|| Shri Hari ||

BCSE0101: DIGITAL IMAGE PROCESSING

MODULE II

Prof. Charul Bhatnagar
Room no. – 325, AB – I
WA: 7055510537

MODULE II

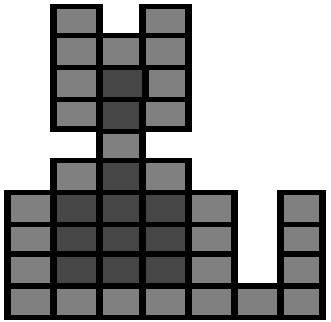
Morphological Image Processing: Introduction, Logical Operations involving Binary Images, Dilation and Erosion, Opening and Closing, The Hit-or-Miss Transformation, Morphological Algorithms – Boundary Extraction, Region Filling, Extraction of Connected Components, Convex Hull, Thinning, Thickening.

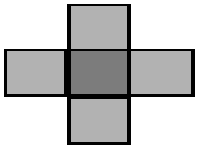
Image Segmentation: Point, Line & Edge detection, Thresholding, Region-based Segmentation, Region Extraction - Pixel Based Approach & Region Based Approach, Edge and Line Detection - Basic Edge Detection, Canny Edge Detection, Edge Linking - Hough Transform.

Representation & Description: Representation - Boundary Following, Chain Codes; Boundary Descriptors – Shape Numbers.

BASIC IDEA OF SE

- for each pixel in binary image:
  - check if SE is "satisfied"
  - output pixel is set to 0 or 1 depending on the operation used





Output is 1 if the SE fits in

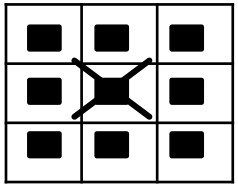
BASIC COMPONENTS IN MORPHOLOGY

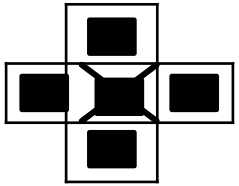
Every Operation has two elements.

1) Input Image

2) Structuring element

The results of the operation mainly depends upon the structuring element chosen.





Origin is not a member of the structuring element

Typical Structuring elements

X denotes the origin of the structuring element

EROSION

- Shrinks or thins objects in a binary image
- Translate structuring element, check if it entirely fits within foreground of image

$$A \ominus B = \{z | b + z \in A \quad for\ every\ b \in B\}$$

- The mathematical defination is

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

Let,
A = {(1,0), (1,1), (1,2), (0,3), (1,3), (2,3), (3,3), (1,4)}
B = {(0,0), (1,0)}

$$A \ominus B = \{(0,3), (1,3), (2,3)\}$$

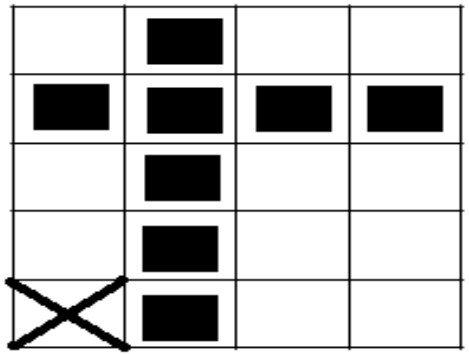
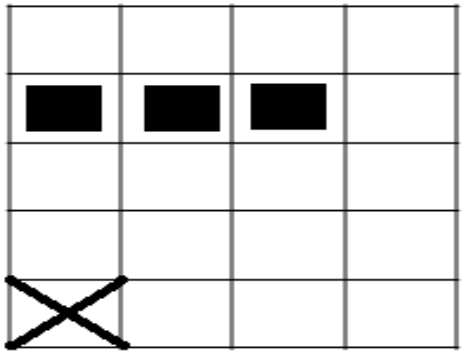


Image A

Image B
Structuring element



$A \ominus B$

Erosion

DILATION

- **Dilation** is an operation that *grows* or *thickens* objects in a binary image.
- The specific manner of this thickening is controlled by the shape of ***structuring element***.

Dilation combines two sets using vector addition

$(a_1, a_2) + (b_1, b_2) = (a_1 + b_1,\ a_2 + b_2)$

$$A \oplus B = \{p \in Z^2 \mid p = a + b, a \in A \ \& \ b \in B\}$$

Let,
A = {(1,0), (1,1), (1,2), (2,2), (0,3), (0,4)}
B = {(0,0), (1,0)}

$$A \oplus B = \{(1,0), (1,1), (1,2), (2,2), (0,3), (0,4), (2,0), (2,1), (2,2), (3,2), (1,3), (1,4)\}$$

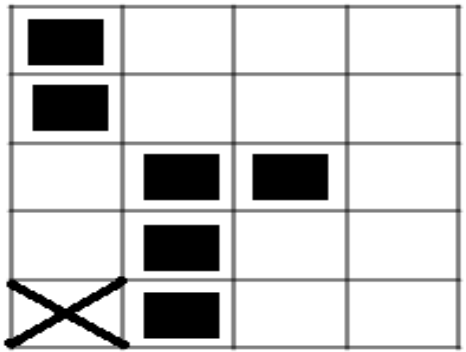
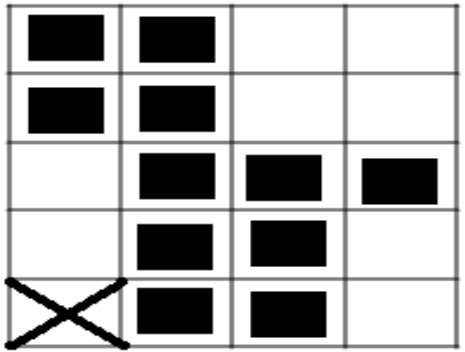


Image A

Image B
Structuring element



$A \oplus B$
Dilation

Dilation
(Another way of defining it)

$$A \oplus B = \{ z \mid (\hat{B})_z \cap A \neq \emptyset \}$$

Theoretical way of generation:

- Obtain the reflection of B about its origin
- Shift this reflection by z
- The dilation of A by B then is set of all displacements z, such that B^ and A overlap by at least one element

- The ***reflected structuring element*** is translated throughout the domain of the image to see where it overlaps with 1-valued pixels. The output image is 1 at each location of the origin s.t. the reflected *structuring element* overlaps at least one 1-valued pixel in the input image.

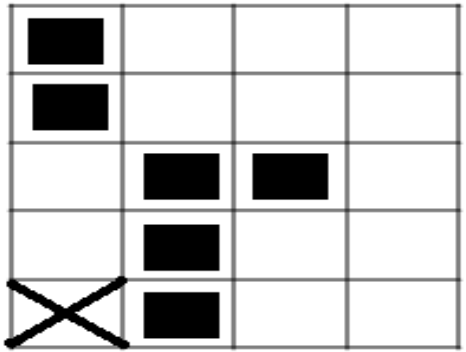
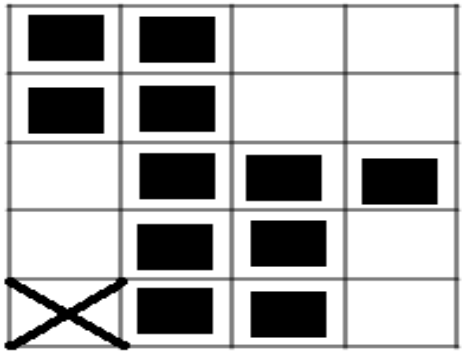


Image A

SE

Reflected SE



$A \oplus B$

Dilation

DILATION & EROSION ARE DUALS OF EACH OTHER WRT SET COMPLEMENTATION & REFLECTION

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

Proof:

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

If $(B)_z$ is contained in set A, then

$$(B)_z \cap A^c = \emptyset$$

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

But the complement of the set of z's that satisfy

$$(B)_z \cap A^c = \emptyset$$

is the set of z's that satisfy

$$(B)_z \cap A^c \neq \emptyset$$

Thus,

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

$$= A^c \oplus \hat{B}$$

Remember???

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

Dilation & Erosion are not inverse transformations

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

Combining Dilation and Erosion

- In practical applications, dilation and erosion are used most often in various combinations.
- Three of the most common combinations of dilation and erosion are
  - *Opening*
  - *Closing*
  - *Hit-or-miss transformation*

OPENING - EROSION FOLLOWED BY DILATION

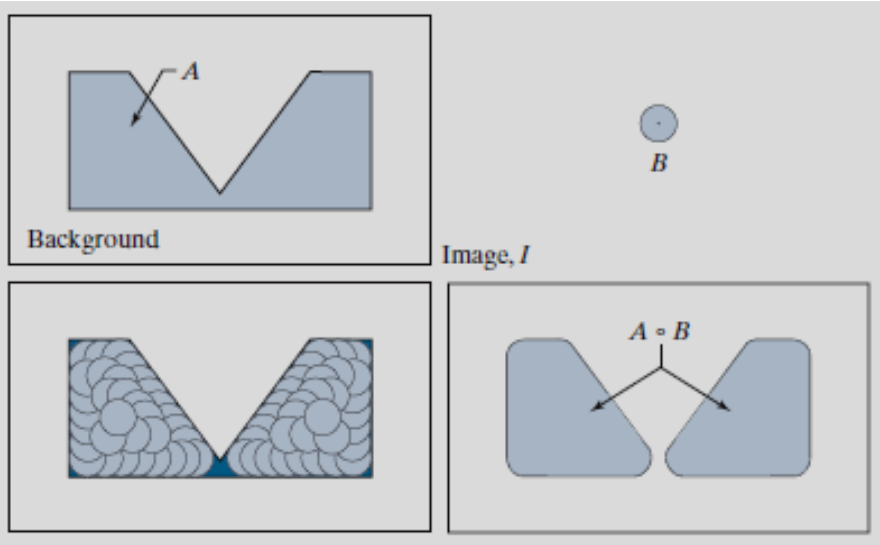
$$A \circ B = (A \ominus B) \oplus B$$

Opening

- Generally smoothes the contours of an object
- Breaks narrow isthmuses
- Eliminates thin protrusions
- It is less destructive than the Erosion

GEOMETRICAL INTERPRETATION OF OPENING

The opening of *A by B is* the union of all the translations of *B so that B fits entirely in A.*



a) Image *I*, composed of set (object) *A and* background.
b) Structuring element, *B*.
c) Translations of *B while being* contained in *A*. *(A* is shown dark for clarity.)
d) Opening of *A* by *B*.

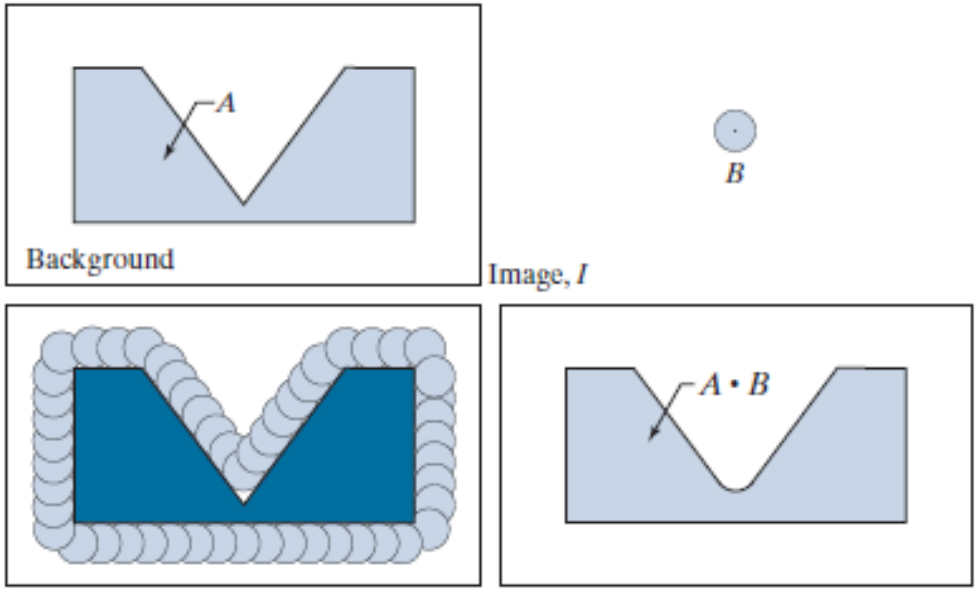
CLOSING - DILATION FOLLOWED BY EROSION

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

Closing

- Also tends to smooth sections of the contours
- Generally fuses narrow breaks & long thin gulfs
- Eliminates small holes & fills gaps in the contour

GEOMETRIC INTERPRETATION OF CLOSING



a) Image *I*, composed of set (object) *A, and* background.

b) Structuring element *B*.

c) Translations of *B* such that *B does not* overlap any part of *A*. *(A is shown* dark for clarity.)

d) Closing of *A* by *B*.

Opening & Closing Are Dual Transformations

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

PROPERTIES

Opening

a) $A \circ B$ *is a subset of* A.

b) If C *is a subset of* D, *then* $C \circ B$ *is a subset of* $D \circ B$.

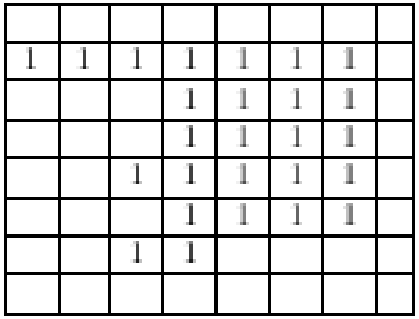
c) $(A \circ B) \circ B = A \circ B$.

Closing

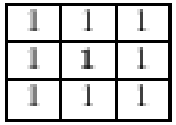
a) A *is a subset of* $A \bullet B$.

b) If C *is a subset of* D, *then* $C \bullet B$ *is a subset of* $D \bullet B$.

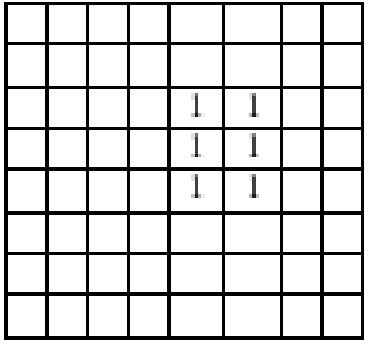
c) $(A \bullet B)\ B = A \bullet B$.



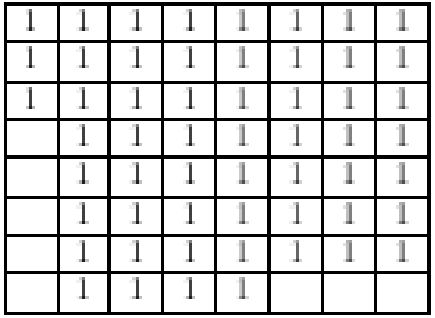
a) Binary image B



b) Structuring Element S



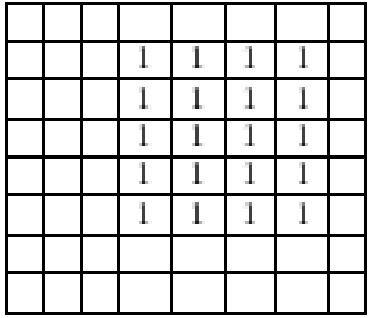
d) Erosion $B \ominus S$



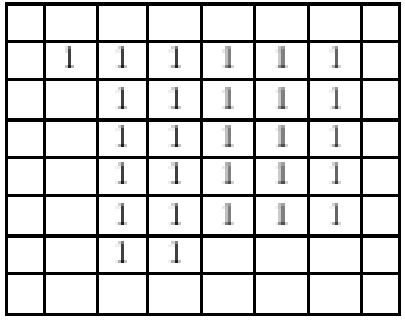
c) Dilation $B \oplus S$

What will be the result of

i. Erosion
ii. Dilation
iii. Opening
iv. Closing



f) Opening $B \circ S$



e) Closing $B \bullet S$

THE HIT-OR-MISS TRANSFORM (HMT)

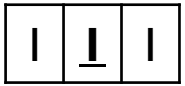
- *HMT is a basic tool for shape detection.*
- Let *I be a binary image composed of foreground (A) and background pixels, respectively.*
- HMT utilizes *two structuring elements:*
  - *B1 for detecting shapes in the foreground*
  - *B2 for detecting shapes in the background*

$$I \circledast B_{1,2} = \left\{ z \middle| (B_1)_z \subseteq A \text{ and } (B_2)_z \subseteq A^c \right\}$$
$$= (A \ominus B_1) \cap \left(A^c \ominus B_2 \right)$$

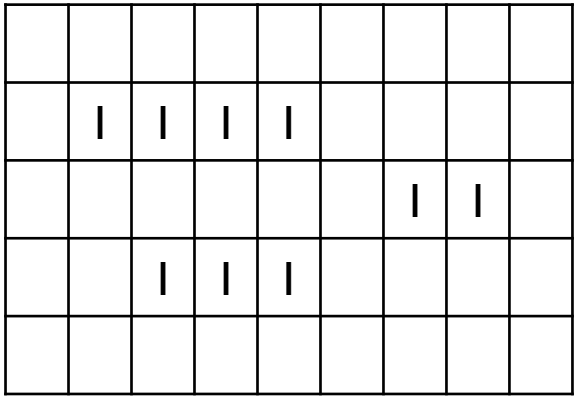
$B_1 = [1\ 1\ 1]$

Eroding the image with a 1×3 *mask that corresponds to the shape* of the object

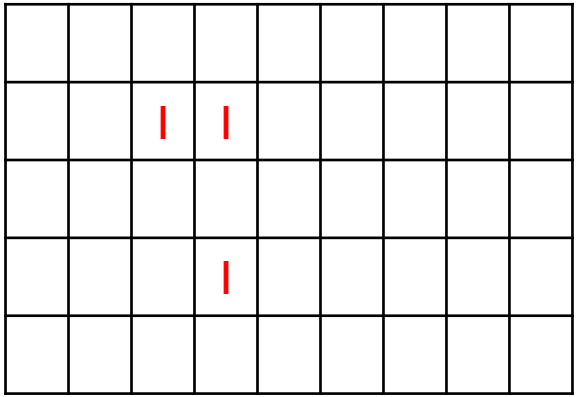
- all objects that are smaller than the target object are removed
- But all objects that are larger than the mask, i. e., where the shifted mask is a subset of the object are retained



SE B_1



A



$A \ominus B_1$

We need a second operation to remove all objects larger than the target object.

- Can be done by analyzing the background of the original binary image.
- Use as a second step an erosion of the background with a 3 × *5 mask* B_2 in which all coefficients are zero except for the pixels in the background, surrounding the object.
- This is a negative mask for the object

1	1	1	1	1
1	0	**0**	0	1
1	1	1	1	1

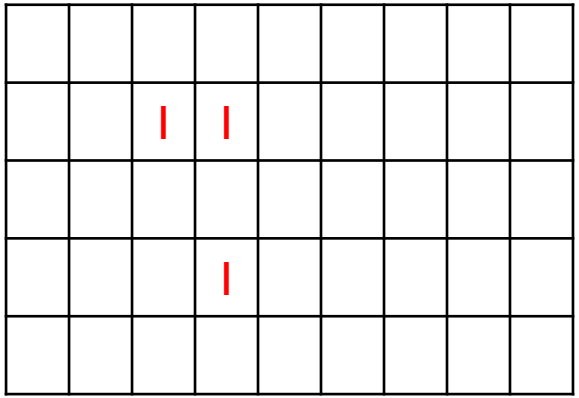
SE B_2

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

A^C

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

$A^C \ominus B_2$

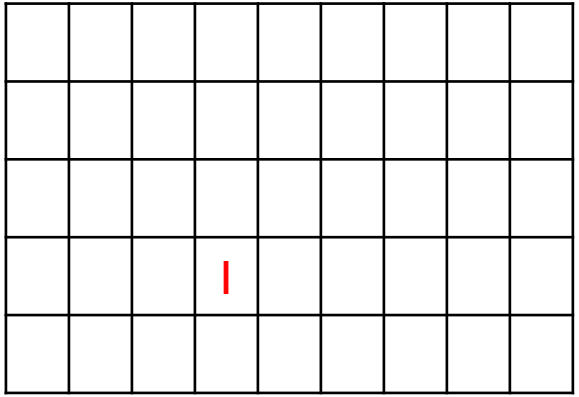


$A \ominus B_1$

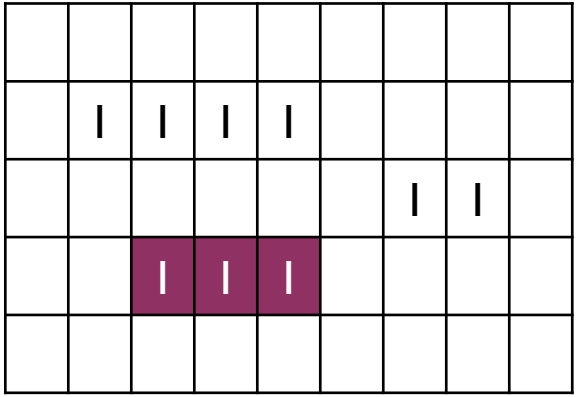
$\cap$

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

$A^C \ominus B_2$



$A \circledast B$



As the hit and miss masks of the hit-miss operator are disjunct, they can be combined into one mask using a hit (1), miss (0), and don't care (*x*) *notation.*

0	0	0	0	0
0	1	**1**	1	0
0	0	0	0	0

What will be the mask for detecting isolated points?

$$\boldsymbol{M}_I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

What will be the mask for detecting objects with horizontal rows of 3 to 5 consecutive pixels?

$$\boldsymbol{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x & 1 & 1 & 1 & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

|| Shri Hari ||

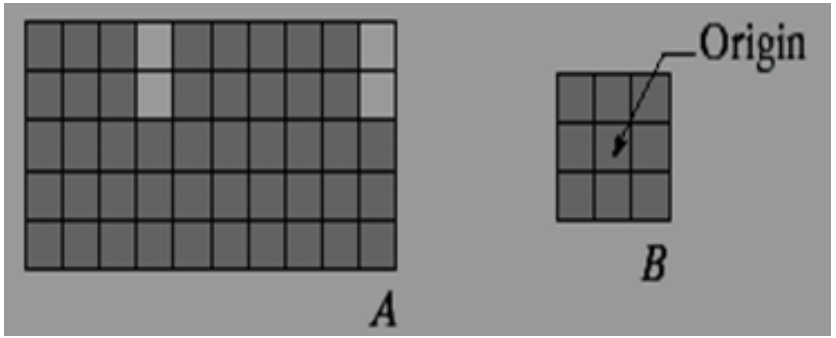
MORPHOLOGICAL ALGORITHMS

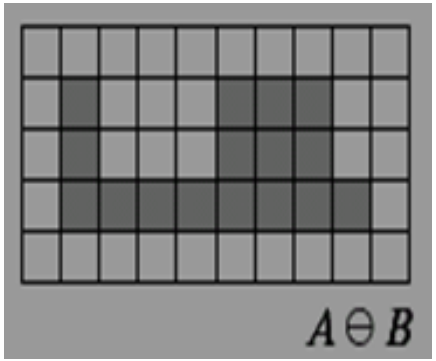
Some practical uses of morphology

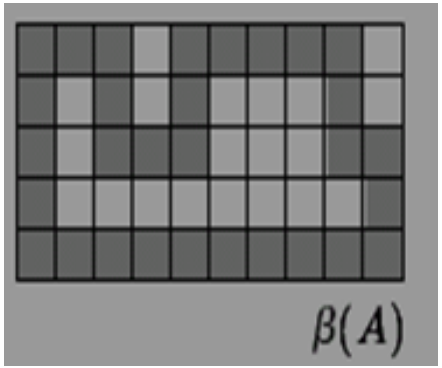
BOUNDARY EXTRACTION

1. Erode A by B
2. Perform set difference between A & its erosion

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







EXAMPLE

Input image

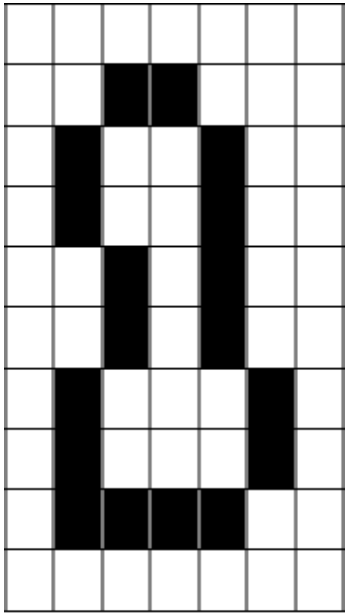
Output Image

Structure element used is a 3X3 all 1s matrix

HOLE FILLING / REGION FILLING

Hole - a background region surrounded by a connected border of foreground pixels.

- Let A denote a set whose elements are 8-connected boundaries, with each boundary enclosing a background region (i.e., a hole).
- **Given a point in each hole, the objective is to fill all the holes with foreground elements (1's).**

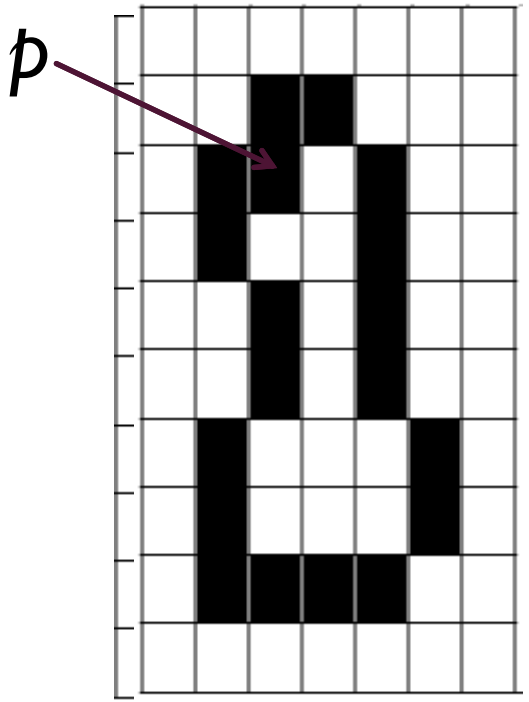


Set A

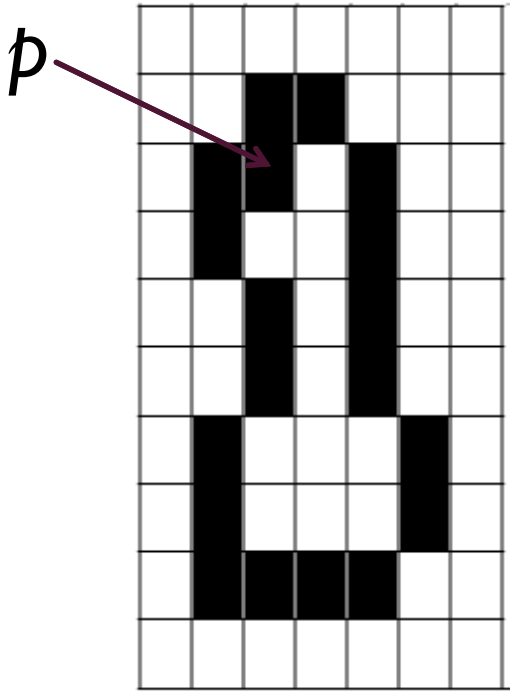
- All non-boundary points are labelled 0.
- Arbitrarily select a point p, inside the boundary and assign it value 1
- The objective is to fill the entire region with 1s.
- The region is filled with 1s by the following procedure.

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

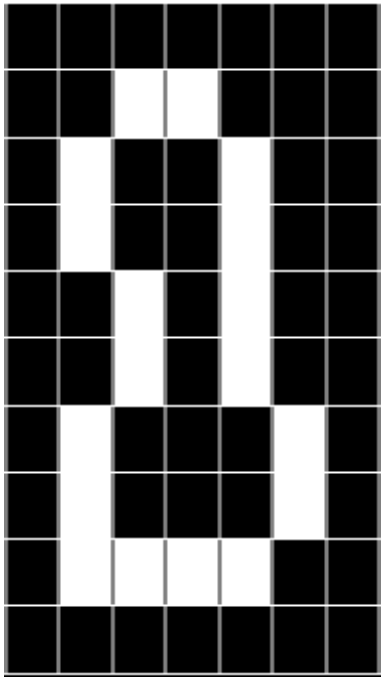
where $X_0 = p$



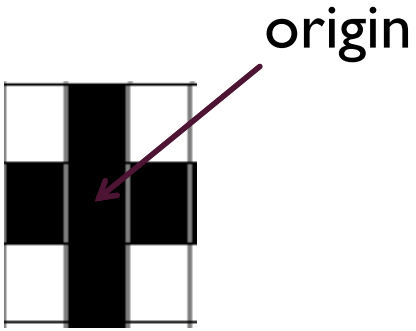
Set A



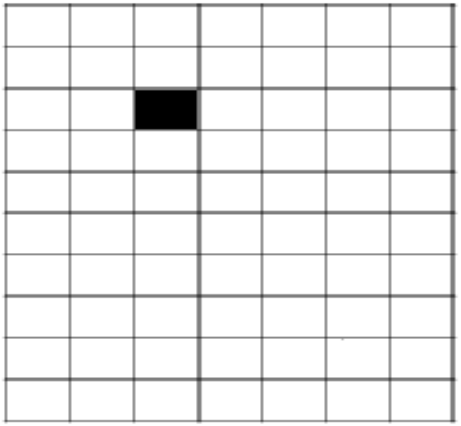
Set A



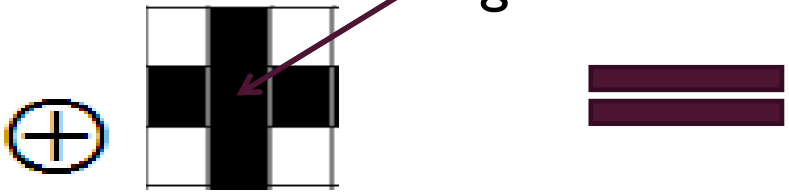
Set A^c



Structuring element B

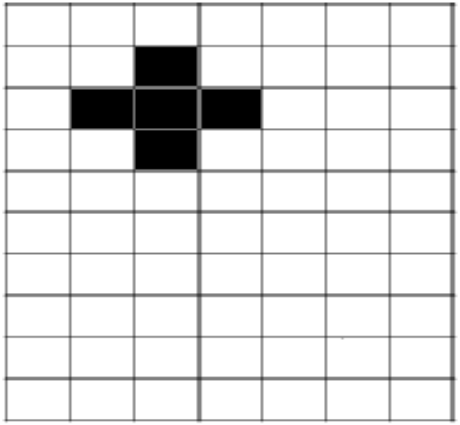


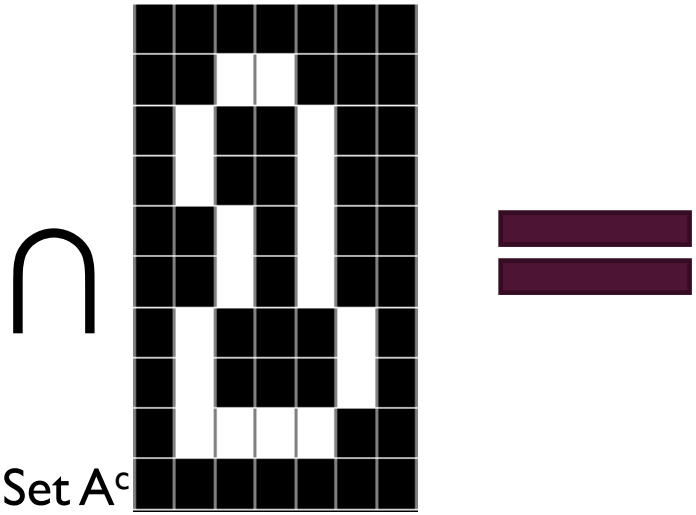
origin



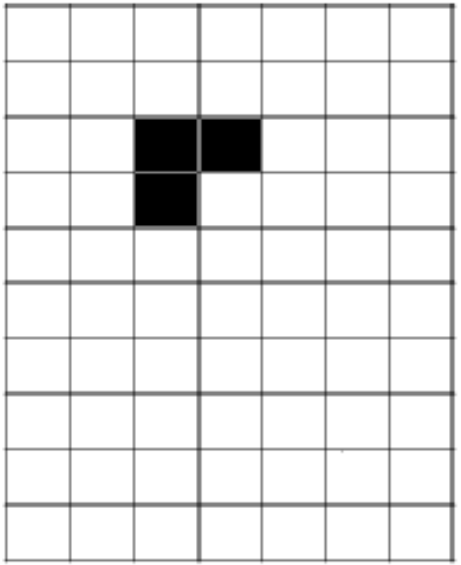
Structuring element B

X_0





Set A^c

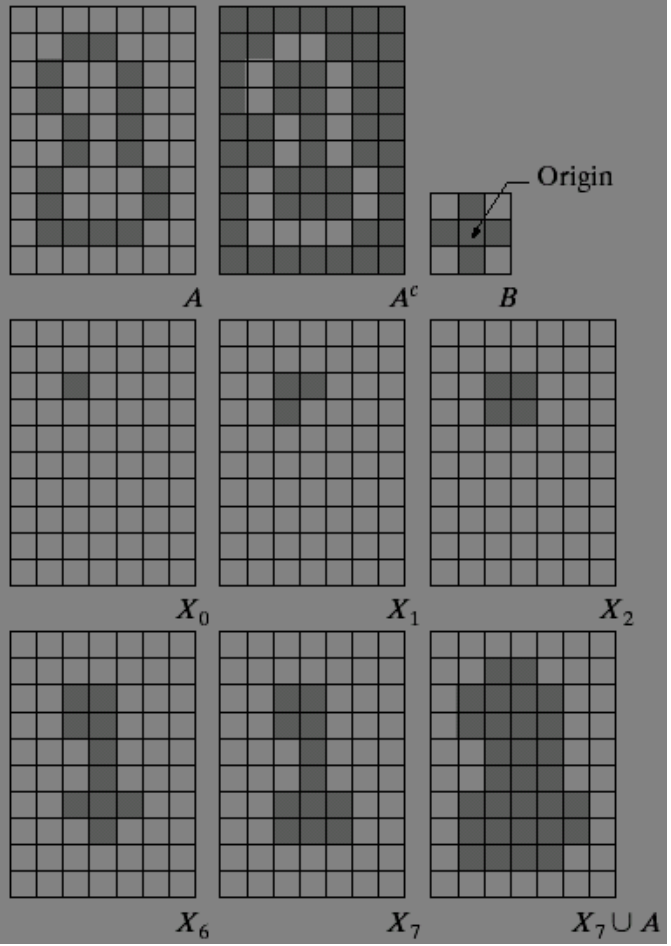


- The algorithm terminates at iteration step k if $X_k = X_{k-1}$.
- The set union of X_k and A contains the filled set & its boundary.

The dilation process would fill the entire area if left unchecked.
The intersection with A^c at each step limits the result to inside the region of interest.

a b c
d e f
g h i

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)].

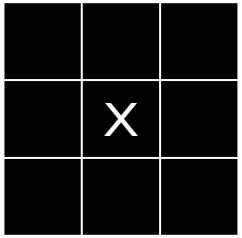


EXTRACTION OF CONNECTED COMPONENTS

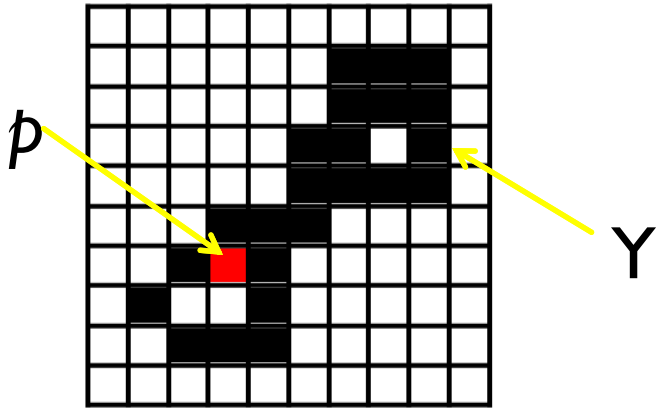
- Extraction of connected components in a binary image is central to many automated image analysis applications
- Let A be a set and Y be a connected component in it.
- Point p of Y is known.
- The following iterative expression yields all the elements of Y.

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

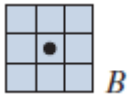
$X_0 = p$



Structuring element B



Set A



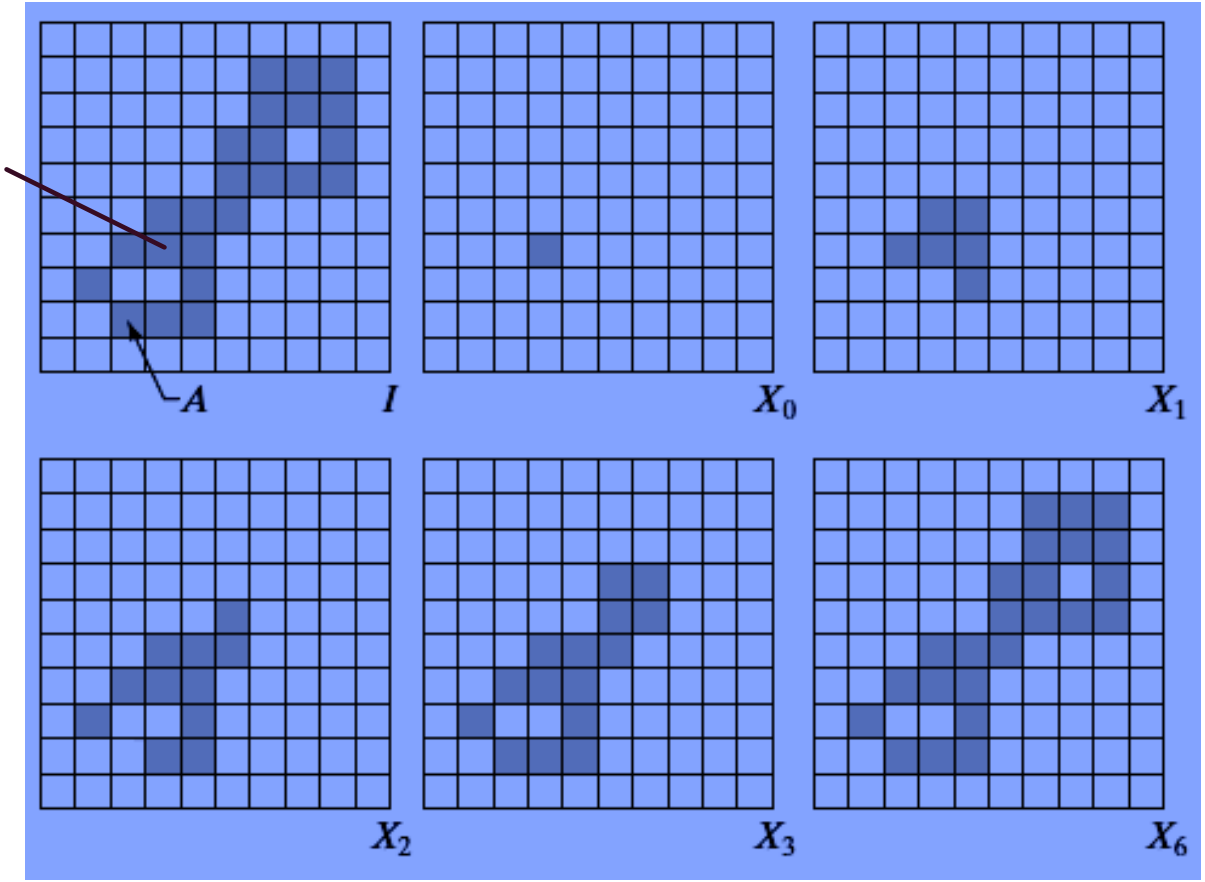
a) Structuring element.

P

b) Image containing a set with one connected component.

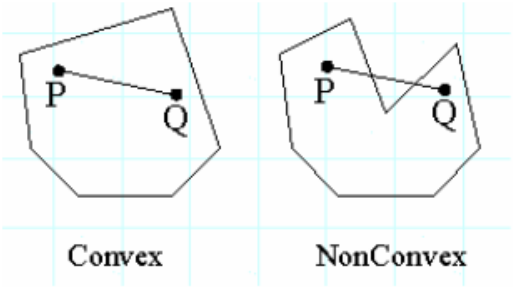
c) Initial array containing a 1 in the region of the connected component.

d) – (g) Various steps in the iteration



CONVEX HULL

- Set A is said to be *convex, if the straight line segment joining any two points in A lies entirely within A*
- The *convex hull H* of an arbitrary set S is the smallest convex set containing S
- The set difference H – S is called the *convex deficiency* of S
- Convex hull & Convex deficiency are useful for object description



ALGORITHM FOR OBTAINING CONVEX HULL, C(A), OF A SET A

- There are four SEs (convex hull masks).

1	x	x
1	0	x
1	x	x

B^1

1	1	1
x	0	x
x	x	x

B^2

x	x	1
x	0	1
x	x	1

B^3

x	x	x
x	0	x
1	1	1

B^4

- Following equations are iteratively applied

  - $X_0^i = A$
  - $X_k^i = \left(X_k^i * B^i\right) \cup A \quad i = 1, 2, 3, 4 \quad \text{and} \quad k = 1, 2, 3, \cdots$

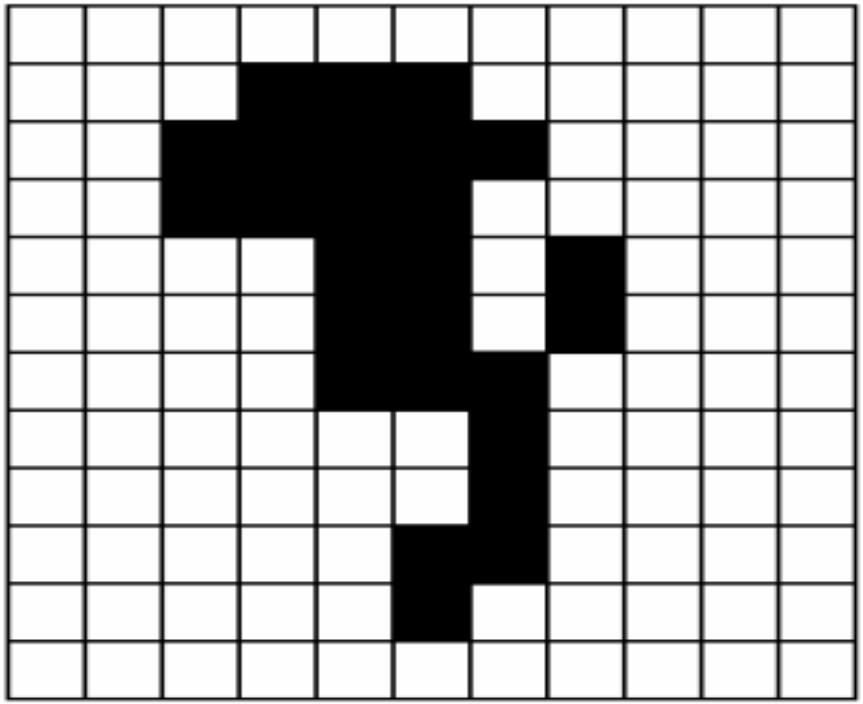
The procedure converges using the i[th] structuring element when

$$\mathbf{X_k^i = X_{k-1}^i \ \ we\ let\ \ D^i = X_k^i}$$

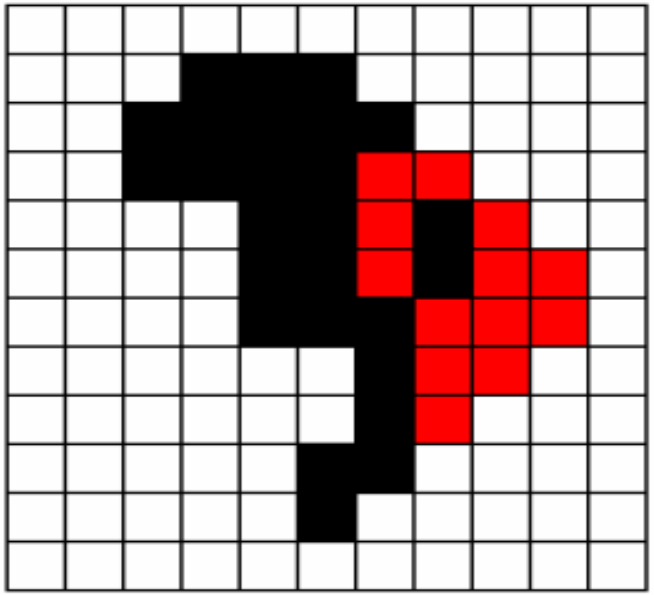
Then the convex hull of *A is the union of the four* results:

$$\mathbf{C(A) = \bigcup_{i=1}^{4} D^i}$$

- Original Black and White Image



- Mask B^1 applied to BW Image

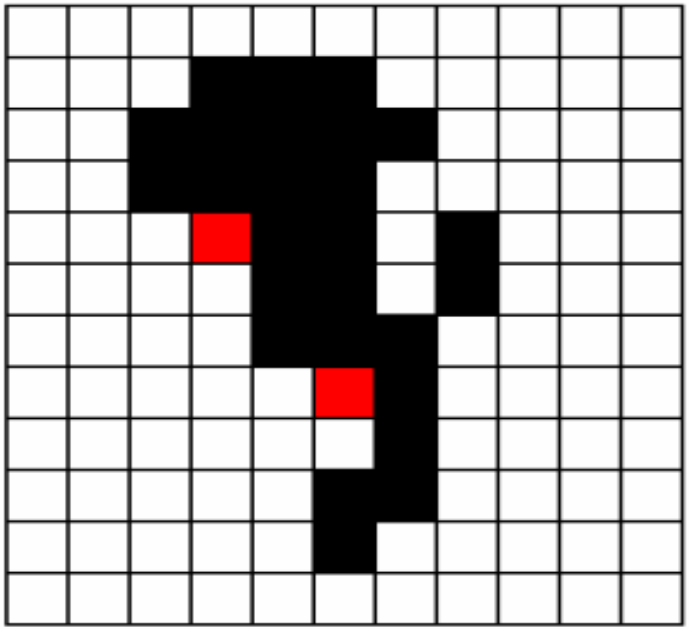


X_4^1

I	x	x
I	0	x
I	x	x

B^I

- Mask B^2 applied to BW Image

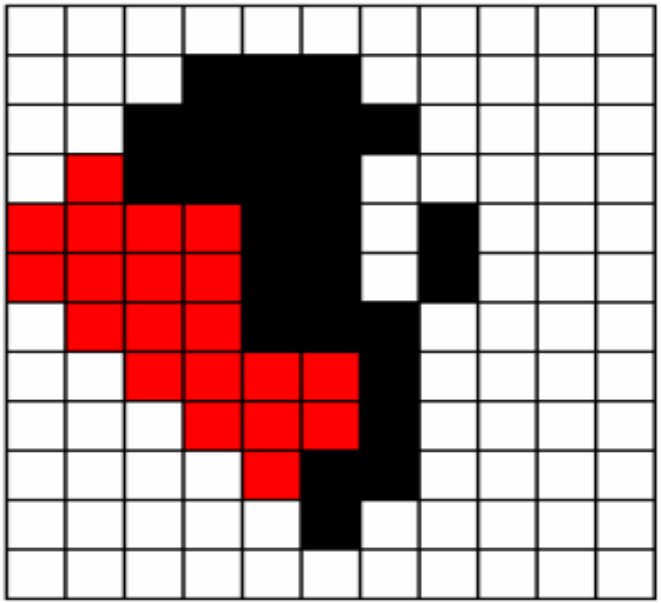


I	I	I
x	0	x
x	x	x

B^2

X_2^2

- Mask B^3 applied to BW Image

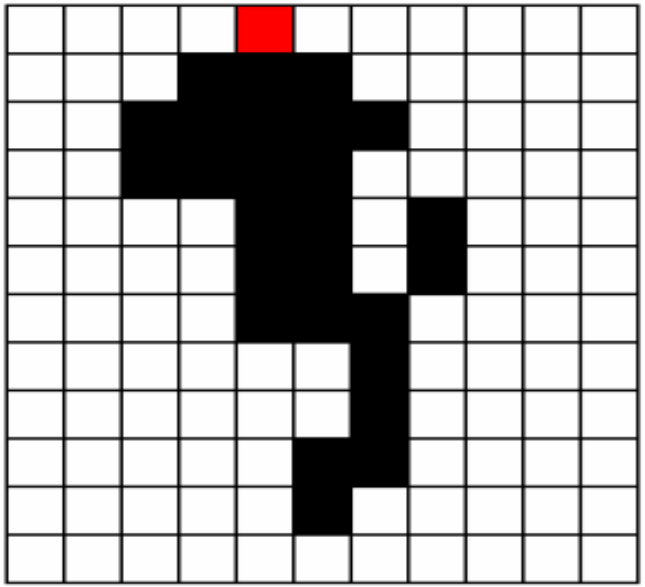


X_8^3

x	x	I
x	0	I
x	x	I

B^3

- Mask B4 applied to BW Image

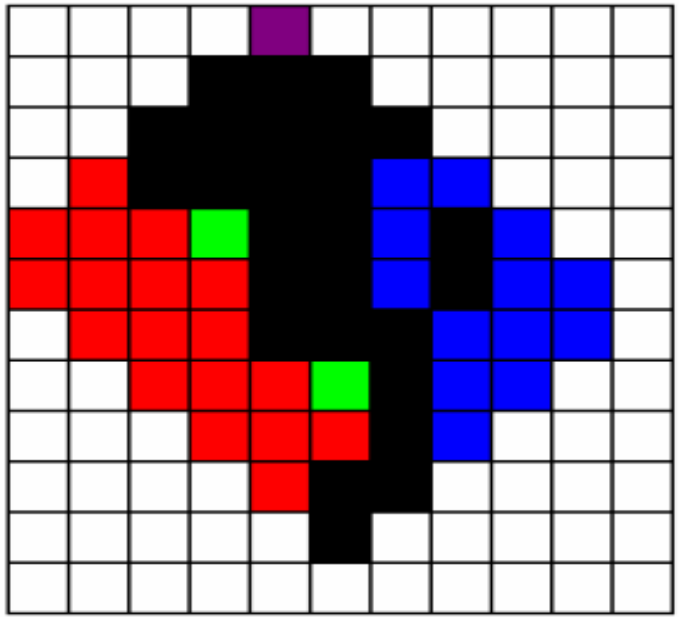


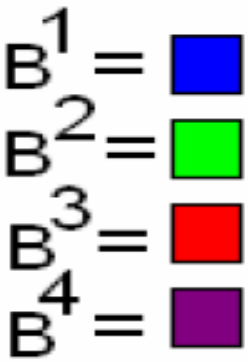
X_2^4

x	x	x
x	0	x
I	I	I

B^4

- Initial Result from Union

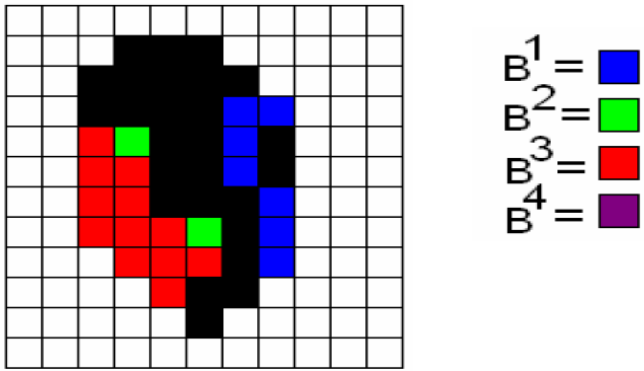




Shortcoming of the procedure - The convex hull can grow beyond the minimum dimensions required to guarantee convexity.

To reduce this effect limit growth so that it does not extend past the vertical & horizontal dimensions of the original set of points

- The Final Image with a bounding box



Boundaries of greater complexity can be used to limit growth even further in images with more detail.

e.g. the maximum dimensions of the original set of points along vertical, horizontal & diagonal directions can be used.

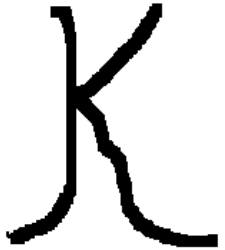
Price paid for such refinements ??
Additional complexity

THINNING

Thinning is used to remove selected foreground pixels from binary images

Erase foreground pixels such that

- **An object without holes** erodes to a minimally connected stroke located equidistant from its nearest outer boundaries
- **An object with holes** erodes to a minimally connected ring midway between each hole & its nearest outer boundary.



Original image

Thinned image

- The thinning of a set A by a structuring element B can be defined in terms of the hit-or-miss transform

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

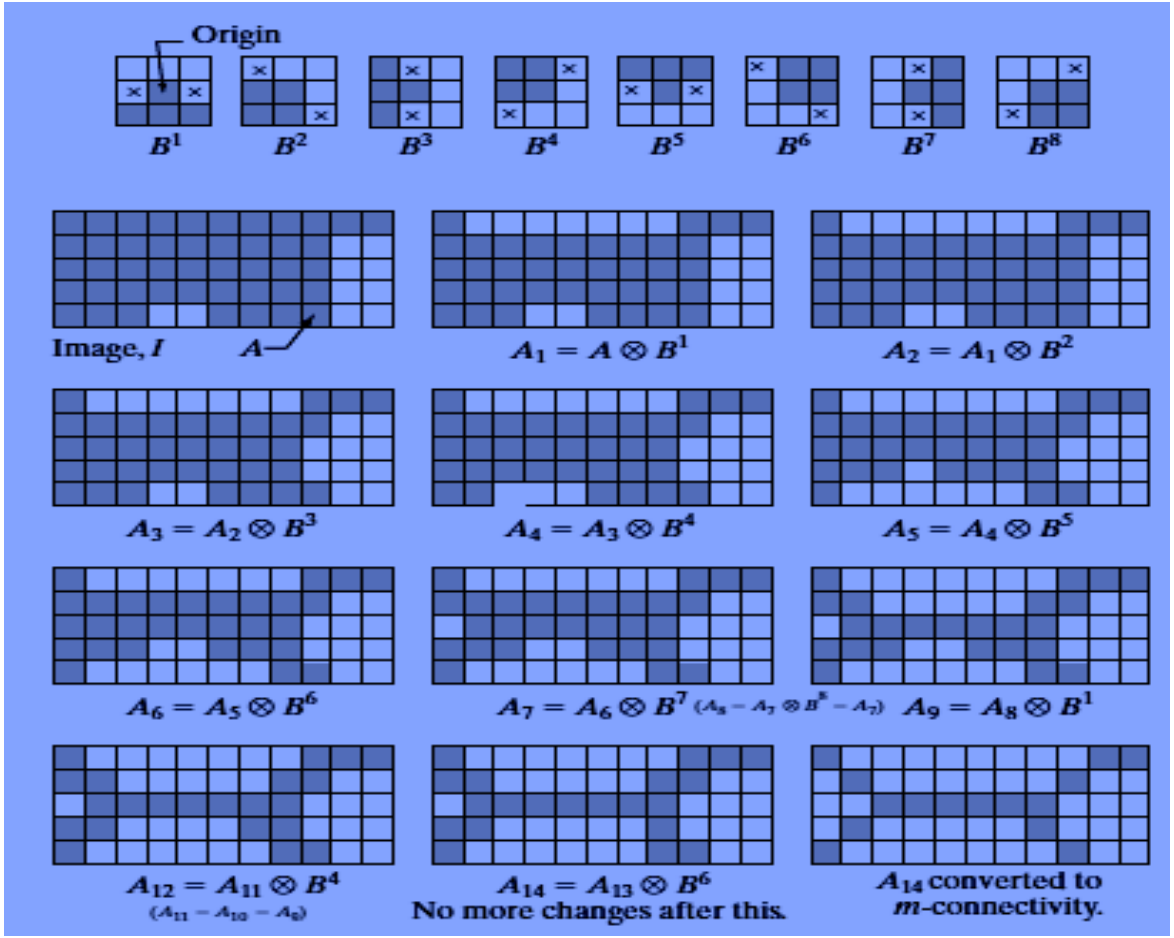
- A more useful expression for thinning *A symmetrically is based on a sequence of* structuring elements:

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

where B^i is the rotated version of B^{i-1}

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

(a) Structuring elements.
(b) Set *A*.
(c) Result of thinning *A with B1 (shaded).*
(d) Result of thinning *A1 with B2.*
(e)–(i) Results of thinning with the next six SEs. (There was no change between *A7* and *A8 .)*
(j)–(k) Result of using the first four elements again.
(l) Result after convergence.
(m) Result converted to *m-connectivity.*



THICKENING

Thickening is the morphological dual of thinning

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

As in thinning, thickening can be defined as a sequential operation:

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

The structuring elements used for thickening have the same form as those for thinning, but with all 1's and 0's interchanged.

A separate algorithm is rarely used for thickening

The usual procedure??

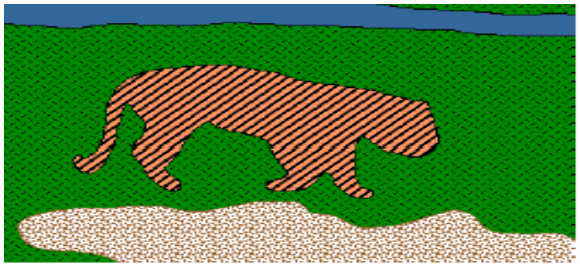
Thin the background of the set in question & then take the complement of the result

To thicken A

- Let $C = A^c$
- Thin C
- Take C^c

IMAGE SEGMENTATION

- Segmentation divides an image into its constituent regions or objects.
- Segmentation of non trivial images is one of the difficult task in image processing. Still under research.
- Segmentation accuracy determines the eventual success or failure of computerized analysis procedure.
- Example Application: Automated inspection of electronic assemblies. (mother boards)



REGION

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

(a) $\bigcup_{i=1}^{n} R_i = R.$

indicates that the segmentation must be *complete, in the sense that* every pixel must be in a region.

(b) R_i is a connected set, for $i = 0, 1, 2, \ldots, n.$

requires that points in a region be connected in some predefined sense

(c) $R_i \cap R_j = \varnothing$ for all i and j, $i \neq j$.

deals with the properties that must be satisfied by the pixels in a segmented region - for example, *Q (Ri) = TRUE* if all pixels in

(d) $Q(R_i) = \text{TRUE}$ for $i = 0, 1, 2, \ldots, n.$

(e) $Q(R_i \cup R_j) = \text{FALSE}$ for any adjacent regions R_i and R_j.

indicates that two adjacent regions *Ri and Rj must be different in the sense of predicate Q*

POINT, LINE AND EDGE DETECTION

- **Point**
  a foreground (background) pixel surrounded by background (foreground) pixels
- **Line**
  a (typically) thin edge segment in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels.
- **Edge**
  a set of connected edge pixels where edge pixels are pixels at which the intensity of an image changes abruptly

DETECTION OF ISOLATED POINTS

- Point detection is based on the second derivative, the Laplacian

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

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

- A point is detected at a location (x, y) on which the kernel is centered if the absolute value of the response of the filter at that point exceeds a specified threshold. Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image.

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

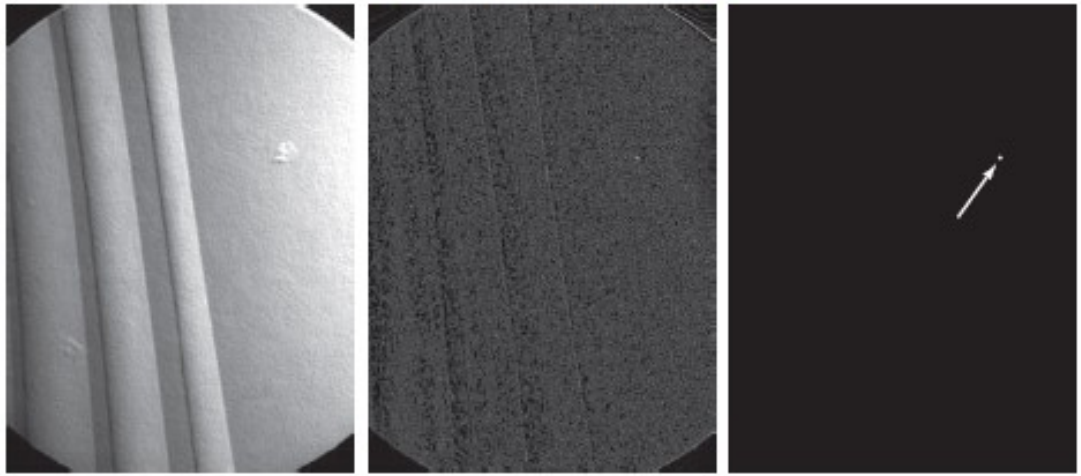
a) Laplacian kernel used for point detection.

b) X-ray image of a turbine blade with a porosity manifested by a single black pixel.

c) Result of convolving the kernel with the image.

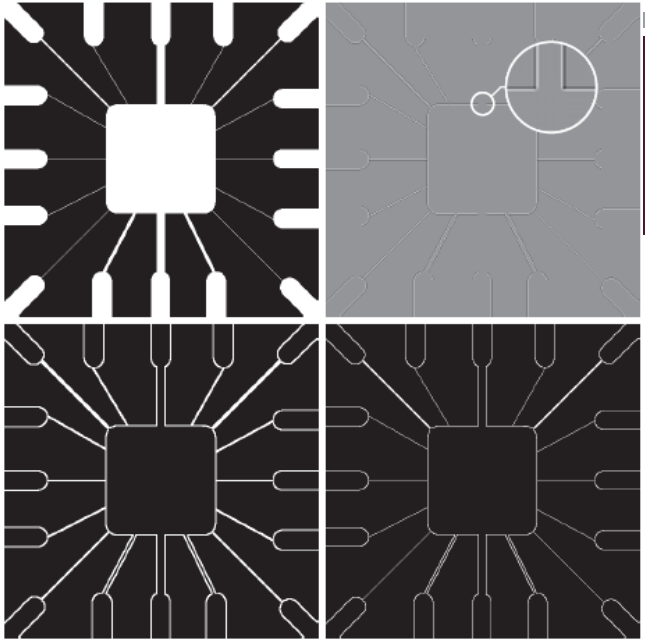
d) Result of thresholding was a single point

1	1	1
1	−8	1
1	1	1



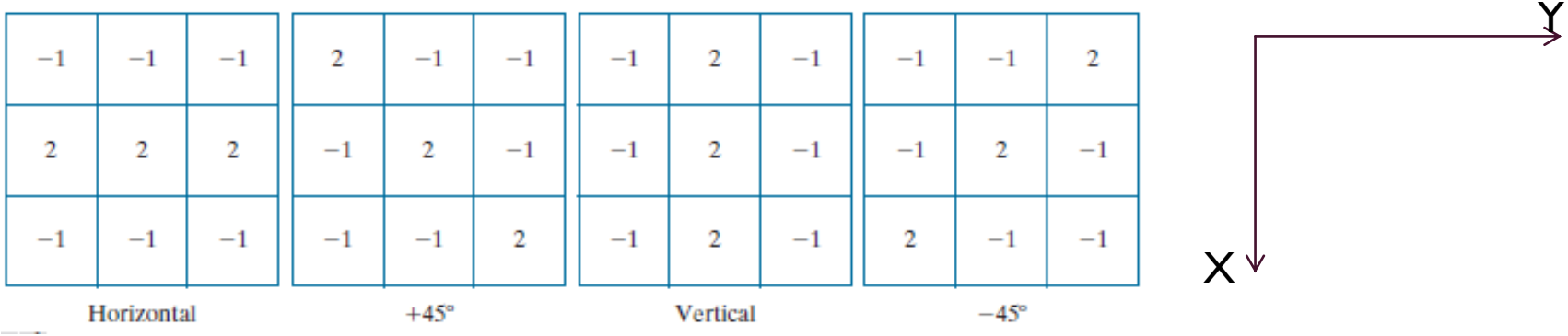
LINE DETECTION

- For line detection the second derivatives result in a stronger filter response and produce thinner lines than first derivatives.
- The Laplacian kernel can be used. The Laplacian image contains negative values, so absolute value can be taken for display
- The double-line effect of the second derivative must be handled properly.



(a) Original image. (b) Laplacian image; the Magnified section shows the positive/negative double-line effect characteristic of the Laplacian. (c) Absolute value of the Laplacian. (d) Positive values of the Laplacian.

- The Laplacian is isotropic, i.e. independent of direction.
- If we would like to detect lines on a certain direction only we might want to use masks that would emphasize a certain direction and be less sensitive to other directions.



Line detection kernels. Detection angles are with respect to the axis system in which positive angles are measured counterclockwise with respect to the (vertical) *x-axis*.

EDGE MODELS

Frequently used for segmenting images based on abrupt (local) changes in intensity.

Edge models are classified according to their intensity profiles.

Step edge

Characterized by a transition between two intensity levels occurring ideally over the distance of one pixel.

Step edges occur, for example, in images generated by a computer for use in areas such as solid modeling and animation.

Ideal representation of a step edge & it's corresponding intensity profile.

In practice, digital images have edges that are blurred and noisy,

- blurring due to limitations in the focusing mechanism (e.g., lenses in the case of optical images)
- noise level determined by the electronic components of the imaging system.

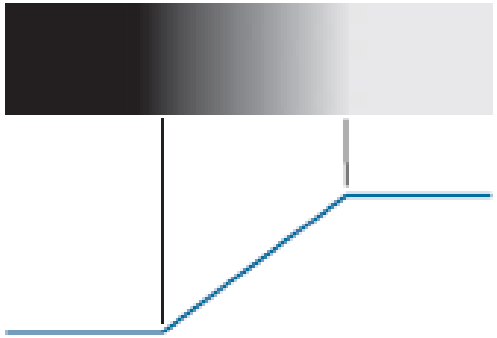
Ramp

The slope of the ramp is inversely proportional to the degree to which the edge is blurred.

No single "edge point" along the profile.

An edge point now is any point contained in the ramp

An edge segment a set of such points that are connected.



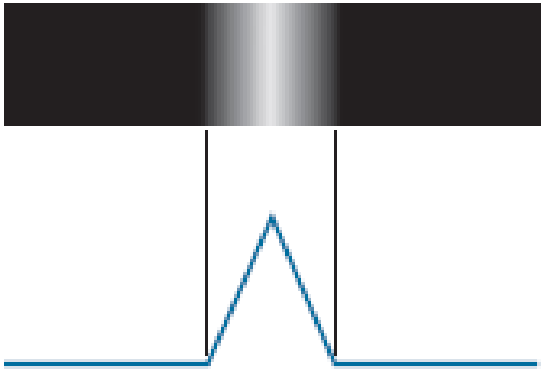
Ideal representation of a ramp and it's corresponding intensity profile.

Roof Edge

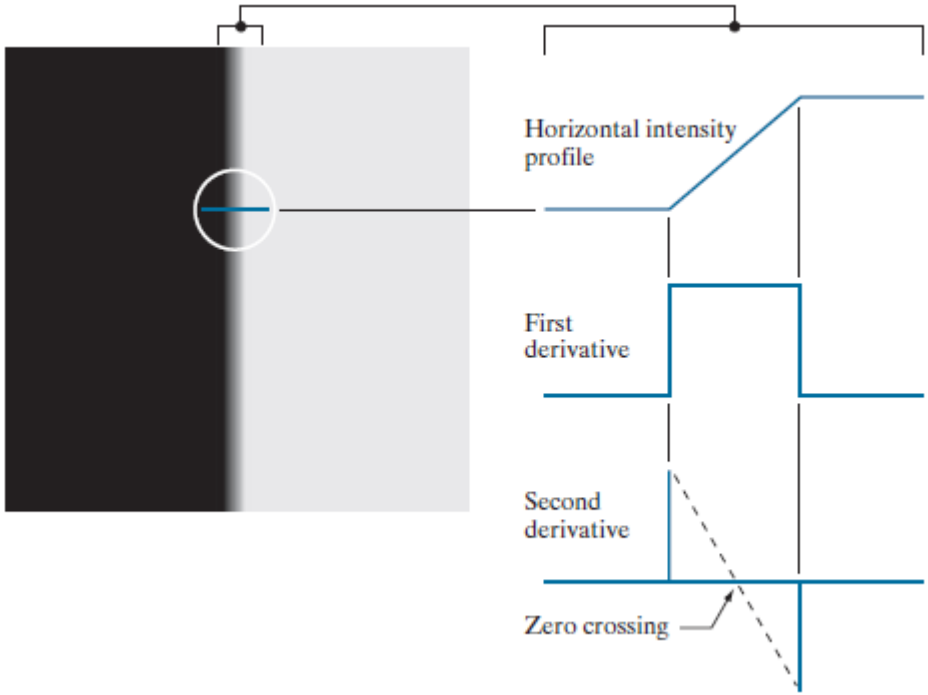
Roof edges are models of lines through a region, with the width of the edge being determined by the thickness and sharpness of the line.

Arise, for example, in range imaging, when thin objects (such as pipes) are closer to the sensor than the background (such as walls). The pipes appear brighter .

Also appear in the digitization of line drawings & in satellite images, where thin features, such as roads, can be modeled by this type of edge.



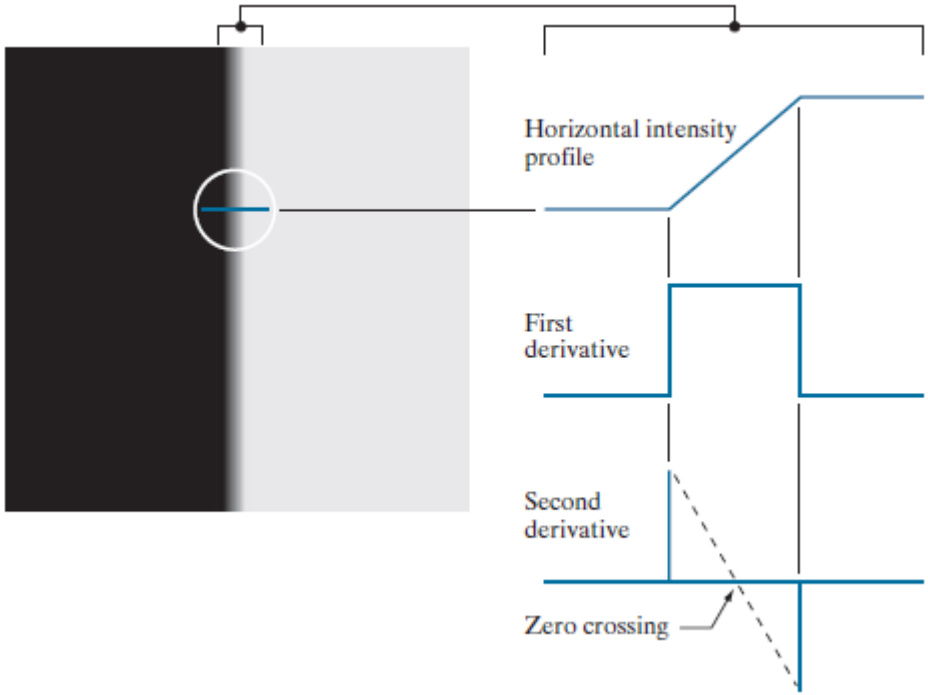
Ideal representation of a roof edge and it's corresponding intensity profile.



a) Two regions of constant intensity separated by an ideal ramp edge.
b) Detail near the edge, showing a horizontal intensity profile & its first & second derivatives.

The First derivative

- is positive at the onset of the ramp and at points on the ramp,
- it is zero in areas of constant intensity.
- the magnitude of the first derivative can be used to detect the presence of an edge at a point in an image



a) Two regions of constant intensity separated by an ideal ramp edge.
b) Detail near the edge, showing a horizontal intensity profile & its first & second derivatives.

The Second derivative

- is positive at the beginning & negative at the end of the ramp
- zero at points on the ramp & at points of constant intensity.
- The intersection between the zero intensity axis and a line extending between the extrema of the second derivative marks a point called the ***zero crossing*** of the second derivative.

Properties of Second Derivative

1. The sign of the second derivative can be used to determine whether an edge pixel lies on the dark or light side of an edge.
2. it produces two values for every edge in an image
3. its zero crossings can be used for locating the centers of thick edges
4. It is very sensitive to noise

Three steps for edge detection are:

1. **Image smoothing**

   for noise reduction.

2. **Detection of edge point**

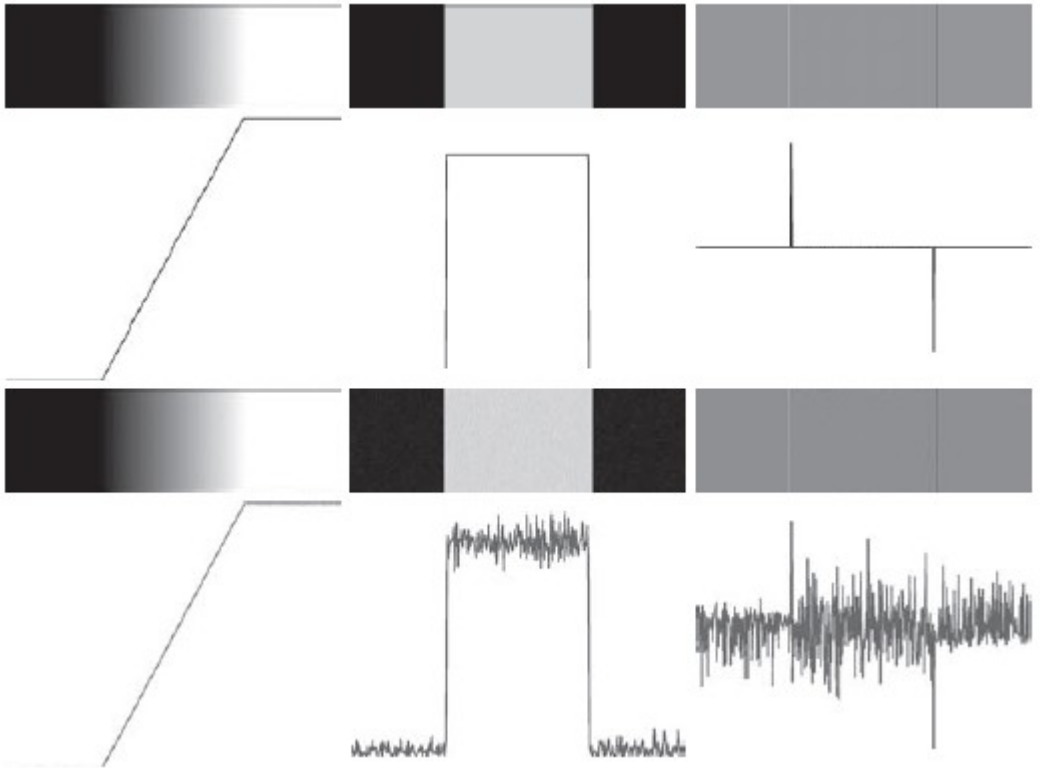
   a local operation that extrac

   edge-point candidates.

3. **Edge localization**

   to select from the candidate

   the set of points comprising

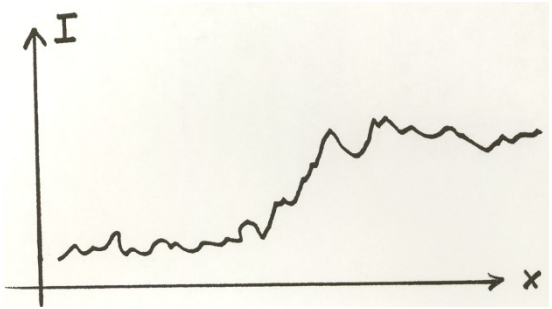


First column: 8-bit images with values in the range [0,255], and intensity profiles of a ramp edge corrupted by Gaussian noise.

Second column: First-derivative images and intensity profiles.

Third column: Second-derivative images and intensity profiles.

HOW CAN YOU TELL THAT A PIXEL IS ON AN EDGE?



Real Edges are Noisy and Discrete!

We want an Edge Operator that produces:

- Edge Magnitude
- Edge Orientation
- High Detection Rate and Good Localization

THE IMAGE GRADIENT AND ITS PROPERTIES

Edge strength and direction at an arbitrary location (x, y) of an image, f, is the gradient, denoted by ∇f and defined as the vector

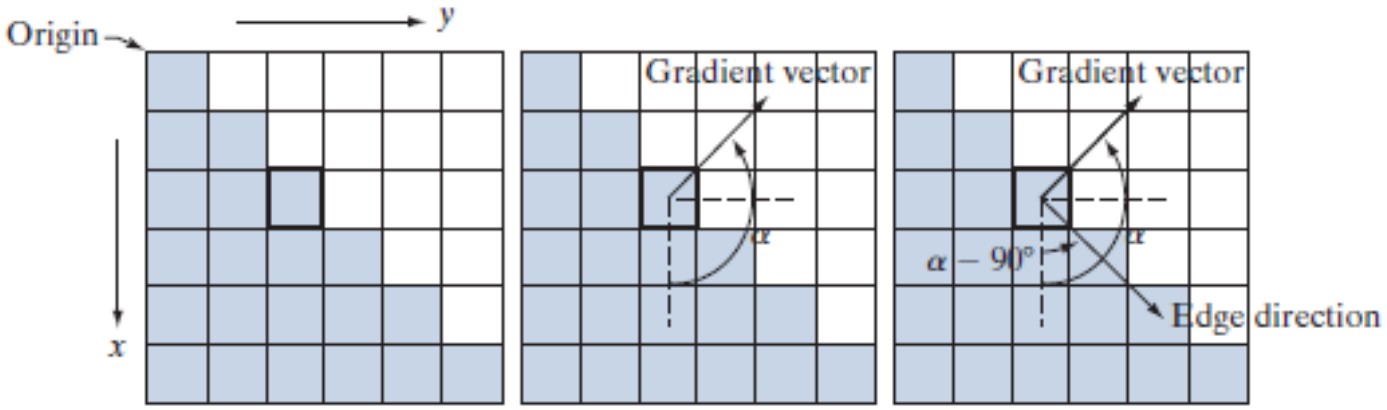
$$\nabla f(x,y) \equiv \text{grad}[f(x,y)] \equiv \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

The **magnitude, M(x, y)**, of this gradient vector at a point (x, y) is given by its Euclidean vector norm

$$M(x,y) = \|\nabla f(x,y)\| = \sqrt{g_x^2(x,y) + g_y^2(x,y)}$$

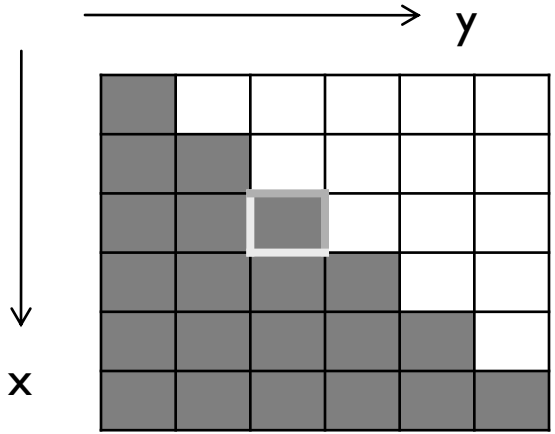
The **direction of the gradient vector** at a point (x, y) is given by

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y(x,y)}{g_x(x,y)}\right]$$



- An image containing a straight edge segment.
- The shaded pixels have value 0, and the white pixels have value 1.
- Angles are measured in the counterclockwise direction with respect to the x-axis
- The edge direction is perpendicular to the direction of the gradient vector at the point where the gradient is computed.

Find the strength & the direction of the edge at the highlighted pixel



Pixels in gray have value 0 & in white have value 1.

Derivative is computed by using a 3x3 neighbourhood – subtract the pixels in the top row from the pixels in the bottom row to get the derivative in the x direction. Similarly obtain the derivative in the y direction.

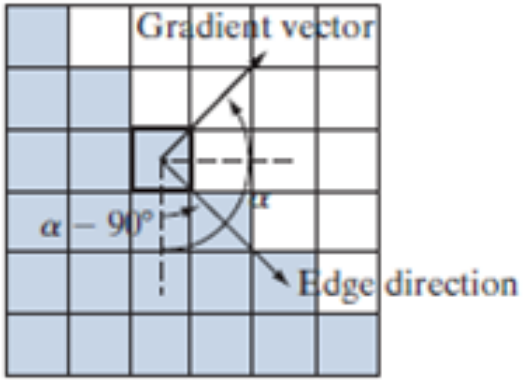
$\partial f / \partial x =$ - 2

$\partial f / \partial y =$ 2

$M(x, y) =$ $2\sqrt{2}$

$\alpha =$ $\tan^{-1}(g_y / g_x)$
-45°
which is the same as
135° measured in
the positive
(counterclockwise)
direction with respect to
the *x-axis*

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

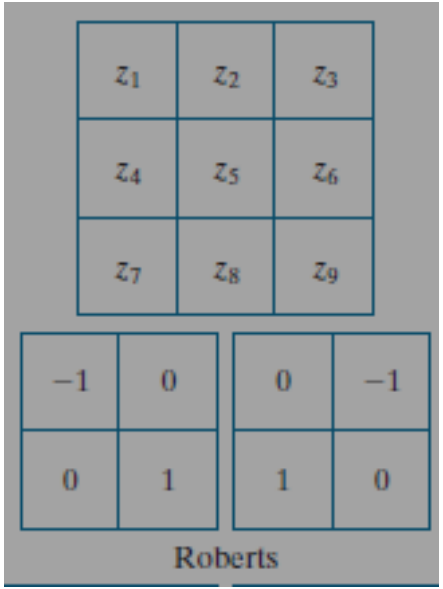


Roberts Operator (Roberts [1965])

- Used when diagonal edge direction is of interest. The Roberts cross-gradient operators use 2-D kernels with a diagonal preference.
- Kernels of size 2 × 2 are simple conceptually, but they are not as useful for computing edge direction as compared with kernels that are symmetric about their centers.
- The smallest such kernel is of size 3 × 3. These kernels take into account the nature of the data on opposite sides of the center point, and thus carry more information regarding the direction of an edge.

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$



Prewitt Operators (Prewitt [1970])

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

Prewitt

Sobel Operators (Sobel [1970])

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Sobel

Examples

A 9x9 original image is given by

9	9	9	9	9	9	9	2	2
9	8	9	9	9	9	2	2	2
9	9	9	9	9	9	3	2	2
9	9	9	9	9	2	2	2	2
7	9	9	9	9	2	2	2	2
9	9	9	9	2	2	2	2	2
9	9	9	9	2	2	2	4	2
9	9	9	2	2	2	2	2	2
9	9	2	2	2	2	1	2	2

d_1

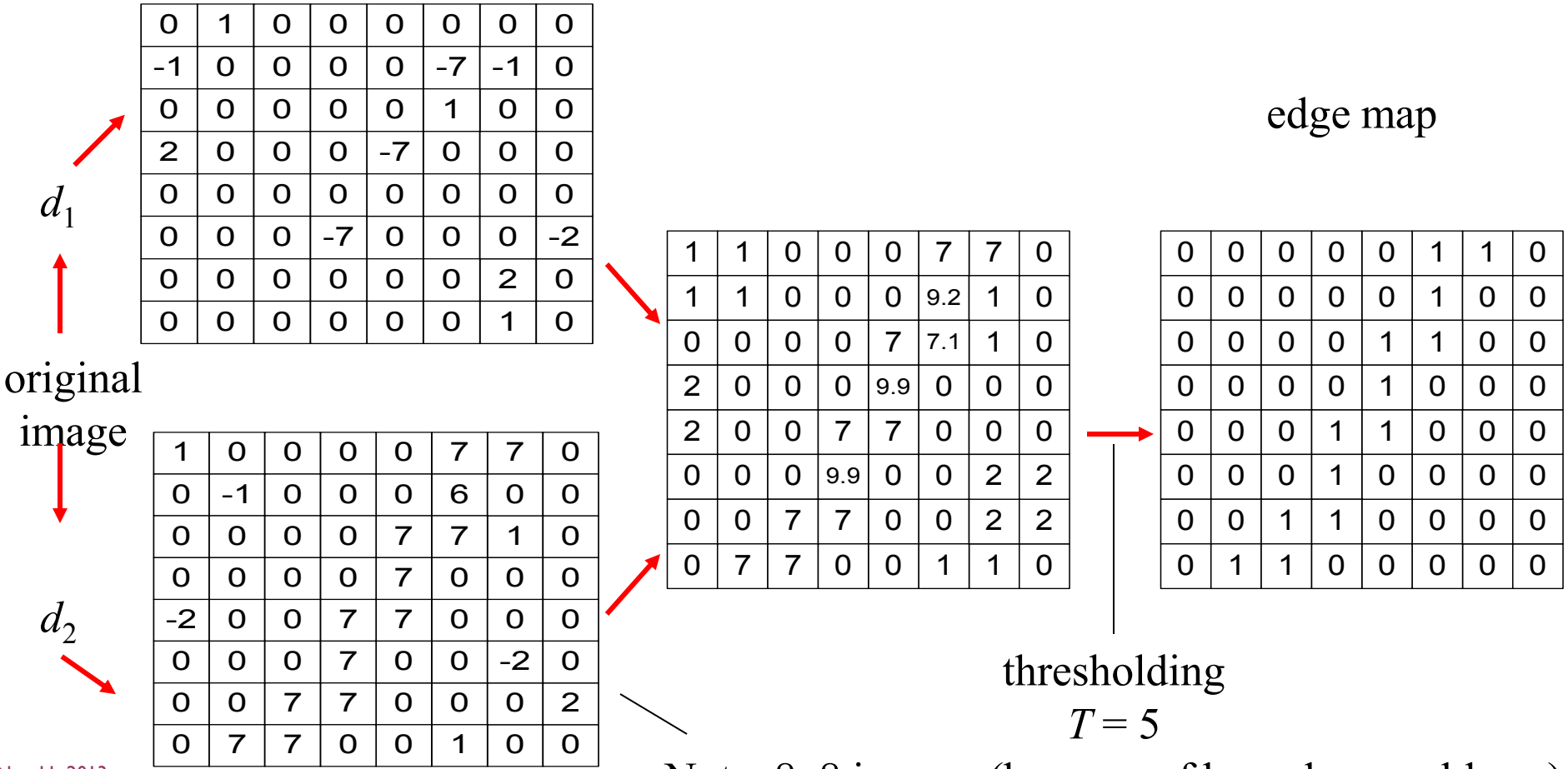
$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

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

d_2

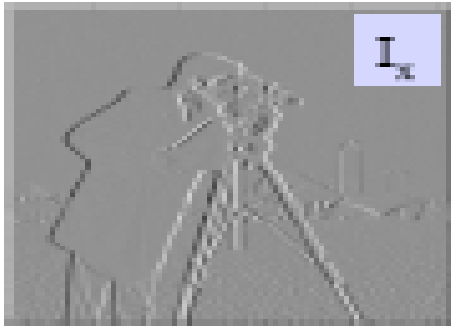
1) Use Robert gradient operator to find its edges

Use $|d(x,y)| = \sqrt{d_1^2(x,y) + d_2^2(x,y)}$ to estimate the gradient magnitude, and use $T = 5$ as the threshold for edge detection



Note: 8x8 images (because of boundary problems)

SIMPLE EDGE DETECTION USING GRADIENTS …



How should we choose the threshold?

> 10 > 30 > 80

THE CANNY EDGE DETECTOR

Algorithm more complex

Performance superior

Canny's approach based on 3 basic objectives:

- **Low error rate**
  - All edges should be found
  - There should be no spurious responses

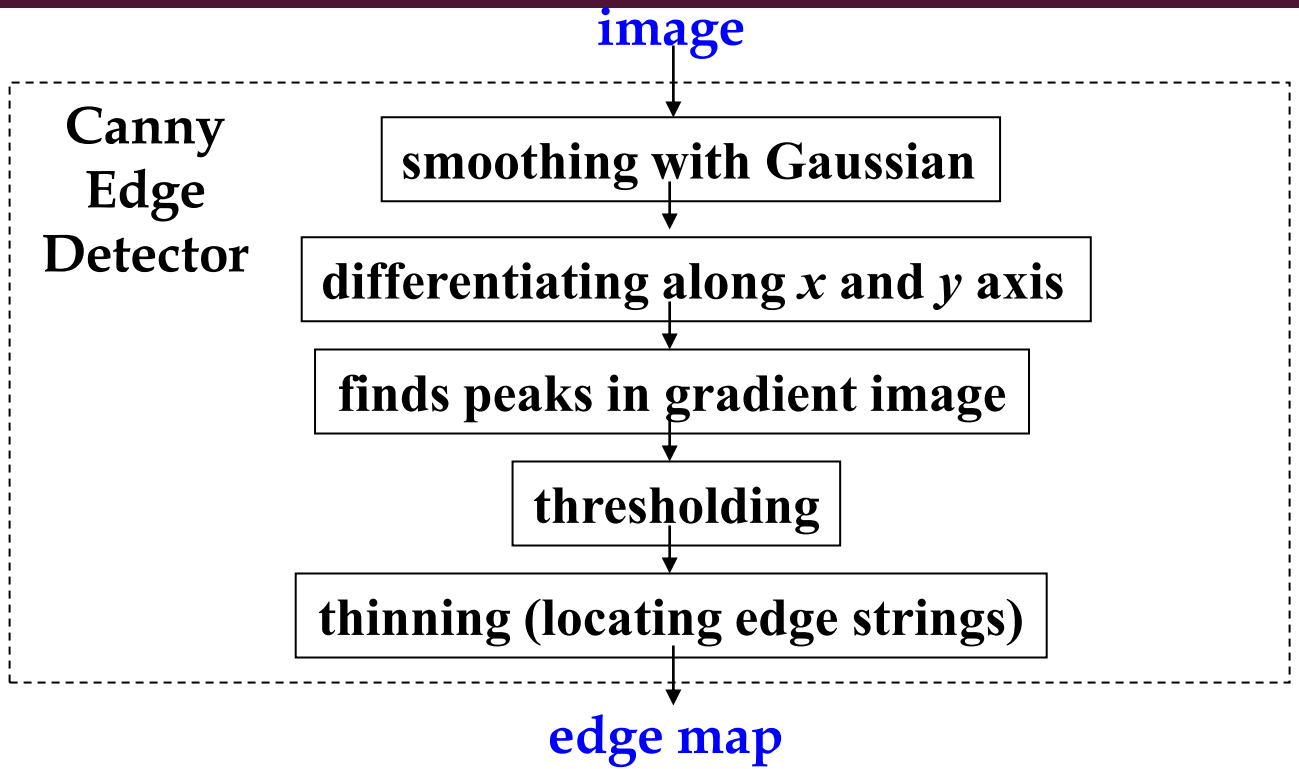
Edge points should be well localized

- The edges detected should be as close as possible to the true edges.
- The distance between a point marked as an edge by the detector & the centre of the true edge should be minimum.

Single Edge point response

- The detector should return only one point for each true edge point.
- The number of local maxima around the true edge point should be minimum.
- The detector should not identify multiple edge pixels where only a single edge point exists.

OPTIMAL EDGE DETECTION: CANNY …



ALGORITHM

Step 1

Let f (x, y) denote the input image and G(x, y) denote the Gaussian function:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

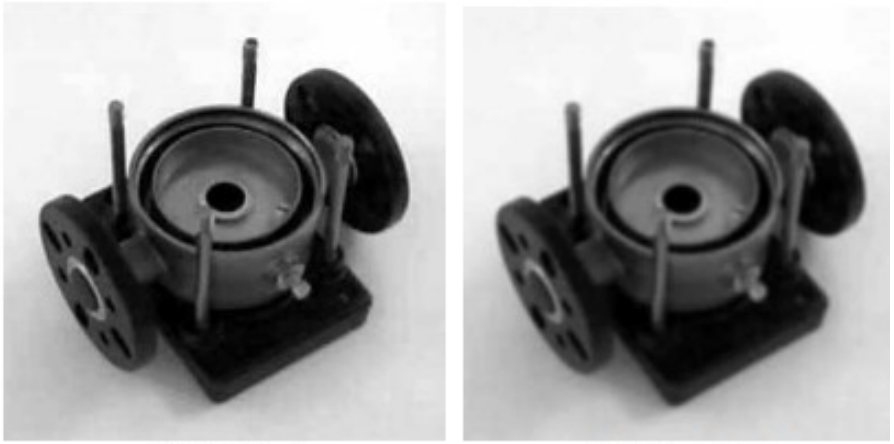
A smoothed image, $f_s(x, y)$, is formed by convolving f and G:

$$f_s(x,y) = G(x,y) \star f(x,y)$$

The image used as example of Canny edge detection.

$$B = \frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

The kernel of a Gaussian filter with a standard deviation of σ = 1.4



(a) Original (b) Smoothed

The original grayscale image is smoothed with a Gaussian filter to suppress noise.

Step 2

Apply first difference gradient operator to compute the edge strength & direction. Any of the filter masks (Roberts, Prewitt etc.) can be used.

$$M_s(x,y) = \|\nabla f_s(x,y)\| = \sqrt{g_x^2(x,y) + g_y^2(x,y)}$$

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y(x,y)}{g_x(x,y)}\right]$$

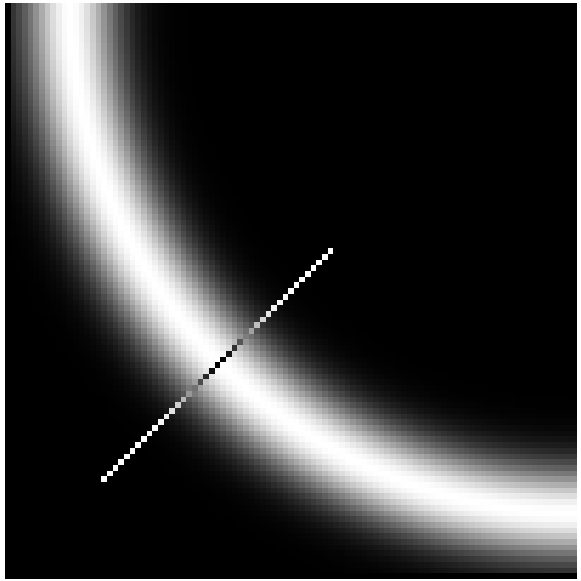
(a) Smoothed (b) Gradient magnitudes

The gradient magnitudes in the smoothed image as well as their directions are determined by applying e.g. the Sobel-operator

Step 3

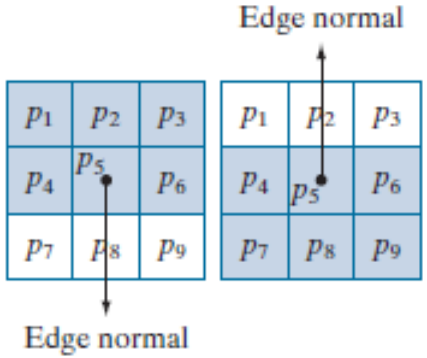
Apply non-maximal suppression to the gradient magnitude.

- The purpose of this step is to convert the "blurred" edges in the image of the gradient magnitudes to "sharp" edges.
- Basically this is done by preserving all local maxima in the gradient image, and deleting everything else.

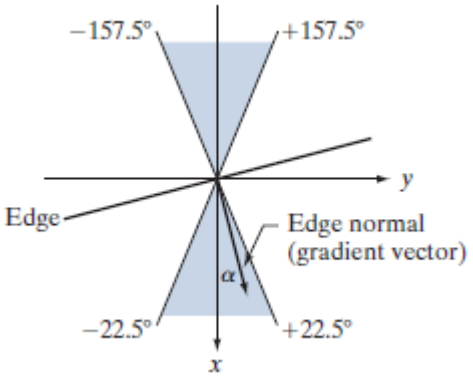


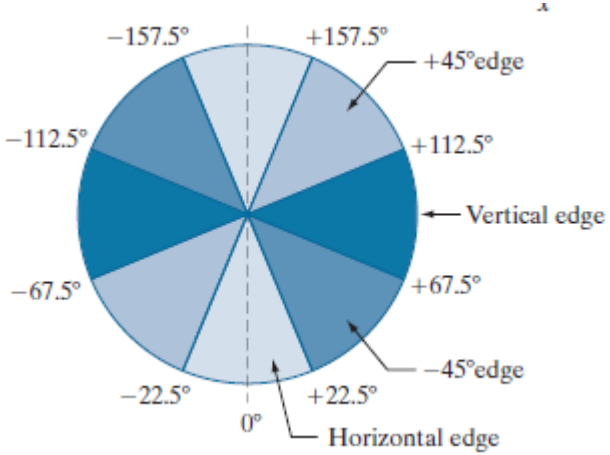
The algorithm is for each pixel in the gradient image:

1. Round the gradient direction θ to nearest 45°
   - in a 3 × 3 region four orientations can be defined for an edge passing through the center point of the region: horizontal, vertical, +45°& −45°.
   - Every edge has two possible orientations.



Two possible orientations of a horizontal edge

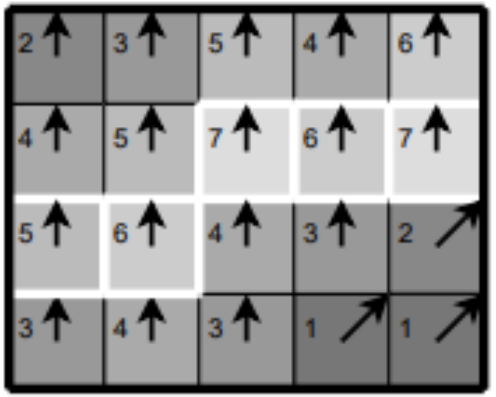




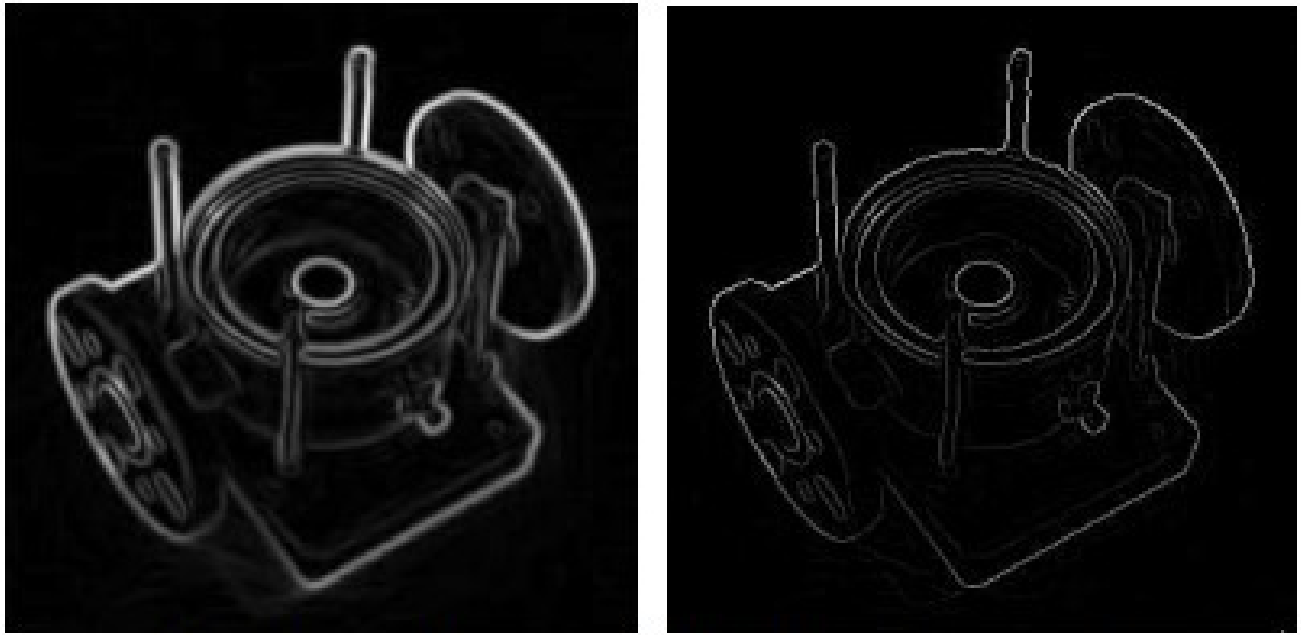
The angle ranges of the edge normals for the four types of edge directions in a 3 × 3 neighborhood. Each edge direction has two ranges

2. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient direction.

- i.e. if the gradient direction is north ($\theta = 180°$), compare with the pixels to the north and south
- If the edge strength of the current pixel is largest; preserve the value of the edge strength. If not, suppress (i.e. remove) the value.
- $g_N(x, y)$ is the non maximal edge suppressed image



Almost all pixels have gradient directions pointing north. They are therefore compared with the pixels above and below. The pixels that turn out to be maximal in this comparison are marked with white borders. All other pixels will be suppressed.



(a) Gradient values

(b) Edges after non-maximum suppression

Non-maximum suppression. Edge-pixels are only preserved where the gradient has local maxima.

The final operation is to threshold $g_N(x, y)$ to reduce false edge points.

False edges can be reduced by applying a threshold T

- all values below T are changed to 0

However,

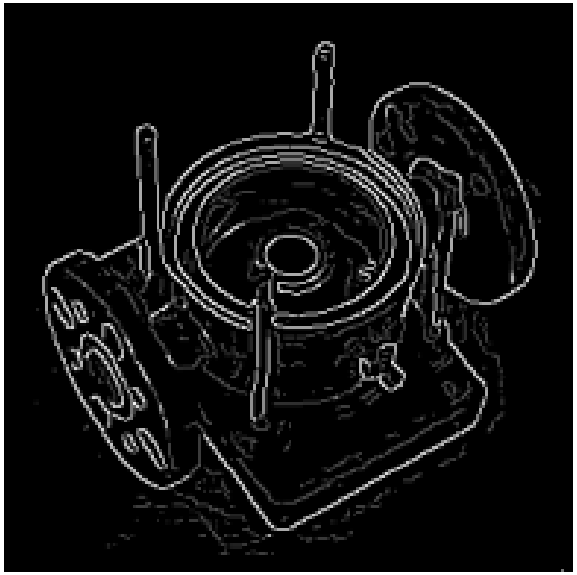
- selecting a good value for T is difficult
- some false edges will remain if T is too low
- some edges will disappear if T is too high

Step 4

Hysteresis thresholding / Edge Linking

- Canny's approach employs double thresholding, known as **hysteresis.**
  - a low threshold, T_L and a high threshold, T_H.
  - Experimental evidence (Canny [1986]) suggests that the ratio of the high to low threshold should be in the range of 2:1 to 3:1.
- Edge pixels stronger than T_H are **marked as strong**; edge pixels weaker than T_L are suppressed and edge pixels between the two thresholds are **marked as weak**.

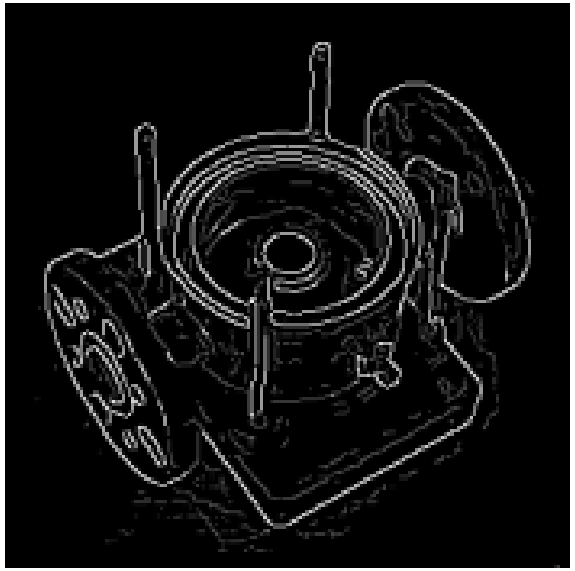
(a) Edges after non-maximum suppression



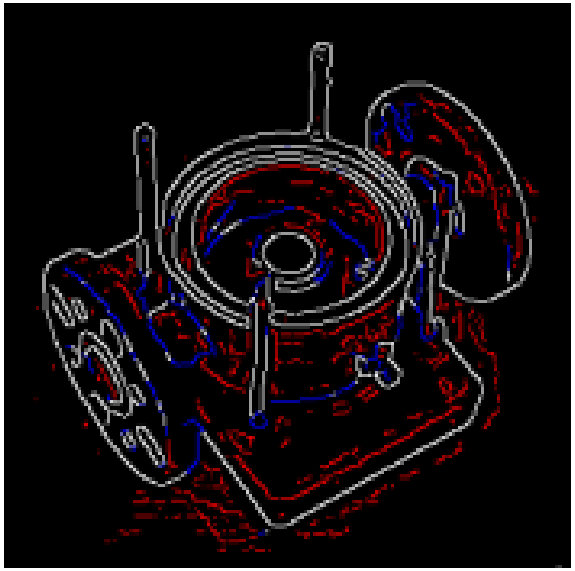
(b) Double thresholding

Thresholding of edges. In the second image strong edges are white, while weak edges are grey. Edges with a strength below both thresholds are suppressed.

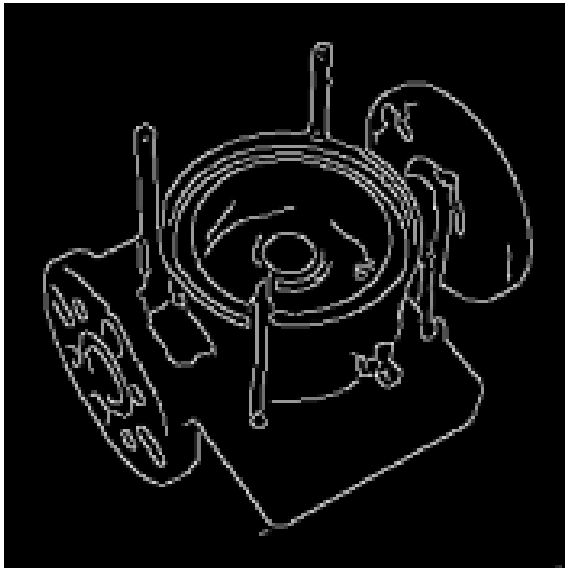
- Strong edges are interpreted as “certain edges” and can immediately be included in the final edge image.
- Weak edges are included if and only if they are connected to strong edges.
  - The logic is that noise and other small variations are unlikely to result in a strong edge. Thus strong edges will (almost) only be due to true edges in the original image.
  - The weak edges can either be due to true edges or noise/color variations. The latter type will probably be distributed independently of edges on the entire image, and thus only a small amount will be located adjacent to strong edges.
  - Weak edges due to true edges are much more likely to be connected directly to strong edges.



(a) Double thresholding



(b) Edge tracking by hysteresis



(c) Final output

Edge tracking and final output. The middle image shows strong edges in white, weak edges connected to strong edges in blue, and other weak edges in red

Finding edge features

An edge element is deemed present at (x,y) if f’(x, y) exceeds a predefined threshold

But we haven’t found edge segments, only edge points

How can we find and describe more complex features?

The Hough transform is a common approach to finding parameterised line segments (here straight lines) or other shapes

HOUGH TRANSFORM

- Performed after Edge Detection
- It is a technique to isolate the curve of a given shape in a given image
- Classical Hough Transform can locate regular curves like straight lines, circles, parabolas, ellipses, etc.
  - Requires that the curve be specified in some parametric form
- Generalized Hough Transform can be used where a simple analytic description of feature is not possible

- Given marked edge pixels, find examples of specific shapes
  - Line segments
  - Circles
  - Generalized shapes
- Basic idea - Patented 1962
  - Every edge pixel is a point that votes for all shapes that pass through it.
  - Votes are collected in "parameter space" - look for peaks

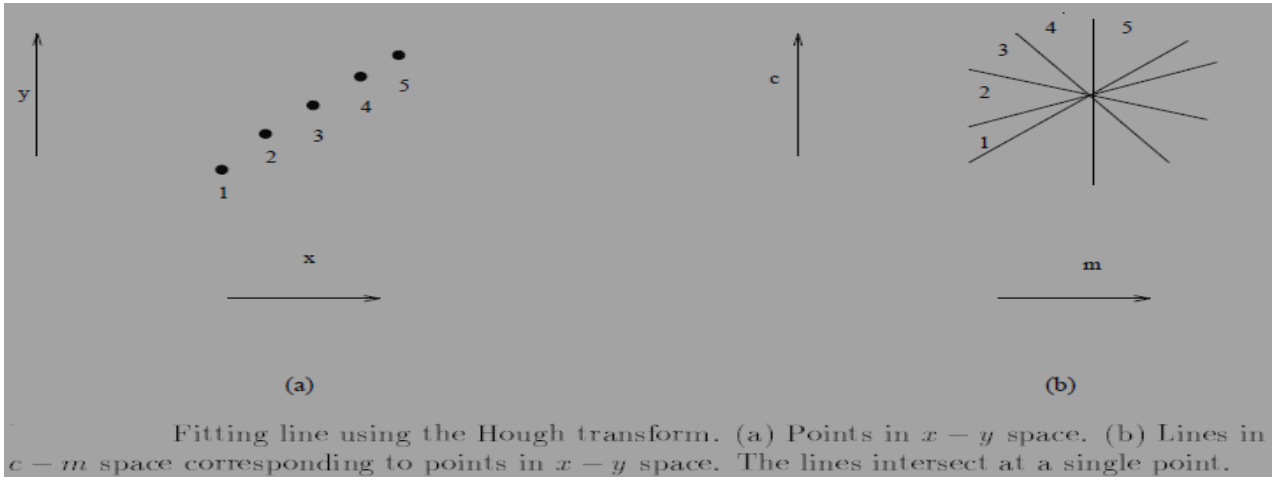
STRAIGHT LINE

$y = mx + c$ where m is the slope & c the y-intercept

The equation can be rewritten in c – m space as: $c = -xm + y$

Suppose we have several edge points $(x_1, y_1), ..., (x_n, y_n)$ in x – y space that we want to fit in a line.

Each point in x – y space maps to a line in c – m space.



Fitting line using the Hough transform. (a) Points in $x - y$ space. (b) Lines in $c - m$ space corresponding to points in $x - y$ space. The lines intersect at a single point.

Properties in slope-intercept space

- Each point (x_i , y_i) defines a line in the $c - m$ space (parameter space).
- Points lying on the same line in the $x - y$ space, define lines in the parameter space which all intersect at the same point.
- The coordinates of the point of intersection define the parameters of the line in the $x - y$ space.

ALGORITHM

1. Quantize parameter space (c, m):

   $P\ [m_{min}, \ldots, m_{max}]\ [c_{min}, \ldots, c_{max}]$

2. For each edge point (x, y)

   For($m = m_{min}; m \leq m_{max}; m$++) {

   $c = -xm + y$; /* round off if needed *

   ($P[m][c]$)++; /* **voting** */ }

3. Find local maxima in $P[m][c]$

if $P[m][c]$ = M, then M points lie on the line $y = m\,x + c$

Find the equation for the edge for the following edge points

- (2,2) $c = -2m + 2$
- (4,7) $c = -4m + 7$
- (5,5) $c = -5m + 5$
- (6,2) $c = -6m + 2$
- (6,6) $c = -6m + 6$

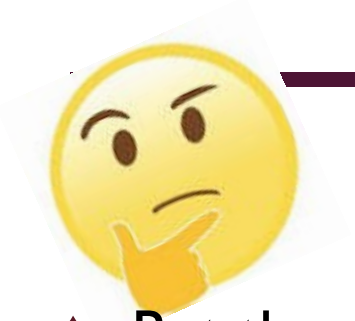
Let $m_{min} = -2$ & $m_{max} = 2$

m	(2,2) c = -2m + 2	(4, 7) c = -4m + 7	(5,5) c = -5m +5	(6, 2) c = -6m + 2	(6,6) c = -6m +6
-2	6	15	15	14	18
-1	4	11	10	8	12
0	2	7	5	2	6
1	0	3	0	-4	0
2	-2	-1	-5	-10	-6

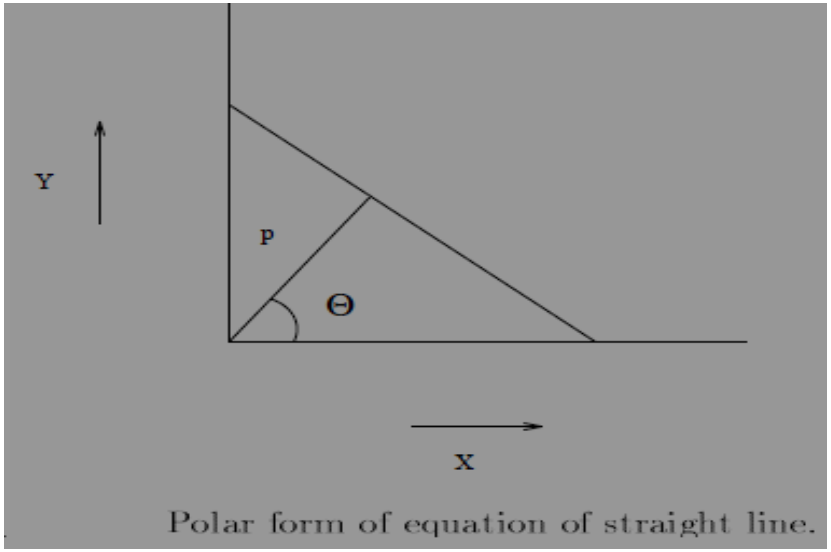
Point	Votes
(-2, 15)	2
(0, 2)	2
(1, 0)	3

So, the equation becomes

y = x



- But there is a Problem
- The slope of line parallel to y – axis is infinity, the computer can not handle it.
- So, we follow another parameterization of a line



Polar form of equation of straight line.

$$p = x \cos\theta + y \sin\theta$$

ALGORITHM

1. Quantize the parameter space (p, θ)

   $P[p_{min}, \ldots, p_{max}][\theta_{min}, \ldots, \theta_{max}]$

2. For each edge point (x, y)

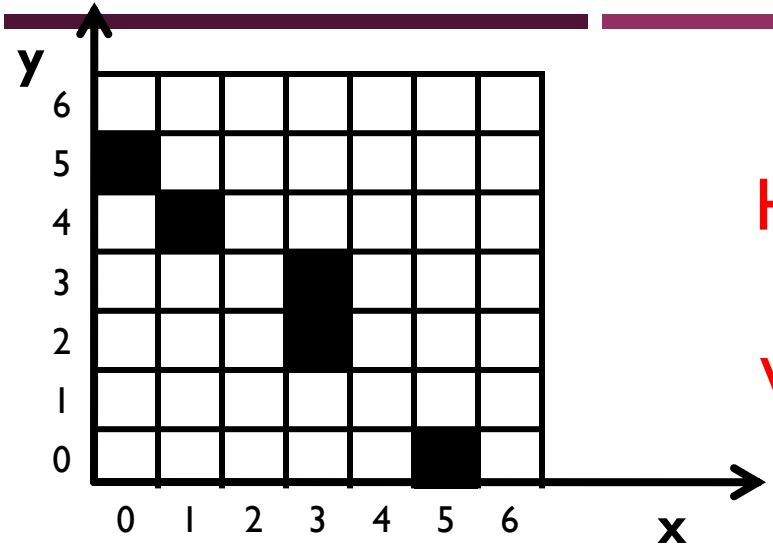
   For($\theta = \theta_{min}$; $\theta \leq \theta_{max}$; θ ++) {

   $p = x\ cos(\theta) + y\ sin(\theta)$; /* round off if needed *

   ($P[p][\theta]$)++; /* voting */

   }

3. Find local maxima in $P[p][\theta]$



Use
Hough Transform to form the main edge.
Work with polar coordinates

$$p = x \cos\theta + y \sin\theta$$

Edgels – (0,5), (1,4), (3,2), (3,3), (5,0)

In the given question, the curve corresponding to the point (3,3) does not intersect at the common point (3.5, 45), where the other curves meet.

The equation for the edge will be:

$$3.5 = x \cos 45 + y \sin 45$$

SEGMENTATION

Different objects or parts of the same object have different characteristics

So

Feature values recorded at pixels belonging to different regions should be different

BASIC IDEA OF IMAGE SEGMENTATION

All the image segmentation methods assume that:

1. the intensity values are different in different regions, and,
2. within each region, which represents the corresponding object in a scene, the intensity values are similar.

THRESHOLDING

Gray value

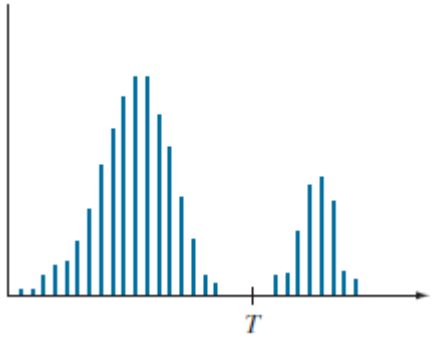
Simplest feature in a gray level image

- Image can be segmented by gray level thresholding
- The simplest segmentation process
- Computationally inexpensive and fast.
- Can easily be done in real time
- Correct thresholding leads to better segmentation.

THE BASICS OF INTENSITY THRESHOLDING

Suppose image f (x, y) is composed of light objects on a dark background.

How will its Histogram look?



To extract the objects from the background select a threshold, T, that separates these modes.

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

Global Thresholding

- T is a constant applicable over an entire image.
- Apply the same threshold to the whole image

Local Thresholding

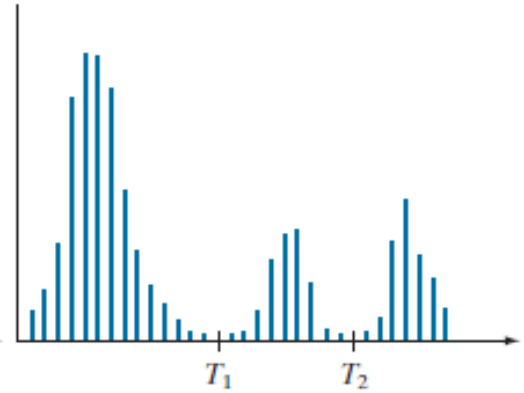
- The value of T changes over an image.
- The value of T at any point (x, y) in an image depends on properties of a neighborhood of (x, y) (for example, the average intensity of the pixels in the neighborhood).

Dynamic (Adaptive) Thresholding

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

Muliple Thresholding

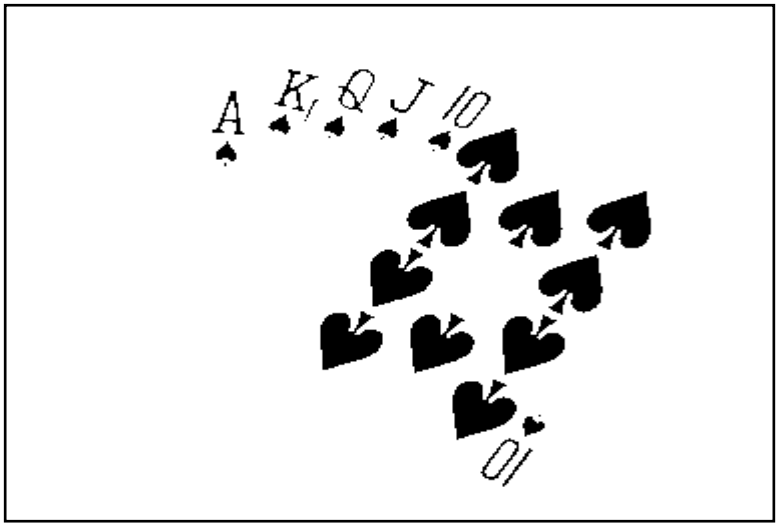
- If an image has more than two dominant modes, then multiple thresholds will be required.



$$g(x,y) = \begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \le T_2 \\ c & \text{if } f(x,y) \le T_1 \end{cases}$$

- Imagine a poker playing robot that needs to visually interpret the cards in its hand

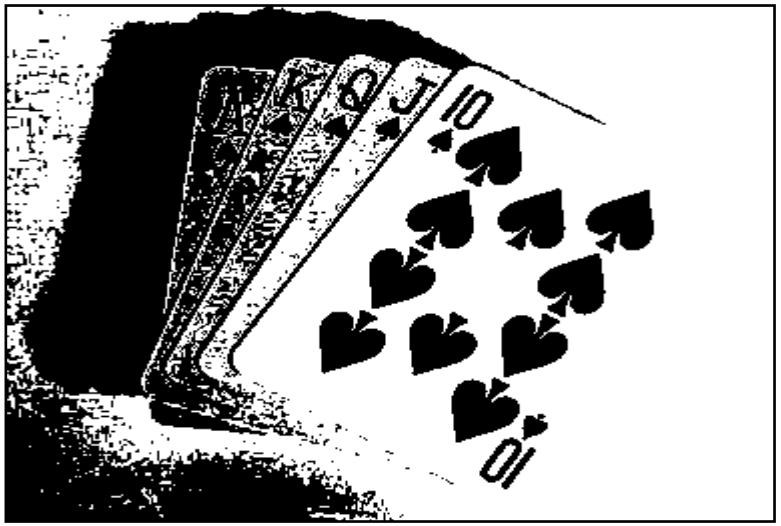
Original Image



Thresholded Image

- If you get the threshold wrong the results can be disastrous

Threshold Too Low



Threshold Too High

BASIC GLOBAL THRESHOLDING

1. Select an initial estimate for T (typically the average grey level in the image)
2. Segment the image using T to produce two groups of pixels: G_1 consisting of pixels with grey levels >T and G_2 consisting pixels with grey levels ≤ T
3. Compute the average (mean) intensity values m_1 and m_2 for the pixels in G_1 and G_2 , respectively.

4. Compute a new threshold value midway between m_1 and m_2 :

$$T = \frac{1}{2}(m_1 + m_2)$$

5. Repeat Steps 2 through 4 until the difference between values of T in successive iterations is smaller than a predefined value, ΔT.

- This algorithm works very well for finding thresholds when the histogram is suitable

The objective of segmentation is to partition an image into regions.

We did it by

- attempting to find boundaries between regions based on discontinuities in intensity levels,
- using thresholds based on the distribution of pixel properties, such as intensity values or color.

We will now discuss segmentation techniques that find the regions directly.

SEGMENTATION BY REGION GROWING

Groups pixels or subregions into larger regions based on predefined criteria for growth.

The basic approach is to start with a set of "seed" points, and from these grow regions by appending to each seed those neighboring pixels that have predefined properties similar to the seed (such as ranges of intensity or color or texture etc).

Let

f (x, y) denote an input image;

S(x, y) denote a seed array containing 1's at the locations of seed points and 0's elsewhere; and

Q denote a predicate to be applied at each location (x, y).

Arrays f and S are assumed to be of the same size.

A basic region-growing algorithm based on 8-connectivity may be stated as follows.

1. Start from a seed marking it as a new pixel for the region.
2. Check for 8 neighbours of the newly added pixels. If they satisfy the homogeneity criteria, add them to the region
3. Once a pixel is accepted as a member of the current region, the nearest neighbours of this new pixel are examined.
4. The process goes on recursively until no more pixels are accepted.
5. All pixels in the current region are marked with a unique label.
6. Then another seed pixel is picked up & the procedure is repeated.

MAIN PROBLEMS OF REGION GROWING

- Large execution time
- The selection of the property to be satisfied
- The selection of the seed points

Homogeneity property for a region can be defined as

$$Prop(R): \max_{(x,y)\in R}\{f(x,y)\} - \min_{(x,y)\in R}\{f(x,y)\} \leq threshold$$

Find the regions for the given seeds.

Threshold value = 3

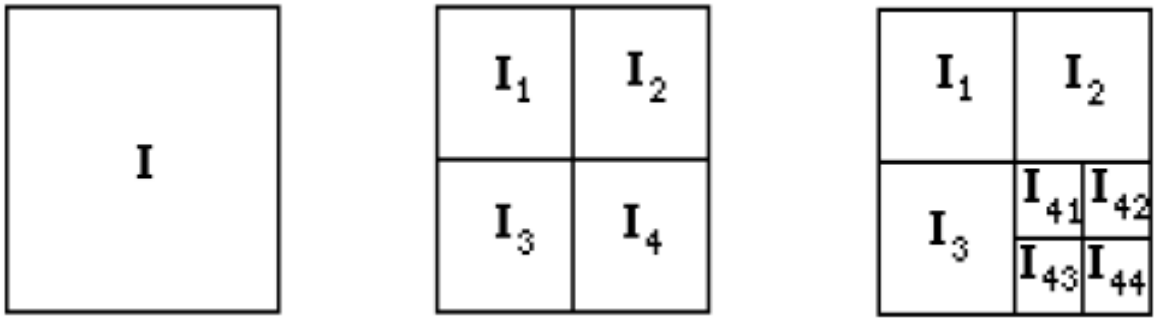
9	6	6	6	7	7	6	6
6	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
1	3	2	3	3	2	4	7
0	0	1	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	6

REGION SPLITTING (TOP DOWN APPROACH)

Start with whole

Split region if not homogeneous

Stop if no region can be split anymore



REGION MERGING (BOTTOM UP METHOD)

- Start from pixel level. Consider each of them as a homogeneous region.
- At any level of merging, check if four adjacent homogeneous regions arranged in 2 x 2 fashion, together satisfy the homogeneity property. If yes, then merge them into a single region; otherwise leave the regions as such.
- Repeat the operation recursively until there are no more regions that can be merged.

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

An 8x8 image with gray levels ranging from 0 to 7

Find the regions based on

1. Region splitting technique
2. Region merging technique

Threshold value = 3

FEATURE EXTRACTION

The resulting sets of segmented pixels have to be converted into a form suitable for further computer processing.

- Step after segmentation - **Feature Extraction**, which consists of
- **Feature detection** - finding the features in an image, region, or boundary.
  - For example, detect corners in a region boundary
- **Feature description** - assigning quantitative attributes to the detected features.
  - E.g. describe those corners by their orientation and location, both of which are quantitative attributes

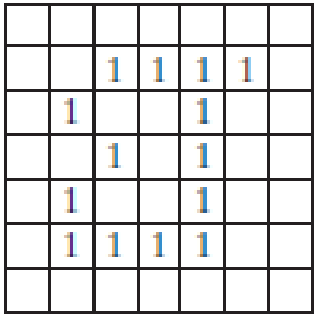
BOUNDARY FOLLOWING (TRACING)

- Many algorithms require that the points in the boundary of a region be ordered in a clockwise or counterclockwise direction.
- **Boundary-following algorithm** gives an ordered sequence of points.
  - **The image should be binary** in which object and background points are labeled 1 and 0, respectively
  - The **image is padded with a border of 0's** to eliminate the possibility of an object merging with the image border.

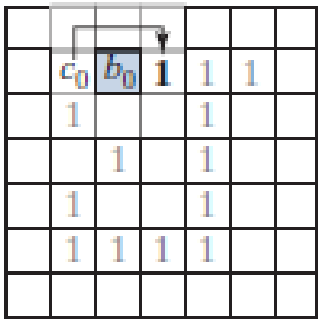
MOORE BOUNDARY TRACING ALGORITHM

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

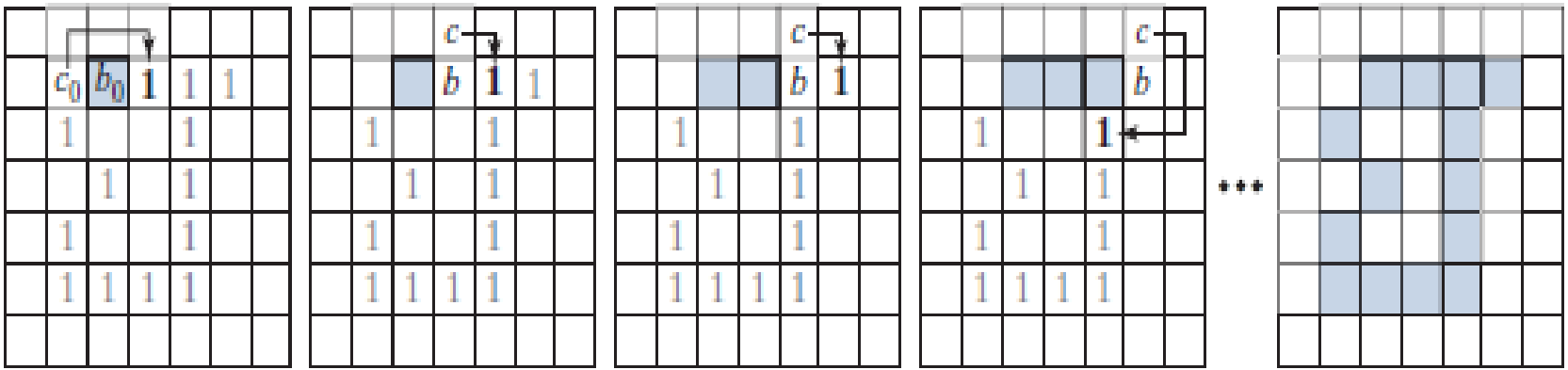
Output: A sequence B (b_0, b_1, ..., b_k) of boundary pixels i.e. the contour.

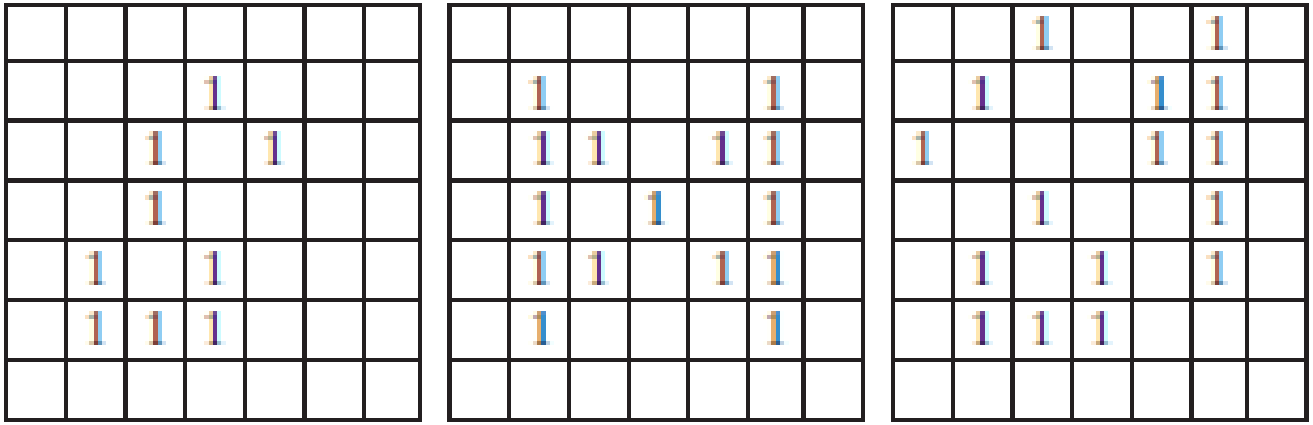


1. Let the starting point, b_0 , be the uppermost-leftmost point in the image that is labeled 1. Let c_0 the west neighbor of b_0 (it is always a background point). Examine the 8-neighbors of b_0 , starting at c_0 and proceeding in a clockwise direction. Let b_1 denote the first neighbor encountered whose value is 1, and let c_1 be the (background) point immediately preceding b_1 in the sequence.



2. Let $b = b_0$ and $c = c_0$. Insert b in B
3. Let the 8-neighbors of b, starting at c and proceeding in a clockwise direction, be denoted by $n_1, n_2, \ldots, n_8$. Find the first neighbor labeled 1 and denote it by n_k .
4. Let $b = n_k$ and $c = n_{k-1}$. Insert b in B
5. Repeat Steps 3 and 4 until $b = b_0$.





Examples of boundaries that can be processed by the boundary-following algorithm. (a) Closed boundary with a branch. (b) Self-intersecting boundary. (c) Multiple boundaries (processed one at a time).

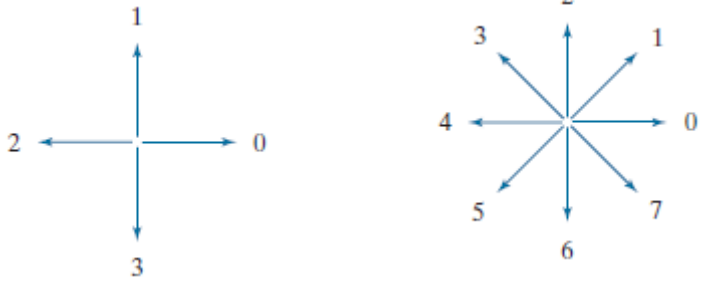
If we start with a binary region instead of a boundary, the algorithm extracts the outer boundary of the region. Typically, the resulting boundary will be one pixel thick, but not always

FREEMAN CHAIN CODES

Why focus on a boundary?
The boundary is a good representation of an object shape and also requires less memory.

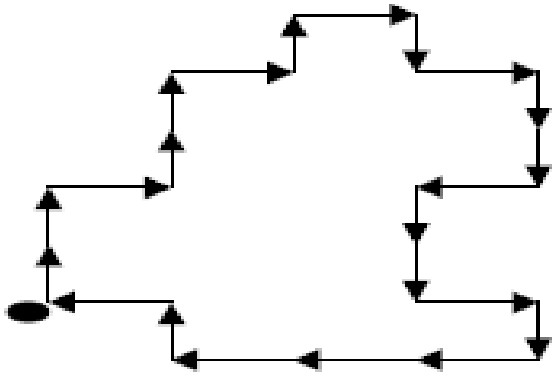
Chain codes: represent an object boundary by a connected sequence of straight line segments of specified length and direction.

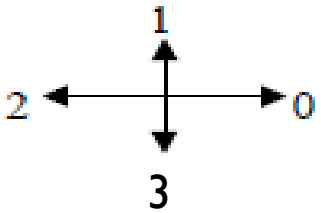
The direction of each segment is coded by using a numbering scheme shown below



Direction numbers for
(a) 4-directional chain code
(b) 8-directional chain code

Find the 4 directional Chain Code





11011010303323303222 12

PROBLEMS WITH CHAIN CODES

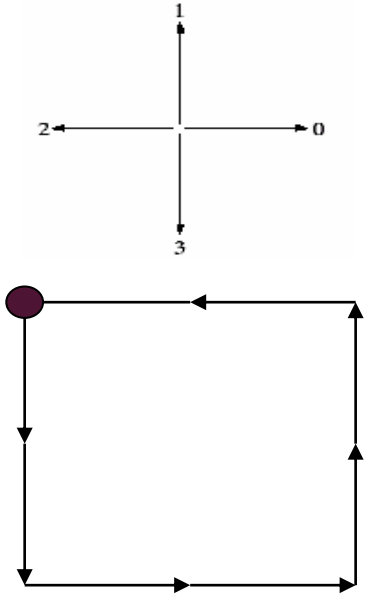
- the resulting chain of codes tends to be quite long
- any small disturbances along the boundary due to noise or imperfect segmentation cause changes in the code that may not be related to the shape of the boundary
- Dependent on the starting point
- Dependent on the orientation

To use boundary representation in object recognition, we need to achieve invariance to starting point and orientation

- **Normalized codes**
- **Differential codes**

NORMALIZATION STRATEGY

TO OVERCOME THE STARTING POINT PROBLEM



33001122

33001122	Sort rows →	00112233
30011223		01122330
00112233		11223300
01122330		12233001
11223300		22330011
12233001		23300112
22330011		30011223
23300112		33001122

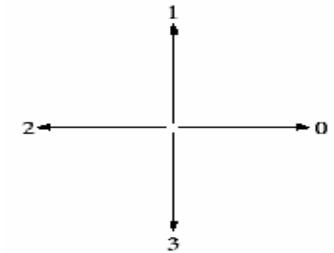
First row gives the normalized chain code

00112233

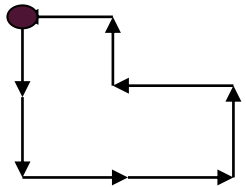
DIFFERENTIAL STRATEGY

NORMALIZATION FOR ROTATION

- Counting (counterclockwise) the number of direction changes that separate two adjacent element of the code
- Assuming the first difference code represents a closed path, rotation normalization can be achieved by circularly shifting the number of the code so that the list of numbers forms the smallest possible integer.



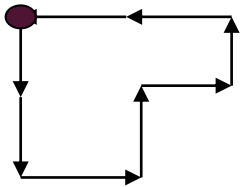
01011311



33001212

01011311

90°



33010122

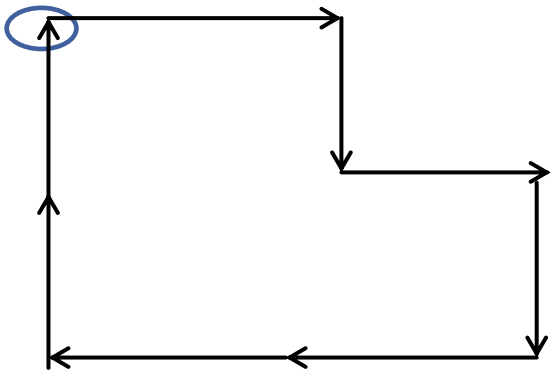
01131101

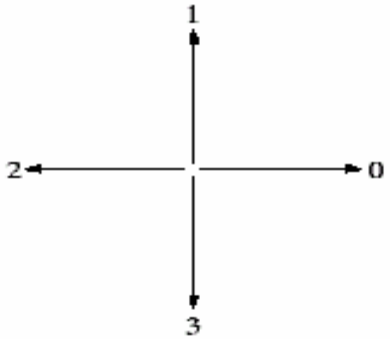
SHAPE NUMBERS

- The **shape number** of a boundary obtained from a chain code is defined as the smallest magnitude of the circular first difference.

- The **order of the shape number** is defined as the number of digits in its representation.

Find the shape number & order of the given boundary.





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