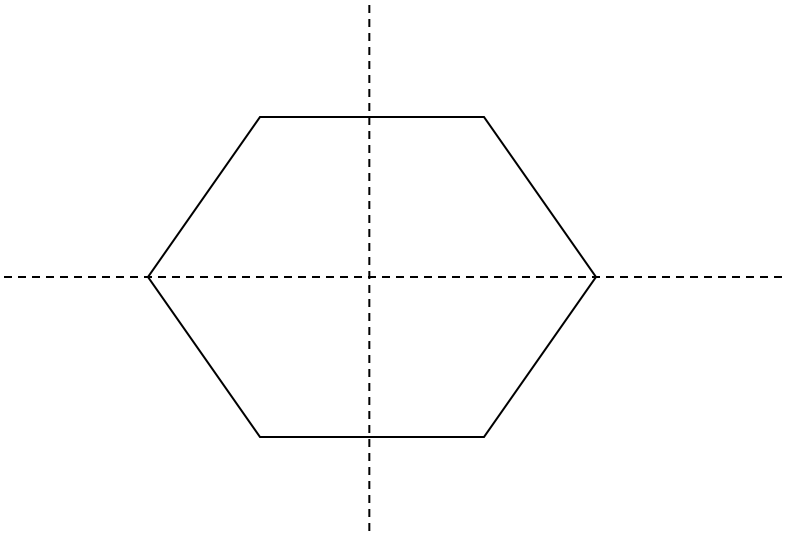
d(a,b) = 10

Neighborhood sequence {1,2}

$d_{oct}(u, v) = \max(x_1, x_2, \lceil 2(x_1 + x_2)/3 \rceil)$

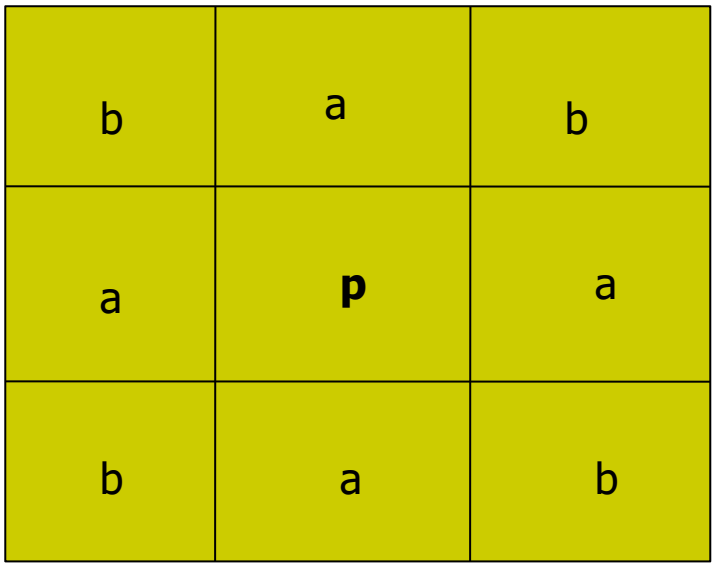
a





Weighted Distance

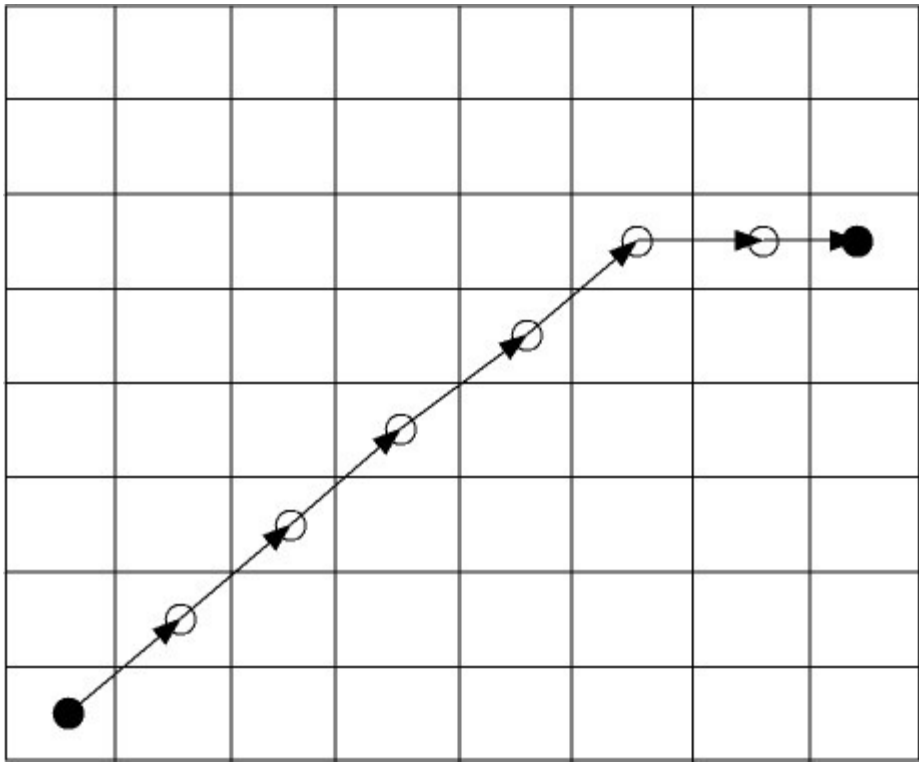
For $\underline{u}, \underline{v} \in Z^2$ and $\underline{x} = |\underline{u} - \underline{v}| = (x_1, x_2)$,



Metric Conditions:
$a \leq b$, and $b \leq 2a$

$$d_{<a,b>}(\underline{u}, \underline{v}) = a.\max(x_1 , x_2) + (b-a).\ \min(x_1 , x_2)$$

Weighted Distance: An example of a path

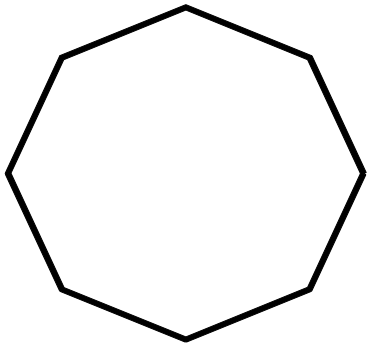


$d_{<a,b>}(p,q)=5b+2a$

e.g.,
For a=1 and b=1.4,
$d_{<1,2>}(p,q)=9$.

Euclidean distance:
$d_e(p,q)= 8.6$

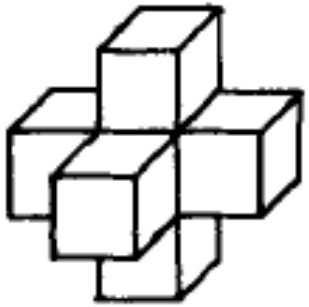
Circle of a weighted distance



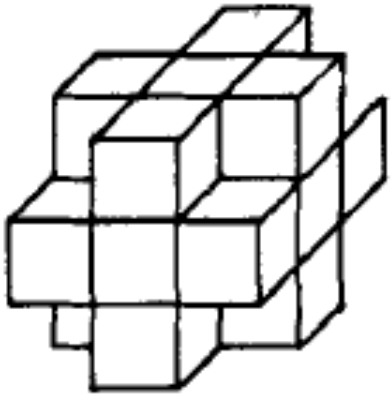
<1, √2>

Neighbors of Voxels in 3-D

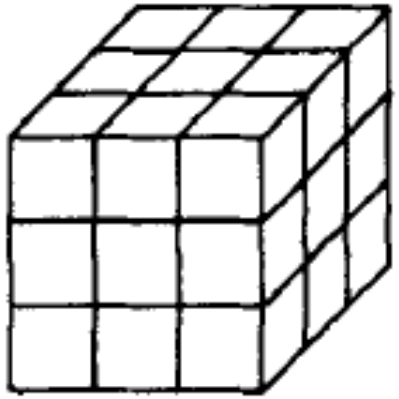
- P = (x,y,z)
  - 6-Neighbors (Face),
  - 18-Neighbors (Edge) &
  - 26-Neighbors (Corner)



6–Neighbors



18–Neighbors



26–Neighbors

J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Digital Distances in 3D

For $\underline{u}, \underline{v} \in Z^3$ and $\underline{x} = |\underline{u} - \underline{v}|$,

$d_6(\underline{u}, \underline{v}) = x_1 + x_2 + x_3$, (grid distance)
$d_{18}(\underline{u}, \underline{v}) = \max(x_1, x_2, x_3, \lceil (x_1 + x_2 + x_3)/2 \rceil)$
$d_{26}(\underline{u}, \underline{v}) = \max(x_1, x_2, x_3)$ (lattice distance).

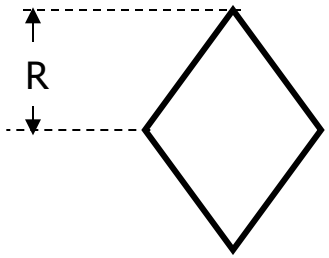
Like 2D, in 3D also Octagonal and Weighted distances are defined.

Yamashita and Ibaraki '86

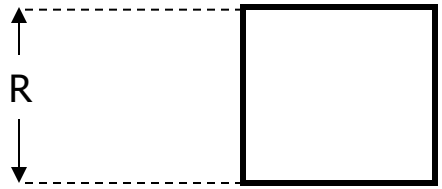
Digital discs

$$B(\underline{O};R)=\{\underline{x}|d(\underline{O},\underline{x})\le R\}$$

2D

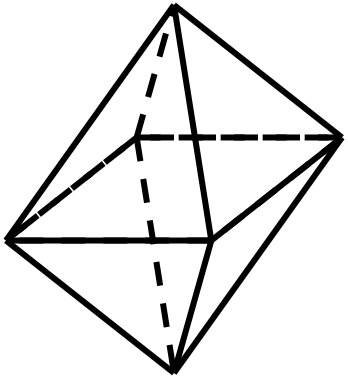


d_4

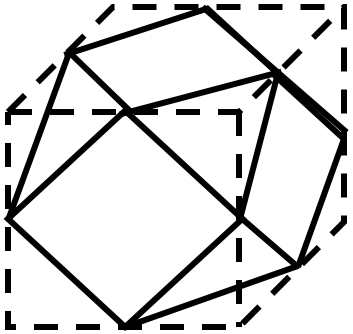


d_8

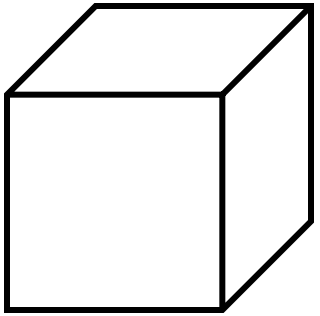
3D



d_6



d_{18}



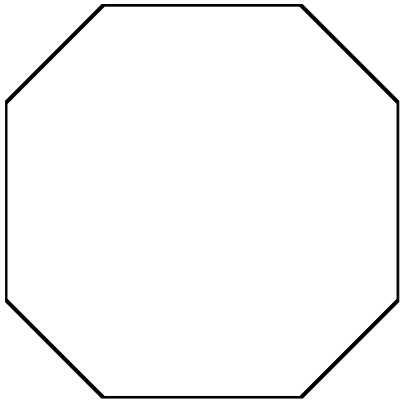
d_{26}

Vertices of octagonal discs:

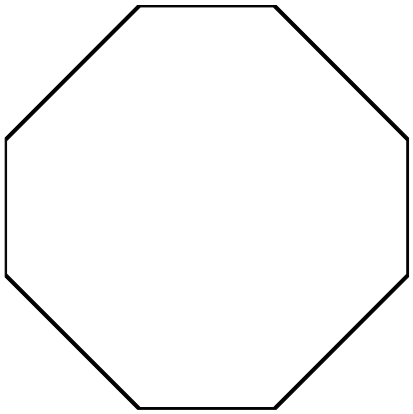
$$\text{2D}: \quad \left\{1^a 2^b\right\} : \text{permutation of} \left(\pm \text{R}, \pm \frac{b}{a+b} \text{R}\right)$$

$$\text{3D}: \quad \left\{1^a 2^b 3^c\right\} : \text{permutation of} \left(\pm \text{R}, \pm \frac{b+c}{a+b+c} \text{R}, \pm \frac{c}{a+b+c} \text{R}\right)$$

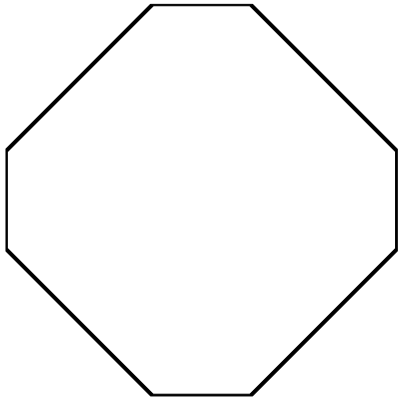
Octagonal Digital Circles



$\{1,2\}$

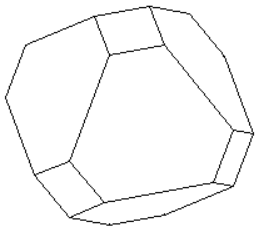


$\{1,1,2\}$

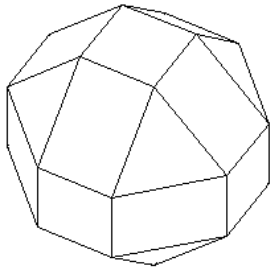


$\{1,1,1,2\}$

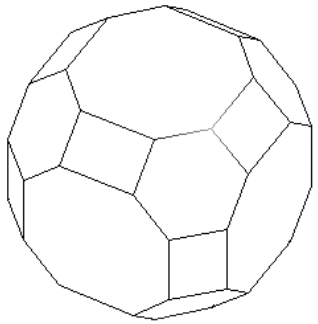
Octagonal Digital Spheres



{1,1,2}



{1,1,3}



{1,2,3}

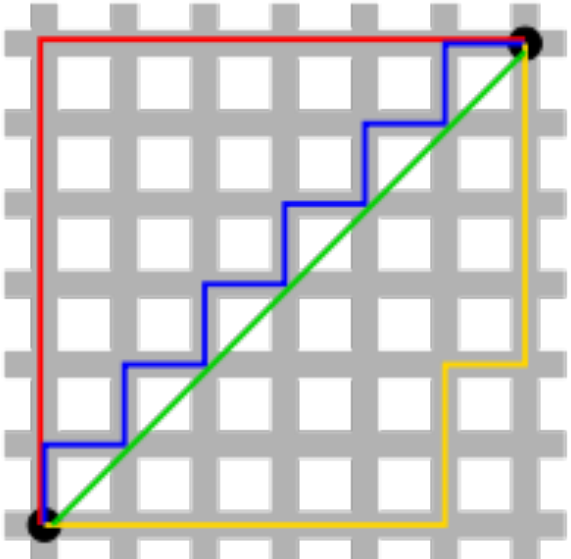
Binary Image

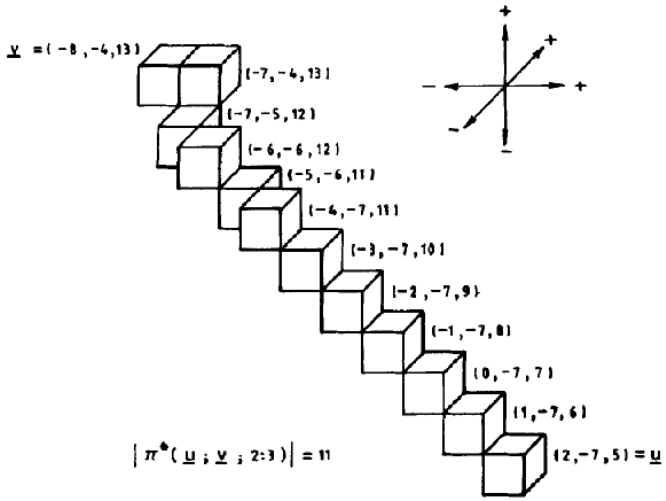
- $f : G^n \rightarrow \{0,1\}$
- Foreground and background
  - $p \in G^n$ is in foreground, $\text{iff } f(p)=1$, otherwise it is in background.
    - Foreground consists of object points.
      - S : foreground
      - $S^c = G^n - S$: background
- Adjacent neighbor
  - A neighboring foreground pixel
    - 2D: 4 / 8 - adjacency
    - 3D: 6 /18 / 26 - adjacency

Path

- A Sequence of distinct pixels satisfying pair-wise adjacency
- Length of the path
  - Number of adjacencies
  - One less than the number of points
  - Shortest Path between two pixels can be non-unique
  - Closed Path: Same Start and End Points

2-D





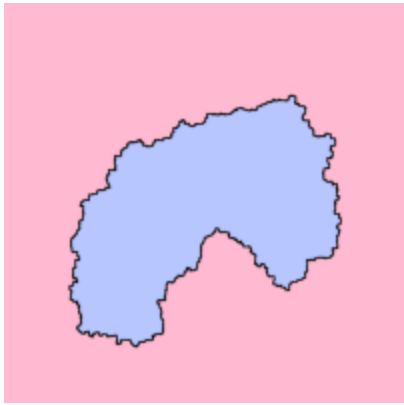
3-D

Connectivity

- Two pixels are **Connected** if there exists a path between them.
- Pixels connected to a given pixel form a **Connected Component.**
  - A **Region** is a **Connected Set of foreground points.**
- **Foreground & Background**
  - Foreground: Union of all disjoint Regions in an Image
  - Background: Complement of Foreground
- **Boundary or Border**
  - Inner Border: Set of points in R adjacent to complement of R
  - Outer Border: Set of points in complement of R adjacent to R

4– and 8–Neighbors Dichotomy

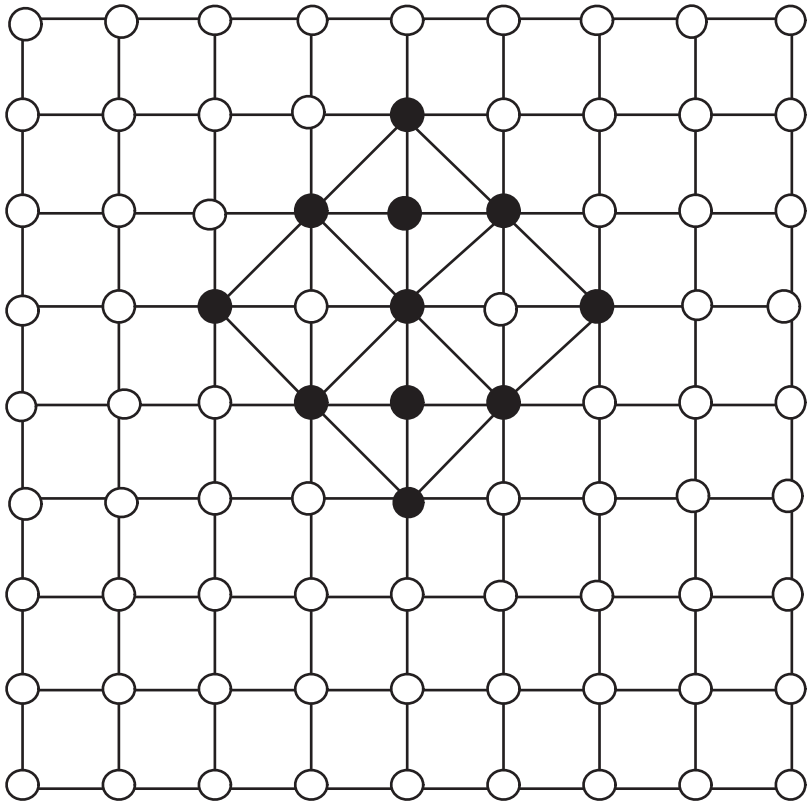
- Jordan's Curve Theorem
  - Every simple closed curve divides the plane into an "interior" region bounded by the curve and an "exterior" region containing all far away points
    - Any continuous path connecting a point of one region to a point of the other intersects that loop somewhere.



- Digital Jordan's Curve Theorem
  - Deviates from the theorem.
  - Use alternate adjacency for interior / exterior

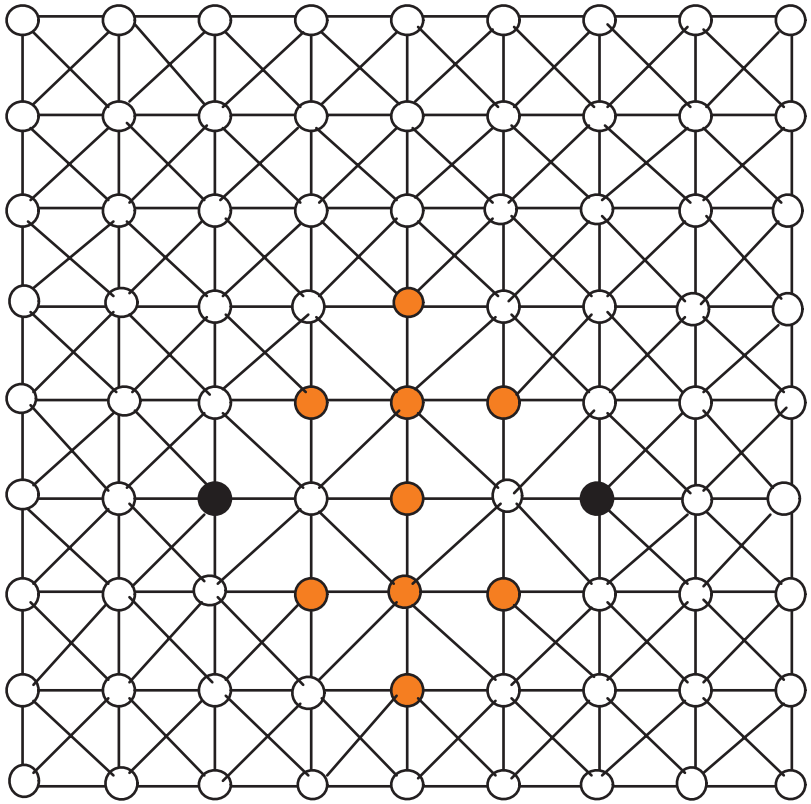
J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Limitation of Jordan's curve theorem in digital space



(8,4)

A single component



(4,8)

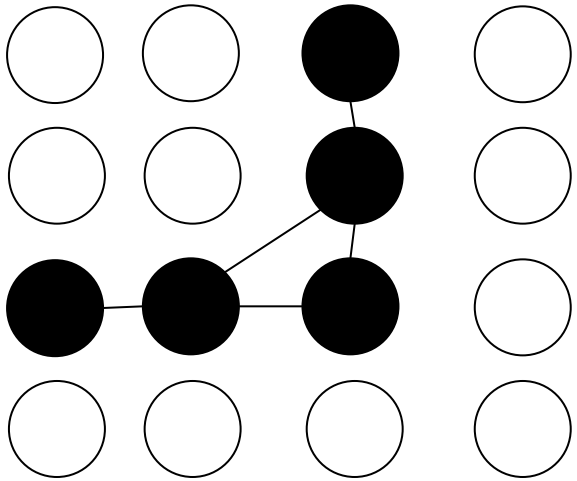
Three components

Foreground points with similar configuration!

Component labeling

- Form a graph with edges between neighboring foreground pixels.
- Compute connected components.
  - Graph traversal algorithms
    - Storage inefficient for large components

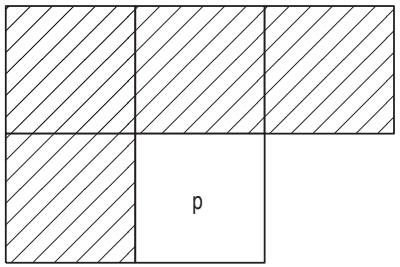
$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



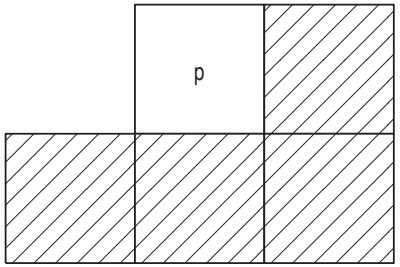
J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Two Scan Labeling

- Use of two masks for
  - Forward scanning: Top to bottom and left to right
  - Backward Scanning: Bottom to top and right to left



(a)



(b)

Chamfering!

For 8-adjacent connected component

  - Check the labels (a number) of neighbors at a foreground point p
  - If <no label> found, create new unique label
  - Else assign the label of a neighbor.
    - For multiple labels declare them equivalent and assign the lower numeral id.

Distance Transforms (DT)

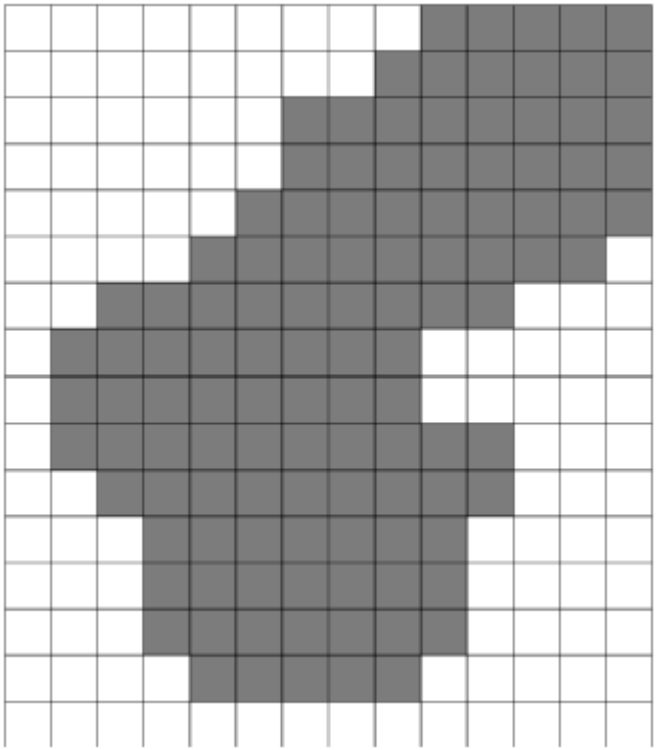
- Distance Transform: At every foreground pixel p (i.e. f(p)=1), it stores the distance of the pixel from the nearest background (f(.)=0) pixel.

- For all pixels p of an image S, the DT algorithm computes:

$$t(p) = \min_k \{d(p, q_k): S(q_k) = 0, 1 \leq k \leq |S|\}$$

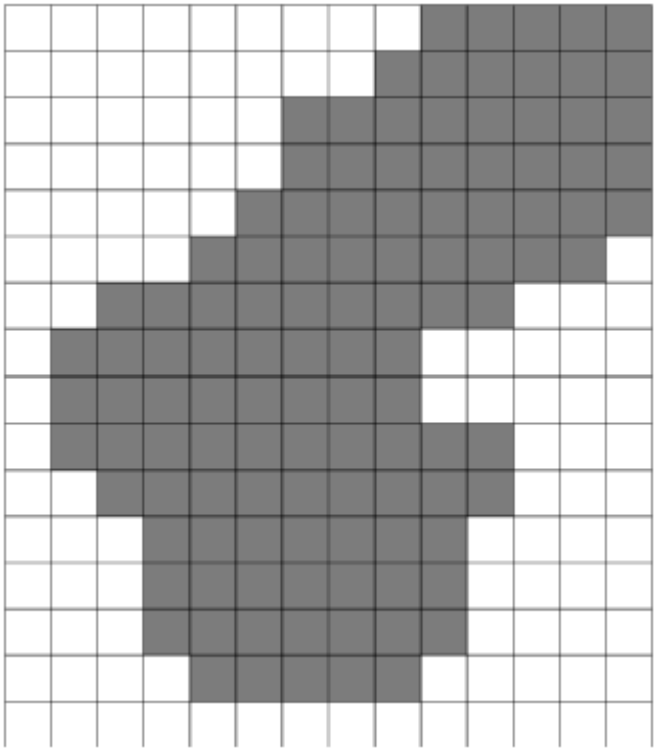
- DT(S): Image of t(p)'s, called the DT image.

Distance Transforms for Binary Images (d_4)



								1	1	1	1	1	
							1	2	2	2	2	1	
						1	1	2	3	3	3	2	1
						1	2	3	4	4	3	2	1
					1	2	3	4	3	3	2	2	1
				1	2	3	4	3	2	2	1	1	
		1	1	2	3	4	3	2	1	1			
	1	2	2	3	4	3	2	1					
	1	2	3	4	4	3	2	1					
	1	2	3	4	5	4	3	2	1	1			
		1	2	3	4	5	4	3	2	1			
			1	2	3	4	3	2	1				
			1	2	3	3	3	2	1				
			1	2	2	2	2	2	1				
				1	1	1	1	1					

Distance Transforms for Binary Images (d_8)



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

Courtesy: R.C. Gonzalez and R.E Woods © 1992-2008

Distance Transforms Algorithms

- Brute Force Algorithm
  - For every pixel in foreground F compute the distance to every pixel in background B and find the minima
  - Let the image be of size n
  - Let f = |F|, b = |B|, f + b = n
  - Often f = O(n) and b = O(n)
  - Number of distance computations = f.b = $O(n^2)$

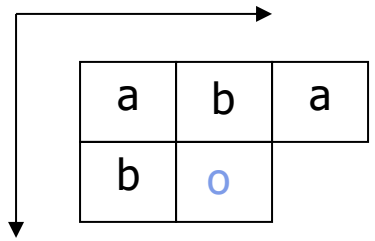
We need a better algorithm

J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

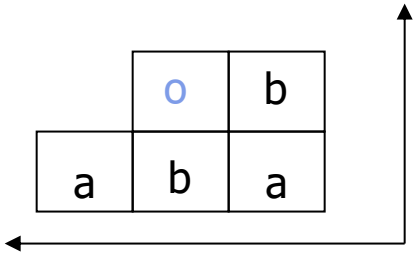
Chamfering Algorithm for Distance Transform

- Initialize all DT values to infinity.
- Perform two scans: Raster & Reverse Raster
- In every scan Chamfer with a mask
- At every pixel position, add the mask weight with the pixel covered and take the minimum

DT(o) = min (DT(Neighboring pixel) + local distance between them)



Forward Scanning
From Left to Right
and Top to Bottom



Backward Scanning
From Right to Left
and Bottom to Top

Computes DT for 'additive' distances

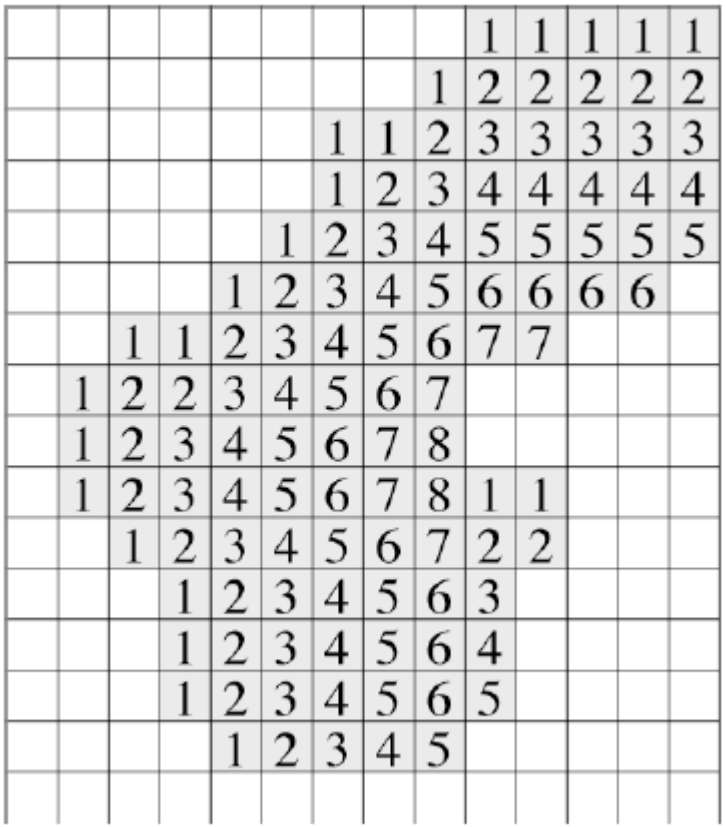
b = 1, a = 2: City Block

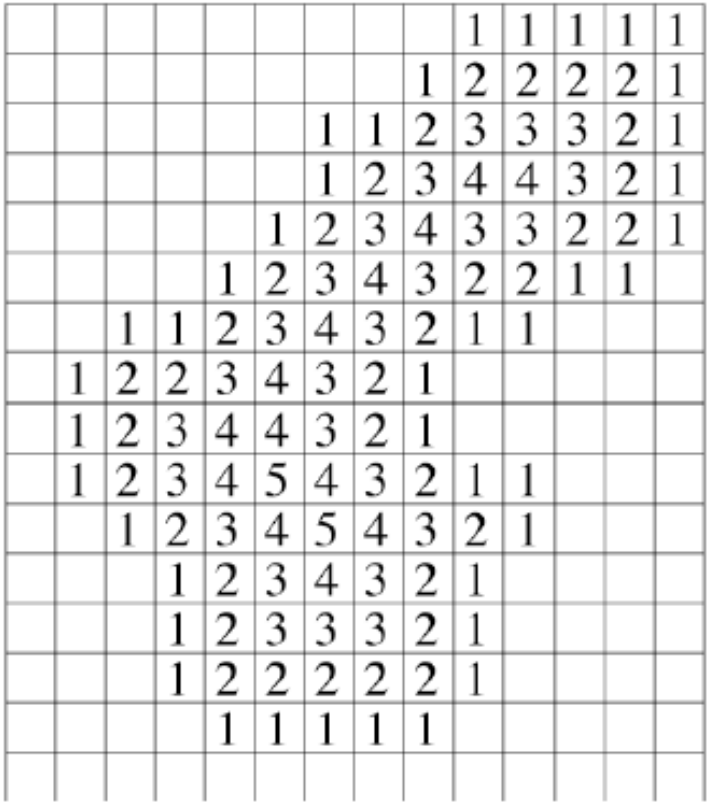
b = 1, a = 1: Chess Board

b = 1, a = √2: Weighted

Complexity = O(n)

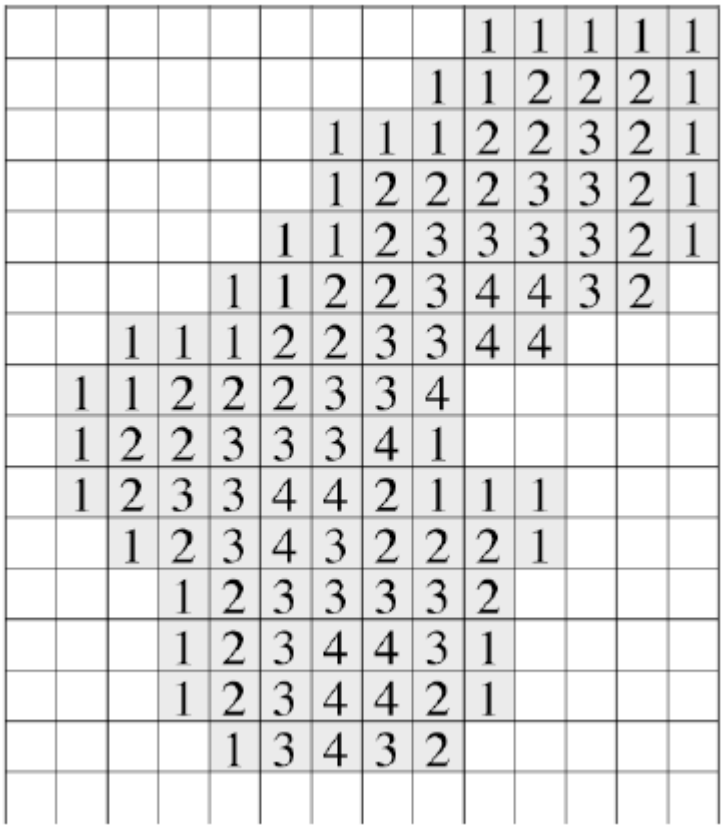
Two pass chamfering algorithm to compute DT with d_4

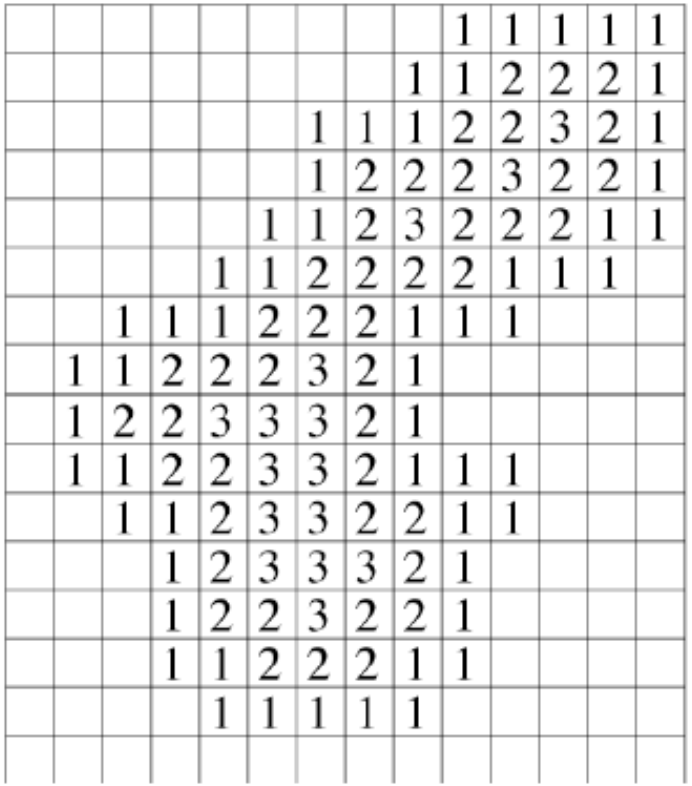




Courtesy: R.C. Gonzalez and R.E Woods © 1992-2008

Two pass chamfering algorithm to compute DT with d_8





Courtesy: R.C. Gonzalez and R.E Woods © 1992-2008

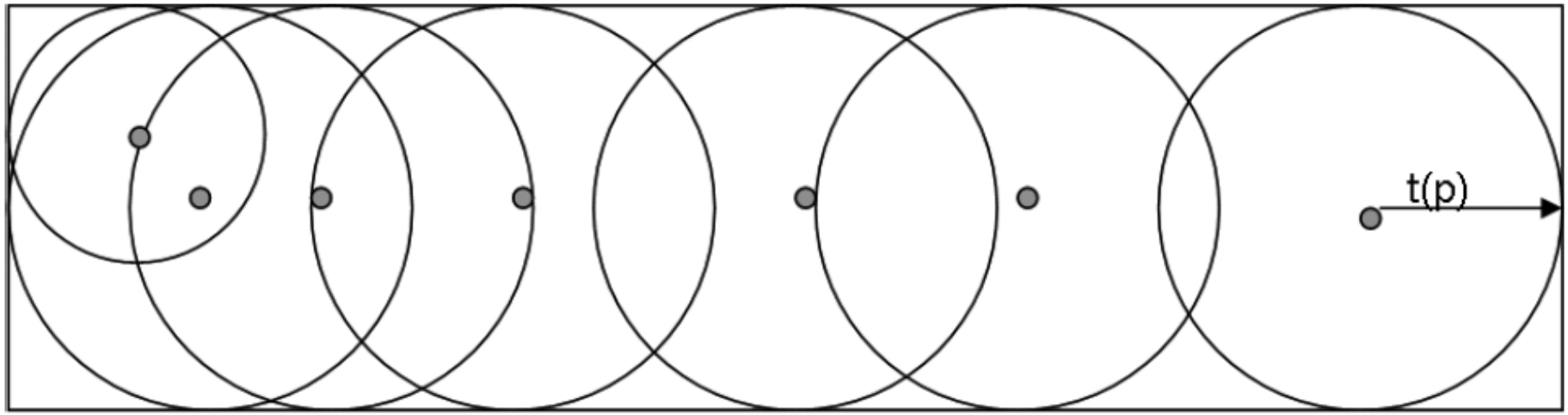
Distance Transform: An example

Input image

Output

DT: Properties -1

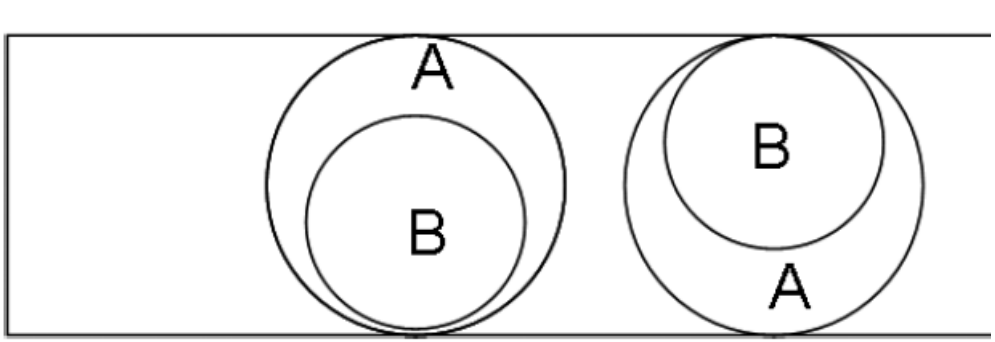
- [1] t(p) represents the radius of the largest disk centered at p and totally contained in F



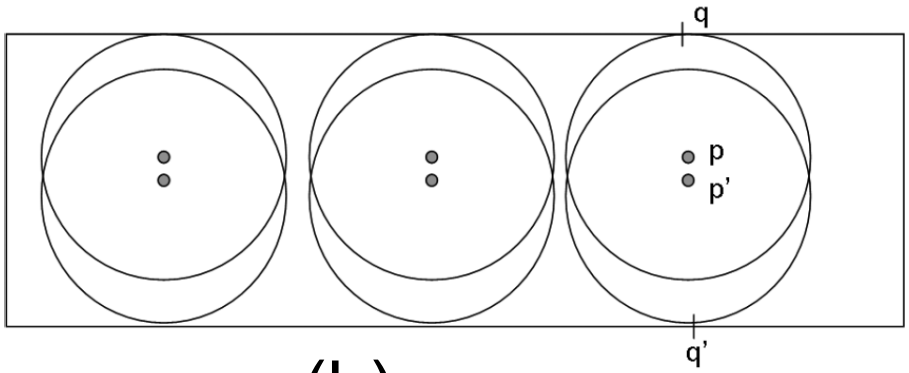
Courtesy: R.C. Gonzalez and R.E Woods © 1992-2008

DT: Properties -2

- If there is only one q ε B with t(p) = d(p, q):
  - (a) an element p' ε F exists such that the disk centered at p' totally contains the disk centered at p (disk B is totally included in disk A)
  - (b) elements p' ε F and q' ε B exist such that d(p, q) = d(p', q') and p is adjacent to p'



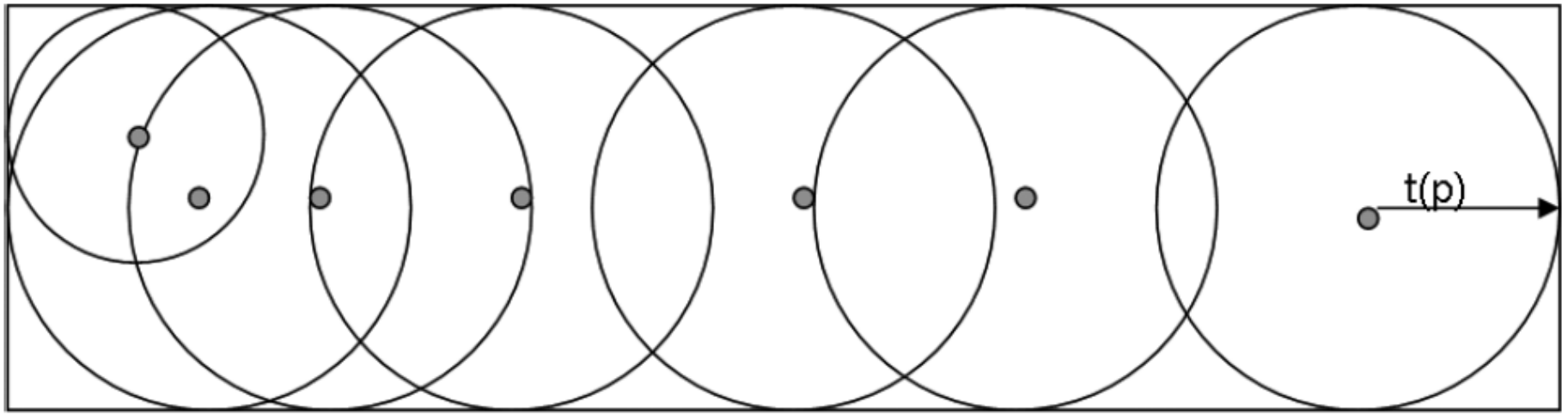
(a)



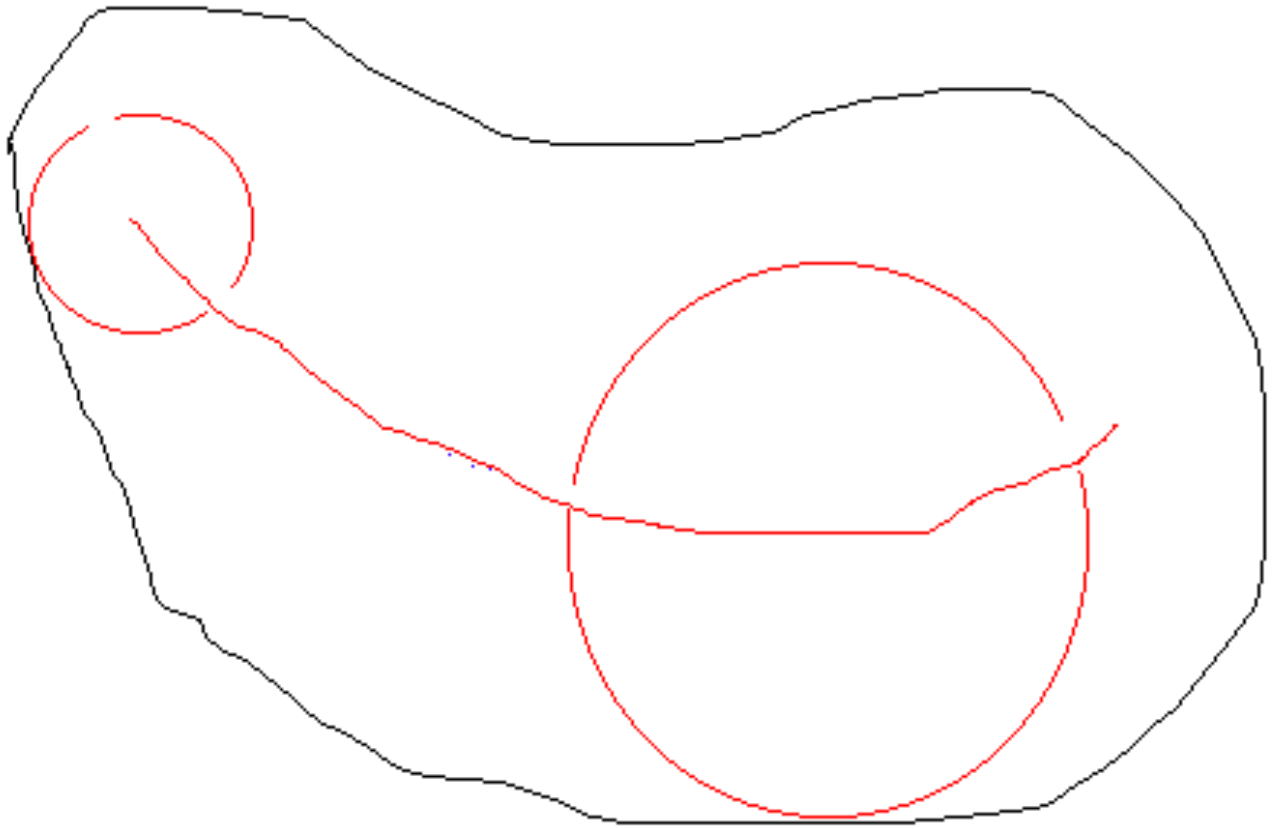
(b)

DT: Properties -3

- If there are two (or more) q, q' ε B such that t(p) = d(p, q) = d(p, q'), then the disk centered at p is a maximal disk in F and p is symmetric



Medial Axis Transform



A set of maximal blocks contained in the pattern.

Computation of Medial Axis Transform

- Compute the distance transform.
- Compute local maxima in the distance transformed image.

Application of MAT in Image Analysis

- Geometric Transformation (Kumar et al '96)
- Computation of Normals (Mukherjee et al '02)
- Thinning of binary pattern (Costa '00, Pudney '98)
- Computation of cross-sections of 3D objects (Mukherjee et al '00)
- Visualization of 3D objects (Mukherjee et al '99, Prevost and Lucas '00)
- Image compression (Kumar et al '95).
- Shape Description (Baja and Svensson '02)

Thinning from Distance Transform

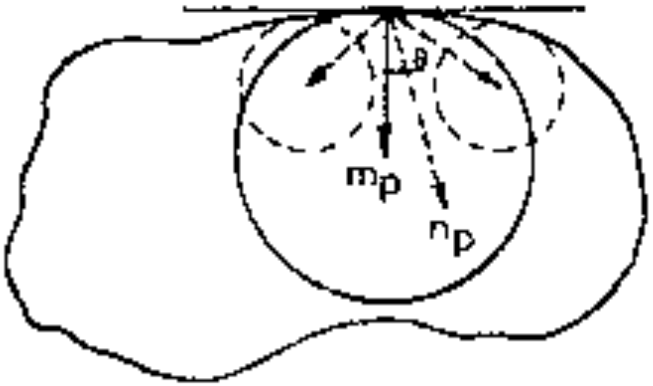
Compute the set of Maximal Blocks.

Use them as anchor points while iteratively deleting boundary points preserving the topology.

Vincent ’91, Ragnemalm ’93, Svensson-Borgefors-Nystrom ’99

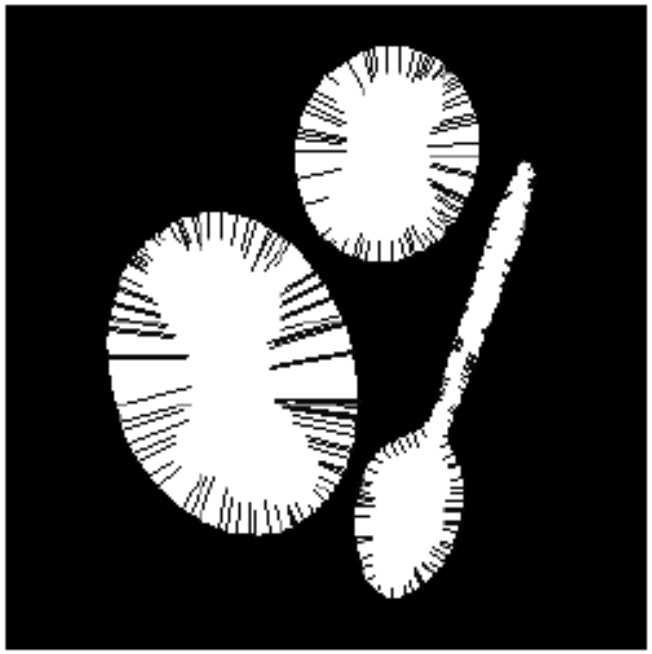
Normal Computation

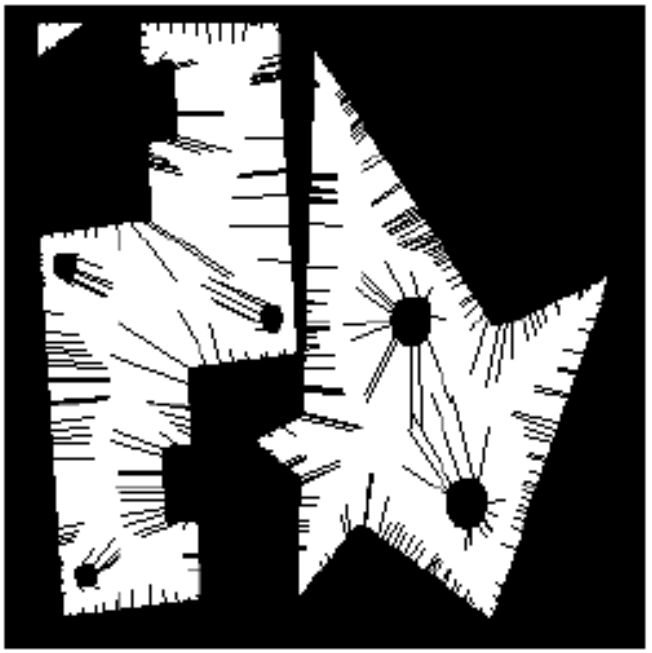
- Normal at a point p computed by computing the resultant vector from that point to the neighboring medial circles.



J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

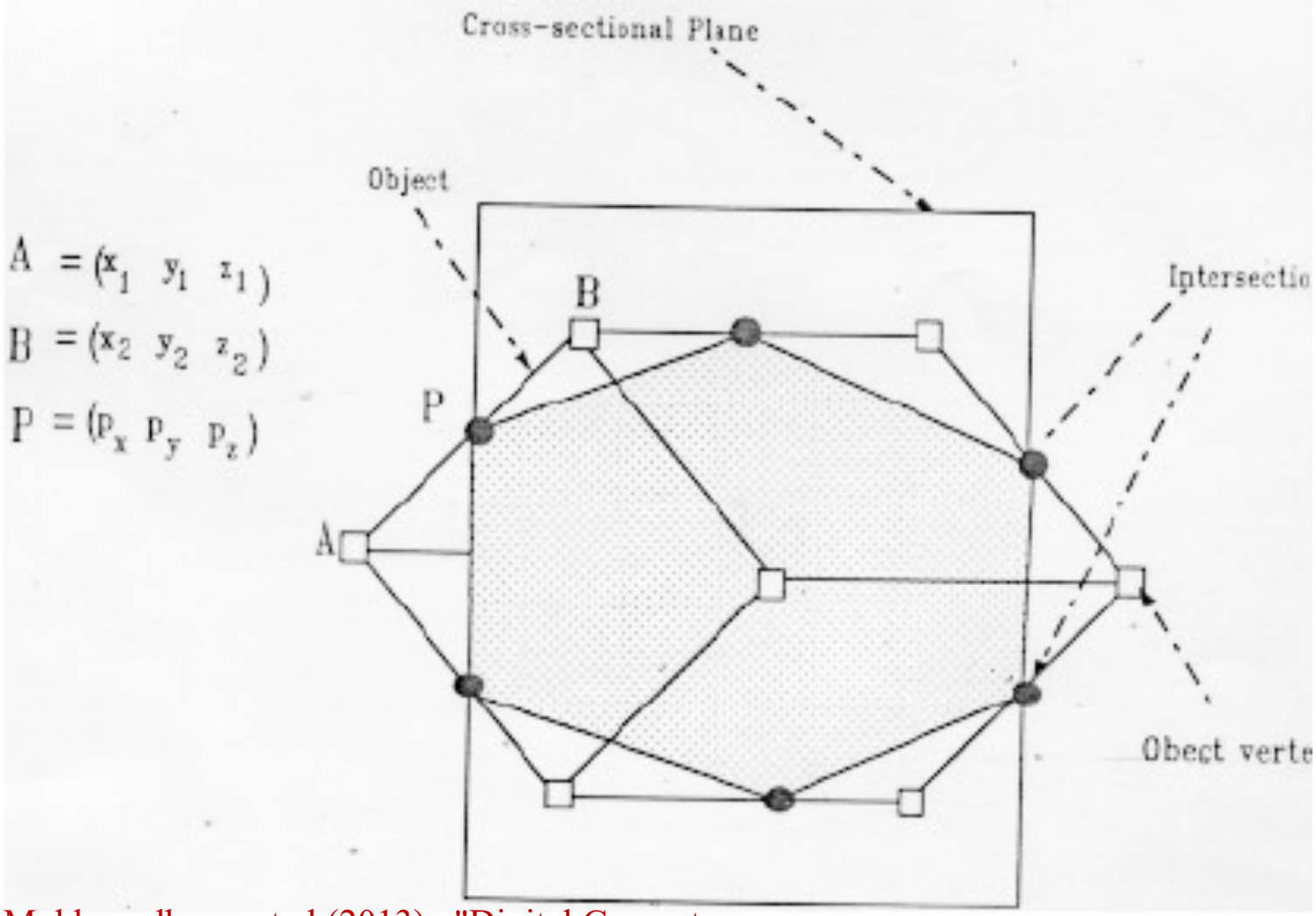
Normal Computations: Examples





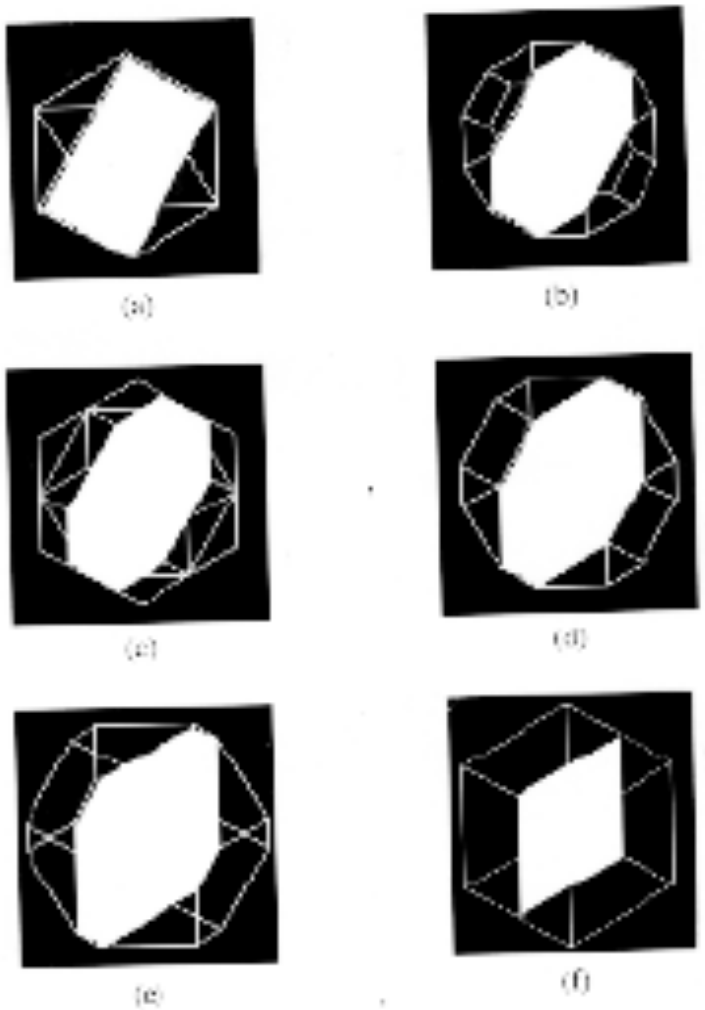
J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Cross-sectioning



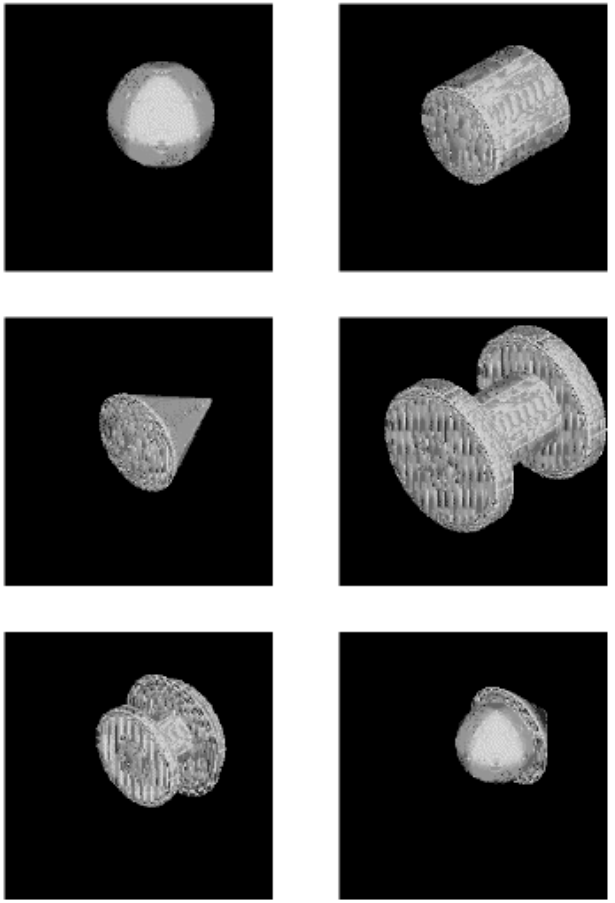
J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Cross-sectioning with different distance functions.



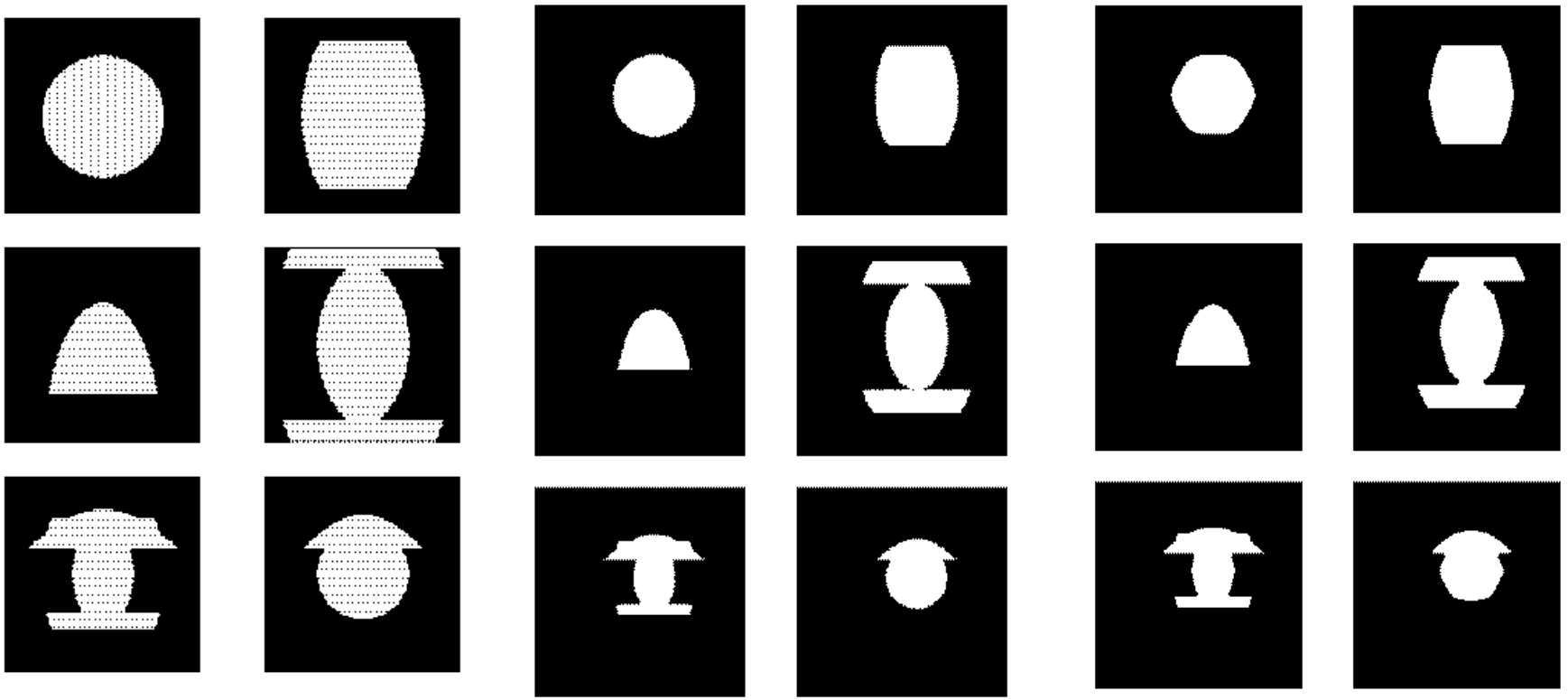
J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

A set of objects for experimentation



J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Cross-sectioning: Voxel data, MAT & Sphere Approx.



Voxel Data

MAT

Euclidean Sphere Approximation

J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Shape representation in 2D

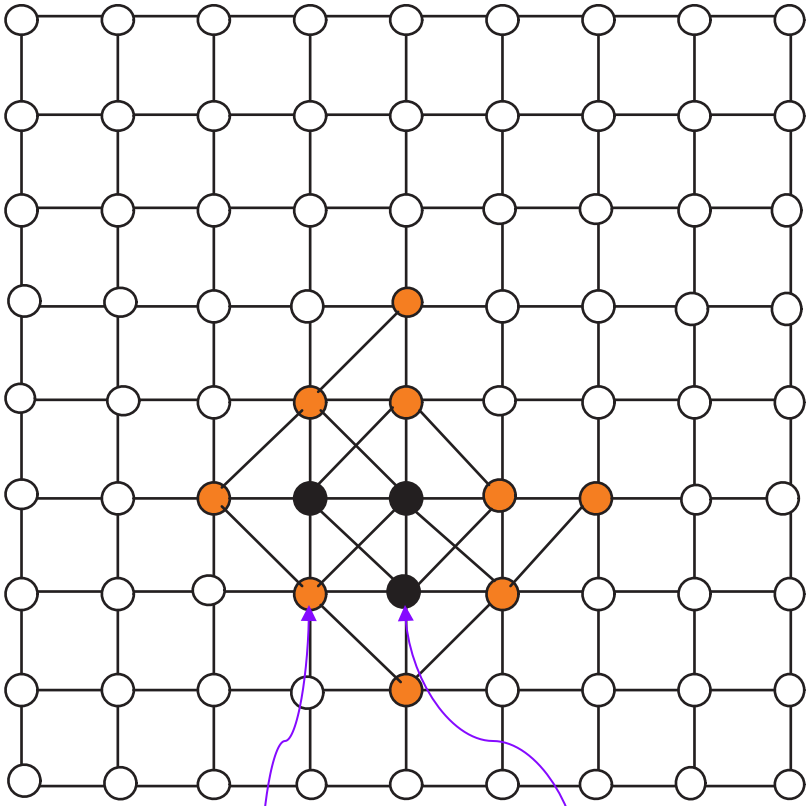
- Boundary (Border) Following
- Chain Codes
- Polygonal Approximation using Minimum-Perimeter Polygons (MPP)
- Polygonal Approximation by Merge/Split
- Signatures
- Boundary Segments
- Skeletons

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Boundary (Contour) Following

- Boundary
  - Good representation of an object shape
  - Requires less memory.
- Input:
  - Border points
    - A foreground (object) point which is m-adjacent to a background point in (m, n) grid.
- Output:
  - Ordered sequence of these points



(8,4) Grid

T. Pavlidis. Algorithms for graphics and image processing. Computer Science Press, Rockville, MD, 1982.

Contour following algorithm

- Start from the leftmost and topmost border point p with its left neighboring background point q. Mark p visited.
- Perform clockwise search among 8-neighbors of q to get the first foreground point *except the preceding border point* .
  - If it is already visited
    - stop and output the sequence.
  - Else
    - include it in the sequence,
    - mark it visited and
    - continue from the recently visited point in the same manner starting from the previous border point but excluding it from consideration .

Assumption: Simple contour (non intersecting)

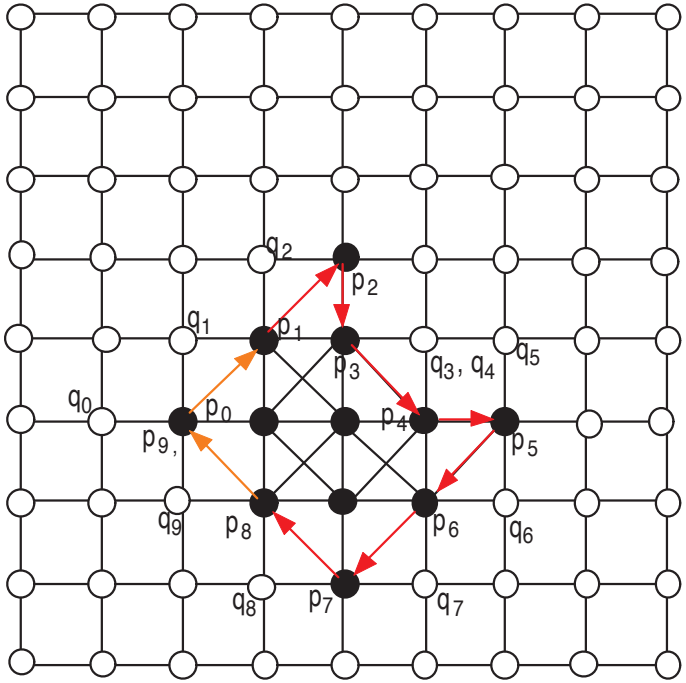
For anti-clockwise contour, search in the anti-clockwise order.

Search order and sequence: An example

r_1	r_2	r_3
$q=r_0$	p	r_4
r_7	r_6	r_5

(a)

The order of searching a foreground pixel in the neighborhood of a border pixel p with a background neighbor at q in an (8 , 4) digital grid.



(b)

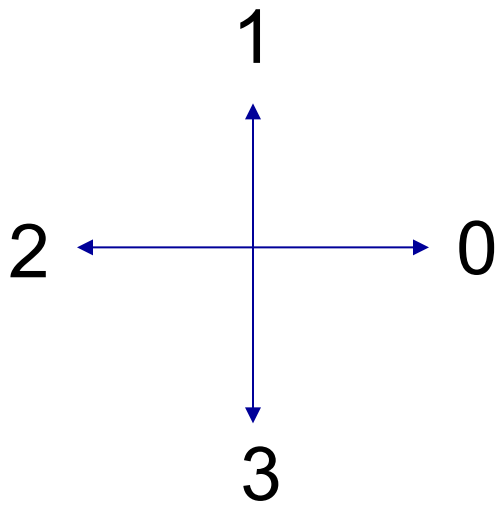
The order follows clockwise movement starting from q

The sequence of pairs of border pixels (p_i ; q_i), where p_i belongs to foreground, and q_i belongs to background, respectively, for the point set

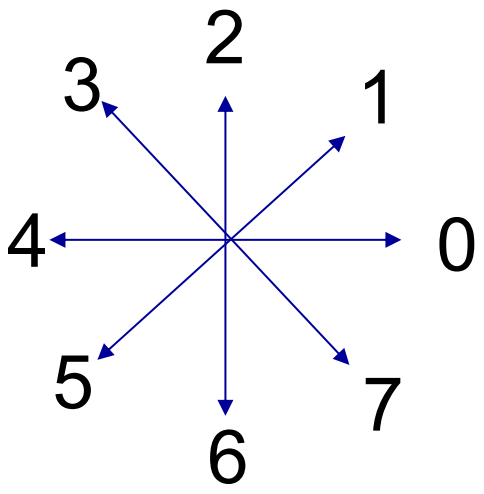
J. Mukhopadhyay, et al (2013), "Digital Geometry in Image Processing", CRC Press, April.

Chain Codes

- Representing a boundary of a connected region
- Based on 4 or 8 connectivity of the segments
- For every point code the direction of the next point



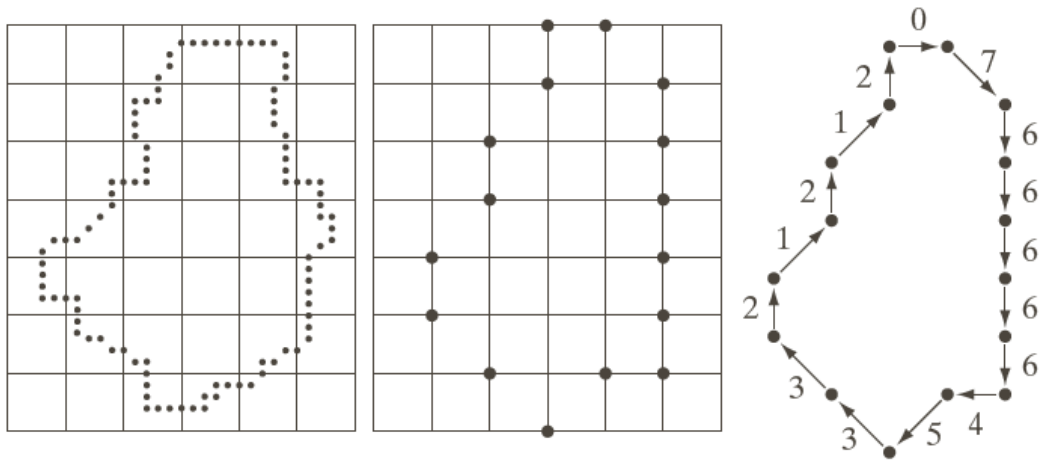
4-Directional Code



8-Directional Code

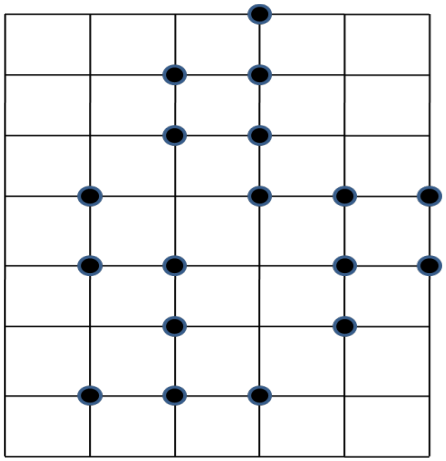
Need for resampling

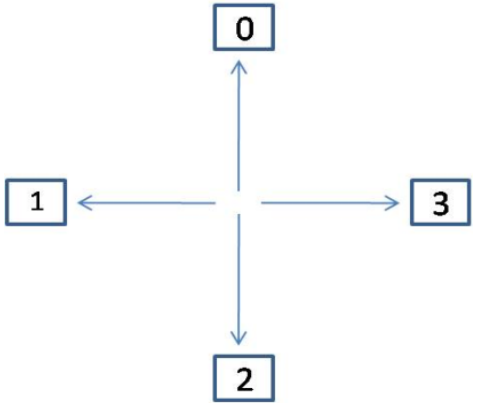
- The chain code of the same length as the perimeter of the object, in many cases too long
- Hence, re-sample the image to a lower resolution before calculating the code
- The re-sampling also reduces noise sensitivity

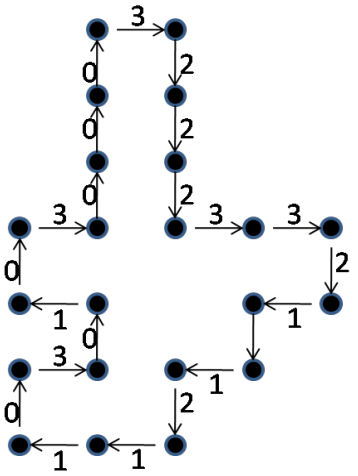


Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

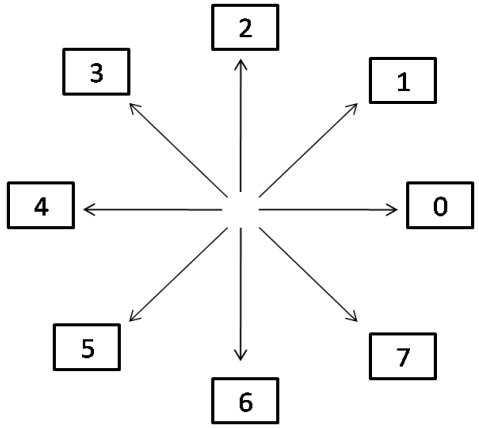
Chain codes: Examples

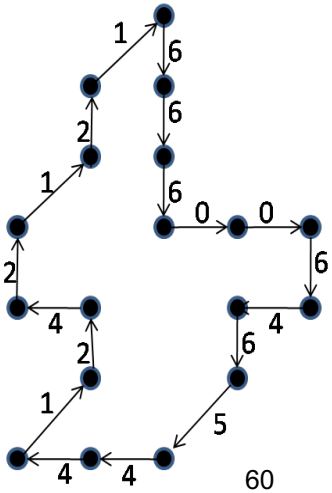






4-connected or
8-connected.





Chain codes: various issues

- Starting Point (various options)
  - Topmost, Leftmost point
  - Context dependent choice
  - Normalize by
    - Assuming the code to be circular (closed curve) and
    - Choose the integer of minimum magnitude
- Rotation Invariance
  - Use first difference of the chain code
    - Obtained by counting the number of direction changes (in a counterclockwise direction)
    - For example, the first difference of the 4-direction chain code 10103322 is 3133030.
  - Use circular shifting for minimal representation
- Size Normalization
  - Achieved by adjusting the size of the re-sampling grid.

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Polygonal approximation

- Polygonal approximations
  - Represent a boundary by straight line segments, and
  - A closed path becomes a polygon
- Number of straight line segments?
  - Accuracy of the approximation
  - Larger number of sides add noise
  - Small number of side result in coarse shapes
- Optimize for minimum number of sides
  - Constraint: Preserve the shape information

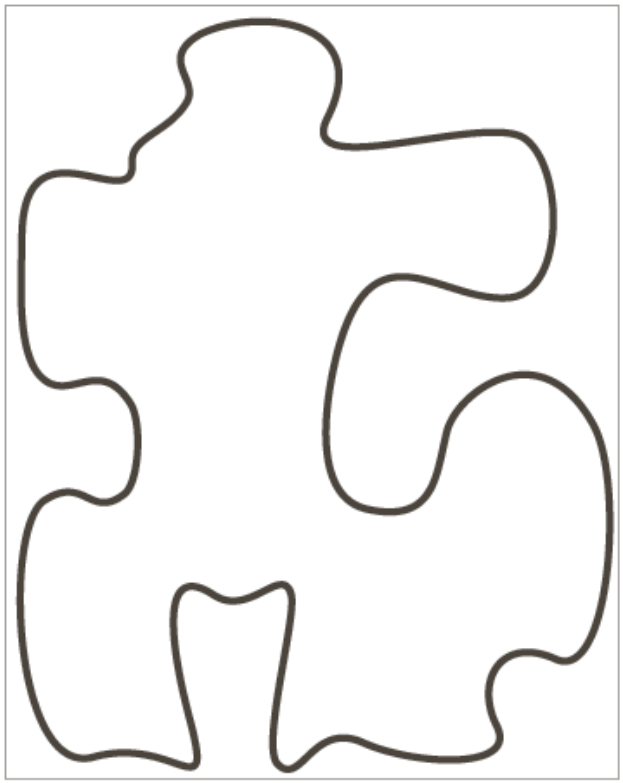
Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Polygonal Approximation using Minimum-Perimeter Polygons

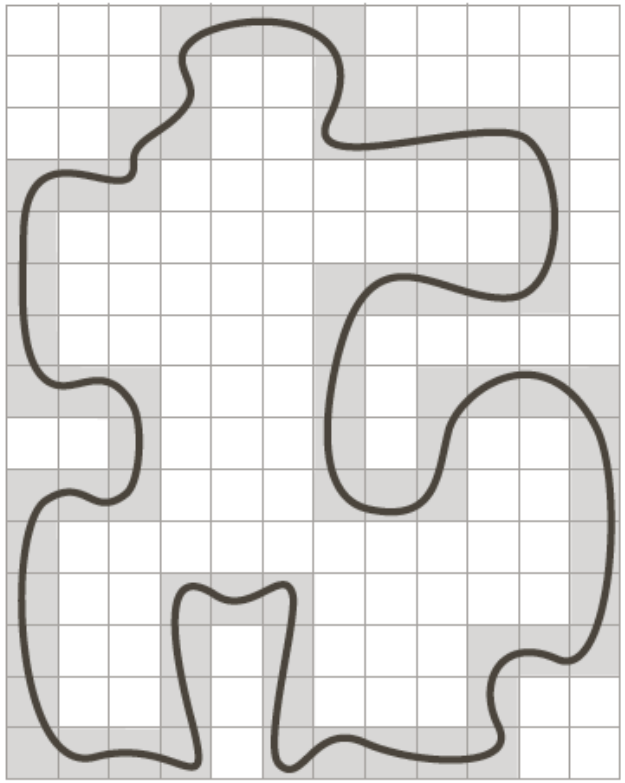
- Optimization involves Iterative Search
  - Computationally Expensive
- Use: Approximate Optimization
  - MPP: Minimum Perimeter Polygons

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

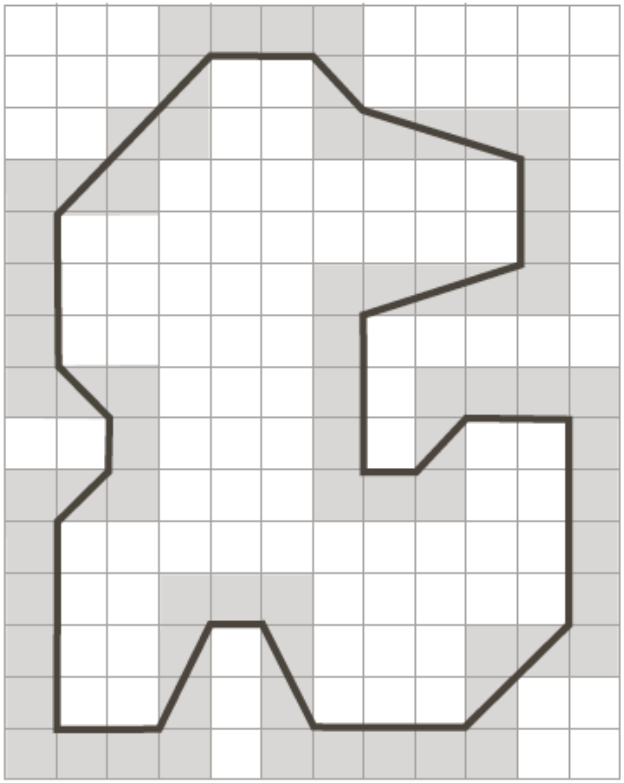
MPP Algorithm: intuition



Boundary of an object



In the discrete grid

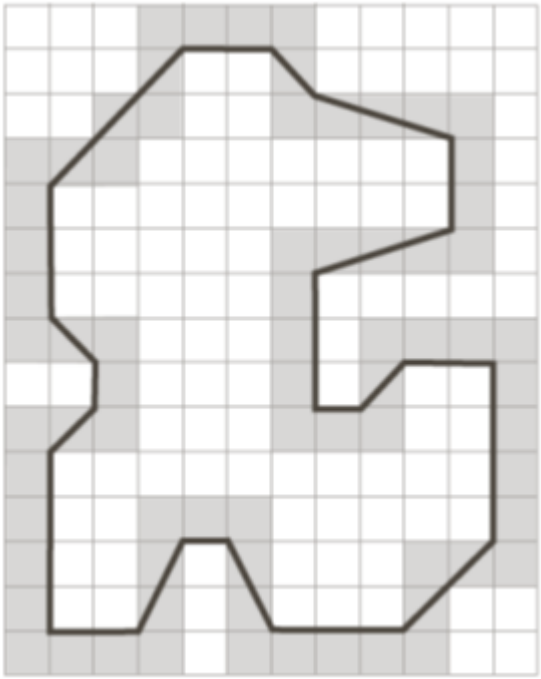


MPP: Vertices from inner and outer walls

Properties of MPP

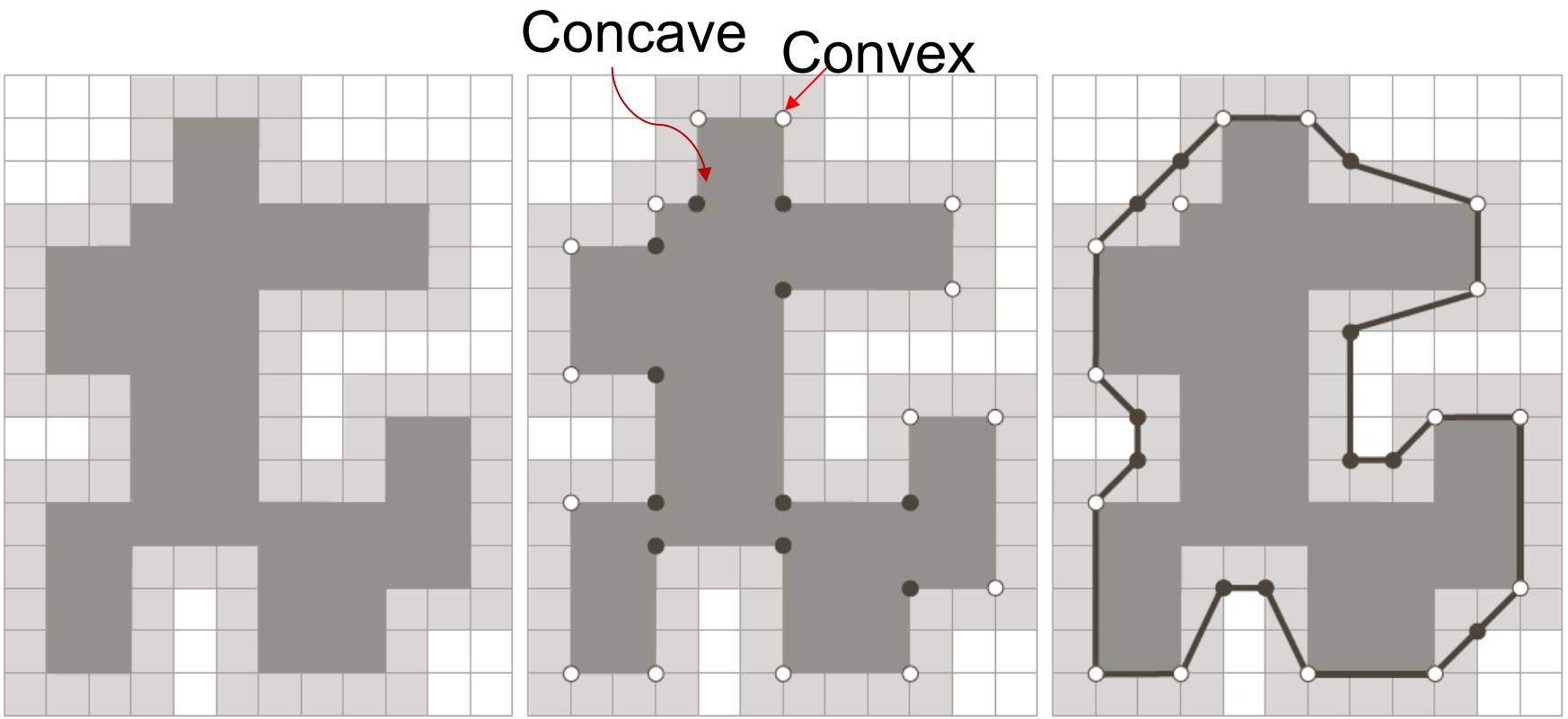
- Size of the Cell ➔
  - Accuracy of Polygonal Approximation
- If size of a cell (dXd)= size of a pixel in the boundary ➔
  - Max error = $\sqrt{2}d$
- Use largest possible cells ➔
  - Fewest number of vertices in MPP

Properties of MPP



- Boundary:
  - 4-connected
  - straight-line segments
- Mark counterclockwise
  - Convex (90 deg turn) Vertices: White
  - Concave (270 deg turn) Vertices: Black
    - Mirror Vertex for every Concave Vertex
- An MPP vertex is
  - Convex Vertex on Inner Wall or
  - Mirror of Concave Vertex on Outer Wall

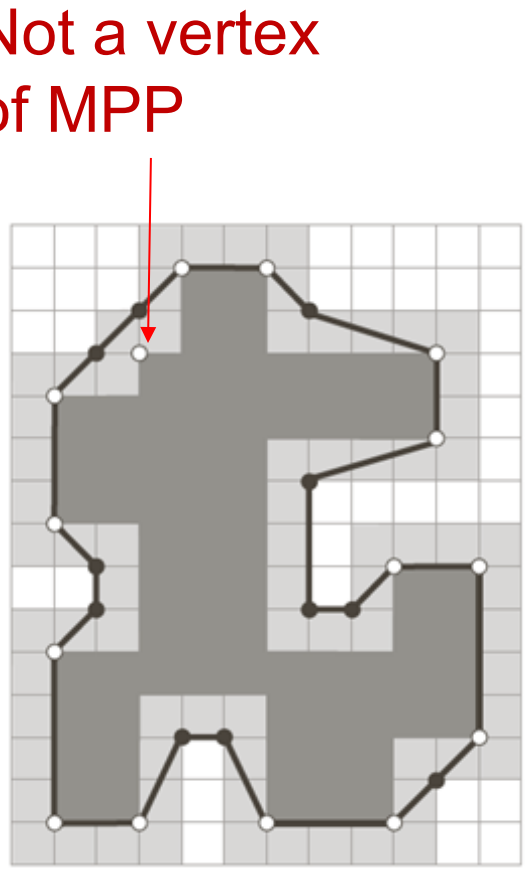
MPP Algorithm: Intuition



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Properties of MPP

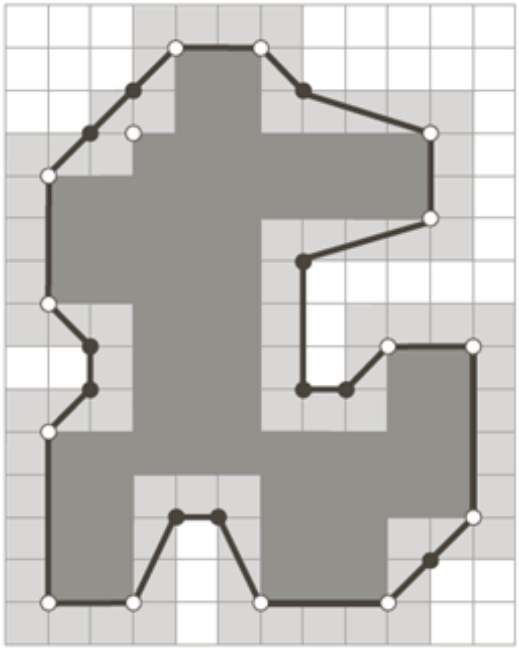
- MPP bounded by a simply connected cellular complex is not self-intersecting
- Every Convex vertex of MPP is a W vertex; but not every W vertex of a boundary is a vertex of MPP
- Every Mirrored Concave vertex of MPP is a B vertex; but not every B vertex of a boundary is a vertex of MPP



Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Properties of MPP

- All B vertices are on or outside MPP and all W vertices are on or inside MPP
- The uppermost, leftmost vertex in a sequence of vertices contained in a cellular complex is always a W vertex of MPP

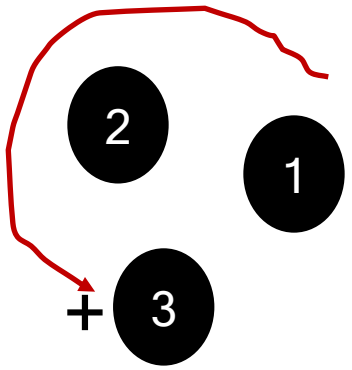


Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Orientation of 3 points

$$a = (x_1, y_1), b = (x_2, y_2), c = (x_3, y_3)$$

$$A = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}$$



$$\text{sgn}(a,b,c) = \det(A) =$$

$$\begin{cases} > 0, \text{if } (a,b,c) \text{ is a counterclockwise sequence} \\ = 0, \text{if the points are collinear} \\ < 0, \text{if } (a,b,c) \text{ is a clockwise sequence} \end{cases}$$

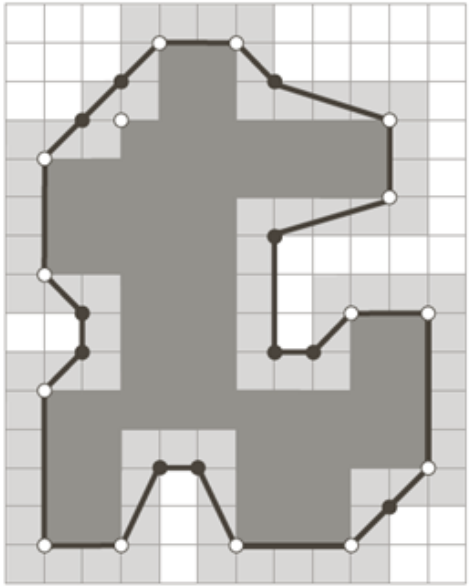
Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

MPP Algorithm

- Input: Vertices (with W/B Markers)
  - In a list in boundary order
- First vertex, V_0:
  - Uppermost, leftmost, a W vertex of MPP
- Crawler Vertices:
  - White crawler: W_C and Black crawler: B_C
- V_L: Last MPP vertex found
- V_k: Current vertex being examined

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

MPP Algorithm



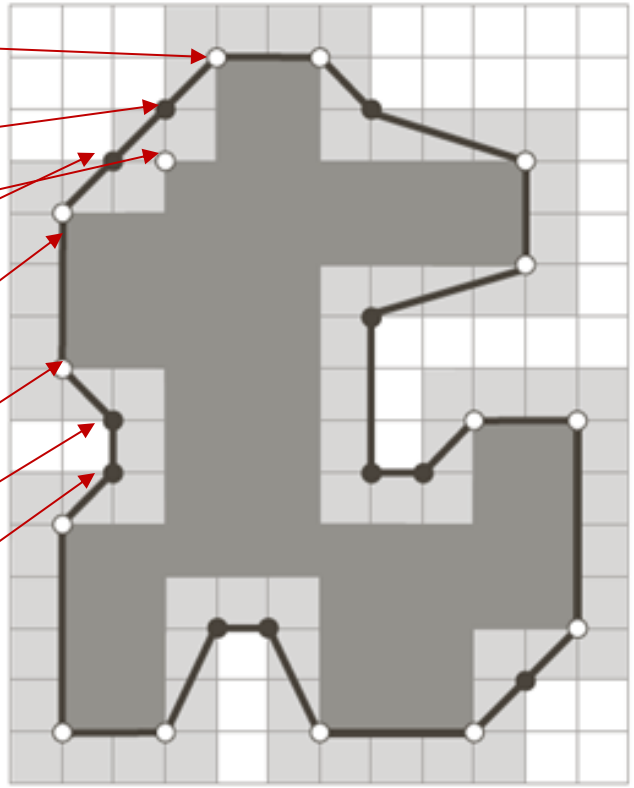
- Let $W_C = B_C = V_0$
- Repeat over the list
  - [A] $sgn(V_L, W_C, V_k) > 0$
    - $V_L \leftarrow W_C$ and $W_C = B_C = V_L$
    - Continue with the next vertex after V_L.
  - [B] $sgn(V_L, W_C, V_k) \leq 0$ & $sgn(V_L, B_C, V_k) \geq 0$
    - $W_C = V_k$, if V_k is convex (W)
    - $B_C = V_k$, otherwise
    - Continue with the next vertex after V_K.
  - [C] $sgn(V_L, B_C, V_k) < 0$
    - $V_L \leftarrow B_C$ and $W_C = B_C = V_L$
    - Continue with the next vertex after V_L.

Move either along W or B as it appears and record a MPP vertex if conditions met. Start again from the new MPP vertex.

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

MPP Algorithm

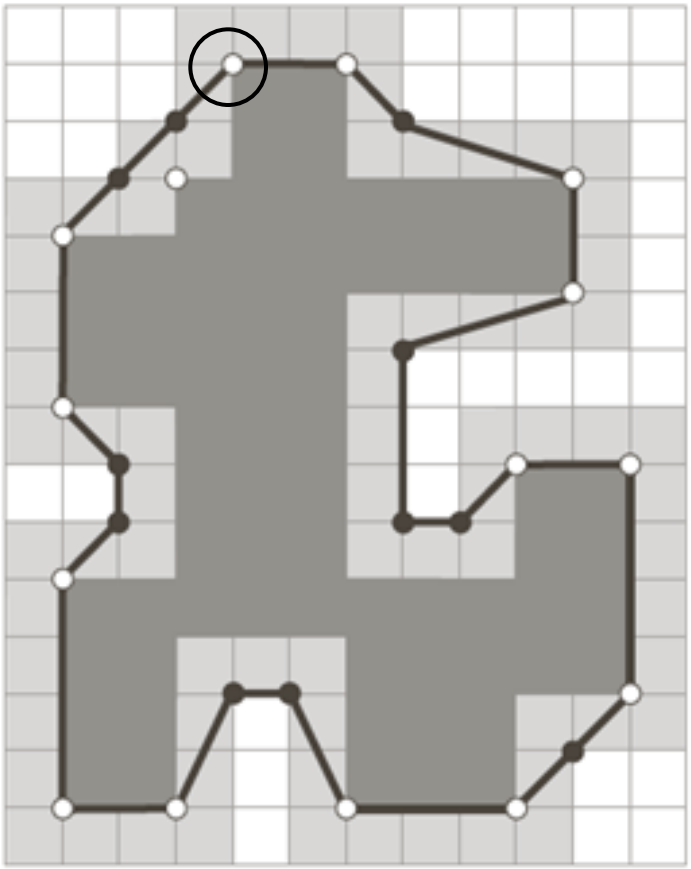
- Vertices:
  - V_0 (1,4) W
  - V_1 (2,3) B
  - V_2 (3,3) W
  - V_3 (3,2) B
  - V_4 (4,1) W
  - V_5 (7,1) W
  - V_6 (8,2) B
  - V_7 (9,2) B



Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

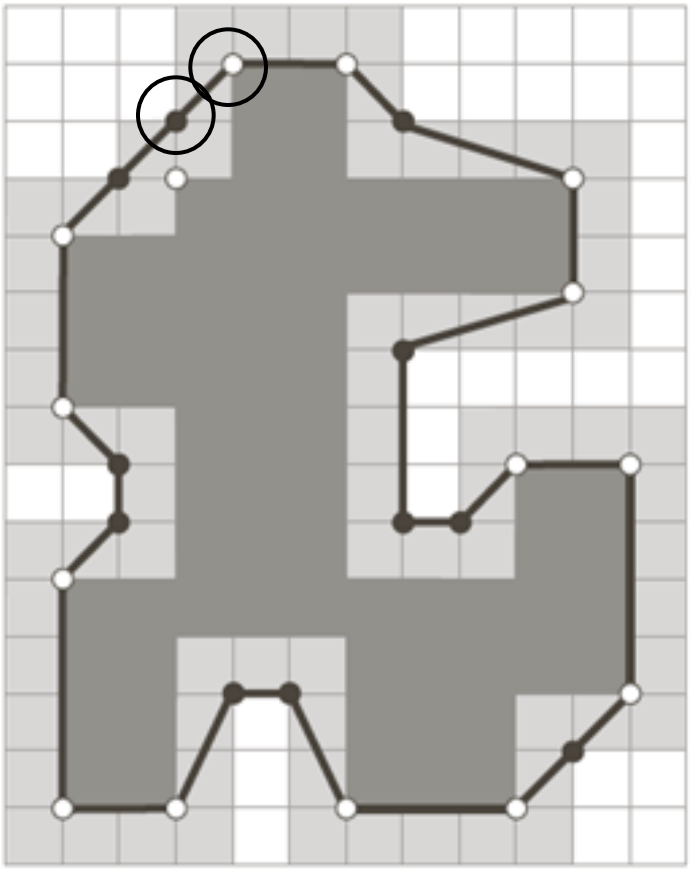
MPP Algorithm: Execution

- $W_C=B_C=V_0=V_L=(1,4)$
- $V_1=(2,4)$ W
- Condition [B]
  - $sgn(V_L,W_c,V_1)=0$ &
  - $sgn(V_L,B_c,V_1)=0$
- $W_C=V_1=(2,4)$
- Next vertex: $V_2=(3,4)$ B



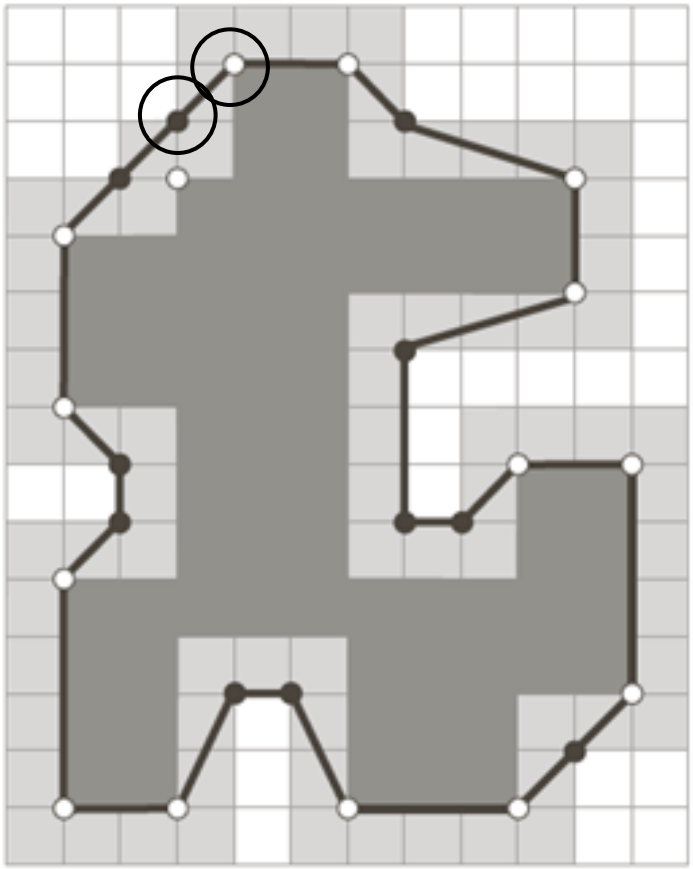
MPP Algorithm: Execution

- B_C=(1,4), W_C=(2,4)
- V_L=(1,4)
- V_2=(3,4) B
- Condition [B]
  - sgn(V_L,W_c,V_2)=0 &
  - sgn(V_L,B_c,V_2)=0
- B_C=V_2=(3,4)
- Next vertex V_3= (3,3) W



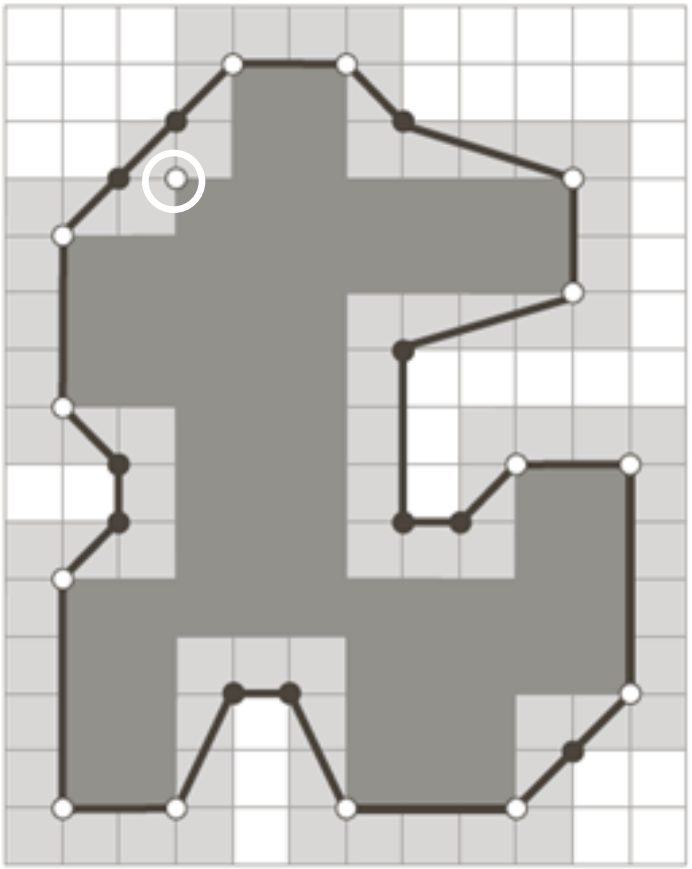
MPP Algorithm: Execution

- W_C=(2,4), B_C=(3,4)
- V_L=(1,4)
- V_3=(3,3) W
- Condition [C]
  - sgn(V_L,B_c,V_3) < 0
- V_L = (3,4) B_C
- W_C= B_C =(3,4)
- Next vertex V_4=(3,3)



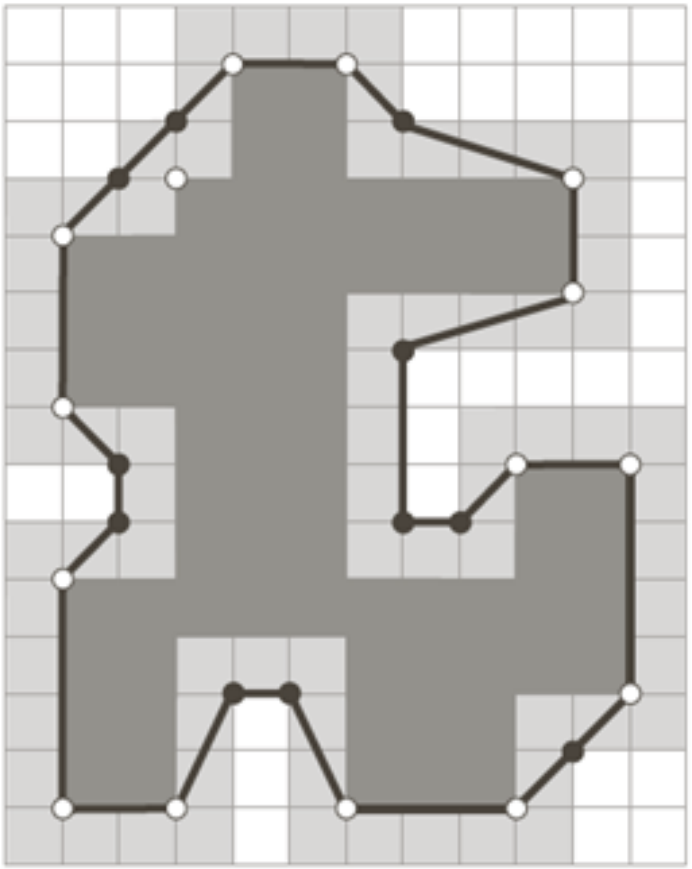
MPP Algorithm: Execution

- $W_C = B_C = (3,4)$
- $V_L = (3,4)$
- $V_4 = (3,3)$ W
- Condition [B]
  - $sgn(V_L, W_c, V_4) = 0$ &
  - $sgn(V_L, B_c, V_4) = 0$
- $W_C = V_4 = (3,3)$
- Next vertex $V_5 = (4,3)$ B



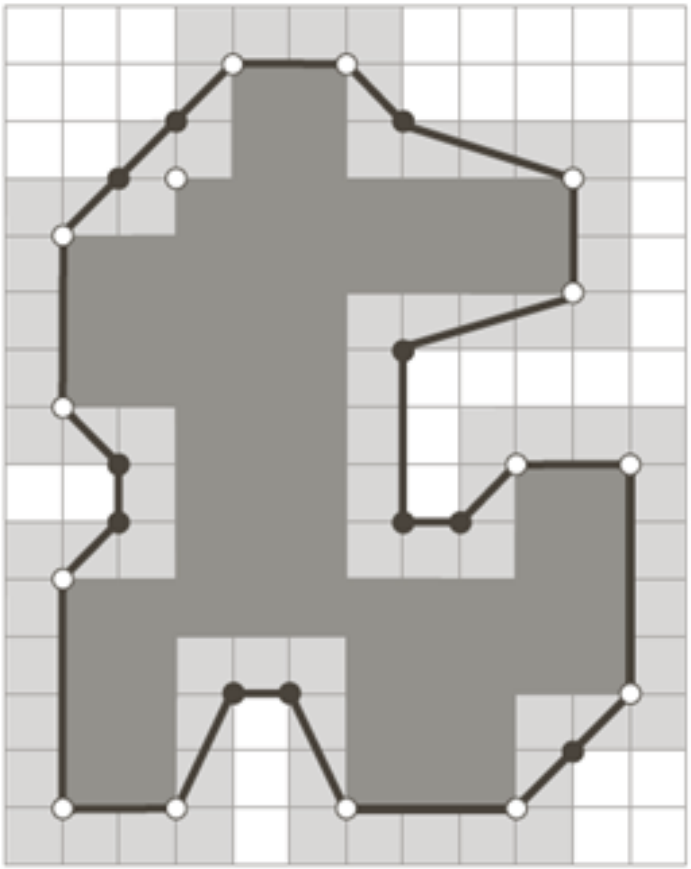
MPP Algorithm: Execution

- V_L=(3,4)
- W_C=(3,3)
- B_C=(3,4)
- V_5=(4,3) B
- Condition [A]
  - sgn(V_L,W_c,V_5) >0
    - $V_L = W_C = (3,3)$
    - $W_C = B_C = V_L$
    - Next vertex: V_6=(4,3) B



MPP Algorithm: Execution

- $W_C = B_C = V_L=(3,3)$
- $V_6=(4,3)$ B
- Condition [B]
  - $sgn(V_L,W_c,V_6)=0$
  - $sgn(V_L,B_c,V_6)=0$
  - $B_C=(4,3)$
  - Next vertex $V_7=(4,2)$ W

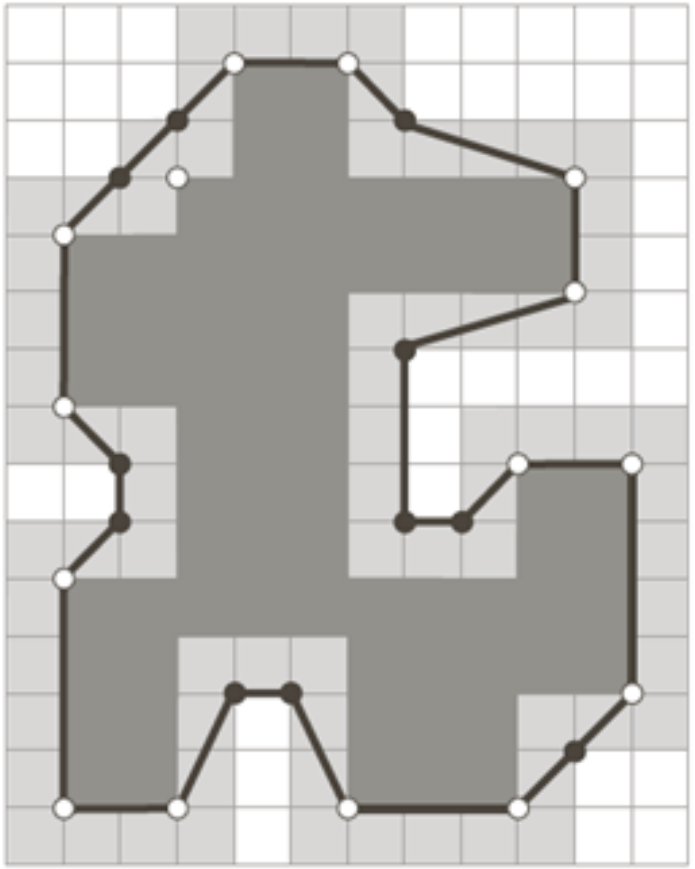


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

MPP Algorithm: Execution

- W_C=(3,3), B_C=(4,3)
- V_L=(3,3)
- V_7=(4,2) W
- Condition [C]
  - sgn(V_L,B_c,V_7) < 0
  - V_L= B_C=(4,3)
  - W_C = B_C = V_L
  - Next vertex V_8= (4,2)

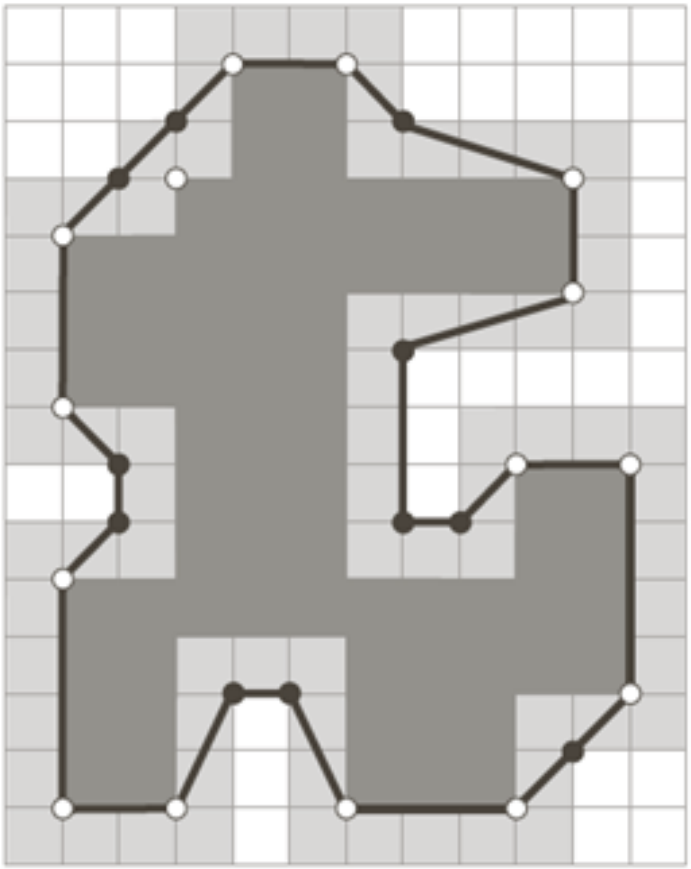


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

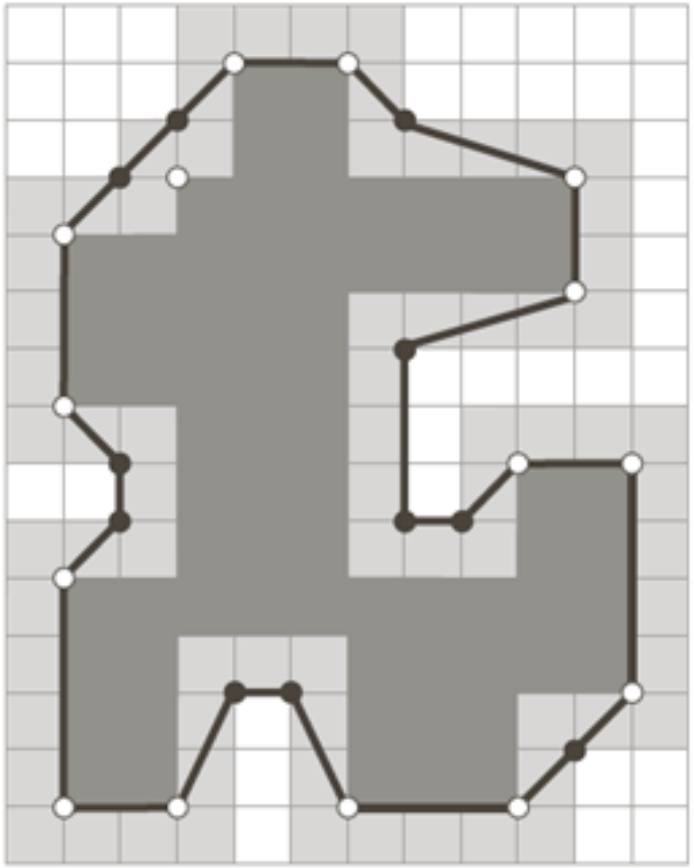
MPP Algorithm: Execution

- W_C=(4,3), B_C=(4,3)
- V_L=(4,3)
- V_8=(4,2) w
- Condition [B]
  - sgn(V_L,W_c,V_8)=0
  - sgn(V_L,B_c,V_8)=0
  - W_C=(4,2)
  - Next vertex V_9= (4,1)



MPP Algorithm: Execution

- W_C=(4,2), B_C=(4,3)
- V_L=(4,3)
- V_9=(4,1) W
- Condition [B]
  - sgn(V_L,W_c,V_9)= 0
  - sgn(V_L,B_c,V_9)= 0
  - W_C= (4,1)
  - Next vertex V_{10}= (5,1)

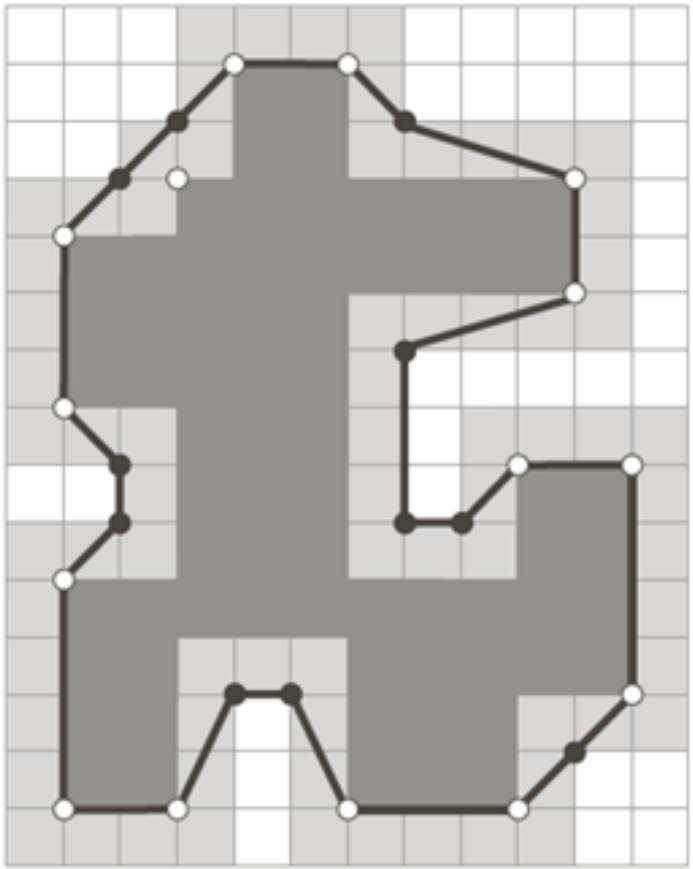


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

MPP Algorithm: Execution

- W_C=(4,1), B_C=(4,3)
- V_L=(4,3)
- V_9=(5,1) W
- Condition [A]
  - sgn(V_L,W_c,V_{10}) > 0
  - V_L = W_C= (4,1) W
  - W_C= B_C= (4,1)
  - Next vertex V_{11}= (5,1)
  - And so on ..

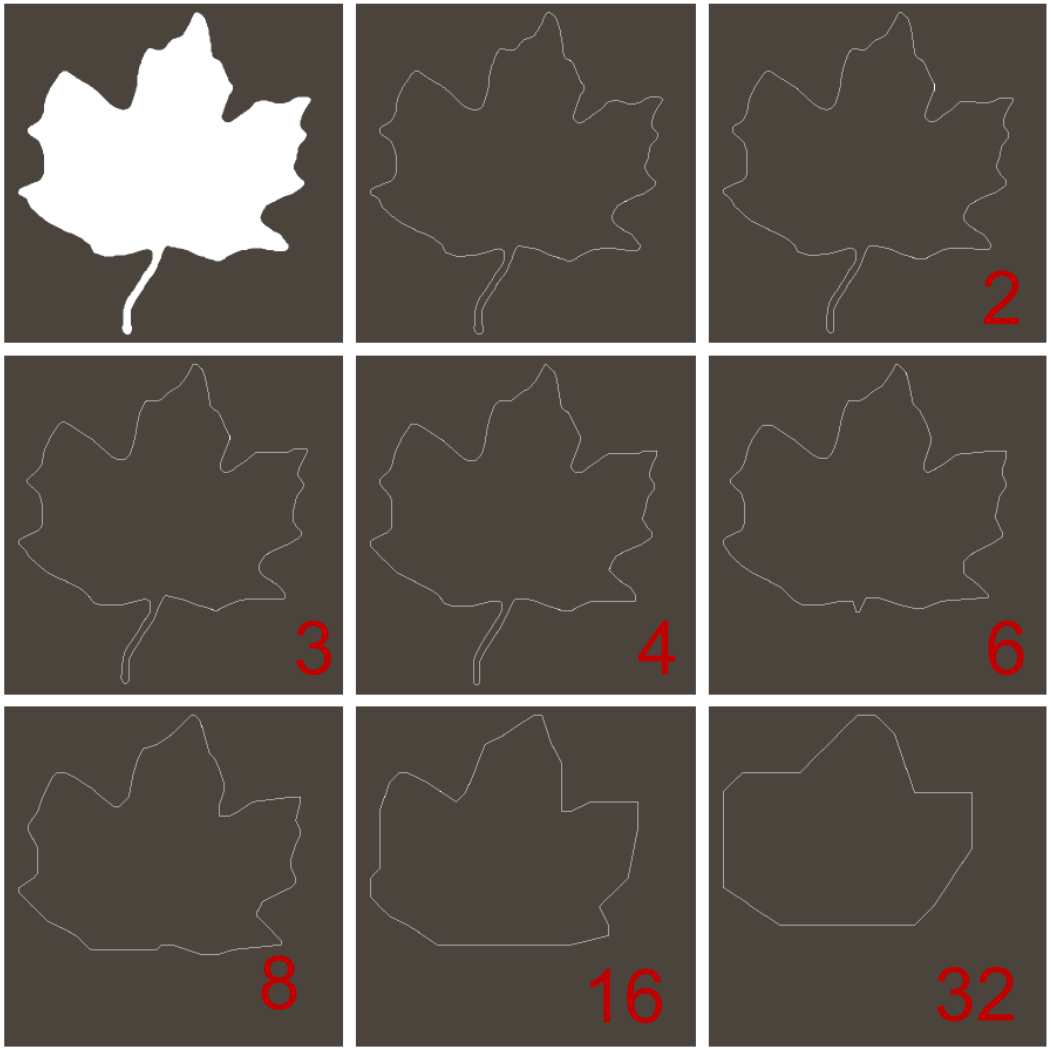


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

MPP : Example

- Image of size 566x566
- 8-connected boundary (1900 points)
- Of different cell size: 2, 3, 4, 6, 8, 16 & 32
  - # of Vertices:
    - 206, 160, 127,92, 66, 32 & 13.



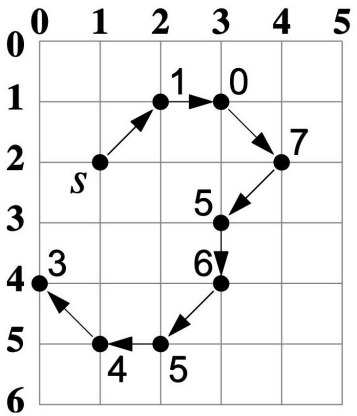
MPP : Example

Input image

Output

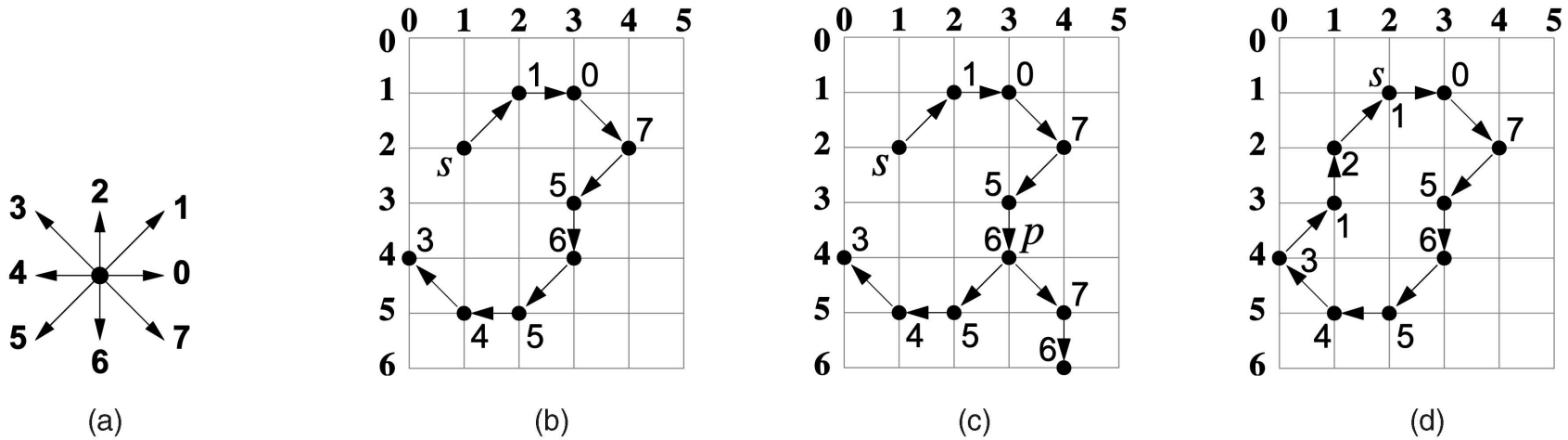
Digital curve and straight segments

- *Digital Curve (DC): A DC C is an ordered sequence of grid points (representable by chain codes) such that each point (excepting the first one) in C is a neighbor of its predecessor in the sequence.*
- *Irreducible Digital Curve: A DC C is said to be irreducible if and only if the removal of any grid point in C makes C disconnected.*



An example of DC and irreducible DC.

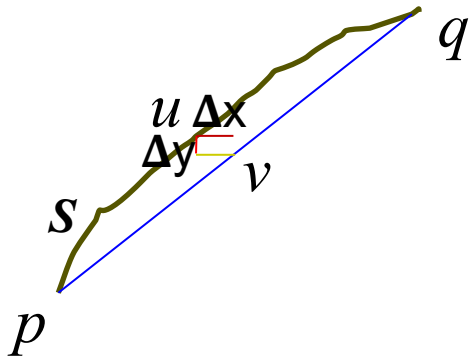
Chain code representations: Examples



Chain codes and their enumeration for defining DC. (a) Chain codes in 8-neighborhood connectivity. (b) (1, 2)10756543.
(c) (1, 2)10756543(3, 4)76. (d) (2, 1)0756543121

Digital Straight Line Segments (DSS)

- *Let **p**, **q** be points of the digital picture subset **S**, and let **pq** denote the (real) line segment between **p** and **q**.*
- ***pq** lies near **S** if, for any (real) point (x,y) of **pq**, there exists a (lattice) point (i,j) of S such that **max** $\{|i - x|, |j - y|\} < 1$.*
- ***S** has the **chord property** if, for every **p, q** in **S**, the chord **pq** lies near S..*



pq lies near **S** if for any v there exists u and such that $\max(\Delta x, \Delta y) \leq 1$.

AZRIEL ROSENFELD, Digital Straight Line Segments. IEEE TRANSACTIONS ON COMPUTERS, VOL. c-23, NO. 12, DECEMBER 1974, 1264-1268.

Digital Straight Line Segments (DSS)

- *A **digital straight line segment (DSS)** is the digitization of a straight line segment.*
- *A DSS is an irreducible DC.*
- *A DC is the **digitization of a straight line segment** if and only if it has the **chord property** .*

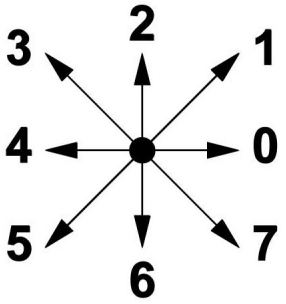
AZRIEL ROSENFELD, Digital Straight Line Segments. IEEE TRANSACTIONS ON COMPUTERS, VOL. c-23, NO. 12, DECEMBER 1974, 1264-1268.

DSS characterization

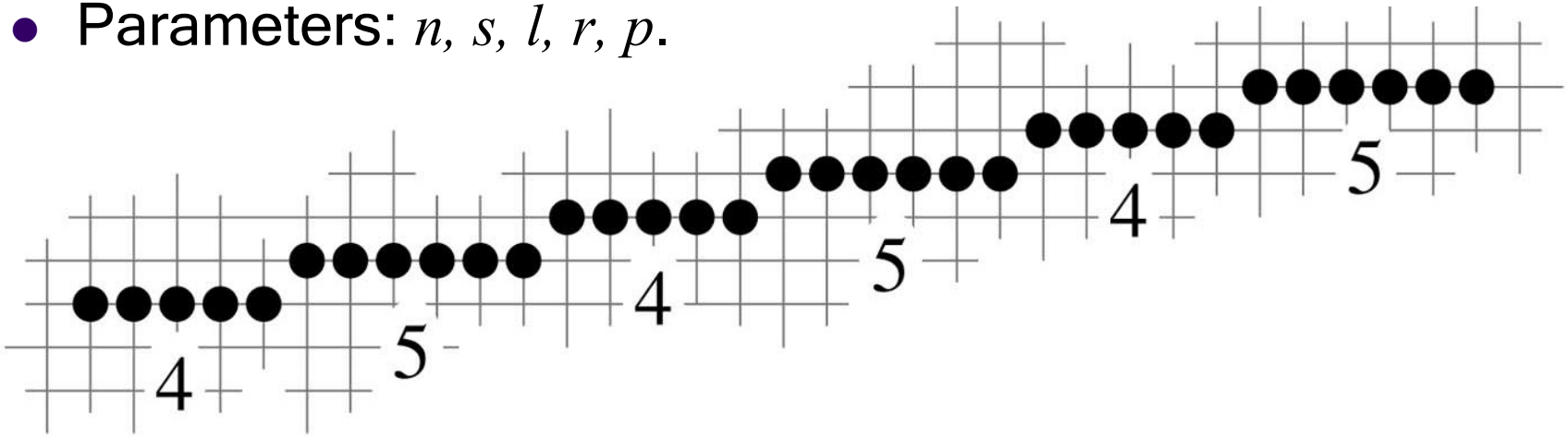
- **R1:** *The runs have at most two directions, differing by* 45 *degrees, and, for one of these directions, the run length must be* 1.
  - *At most two types of elements and they differ only by unity, modulo eight.*
  - *One of the two element values always occurs singly.*
- **R2:** *The runs can have only two lengths*: *consecutive integers.*
  - *Successive occurrences of the element occurring singly are as uniformly spaced as possible.*
- **R3:** *One of the run lengths can occur only once at a time.*
- **R4:** *For the run length that occurs in runs, these runs can themselves have only two lengths p and p+1, which are consecutive integers, and so on.*
  - *Except run lengths at two extreme ends (l and r, $l, r \leq (p+1)$)*

AZRIEL ROSENFELD, Digital Straight Line Segments. IEEE TRANSACTIONS ON COMPUTERS, VOL. c-23, NO. 12, DECEMBER 1974, 1264-1268.

An example of a DSS



- Singular element (s) : 1
- Non singular element (n): 0
- Parameters: n, s, l, r, p.



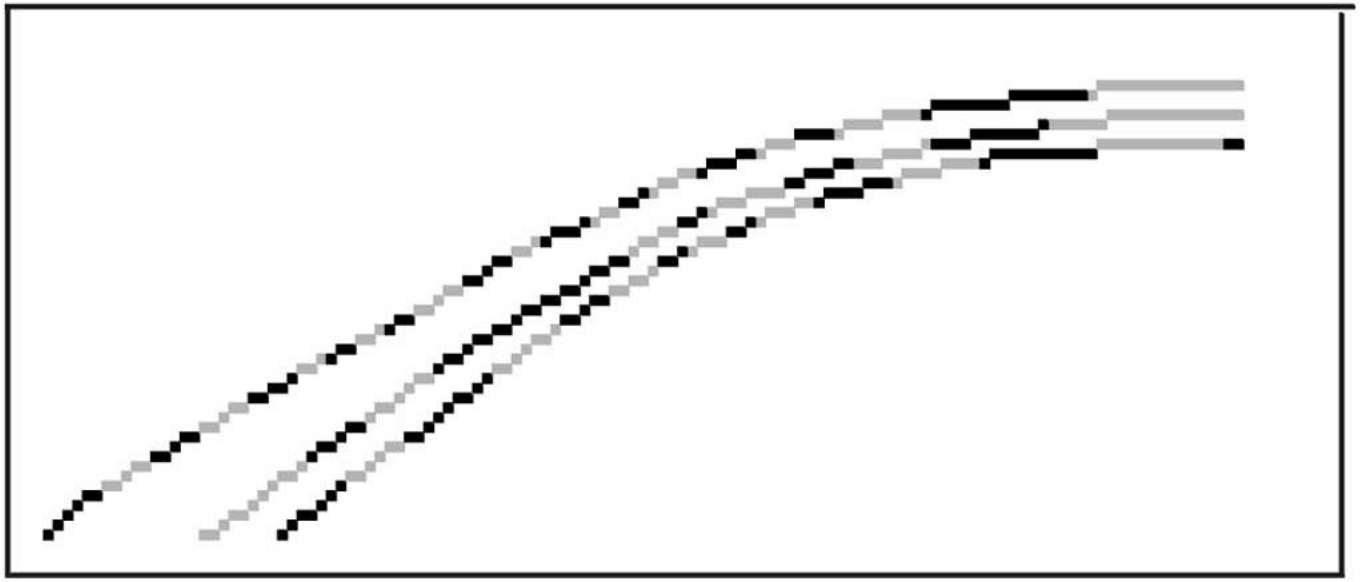
$0^4\ 1\ 0^5\ 1\ 0^4\ 1\ 0^5\ 1\ 0^4\ 1\ 0^5$

l=4, r=5, p=4

P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

Digital curve segmented by DSS's

- Segments shown with alternate black and grey fragments.
- Of small lengths.
- Too many fragmentation.



P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

Approximate DSS (ADSS)

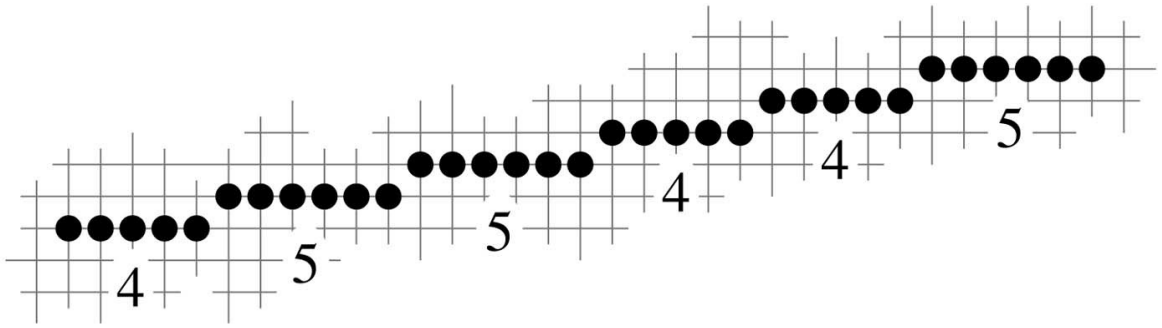
- Used R1
- Modified R2
  - Run lengths of non-singly occurred (n) element may vary more than unity, depending on the minimum length (p).
- But dropped R3 and R4
  - To allow longer fragments of DC approximating a straight line segment.
  - DSS is also accepted in the criteria of ADSS
- New Parameters: Run length interval parameters $[p,q]$ excepting l and r.
- Other hyperparameters: Tolerances
  - $q\text{-}p \leq d = \lfloor (p+1)/2 \rfloor$, and $l\text{-}p,\ r\text{-}p \leq \epsilon = \lfloor (p+1)/2 \rfloor$

Integer computation

P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

Not DSS but ADSS

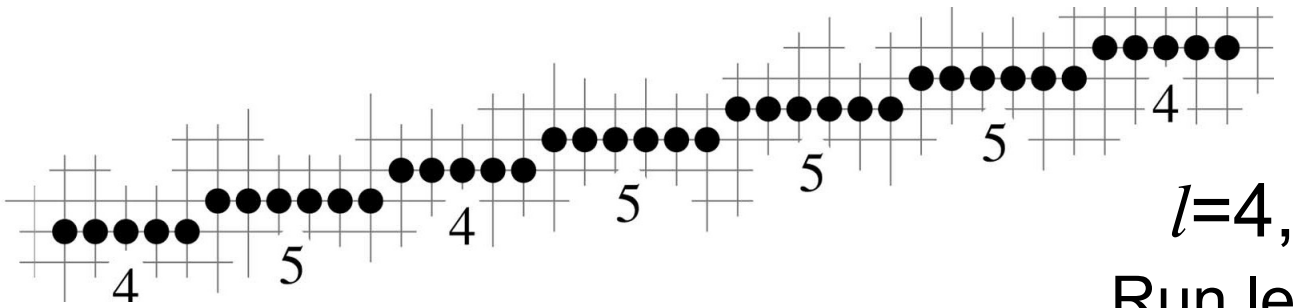


l=4, r=5, p=4, q=5

$0^4\ 1\ 0^5\ 1\ 0^5\ 1\ 0^4\ 1\ 0^4\ 1\ 0^5$

Run length seq: 455445

Both having non-singular occurrence



l=4, r=4, p=4, q=5

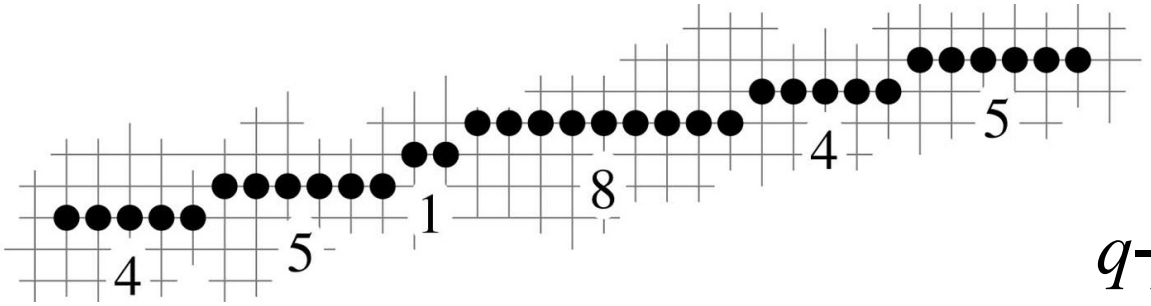
Run length seq: 4545554

$0^4\ 1\ 0^5\ 1\ 0^4\ 1\ 0^5\ 1\ 0^5\ 1\ 0^5\ 1\ 0^4$

Run lengths of 5 nonconsecutive 1 & 3

P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

Not ADSS



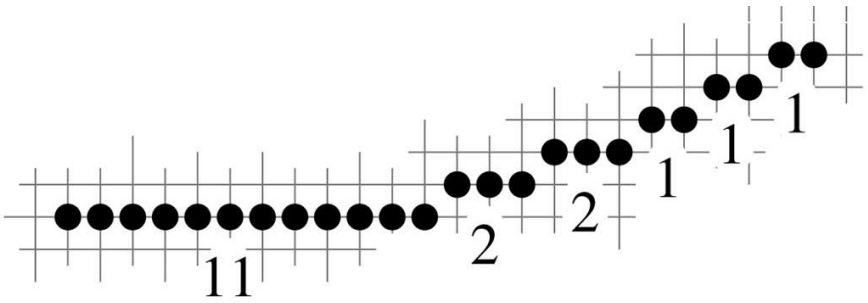
$0^4\ 1\ 0^5\ 1\ 0^1\ 1\ 0^8\ 1\ 0^4\ 1\ 0^5$

l=4, r=5, p=1, q=8

q-p > L (p+1)/2 ⌟ =1

l-p > L (p+1)/2 ⌟ =1

r-p > L (p+1)/2 ⌟ =1



$0^{11}\ 1\ 0^2\ 1\ 0^2\ 1\ 0^1\ 1\ 0^1\ 1\ 0^1$

l=11, r=1, p=1, q=2

l-p > L (p+1)/2 ⌟ =1

P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

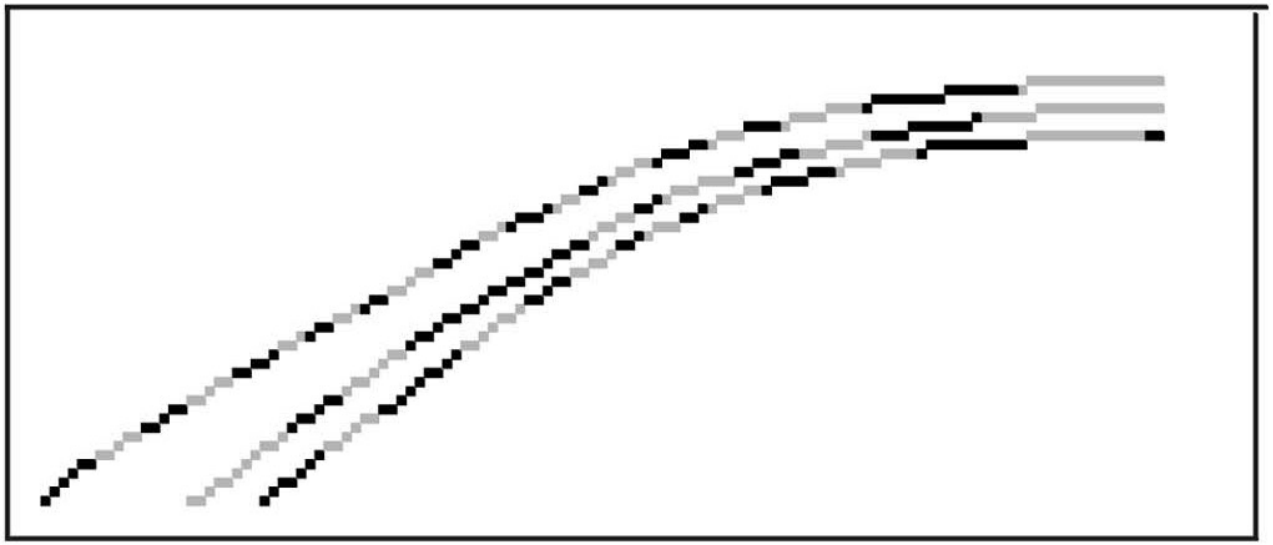
Extraction of a sequence of ADSS from a DC

- Start from the first point by including it as a vertex of the first ADSS.
- Extract parameters l, n, and s
  - l: leftmost run length of non-singularly occurring element,
  - n: non-singularly occurring element, and
  - s: non-singularly occurring element.
- Compute runs of n till it breaks conditions of ADSS.
- Include the point before the breaking condition emerges in the sequence of ADSS in the DC.
- Repeat above steps from the extraction of parameters and continue till the end of DC.

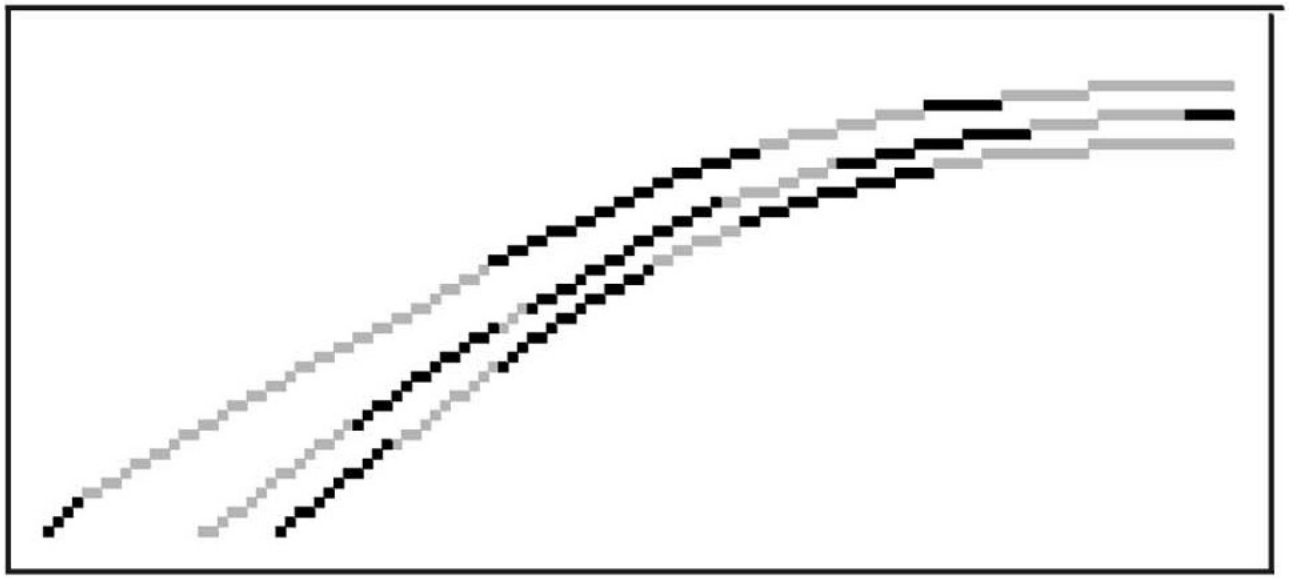
A linear time algorithm

P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

Example



DSS

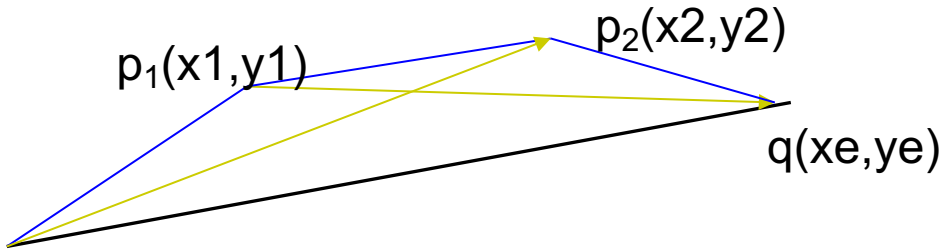


ADSS

P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

Polygonization / Polylineation

- Input: A sequence of ADSS.
- Output: Vertices of polygon.
- Algo:
  - Merge consecutive sequences following an error criteria.
    - Cumulative (Max) area of triangles formed by end points of the ADSS and the line segment of the merged segment should remain within a fraction of maximum iso-thetic distance of the merged line segment.
  - Represent merged segments as a straight line segment with start and end points of the start and end ADSS of the sequence.
  - Continue till all the ADSS's are covered.

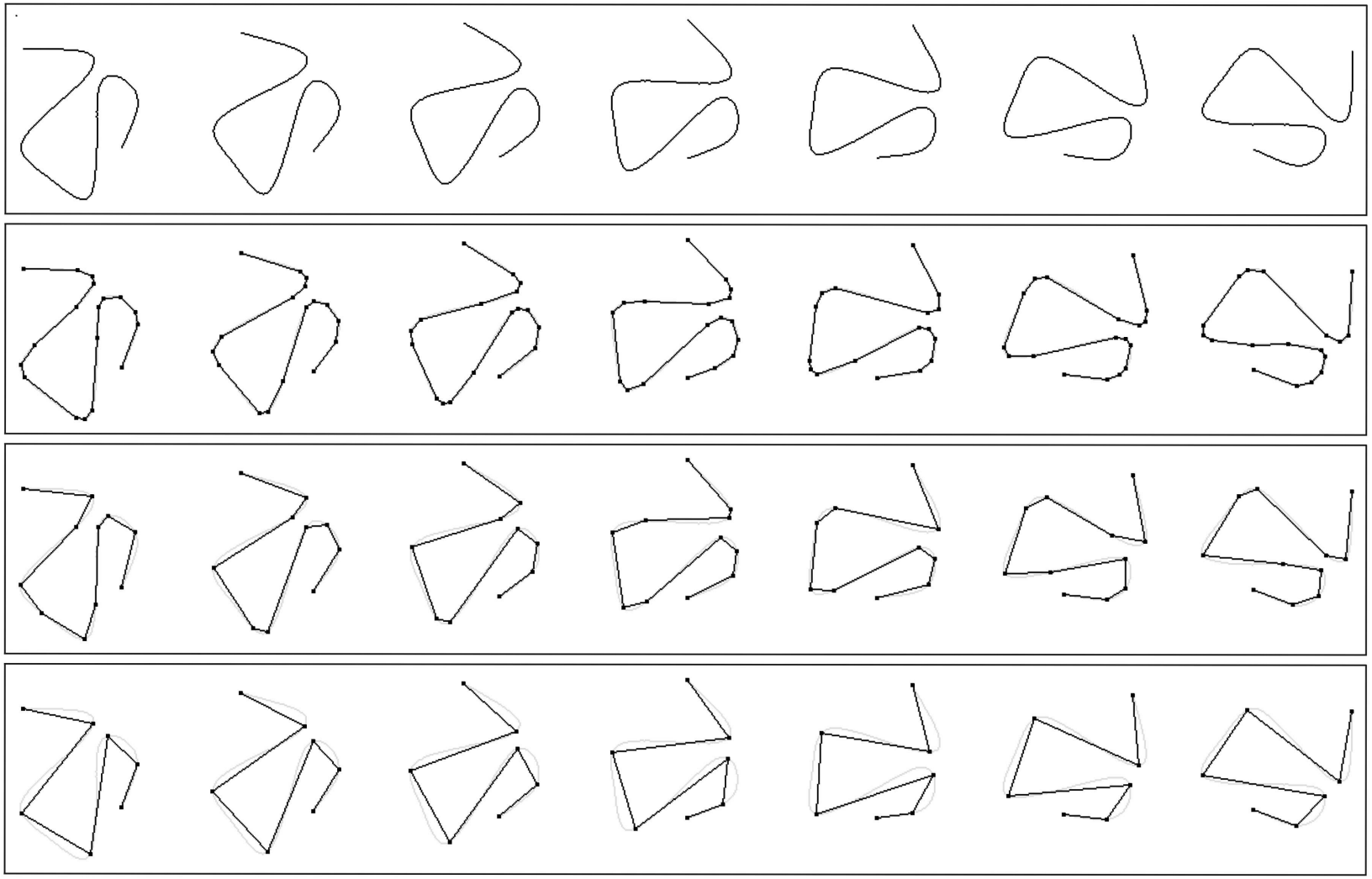


$d_{iso}(p,q) = \max\{|xe-xs|,|ye-ys|\}$

Cumulative area $= \Delta pp_1q + \Delta pp_2q < \tau\, d_{iso}(p,q)$

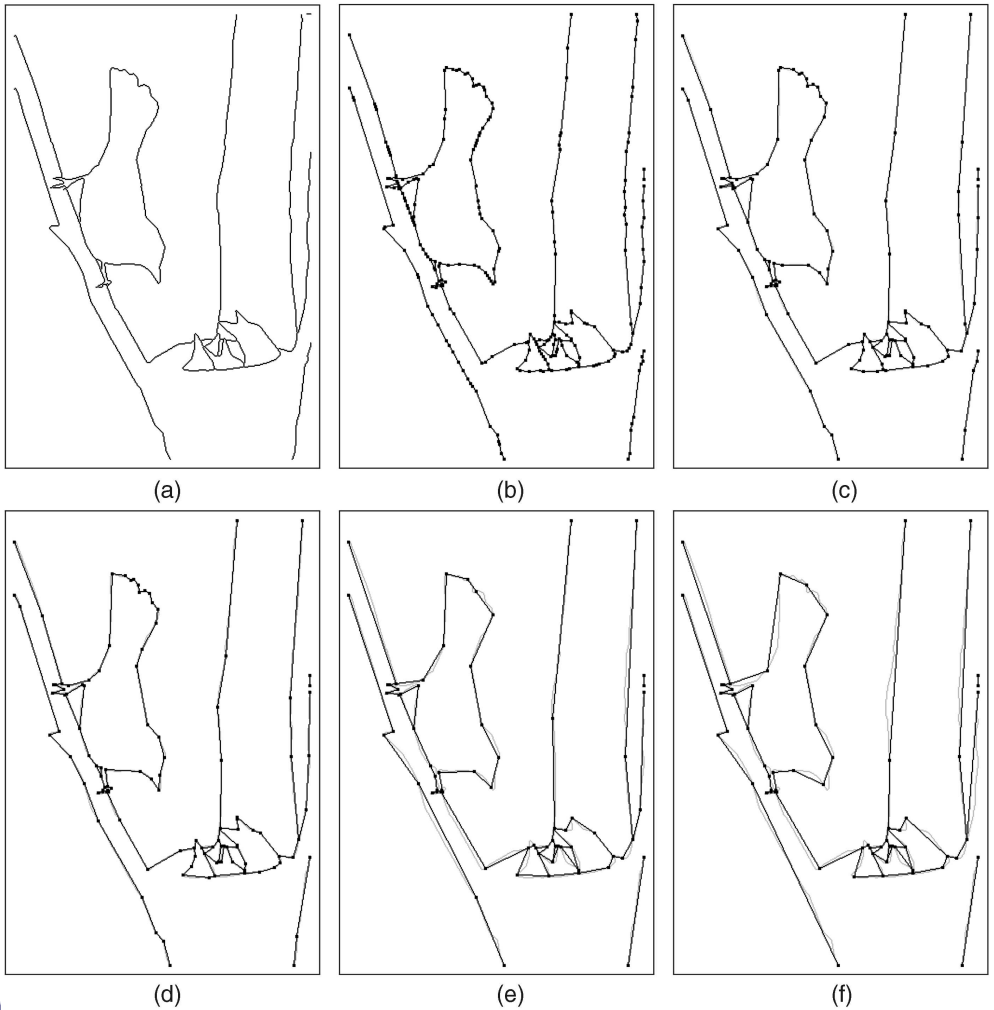
P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

A few results



P. Bhowmick and B. B. Bhattacharya, "Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1590-1602, Sept. 2007

Another example



(a) input set of DC. (b) ADSS. (c) C_{cum} : τ= 2. (d) C_{max} : τ =2. (e) C_{cum}: τ =8. (f) C_{max} : τ =8.

Polygonal approximation by merging

1. Merge points along boundary until LSE of line fit exceeds a threshold. Output a line (side of polygon) with parameters of LSE
2. Repeat Step 1 as long as there are points on the boundary
3. Intersections of line segments give vertices

Disadvantage

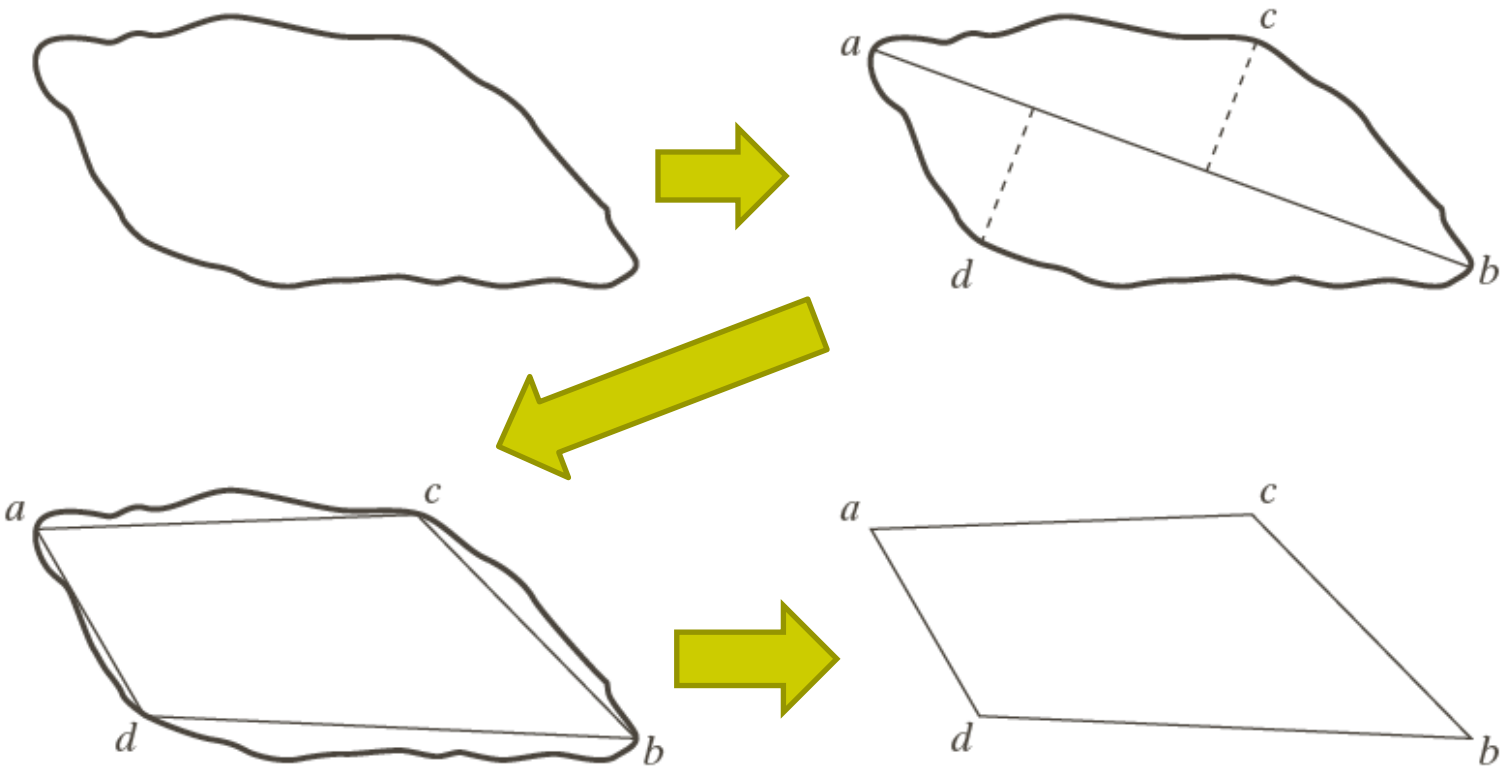
May not produce vertices at inflection points as 'long' consume these 'outlier' points within the threshold.

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Polygonal approximation by splitting

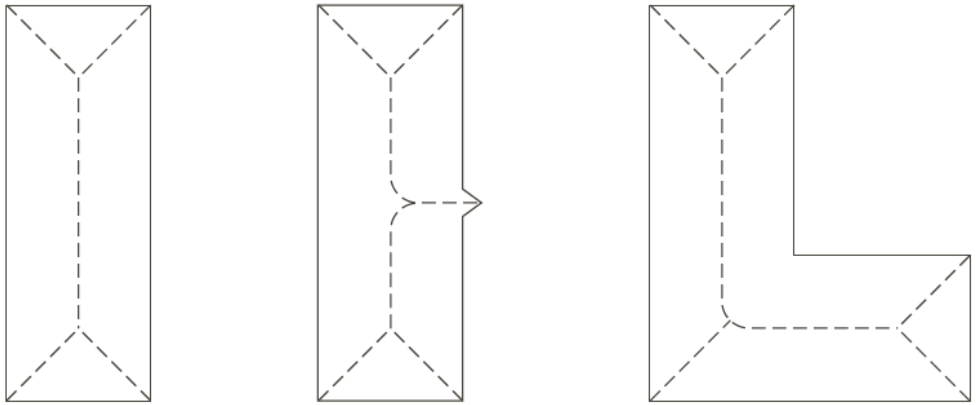
- Boundary into two parts using extreme points.
  - Diameter for closed contour providing two points.
  - Most distant point from the straight line connecting two end points of an open contour segment.
- Start with the closed contour and determine two extreme points.
  - Output two open segments in counter-clockwise order.
- Recursively process every open segment and form a vertex at each stage connecting end points of open contours
  - till the max. dist. less than a threshold.

Polygonal approximation by splitting



Skeletonization: Medial Axis Transform (MAT)

- MAT of region R with border B
  - For each point p in R, find its closest neighbor in B
    - "Closest" use the concept of distance – often Euclidean
  - If p has more than one such neighbor, it belongs to the Medial Axis (Skeleton) of R
- Intuitively MAT is defined by Grassfire analogy



Skeletonization: Medial Axis Transform (MAT)

- Intuitive approach is computationally expensive as it needs the computation of the distance from every interior point to every point on the boundary of a region
- Alternate approach is to remove non-MAT points
- Typical fast (thinning) algorithms iteratively delete boundary points of a region provided the deletion:
  - Does not remove the end points
  - Does not break connectivity
  - Does not cause excessive erosion of the region

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Skeletonization: Thinning Algorithm

- Border / Contour Point: An object (region) point (1) with at least one background (0) neighbor
- Step 1
  - Repeat for all contour points
    - Flag contour points for deletion by Condition 1
  - Remove all flagged points (change 1 → 0)
- Step 2
  - Repeat for all contour points
    - Flag contour points for deletion by Condition 2
  - Remove all flagged points (change 1 → 0)
- Repeat Steps 1 & 2 till no deletion is possible

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Skeletonization: Thinning Algorithm

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

- Condition 1
  a) $2 \le N(p_1) \le 6$
  b) $T(p_1) = 1$
  c) $p_2.p_4.p_6=0$
  d) $p_4.p_6.p_8=0$
- Condition 2
  A. $2 \le N(p_1) \le 6$
  B. $T(p_1) = 1$
  C. $p_2.p_4.p_8=0$
  D. $p_2.p_6.p_8=0$

$N(p_1)$ =Non-zero neighbors of p_1= $p_2+p_3+p_4+p_5+p_6+p_7+p_8+p_9$

$T(p_1)$ =Number of 0-1 transitions in the sequence $p_2,p_3,p_4,p_5,p_6,p_7,p_8, p_9,p_2$

0	0	1
1	p_1	0
1	0	1

$N(p_1)=4$

$T(p_1)=3$

Skeletonization: Thinning Algorithm

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

- Condition 1
  a) $2 \le N(p_1) \le 6$
  b) $T(p_1) = 1$
  c) $p_2.p_4.p_6=0$
  d) $p_4.p_6.p_8=0$
- Condition 2
  A. $2 \le N(p_1) \le 6$
  B. $T(p_1) = 1$
  C. $p_2.p_4.p_8=0$
  D. $p_2.p_6.p_8=0$

$N(p_1)$ =Non-zero neighbors of p_1= $p_2+p_3+p_4+p_5+p_6+p_7+p_8+p_9$

$T(p_1)$ =Number of 0-1 transitions in the sequence $p_2,p_3,p_4,p_5,p_6,p_7,p_8, p_9,p_2$

0	0	1
1	p_1	0
1	0	1

$N(p_1)=4$

$T(p_1)=3$

Skeletonization: Thinning Algorithm

- If p_1 satisfies [a,b,c,d] (or [A,B,C,D]) it should be safe to remove it (flag it for removal) while preserving the structure of the skeleton intact
  - **Note**: The actual removal is lazy so that the order of checking for these conditions does not impact the actual application

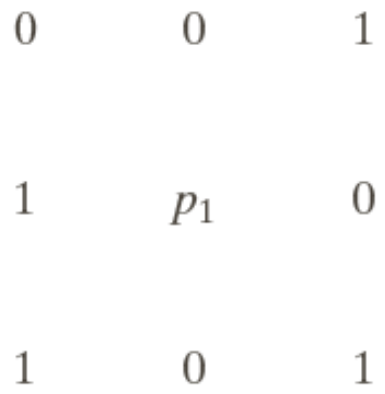
Skeletonization: Thinning Algorithm

- Interpretations of conditions
  - [a, A]: Guard Condition for
    - Protecting end point
    - Limiting excessive erosion
  - [b, B]: Guard Condition for
    - Preserving connectivity
  - [c, C] / [d, D]: Candidate Condition for
    - Border points
    - Corner points

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Skeletonization: Thinning Algorithm

- How do the conditions guarantee a MAT?
  - [a, A] $2 \le N(p_1) \le 6$ is violated if p_1 has
    - 1 neighbor → End point of a stroke
    - 7 neighbors → Causes excessive erosion
  - [b, B] $T(p_1) = 1$ is violated if p_1 is
    - On a 1-pixel thick stroke (bridge) → Preserve connectivity



$N(p_1)=4$

$T(p_1)=3$

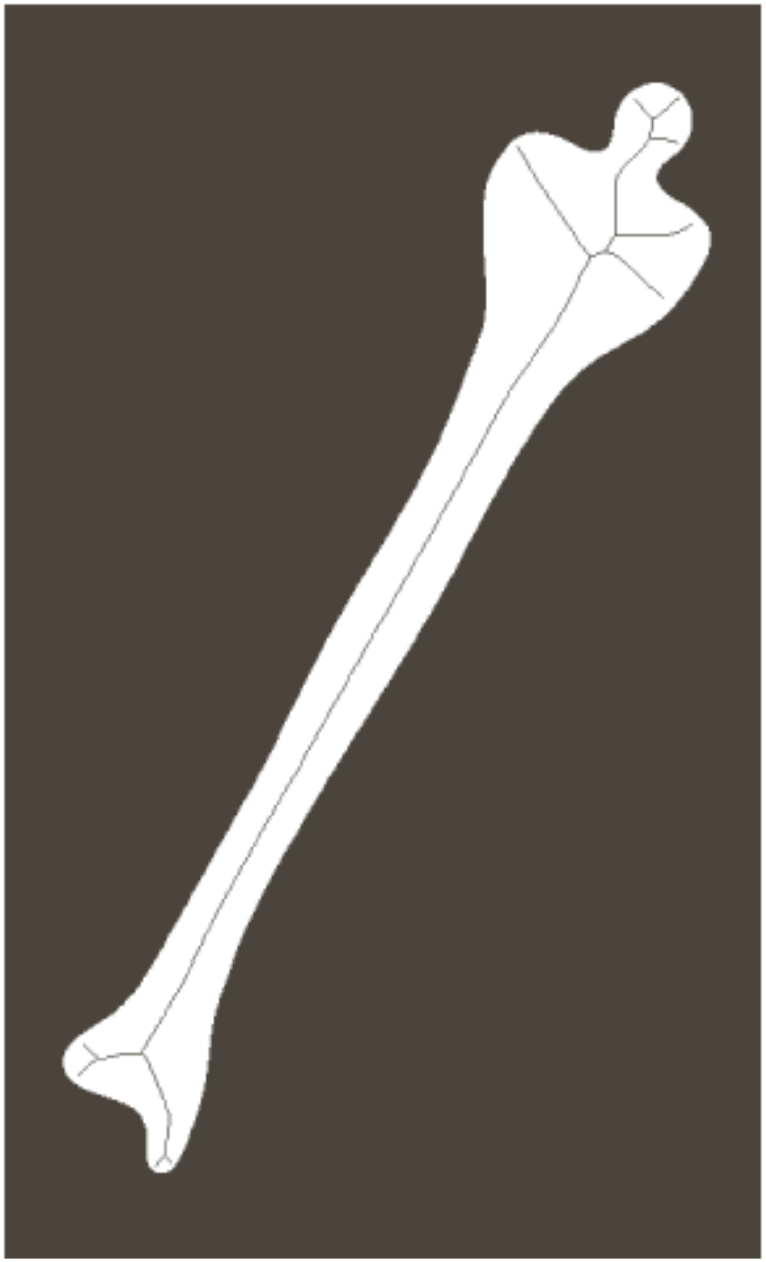
Skeletonization: Thinning Algorithm

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

- [c,d] $p_2.p_4.p_6=0$ and $p_4.p_6.p_8=0$ are satisfied simultaneously by minimal set of values:
  - $p_4=0$ (East boundary point) *or*
  - $p_6=0$ (South boundary point) *or*
  - $p_2=0$ *and* $p_8=0$ (NW corner point)
- [C,D] $p_2.p_4.p_8=0$ and $p_2.p_6.p_8=0$ are satisfied simultaneously by minimal set of values:
  - $p_2=0$ (North boundary point) *or*
  - $p_8=0$ (West boundary point) *or*
  - $p_4=0$ *and* $p_6=0$ (SE corner point)
- NE corner point ($p_2=0$ *and* $p_4=0$) satisfy [c,d] and [C,D]
- SW corner point ($p_6=0$ *and* $p_8=0$) satisfy [c,d] and [C,D]

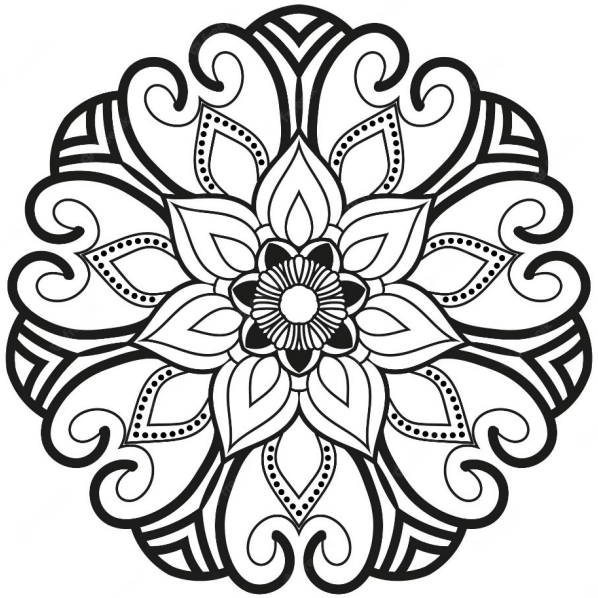
Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Skeleton: Example

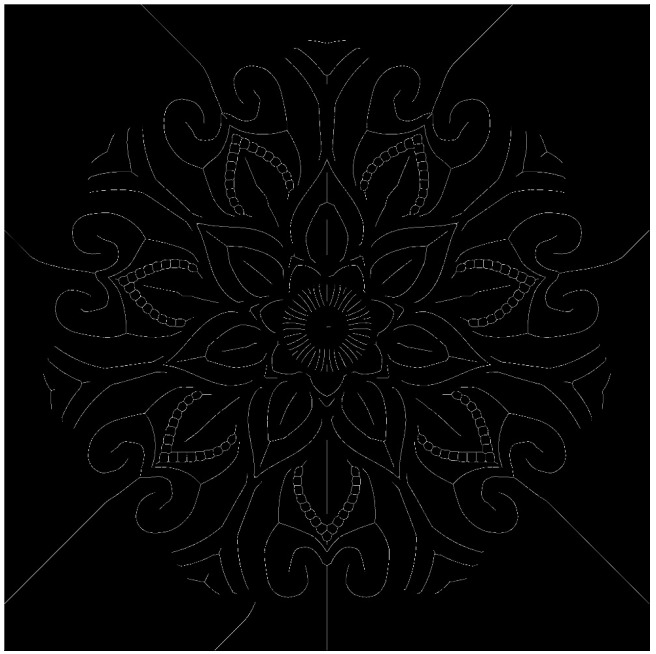


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Thinning : Example



Input image



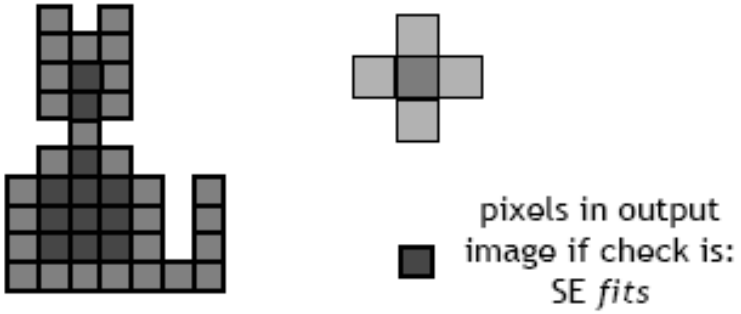
Output

Morphological Operations

- Operations on binary images with a structuring element to perform morphological (structural) changes or extract the features
- Structural element (SE):
  - a set of points in the digital grid with a reference integral coordinate system.
  - Performs 'Hit' or 'Fit' operation at any point in the digital grid.

Examples of SE

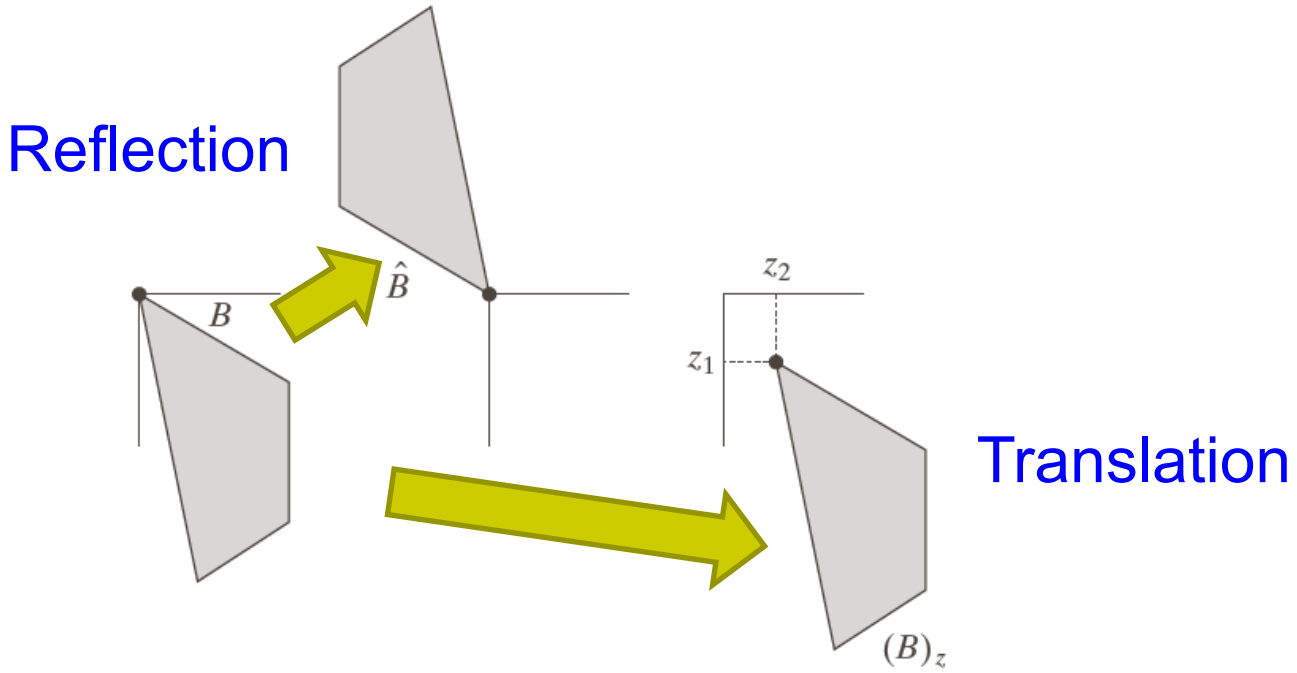
Points conforming to 'Fit' operations



Reflection and Translation

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

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



Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Basic morphological operations

- Erosion 

- Dilation 

Keep general shape but smooth with respect to object / background

- combine to
  - Opening → object
  - Closing → background

Erosion

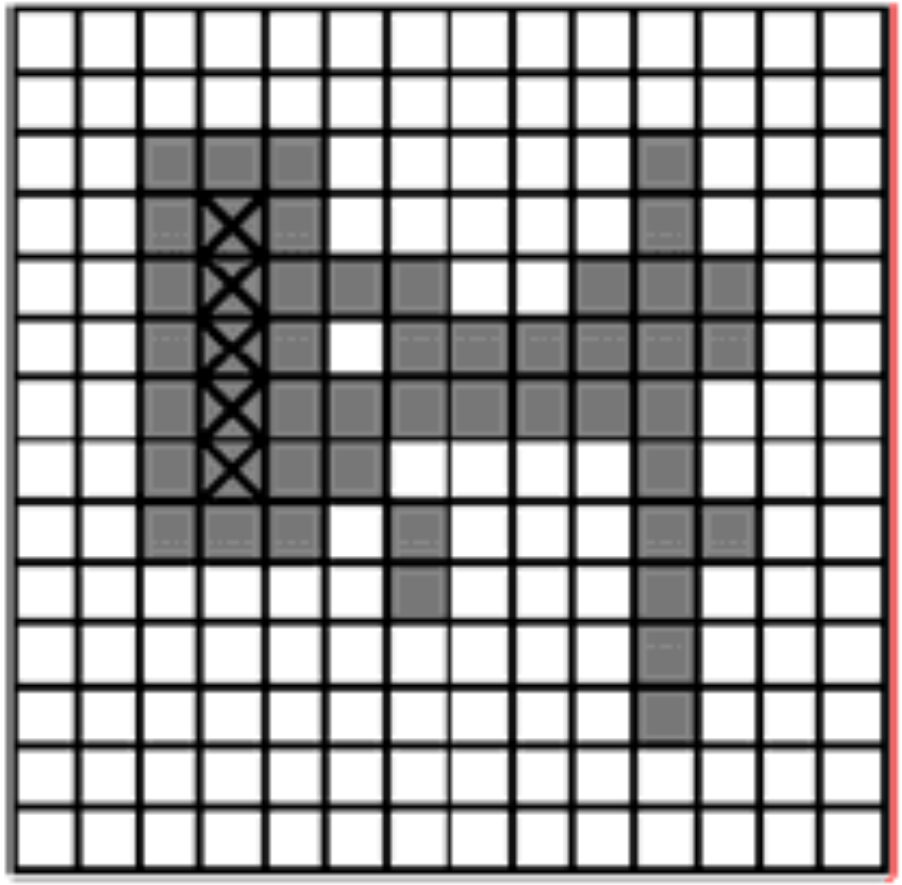
- Does the structuring element **fit the set?**
- erosion of a set A by structuring element B: all z in A such that B is in A when origin of B=z

$$A \ominus B = \{z | (B)_z \subseteq A\}$$

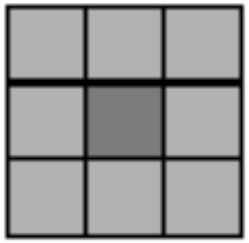
$$A \ominus B = \{z | (B)_z \cap A^c = \Phi\}$$

- **shrink the object**

Erosion

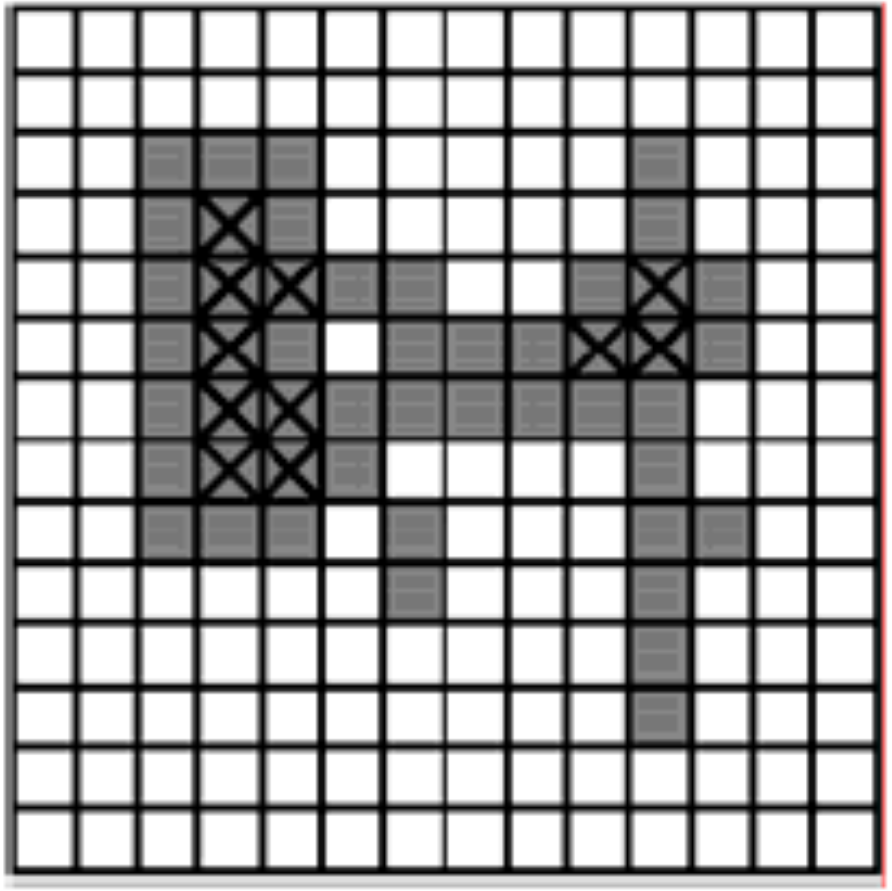


SE=

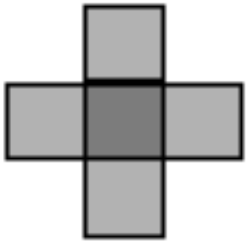


Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Erosion



SE=



Erosion Example 1

Original image

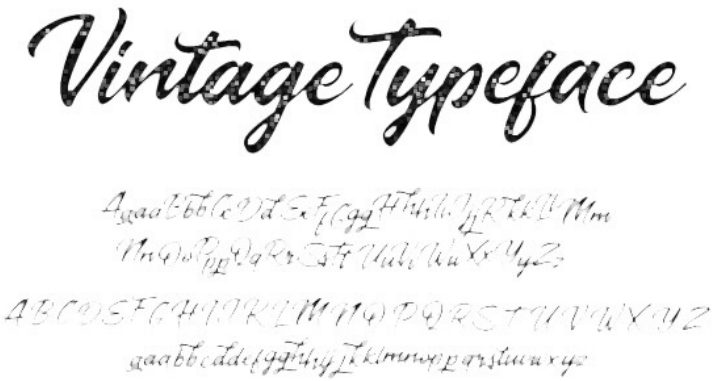
Erosion by 3*3 square structuring element

Erosion by 5*5 square structuring element

Watch out: In these examples a 1 refers to a black pixel!

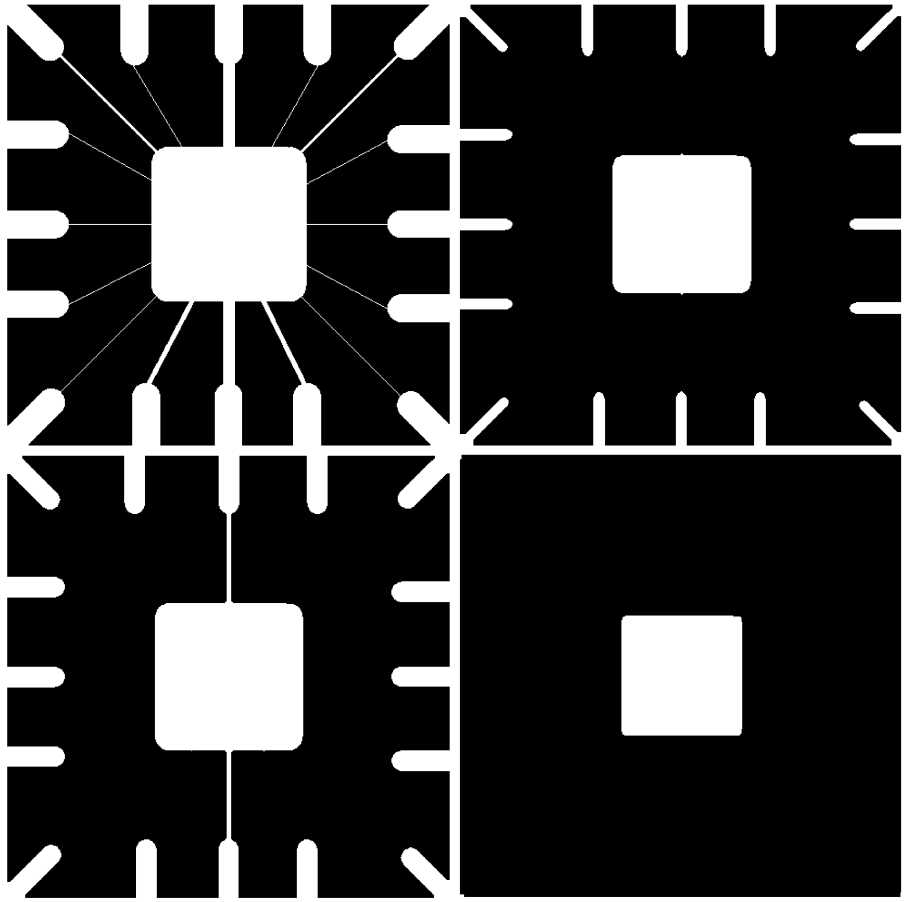
Erosion: Example

Input image



Eroded image

Erosion Example 2



Original image

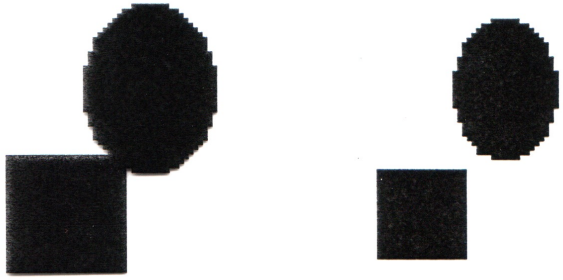
After erosion with a disc of radius 10

After erosion with a disc of radius 5

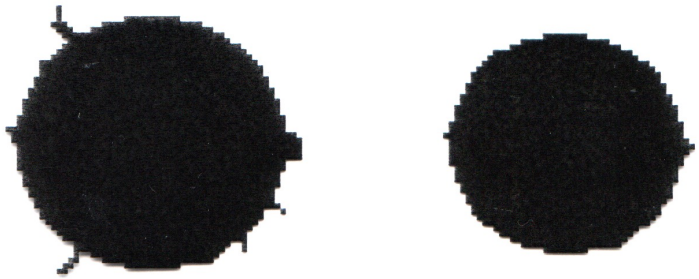
After erosion with a disc of radius 20

What Is Erosion For?

Erosion can split apart joined objects



Erosion can strip away extrusions



Watch out: Erosion shrinks objects!

Dilation

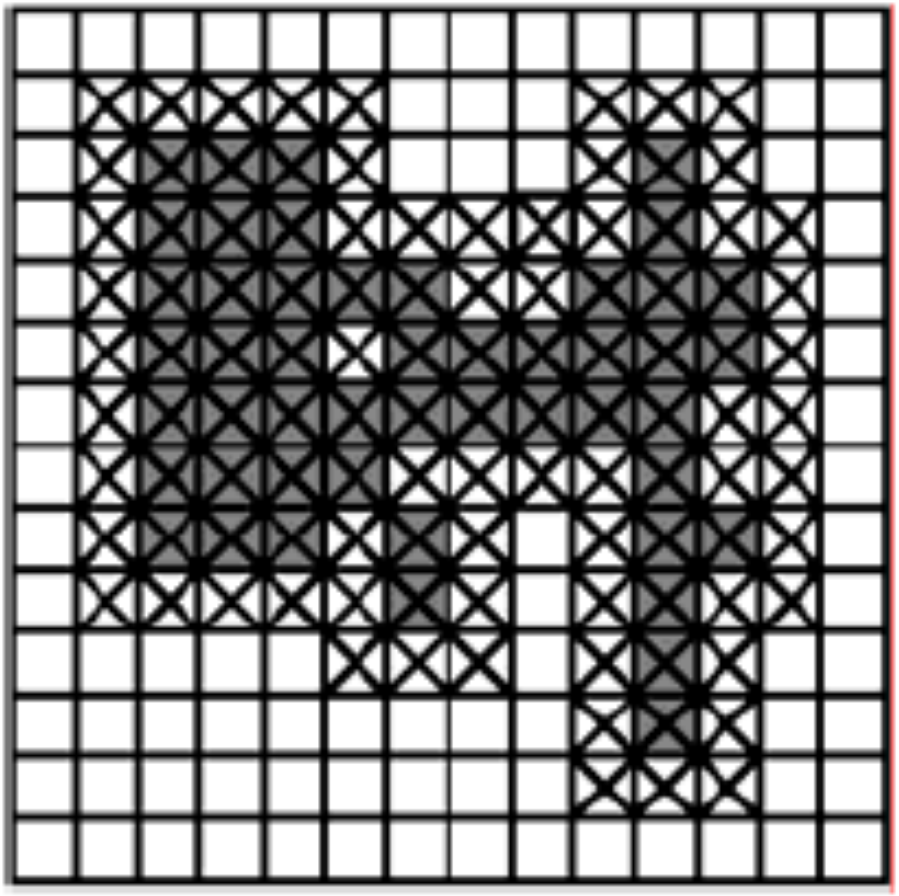
- Does the structuring element **hit the set?**
- dilation of a set A by structuring element B: all z in A such that B hits A when origin of B=z

$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \Phi\}$$

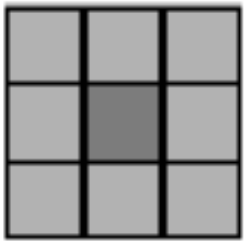
$$A \oplus B = \{z | [(\hat{B})_z \cap A] \subseteq A\}$$

- **grows the object**

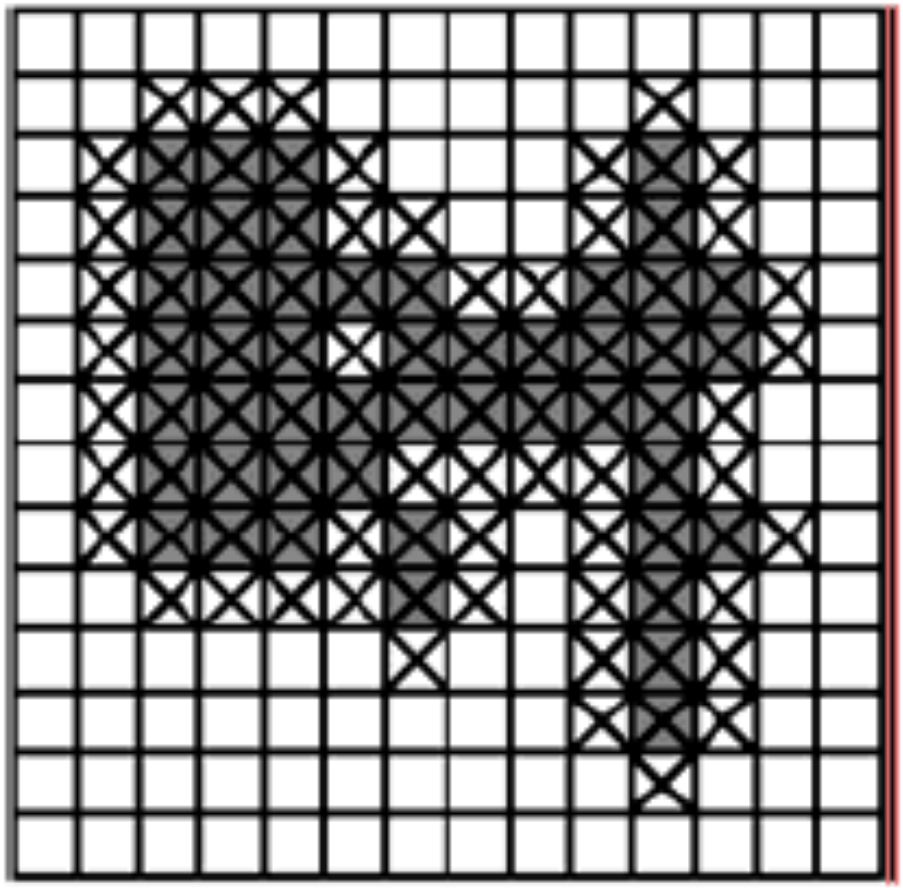
Dilation



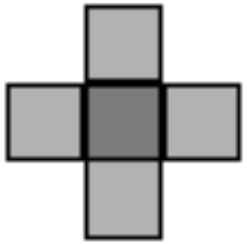
SE=



Dilation



SE=



Dilation Example

Original image

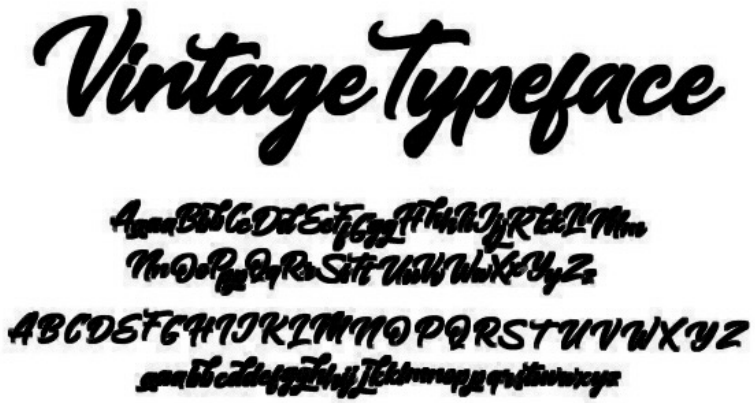
Dilation by 3*3 square structuring element

Dilation by 5*5 square structuring element

Watch out: In these examples a 1 refers to a black pixel!

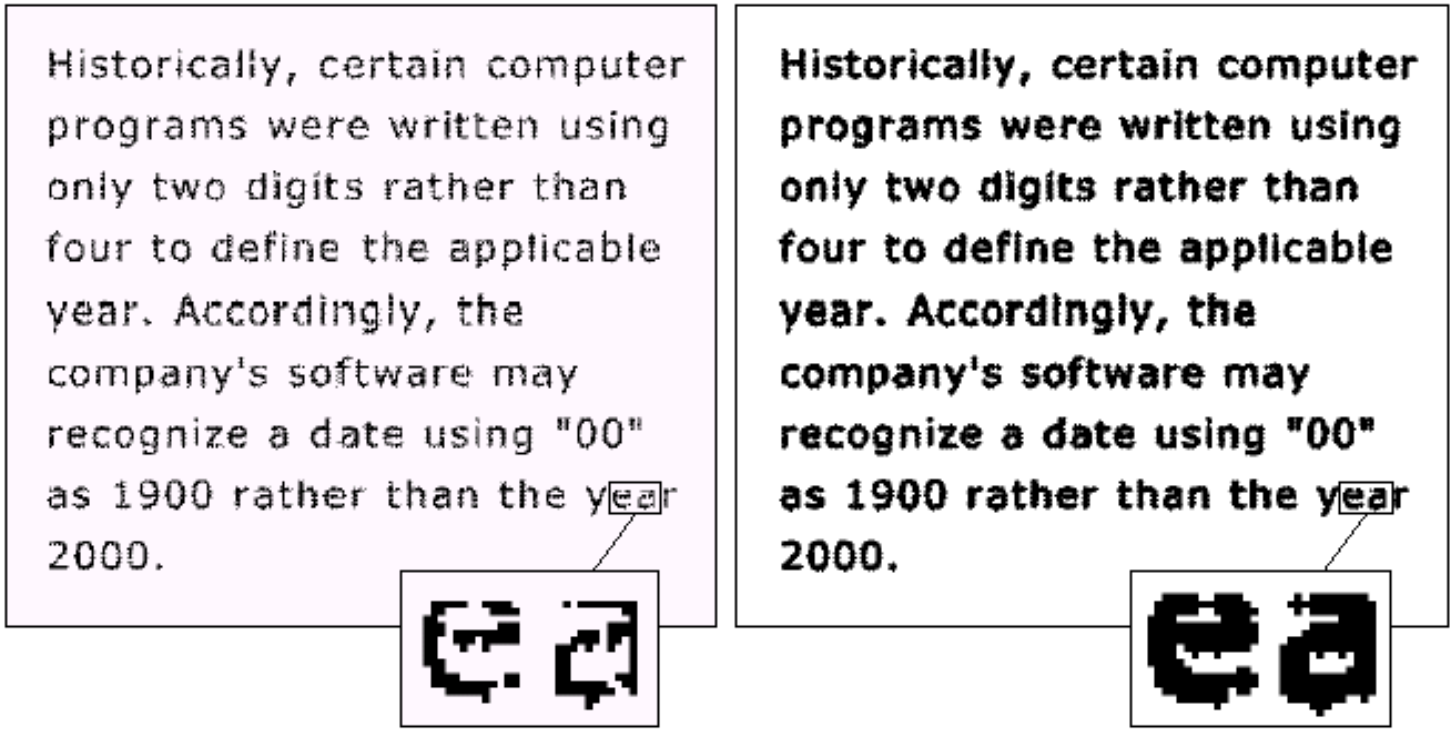
Dilation: Example

Input image



Dilated Image

Dilation : Bridging gaps



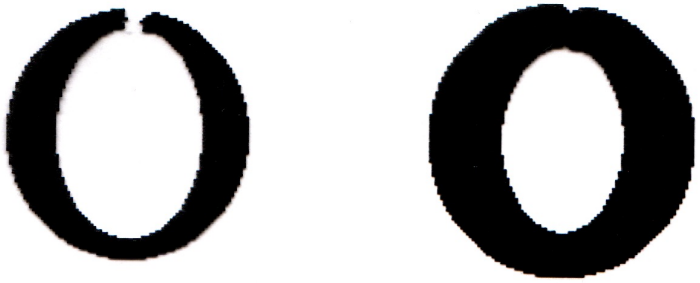
a c
b

FIGURE 9.5
(a) Sample text of poor resolution with broken characters (magnified view).
(b) Structuring element.
(c) Dilation of (a) by (b). Broken segments were joined.

0	1	0
1	1	1
0	1	0

What Is Dilation For?

Dilation can repair breaks



Dilation can repair intrusions

Watch out: Dilation enlarges objects

Useful

- Erosion
  - removal of structures of certain shape and size, given by SE
- Dilation
  - filling of holes of certain shape and size, given by SE

Duality

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

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

$$(A \oplus B)^c = A^c \ominus \hat{B}$$

5 basic structuring elements

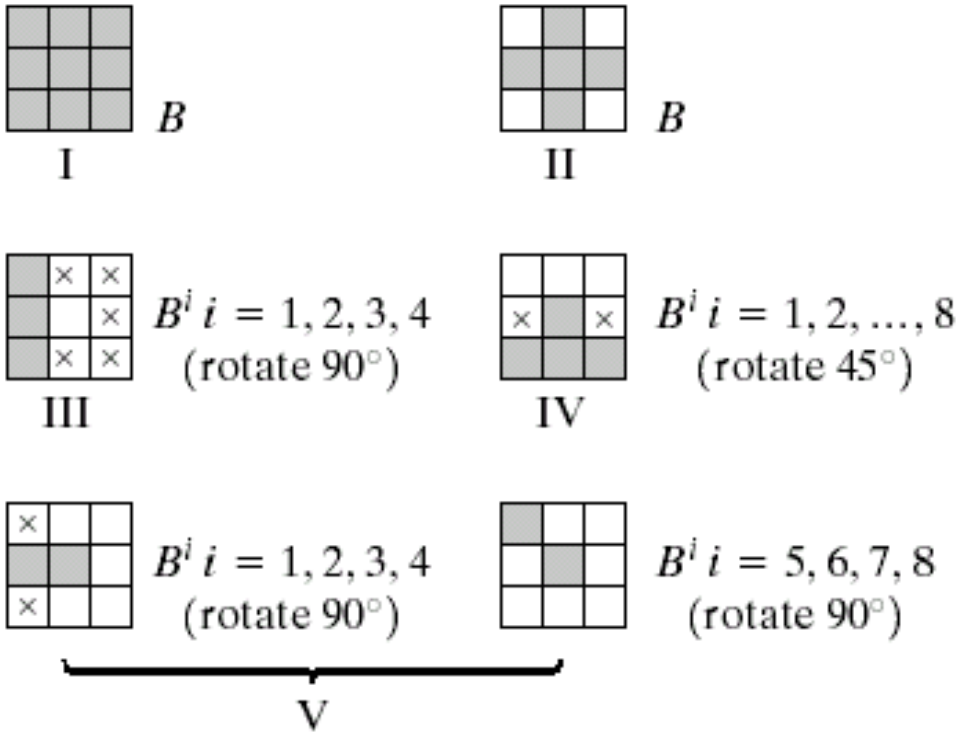


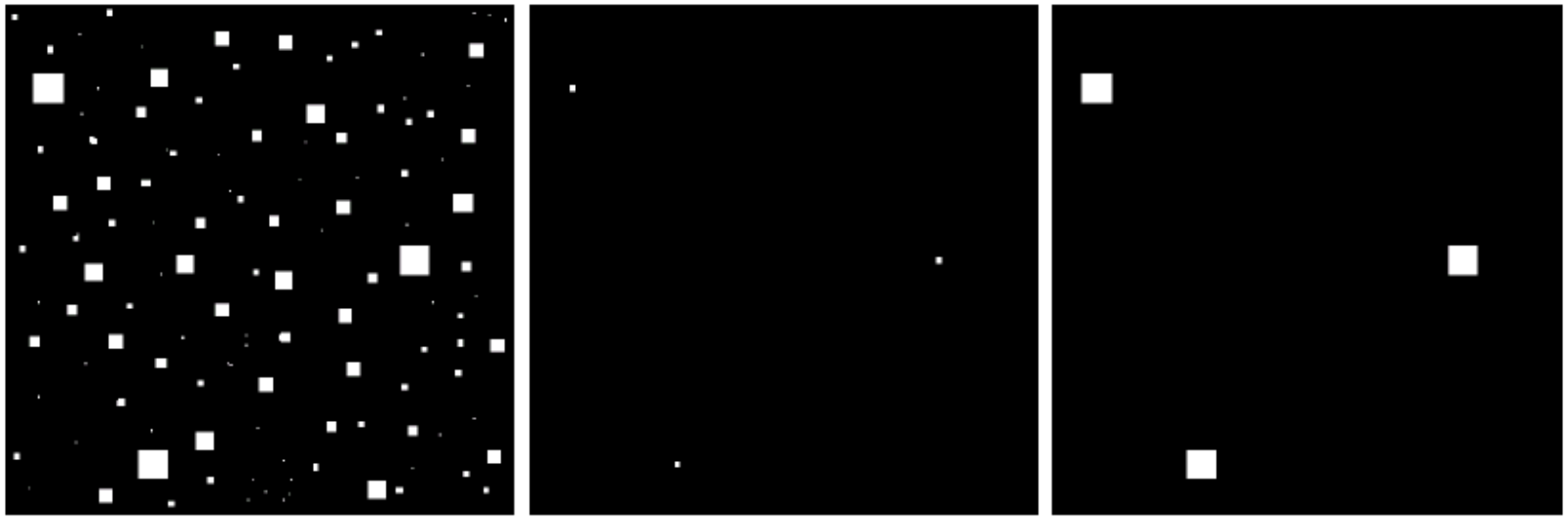
FIGURE 9.26 Five basic types of structuring elements used for binary morphology. The origin of each element is at its center and the ×'s indicate "don't care" values.

Combining erosion and dilation

- WANTED:
  - remove structures / fill holes
  - without affecting remaining parts

- SOLUTION:
  - combine erosion and dilation
  - (using same SE)

Erosion followed by dilation: eliminating irrelevant detail



a b c

FIGURE 9.7 (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

structuring element B = 13x13 pixels, each set to 1.

Opening

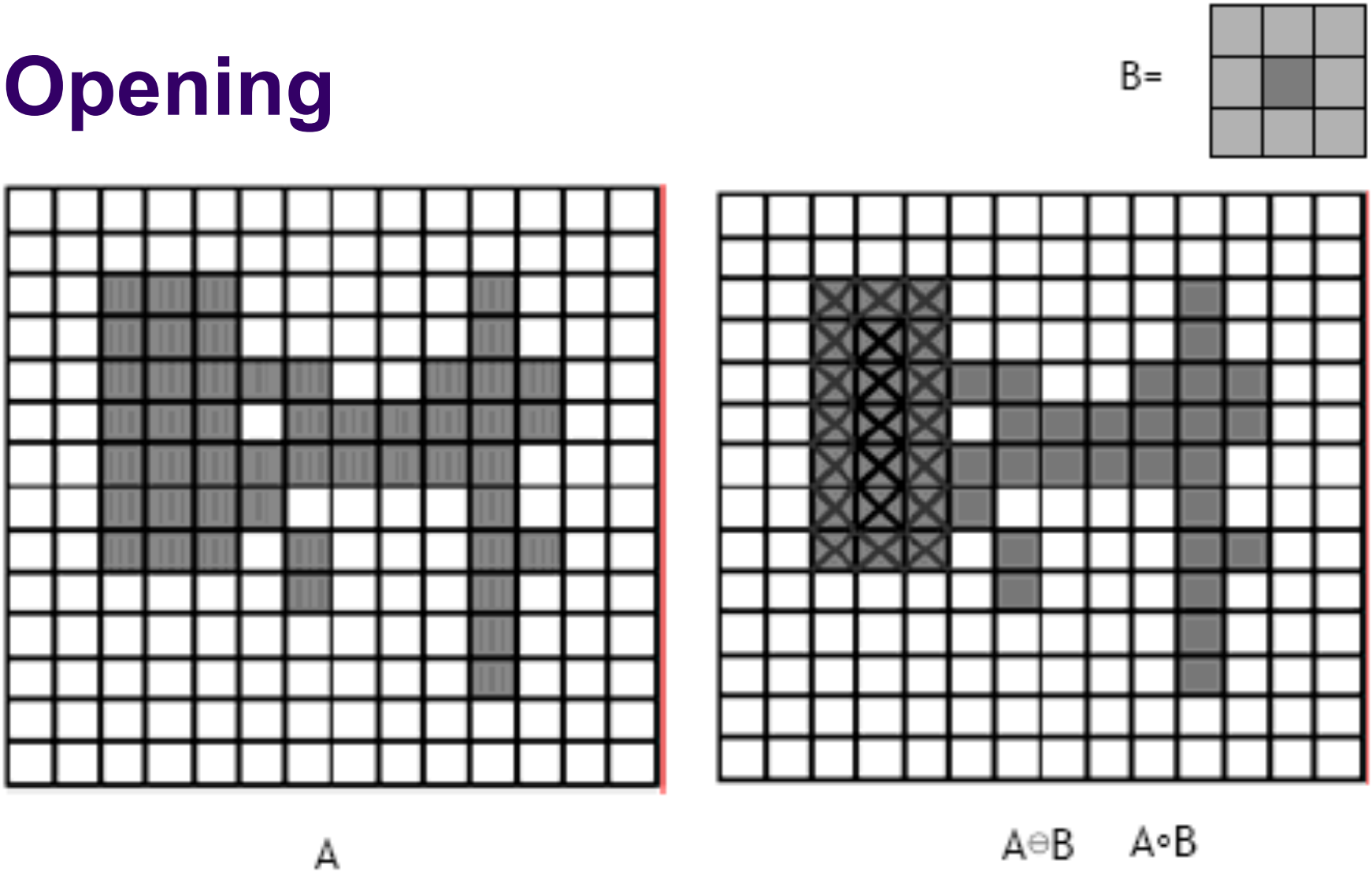
erosion followed by dilation, denoted $\circ$

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

- eliminates protrusions
- breaks necks
- smoothes contour

Opening

B=

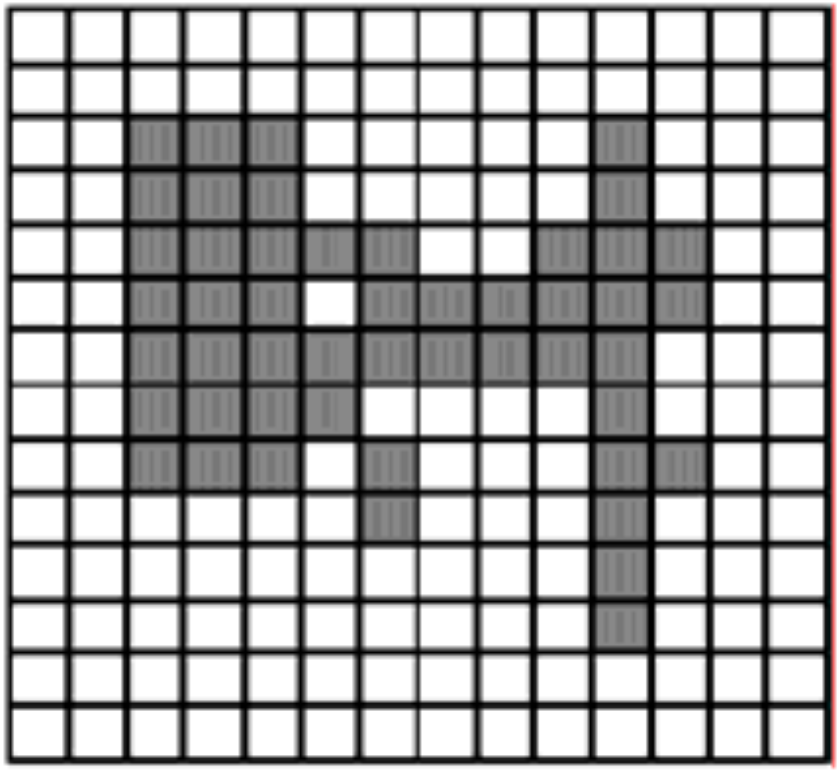


A

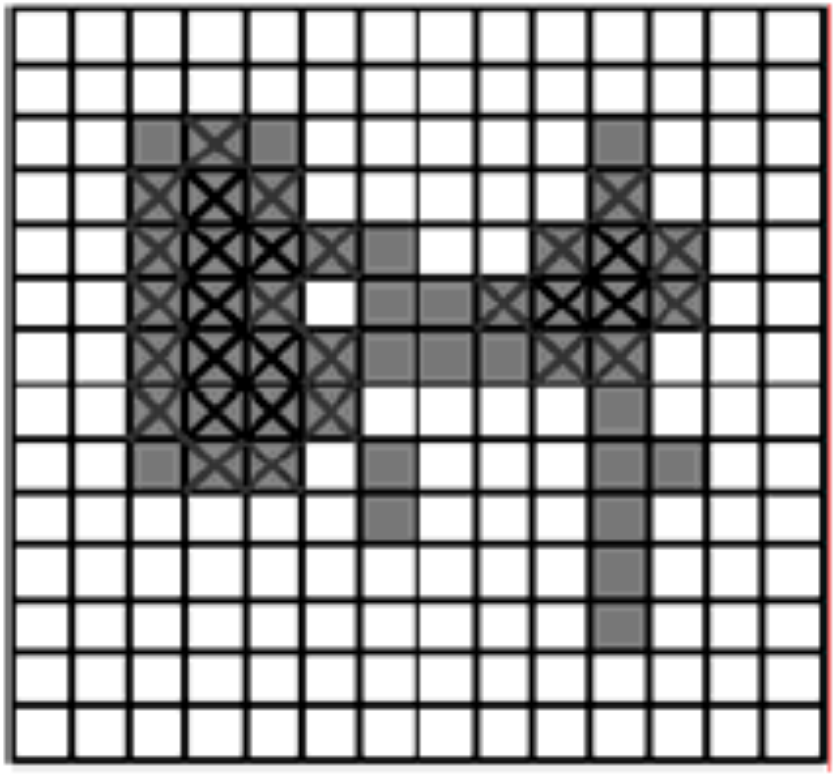
A⊖B A∘B

Opening

B=

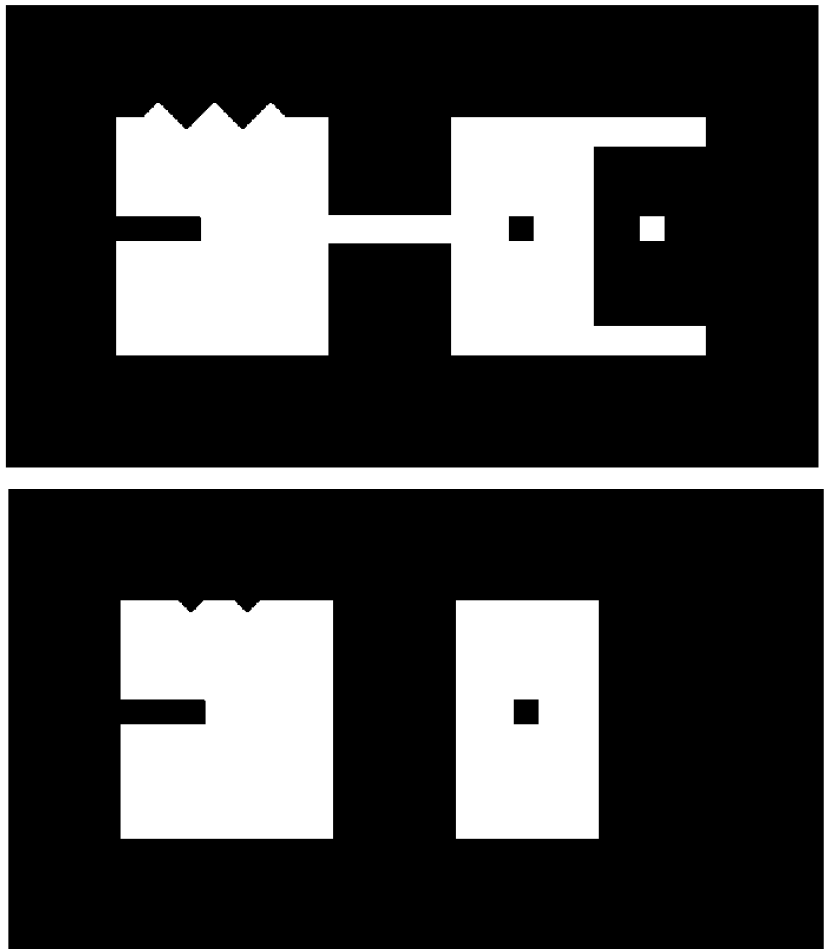


A



A⊖B A∘B

Opening Example



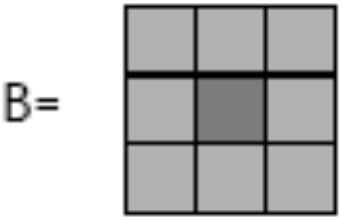
Closing

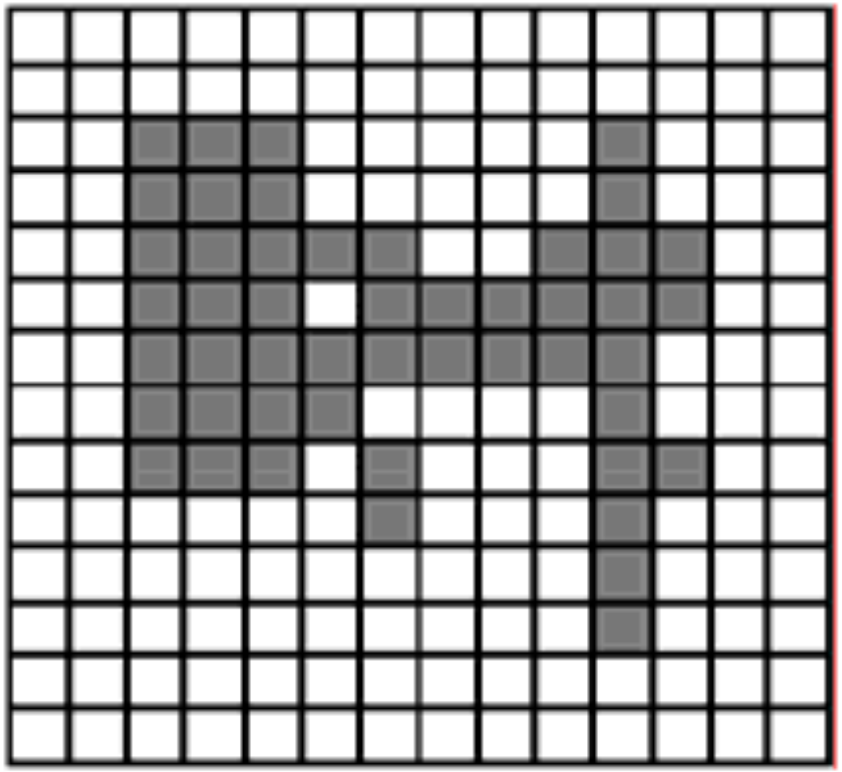
dilation followed by erosion, denoted •

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

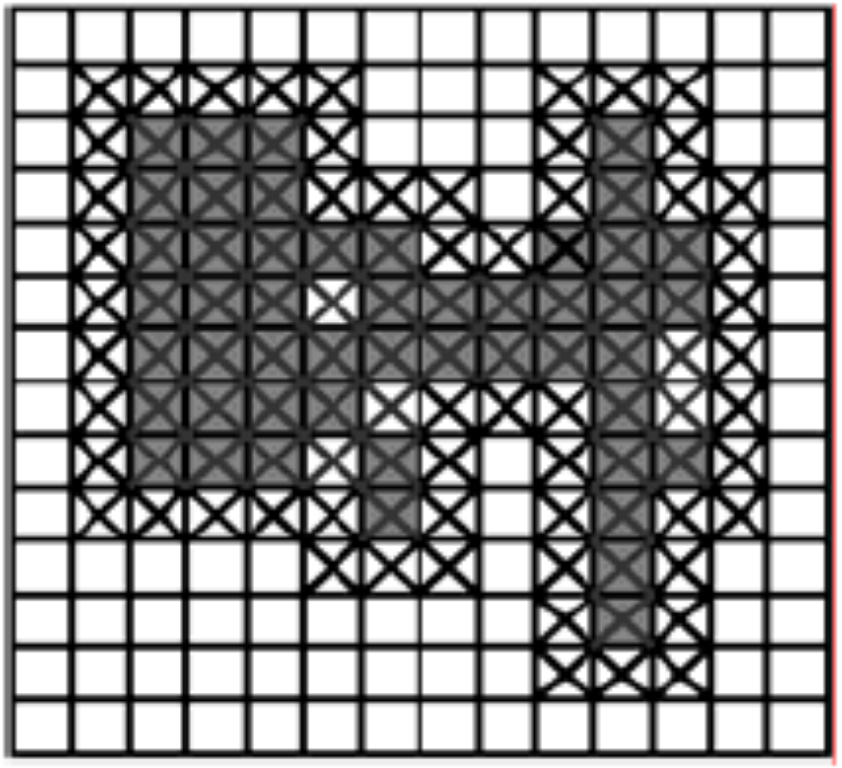
- smooth contour
- fuse narrow breaks and long thin gulfs
- eliminate small holes
- fill gaps in the contour

Closing



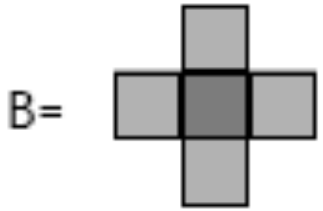


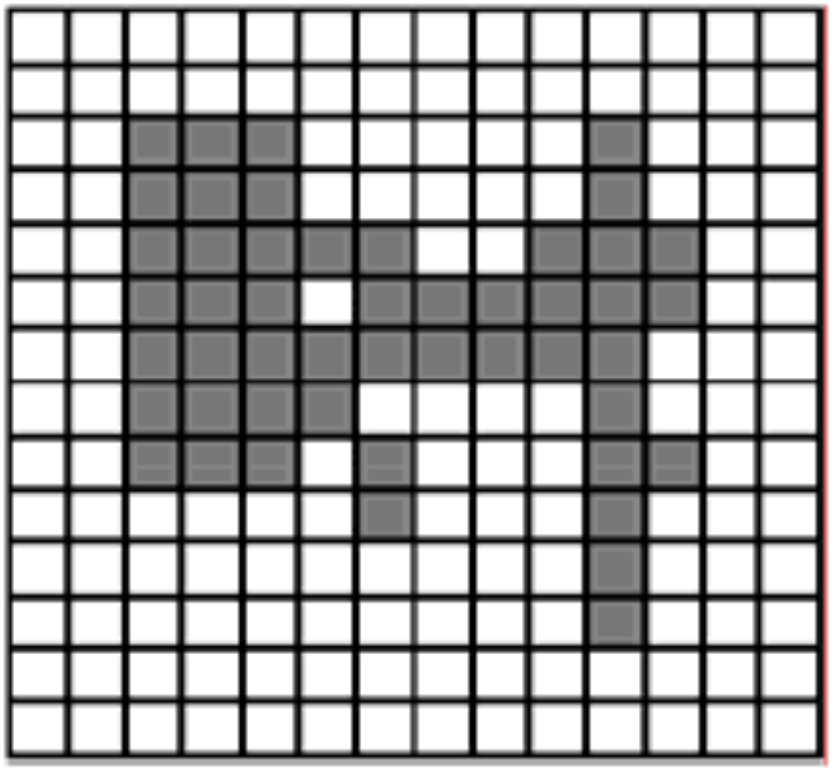
A



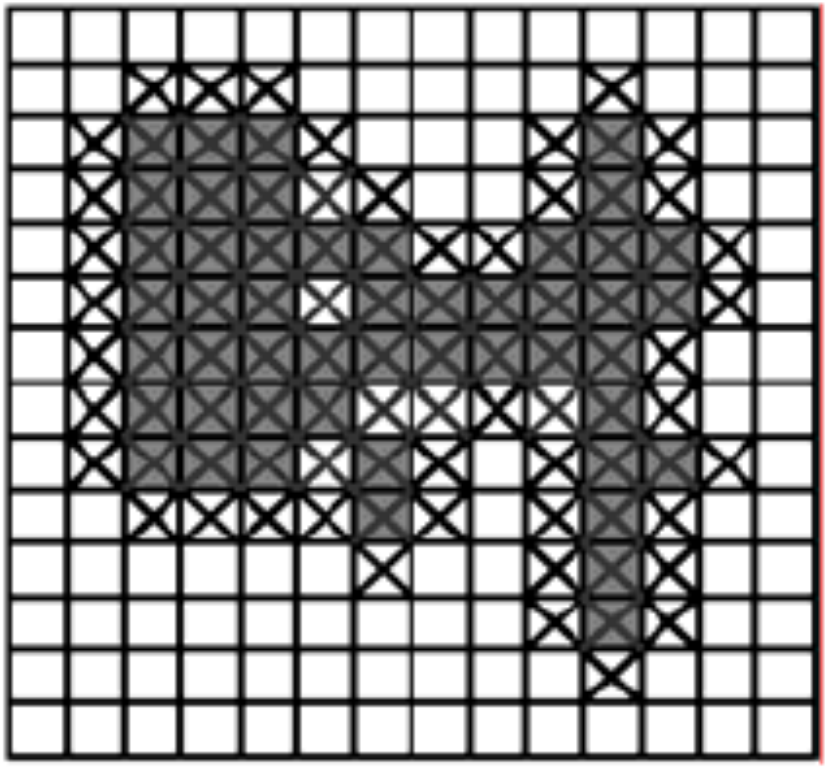
A⊕B A • B

Closing





A



A⊕B A • B

Closing Example

Original Image

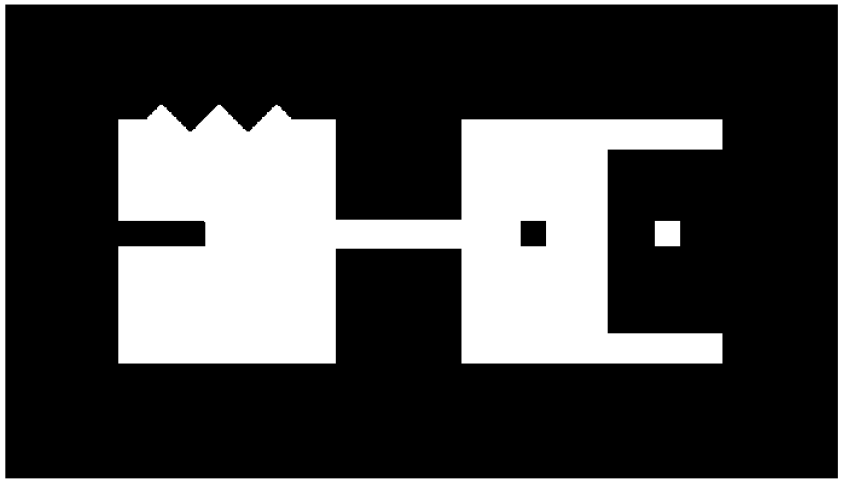
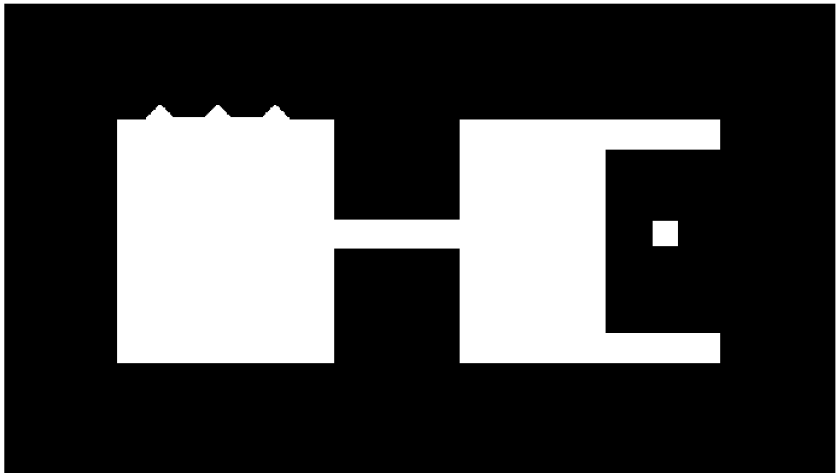


Image After Closing



Properties

Opening

(i) $A \circ B$ is a subset (subimage) of A

(ii) If C is a subset of D, then $C \circ B$ is a subset of $D \circ B$

(iii) $(A \circ B) \circ B = A \circ B$

Closing

(i) A is a subset (subimage) of $A \bullet B$

(ii) If C is a subset of D, then $C \bullet B$ is a subset of $D \bullet B$

(iii) $(A \bullet B) \bullet B = A \bullet B$

Note: repeated openings/closings has no effect!

Duality

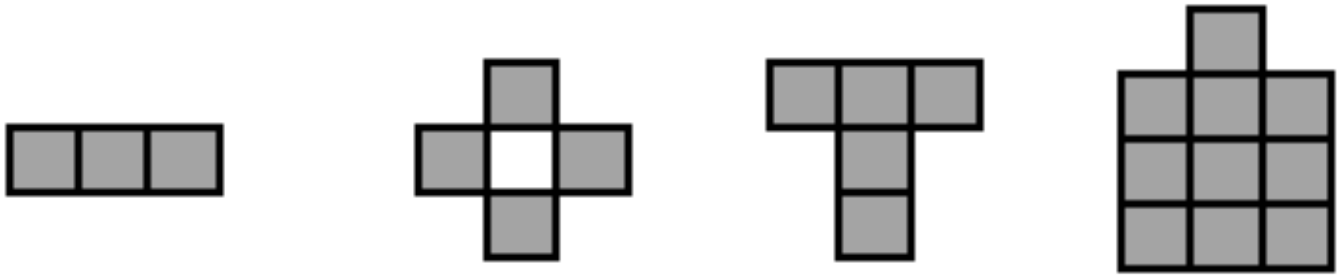
- Opening and Closing are duals of each other with respect to set complementation and reflection

$$(A \bullet B)^c = A^c \circ \hat{B}$$

$$(A \circ B)^c = A^c \bullet \hat{B}$$

Hit-or-Miss Transformation ⊛ (HMT)

- find location of one shape among a set of shapes
- ”template matching”



- composite SE: object part (B1) and background part (B2)
- does B1 ***fits the object while, simultaneously,*** B2 misses the object, i.e., ***fits the background?***

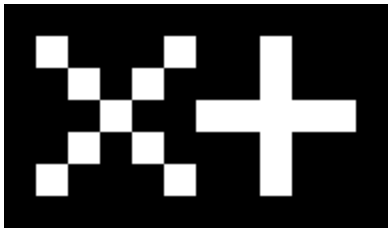
Hit-or-Miss Transformation

$$A \circledast B = (A \ominus B_1) \cap [A^c \ominus B_2]$$

$$A \circledast B = (A \ominus B_1) - (A \oplus \hat{B}_2)$$

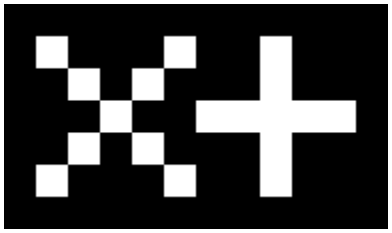
Hit-or-Miss transformation

Searching for white pixels, that do not have 4-connected neighboring pixels.

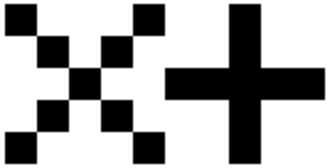


Original image (white pixels)

B^1

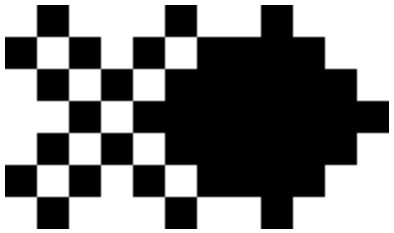


Erosion with B1

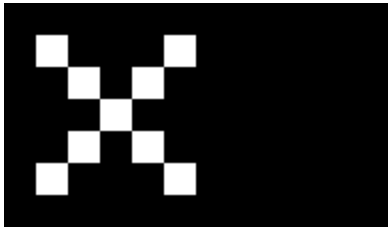


Complement

B^2



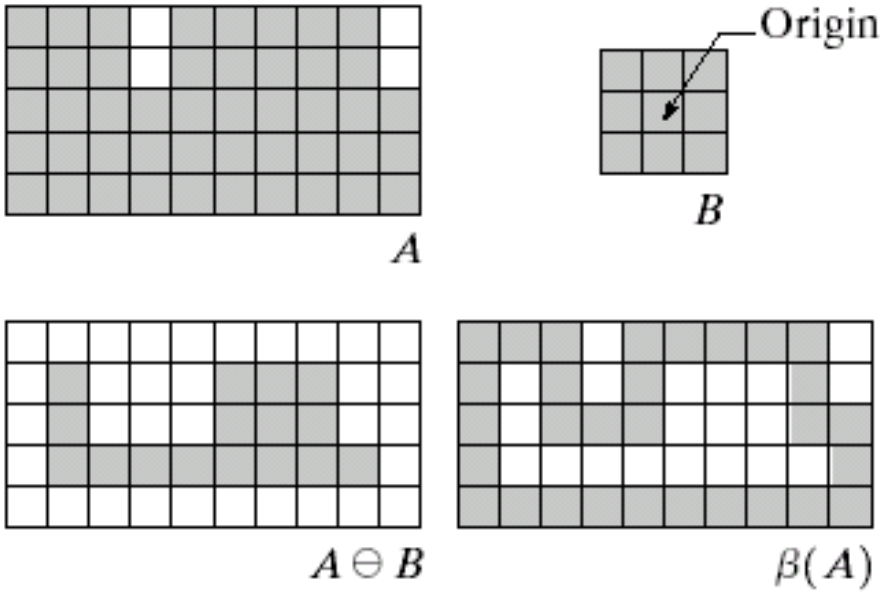
Erosion with B2



Application: Boundary Extraction

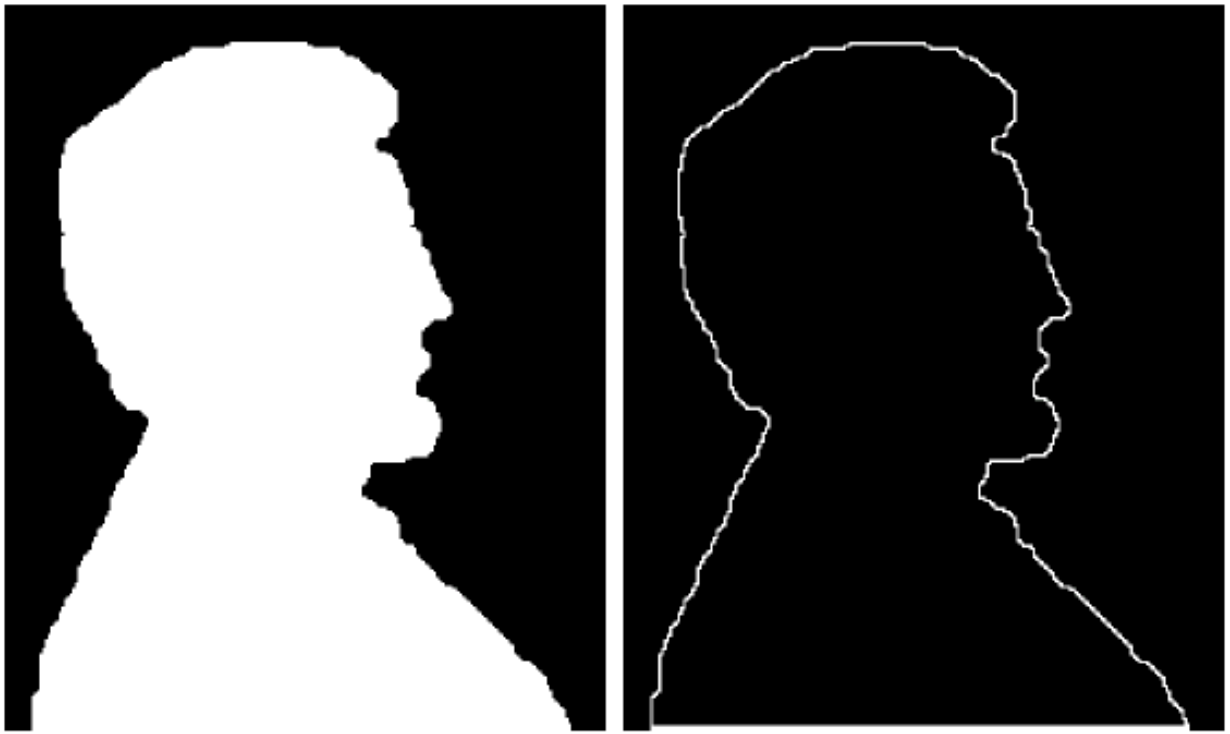
a b
c d

FIGURE 9.13 (a) Set A. (b) Structuring element B. (c) A eroded by B. (d) Boundary, given by the set difference between A and its erosion.



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

Boundary / Contour Extraction



a b

FIGURE 9.14
(a) A simple binary image, with 1's represented in white. (b) Result of using Eq. (9.5-1) with the structuring element in Fig. 9.13(b).

Application: Hole Filling

The key equation for region filling is

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

Where X_0 contains a starting point inside in each hole hole, B is a symmetric structuring element and A^c is the complement of A

This equation is applied repeatedly until X_k is equal to X_{k-1}

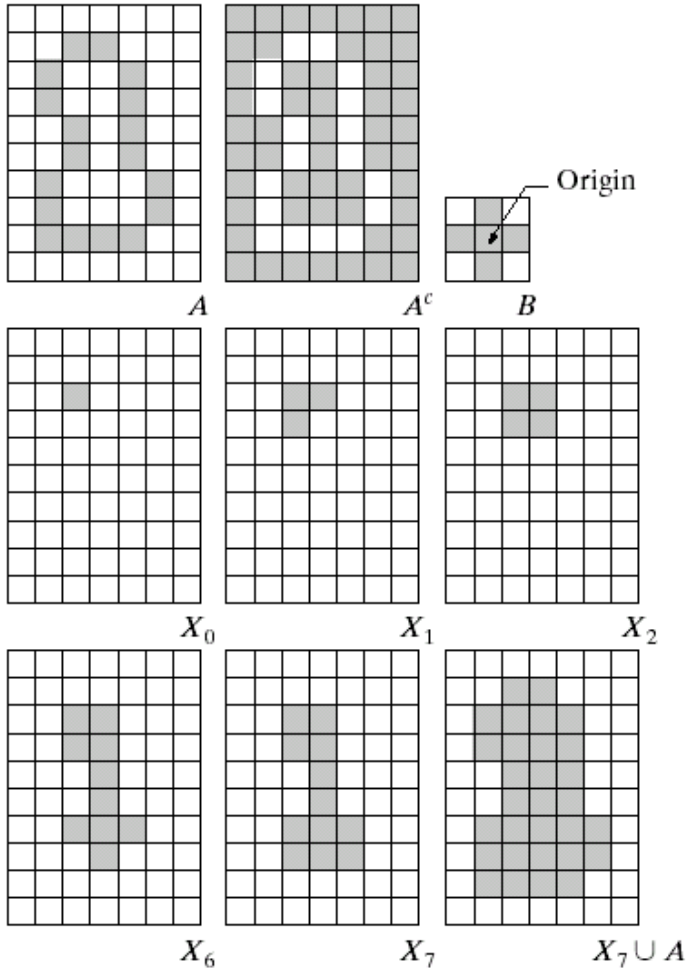
Finally, union of the result with A is performed.

Region Filling

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1,2,3,\ldots$$

a b c
d e f
g h i

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



Application: Extraction of connected components

- Y: A connected component in a set A.
- p: A point in Y.
- For extracting connected component Y perform the following iteratively with $X_0=p$:

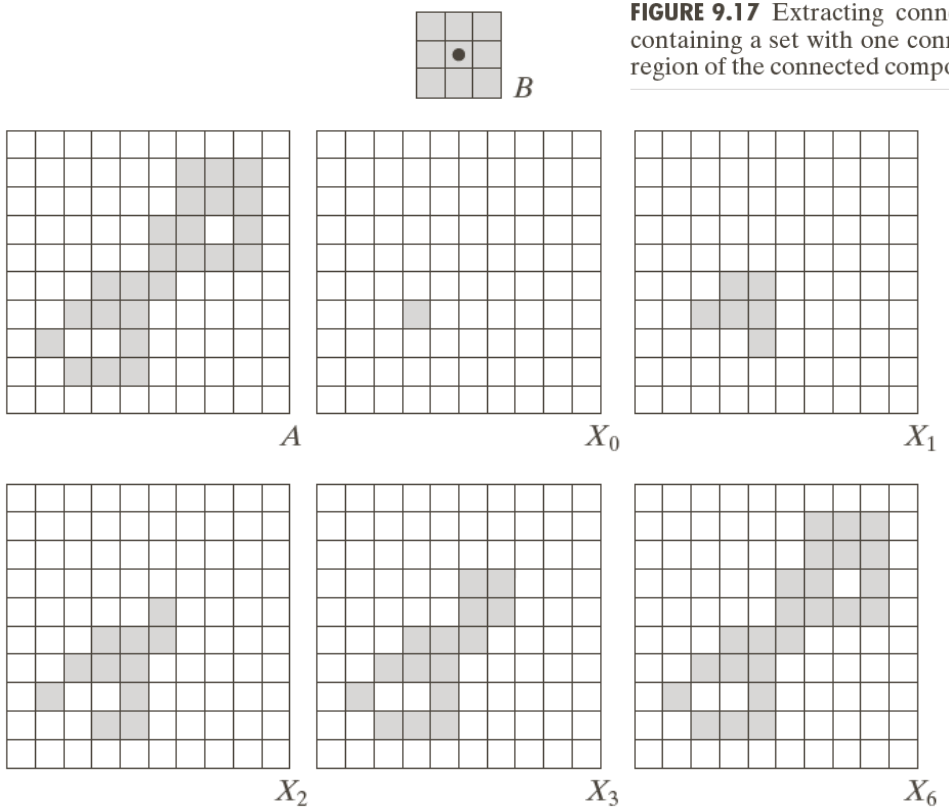
$$X_k = (X_{k-1} \oplus B) \cap A, \qquad k = 1,2,3$$

Terminates when $X_k=X_{k-1}$ to provide Y.

Connected components

a
b c d
e f g

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



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

Application: Convex Hull

- Convex Set
  - A set *A* is said to be convex if the straight line segment joining any two points in A lies entirely within A.
- Convex Hull: H = CH(S)
  - Minimal convex superset of S
- Continuous Algorithm
- Convex Deficiency: H - S

Courtesy: P.P. Das, Professor, Dept. of CSE, IIT Kharagpur

Convex Hull

Approximate solution!

- B^i, i=1,2,3, 4 : Four structuring elements
- Perform the following iterative construction for each structuring element to provide D^i at convergence.

$$X_k^i = \left(X_{k-1}^i \circledast B^i\right) \cup A, i = 1,2,3, and\ 4; k = 1,2,3, \ldots$$

The convex hull of A: $C(A) = \bigcup_{i=1}^{4} D^i$

Constraints: Length of vertical and horizontal to be less than 3.

Convex hull

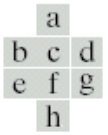
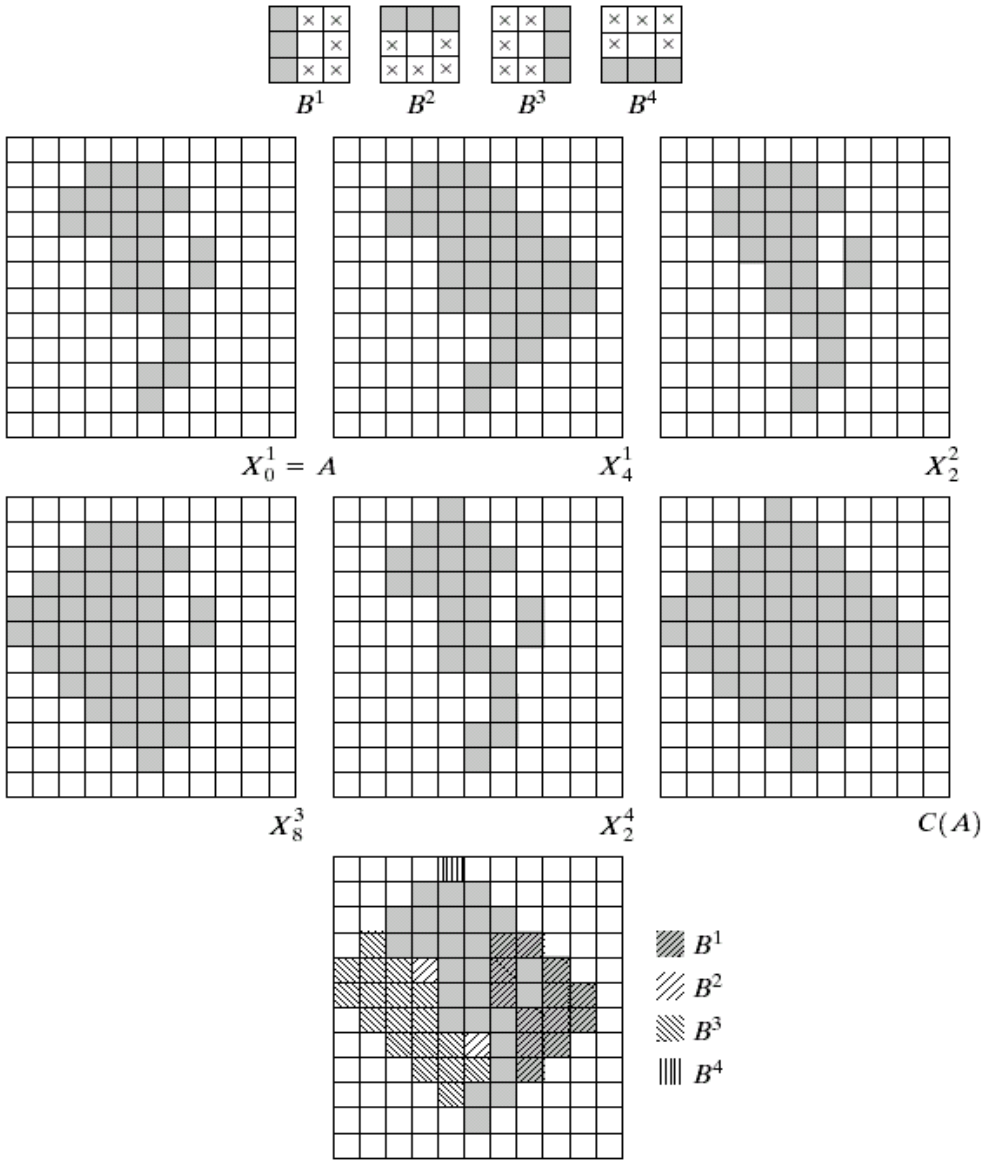


FIGURE 9.19
(a) Structuring elements. (b) Set A. (c)–(f) Results of convergence with the structuring elements shown in (a). (g) Convex hull. (h) Convex hull showing the contribution of each structuring element.



Convex Hull

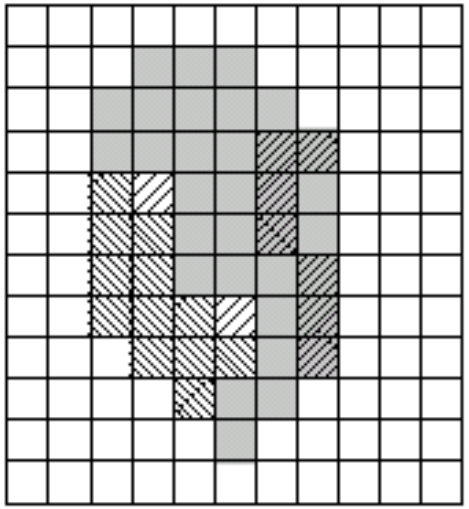


FIGURE 9.20 Result of limiting growth of convex hull algorithm to the maximum dimensions of the original set of points along the vertical and horizontal directions.

Thinning

- The thinning of a set A by a structuring element B.

$$A \otimes B = A - (A \circledast B)$$

- A more useful expression using an alternate sequence of structuring elements till convergence.

$$\{B\} = \{B^1, B^2, \ldots., B^n\}$$

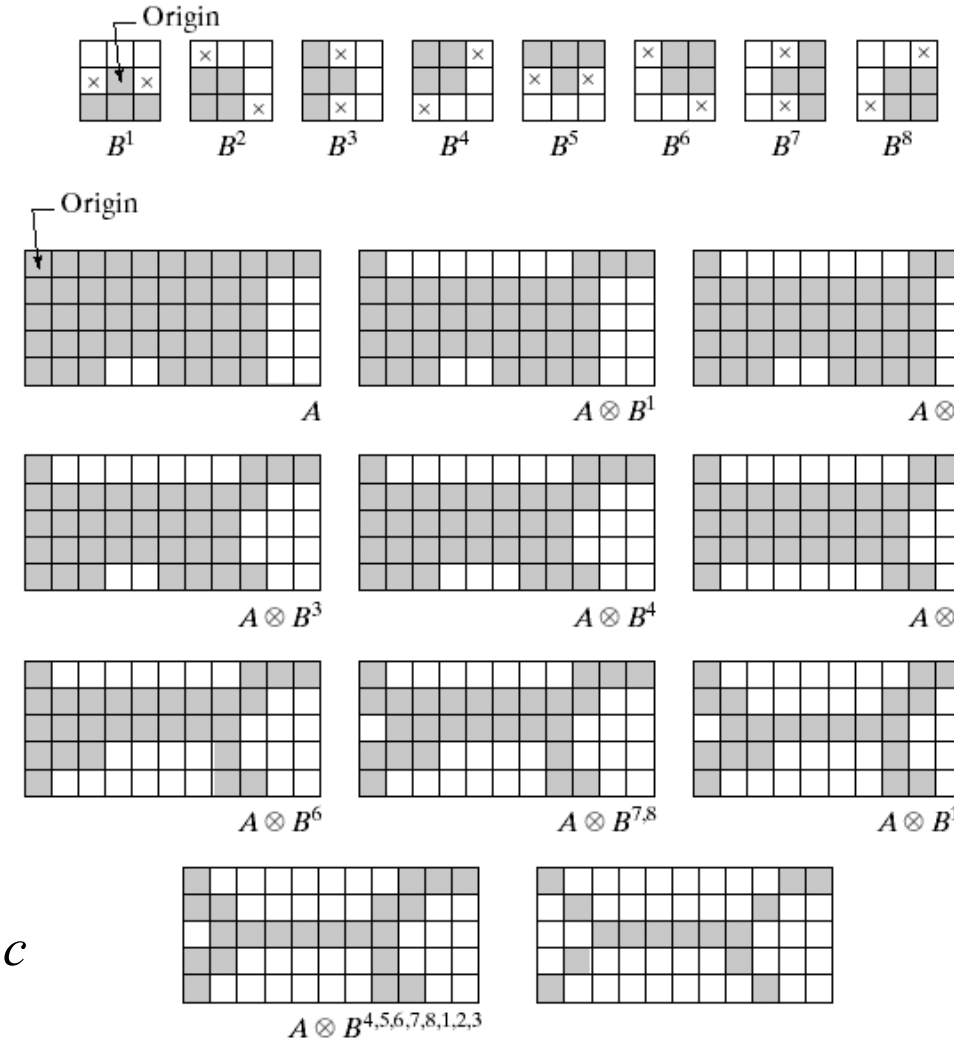
  - Where B^i is a rotated version of B^{i-1}.

$$A \otimes \{B\} = \left(\left(\cdots \left((A \otimes B^1) \otimes B^2 \right) \cdots \right) \otimes B^n \right)$$

Thinning

$$A \otimes B = A - (A \circledast B)$$

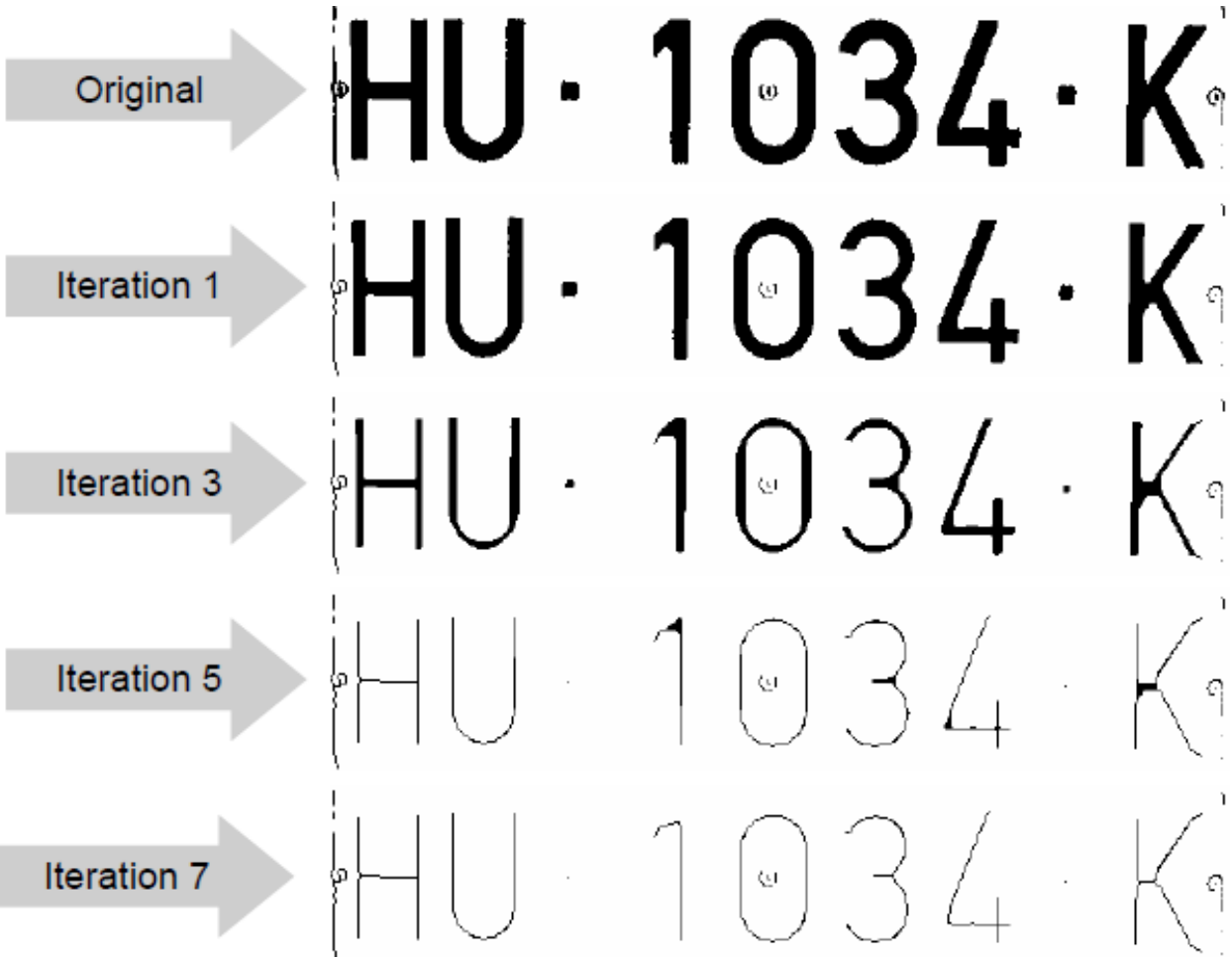
$$= A \cap (A \circledast B)^c$$



a
b c d
e f g
h i j
k l

FIGURE 9.21 (a) Sequence of rotated structuring elements used for thinning. (b) Set A. (c) Result of thinning with the first element. (d)–(i) Results of thinning with the next seven elements (there was no change between the seventh and eighth elements). (j) Result of using the first element again (there were no changes for the next two elements). (k) Result after convergence. (l) Conversion to m-connectivity.

Thinning



Courtesy: R.C. Gonzalez and R.E Woods © 1992-2008

Thickening

- Thickening morphological dual of thinning.

$$A \odot B = A \cup (A \circledast B)$$

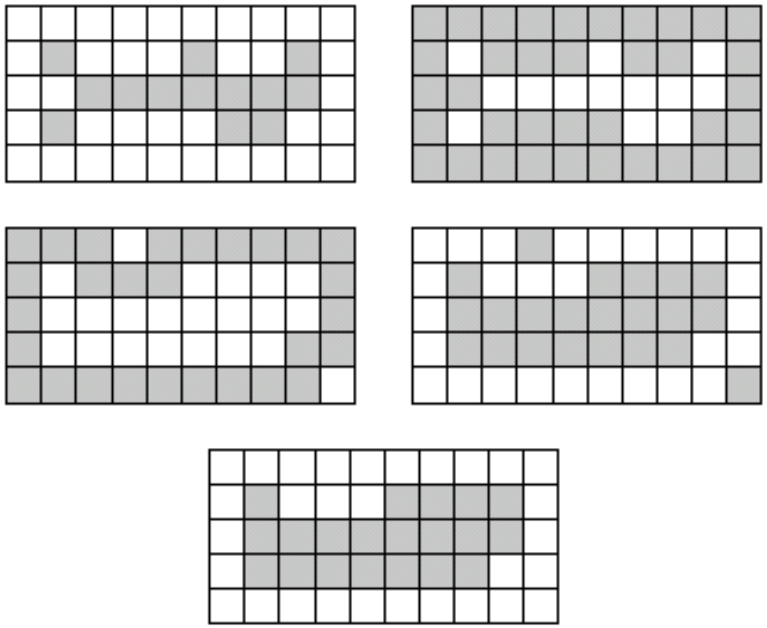
  - The SE has the same form but 1's (foreground) and 0's (background) interchanged.
- As in thinning, thickening also performed using an alternate sequence of rotated SEs till convergence.

$$A \odot \{B\} = \left(\left(\cdots \left((A \odot B^1) \odot B^2 \right) \cdots \right) \odot B^n \right)$$

- Equivalent to thinning background of the thinned pattern and then taking the complement.
  - The thinned background forms the boundary of the thickened object.

Thickening

$$A \odot B = A \cup (A \circledast B)$$

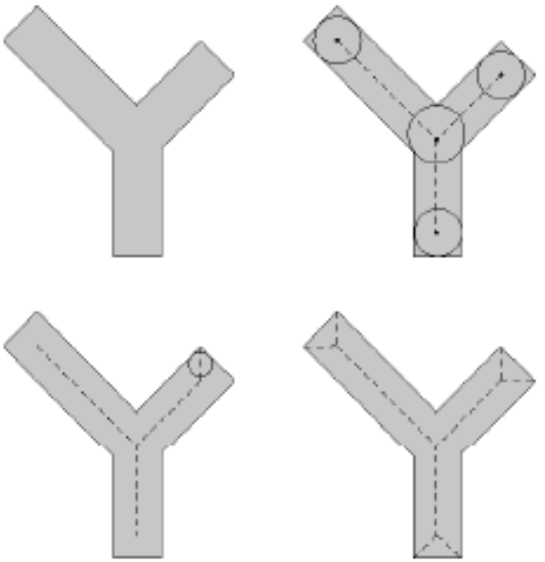


a b
c d
e

FIGURE 9.22 (a) Set A. (b) Complement of A. (c) Result of thinning the complement of A. (d) Thickened set obtained by complementing (c). (e) Final result, with no disconnected points.

Skeletons

- Skeleton, S(A), of a set A
  - (a) If **z** is a point of S(A) and $(D)_z$ is the largest disk centered at **z** and contained in A, one cannot find a larger disk containing $(D)_z$ and included in A. The disk $(D)_z$ is called a maximum disk
  - (b) The disk $(D)_z$ touches the boundary of A at two or more different places
- An inner point belongs to the skeleton if it has at least two cloest boundary points



a b
c d

FIGURE 9.23
(a) Set *A*.
(b) Various positions of maximum disks with centers on the skeleton of *A*.
(c) Another maximum disk on a different segment of the skeleton of *A*.
(d) Complete skeleton.

Skeletons

- Morphological Skeleton: $S(A) = \bigcup_{k=0}^{K} S_k(A)$
  - Where, $S_K(A) = (A \ominus kB) - (A \ominus kB) \circ B$

    k successive erosion.
  - k is the last iterative step before A erodes to an empty set.
- The set A can be reconstructed by

$$A = \bigcup_{k=0}^{K} (S_k(A) \oplus kB)$$

k successive dilation.

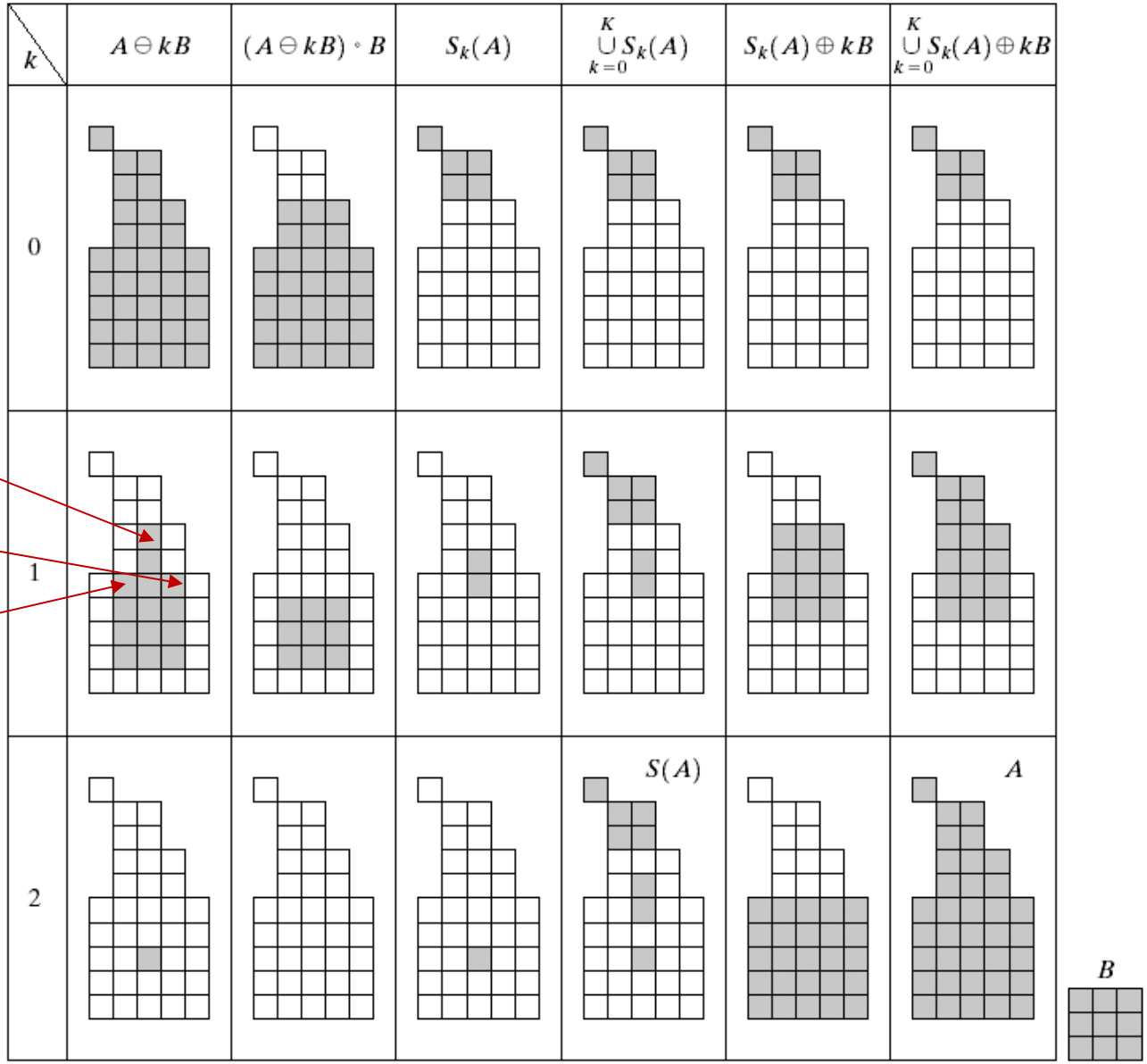


FIGURE 9.24 Implementation of Eqs. (9.5-11) through (9.5-15). The original set is at the top left, and its morphological skeleton is at the bottom of the fourth column. The reconstructed set is at the bottom of the sixth column.

These pixels wrongly shown as object points after the erosion with radius 1.

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Summary

- Binary or Bilevel images: f: Z^2 (or Z^3 or Z^n)→ {0/1}
  - Background (0) and Foreground or Object Point (1)
- Neighbors, Connectivity, Paths and Distances
  - 2D: 4 / 8 -neighbors 3D: 6 / 18 / 26 -neighbors.
  - Multiple paths of shortest distances exist.
  - Jordan's Curve Theorem breaks.
    - 4-8 or (8-4) Grids for complementary adjacency in background and foreground.
  - Component Labeling: Chamfering (Linear Time)
  - Different types of distances
    - 4 (/8)-Neighbor in 2D and 6(/18/26)-Neighbor in 3D
    - Octagonal distances
    - Weighted distances

Summary

- Distance Transform
  - Chamfering Algorithm for Additive Distances
- Medial Axis Transform
  - Applications: Transformation, Cross-sections, Skeleton
- Shape representation.
  - Contour following, Chain Codes
  - Polygonization
    - Minimum-perimeter polygon (MPP), ADSS algorithm, Merging and Splitting
  - Skeletonization: Thinning
- Morphological Operations
  - Dilation, Erosion, Opening, Closing, Hit-&-Mis Transform
  - Applications: Smoothing, Convex Hull, Region Filling, Thinning, Connected Component Extraction, Thinning

Thank You