Image Segmentation Algorithms: An Overview

by

Dr. Amitava Chatterjee
Electrical Measurement and Instrumentation Laboratory,
Electrical Engineering Department,
Jadavpur University, Kolkata, India.

Image Segmentation Algorithms

✓ **The principal objective of image segmentation is to subdivide an image into its constituent regions or objects. The level of detail to which the subdivision is carried out depends on the problem under consideration.**

Image Segmentation Algorithms

Discontinuity based Approach

Partition/segment an image into regions based on abrupt changes in intensity

Similarity based Approach

Partition/segment an image into regions that are similar according to a set of predefined criteria

Image Segmentation Algorithms

- **Segmentation algorithms**
  - **Based on Gray Level Discontinuity**
    - Isolated Point Detection
    - Line Detection
    - Edge Detection
    - Edge Linking and Boundary Detection
  - **Based on Gray Level Similarity**
    - Thresholding Technique
    - Region Growing
    - Region Splitting and Merging

Image Segmentation Algorithms

Fundamental Concepts

Let *R* represent the entire spatial region occupied by an image.

Problem in Hand …

Partition/segment *R* into n subregions $R_1, R_2, \ldots, R_n$ such that:

(i): $\bigcup_{i=1}^{n} R_i = R$ → *Segmentation must be complete.*

(ii): R_i is a connected set, $i = 1, 2, \cdots, n$ → *Points in a region must be connected.*

(iii): $R_i \cap R_j = \Phi$ for all i and j, $i \neq j$ → *Region must be disjoint.*

(iv): $Q(R_i) = \text{TRUE}$ for $i = 1, 2, \cdots, n$ → *All pixels in R_i have the same intensity level.*

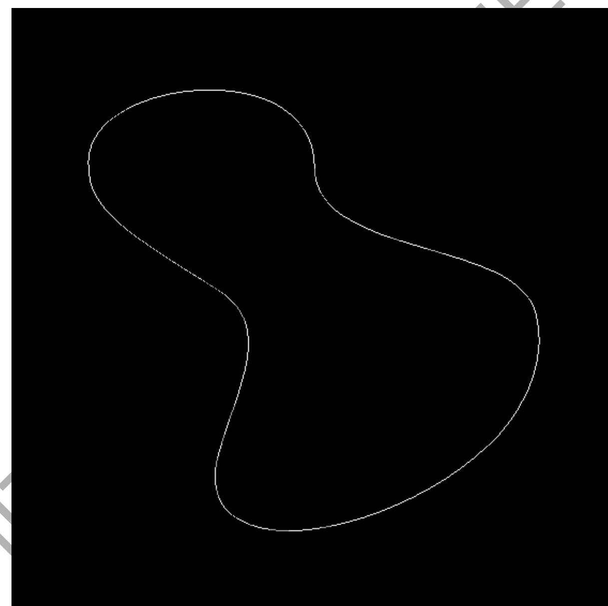
(v): $Q(R_i \cup R_j) = \text{FALSE}$ for adjacent regions R_i and R_j → *Two adjacent regions R_i and R_j must be different.*

Image Segmentation Algorithms

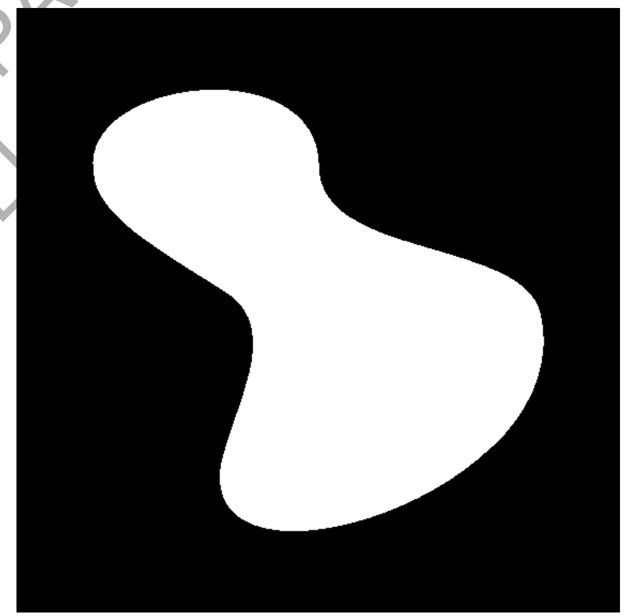
Fundamental Concepts (contd...)

A Basic Example …

An Original Image Containing a Region of Constant Intensity



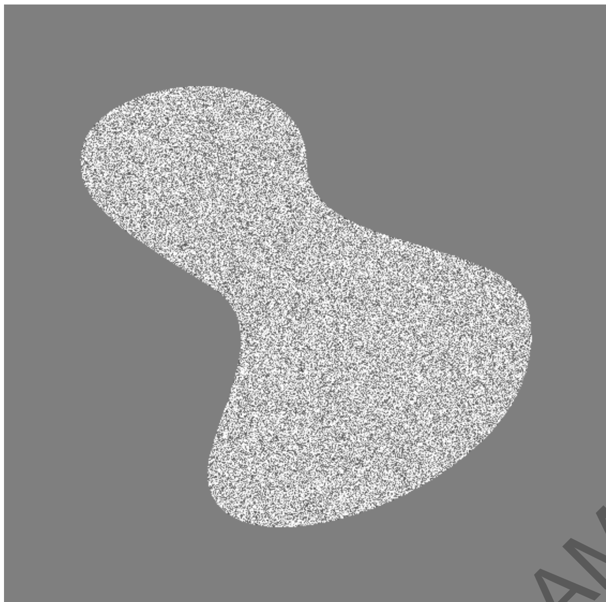
Boundary of the Inner Region obtained from Intensity Discontinuities



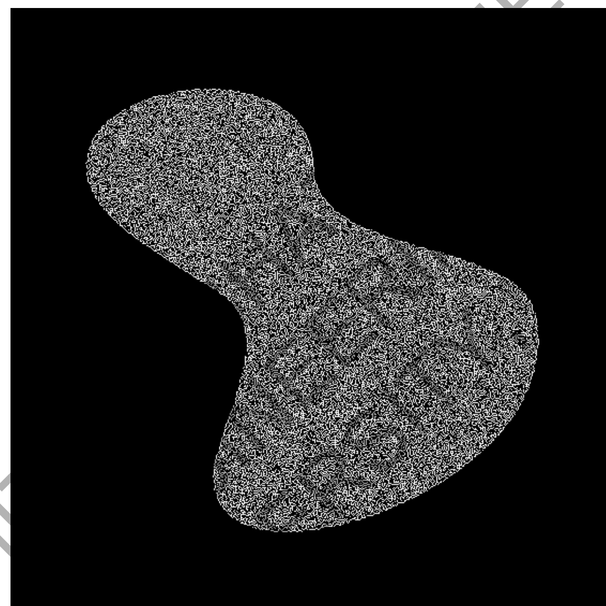
Segmented Image employing Thresholding

Image Segmentation Algorithms
Fundamental Concepts (contd…)

Basic Example contd …



Same Original Image Containing a Noisy (textured) Region



Processed image obtained from Edge Computations

Segmented Image employing Region Splitting and Merging

Discontinuity based Algorithms

Point, Line and Edge Detection

What is the Primary Objective ??

To detect *sharp, local* changes in intensity.

Three Types of Image Features under Consideration …

Isolated points, lines and edges.

What are Edge Pixels and Edges ??

Edge pixels are those pixels where the intensity of an image function changes abruptly. Edges (or edge segments) are sets of connected edge pixels.

What are Edge Detectors ??

Edge detectors are local image processing methods utilized to detect edge pixels.

Discontinuity based Algorithms
Point, Line and Edge Detection (contd…)

How to Detect Local Changes in Intensity ??

By using Derivatives.

Why Derivatives are Prime Candidates for this Operation ??

We know Averaging, which is analogous to integration, smoothes an image. Hence the operation of differentiation should logically be effective to detect abrupt, local changes in intensity.

Both First and Second-Order Derivatives are well suited for this purpose.

Discontinuity based Algorithms

Point, Line and Edge Detection (contd...)

Constraints of using an Approximation for 1st Derivative…

- ✓ **Must be zero in areas of** ***constant intensity.***
- ✓ **Must be non-zero at the** ***onset of an intensity step or ramp.***
- ✓ **Must be non-zero** ***along ramps.***

Constraints of using an Approximation for 2nd Derivative…

- ✓ **Must be zero in areas of** ***constant intensity.***
- ✓ **Must be non-zero at the** ***onset and end of an intensity step or ramp.***
- ✓ **Must be zero** ***along ramps of constant slope.***

Discontinuity based Algorithms

Point, Line and Edge Detection (contd...)

Implementing a 1st Derivative…

$$\frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x)$$

$$\frac{\partial f}{\partial x} = \frac{df}{dx}$$

when f is a function of only one variable.

Implementing a 2nd Derivative…

$$\frac{\partial^2 f}{\partial x^2} = f''(x) = \frac{\partial f'(x)}{\partial x} = f'(x+1) - f'(x)$$

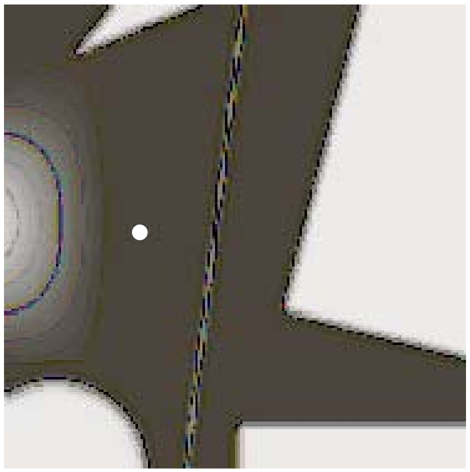
$$= f(x+2) - f(x+1) - f(x+1) + f(x)$$

$$= f(x+2) - 2f(x+1) + f(x)$$

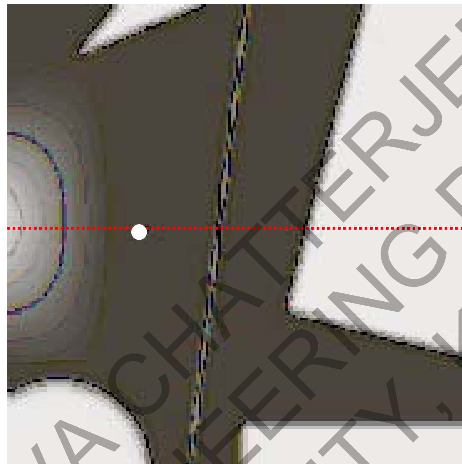
Discontinuity based Algorithms

Point, Line and Edge Detection (contd...)

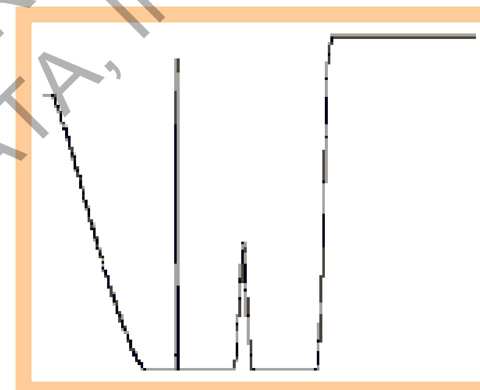
Comparison of 1^{st} and 2^{nd} Derivatives Illustrated…



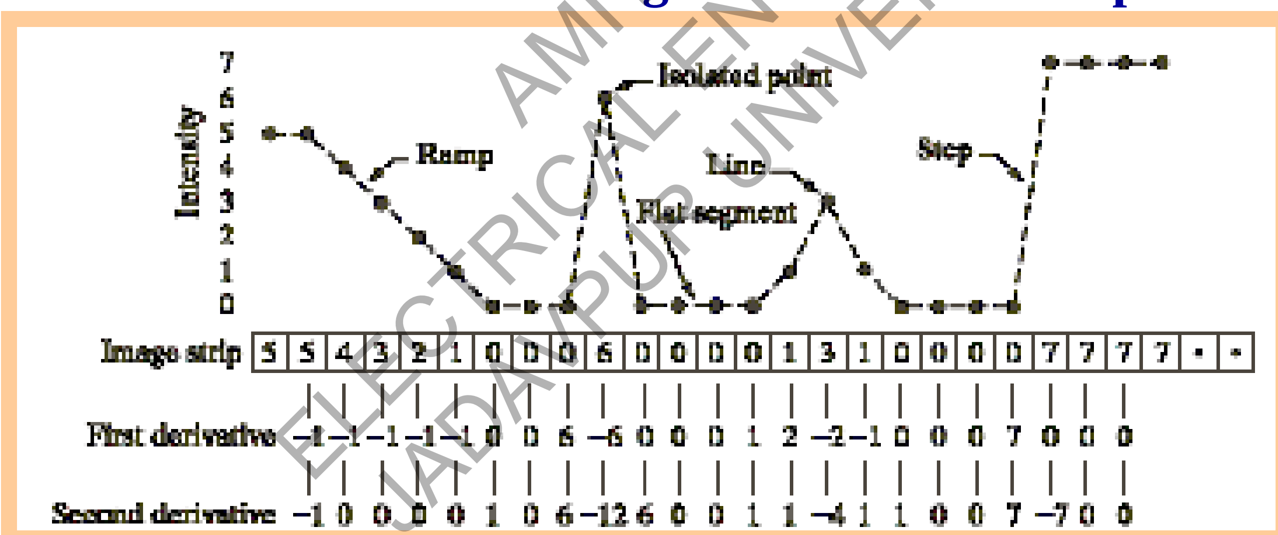
Original image



Same image with a horizontal line through the isolated noise point



Horizontal intensity profile

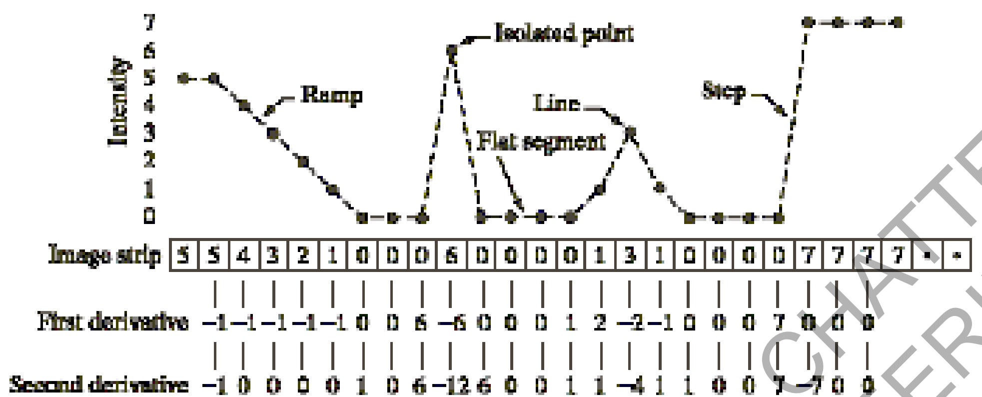


Simplified horizontal intensity profile

Discontinuity based Algorithms

Point, Line and Edge Detection (contd...)

Comparison of 1^{st} and 2^{nd} Derivatives Illustrated (contd…)



Simplified horizontal intensity profile

Conclusions…

First order derivatives **produce thick edges and *second order derivatives* produce finer edges.**

For both ramp and step edges second derivative produce double-edge effect.

The *sign* of the second order derivative is used to determine whether an edge is a transition from light to dark or dark to light.

Discontinuity based Algorithms

Point, Line and Edge Detection (contd...)

Comparison of 1st and 2nd Derivatives Illustrated (contd…)

Salient Points to be Remembered…

First order derivatives generally produce thicker edges in an image.

Second order derivatives have a stronger response to fine detail e.g. thin lines, isolated point and noise.

Second order derivatives produce a double-edge response at ramp and step transitions of intensity.

The *sign* of the second order derivative can be used to determine whether an edge is a transition from light to dark or dark to light.

Discontinuity based Algorithms

Point, Line and Edge Detection (contd...)

How to Compute 1st and 2nd Derivatives at Every Pixel Location ??

By using Spatial Filters.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

A 3 × 3 image section under consideration

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

A 3 × 3 spatial filter mask employed for this image section

Response of the mask at the center point of the region = *R*

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

Isolated Point Detection

This should be based on computation of second derivative.

Apply the *Laplacian*.

The Laplacian of an image function *f*(*x*, *y*) of two variables:

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

Common digital approximations of the second derivatives:

$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

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

Isolated Point Detection

The masks for 2nd derivative operators of size 3×3:

0	1	0
1	-4	1
0	1	0

Mask 1

Mask employing Laplacian considering conventional horizontal and vertical directions

1	1	1
1	-8	1
1	1	1

Mask 2

Mask employing Laplacian considering four directions: (a) horizontal, (b) vertical, (c) +45° and (c) -45° directions

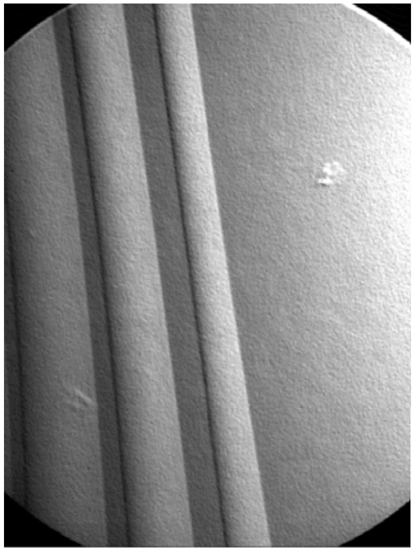
A point is said to be detected at point (*x*, *y*), on which the mask is centered, if |*R*(*x*,*y*)| at that point exceeds a threshold.

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

***T*: a non-negative threshold**

Isolated Point Detection

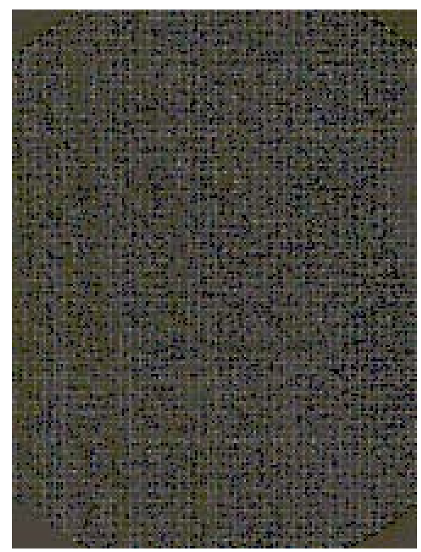
An Example …



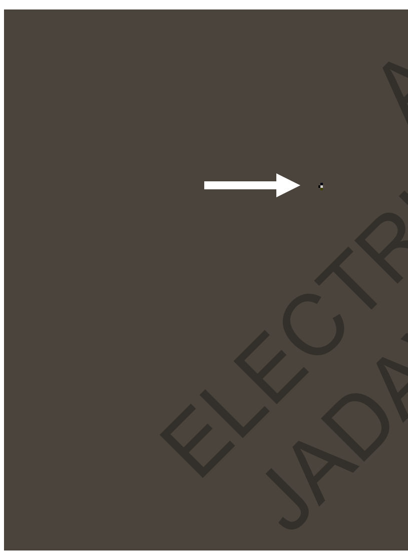
X-Ray Image of Turbine Blade with a Porosity (contains a single black pixel)

1	1	1
1	-8	1
1	1	1

A Laplacian Mask



Result of Convolution



Result of Thresholding the Response (single point detected shown enlarged)

Line Detection

Second derivatives should produce stronger response and thinner lines than first derivatives.

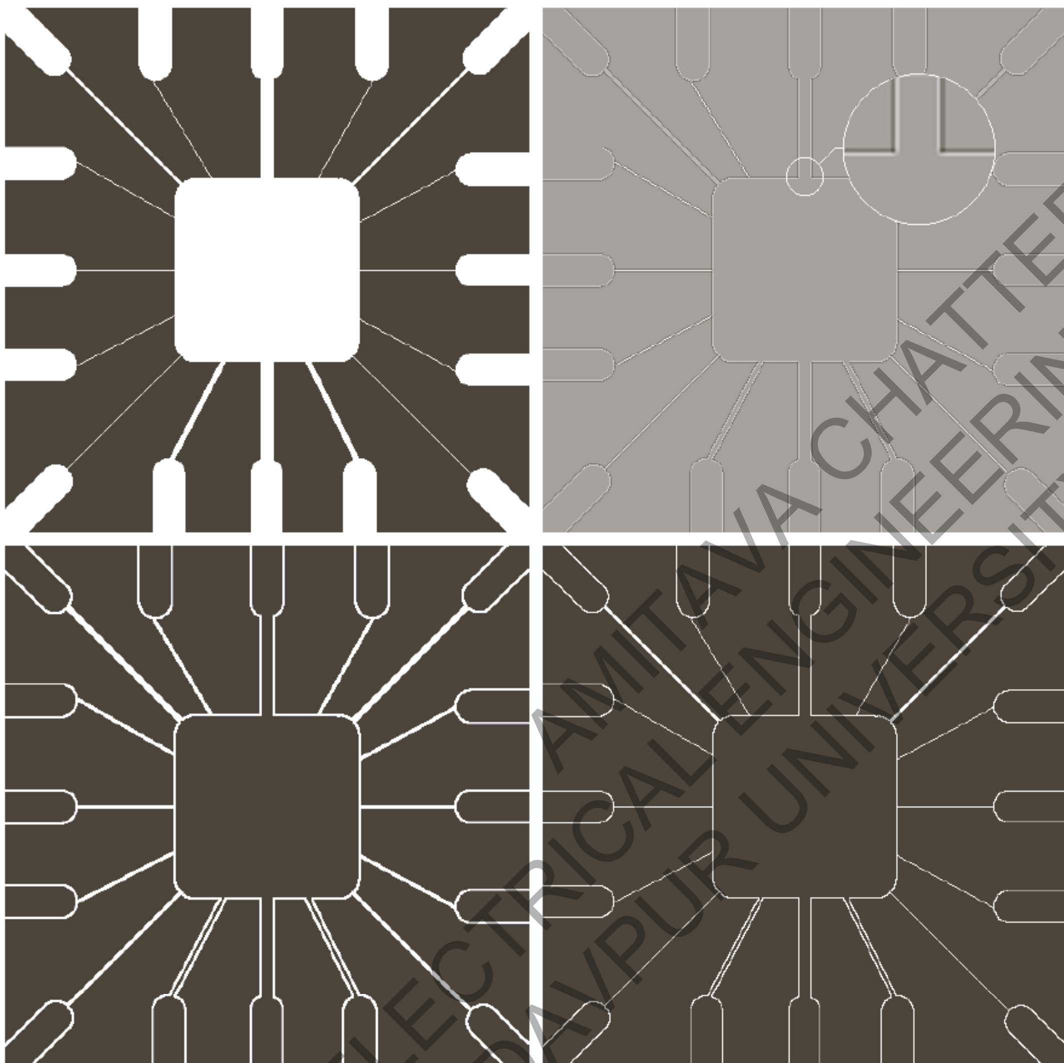
We can utilize the *Laplacian* masks as we did in case of point detection.

An Important Note…

The double-line effect of the second derivative has to be handled properly.

Line Detection

An Example…



(a) Original Image of a wire-bond mask

(b) Laplacian Image considering the mask in the previous example

(c) Absolute value of the Laplacian

(d) Positive values of the Laplacian

Line Detection

An Important Observation …

The Laplacian detector used in the previous example gives a response which is independent of direction.

If we are Interested in Detecting Lines in Specified Directions, what should we do ???

We have to utilize *Special Line Detection Masks*.

Horizontal

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

+45°

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

Vertical

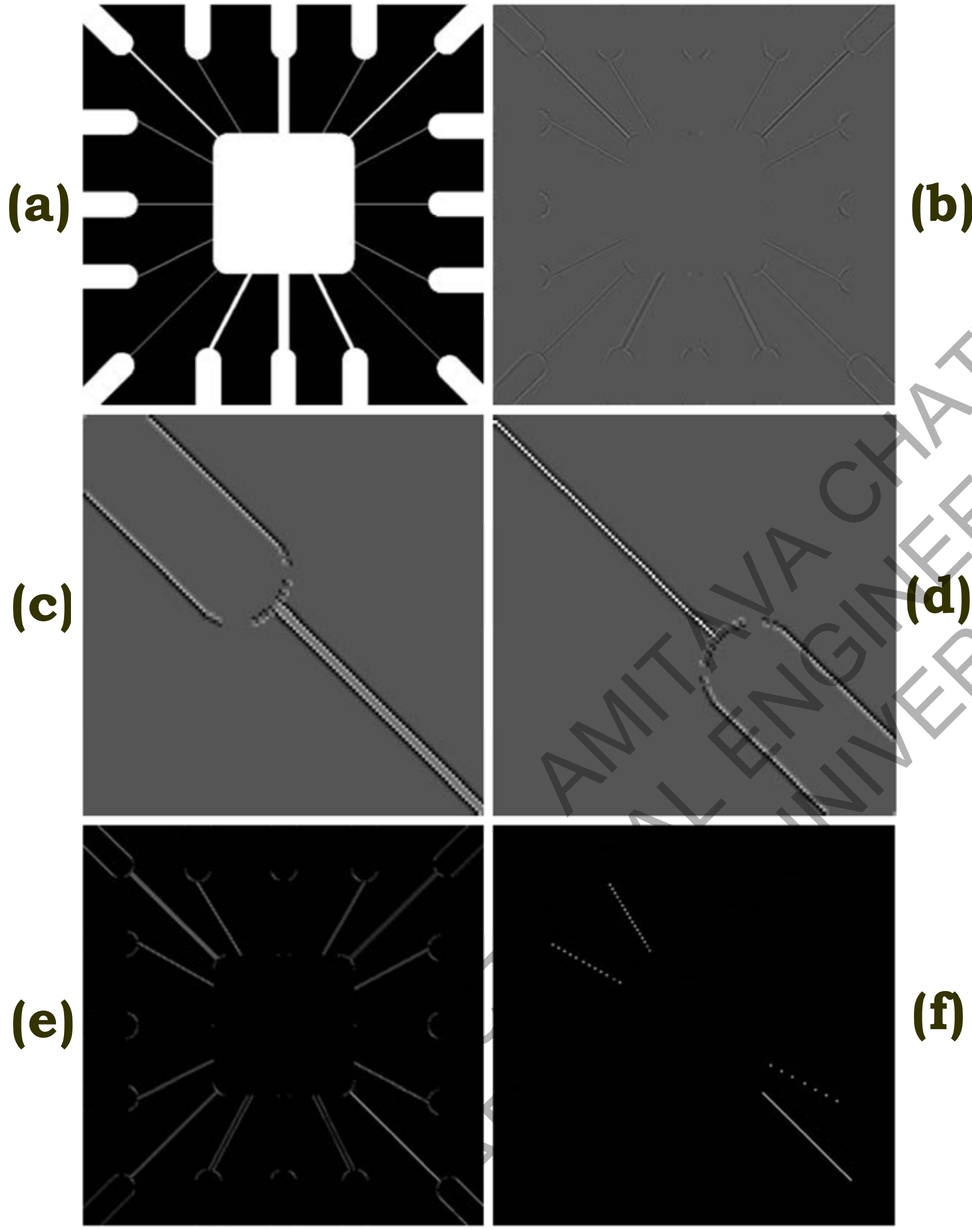
-1	2	-1
-1	2	-1
-1	2	-1

-45°

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

Line Detection

Previous Example Revisited …



(a) Original Image of the same wire-bond mask

↓

(b) Processed Image utilizing the 45° line detection mask

↓

(c) Zoomed view of the top left region of (b)

↓

(d) Zoomed view of the bottom right region of (b)

↓

(e) Image in (b) with All negative values set to zero

↓

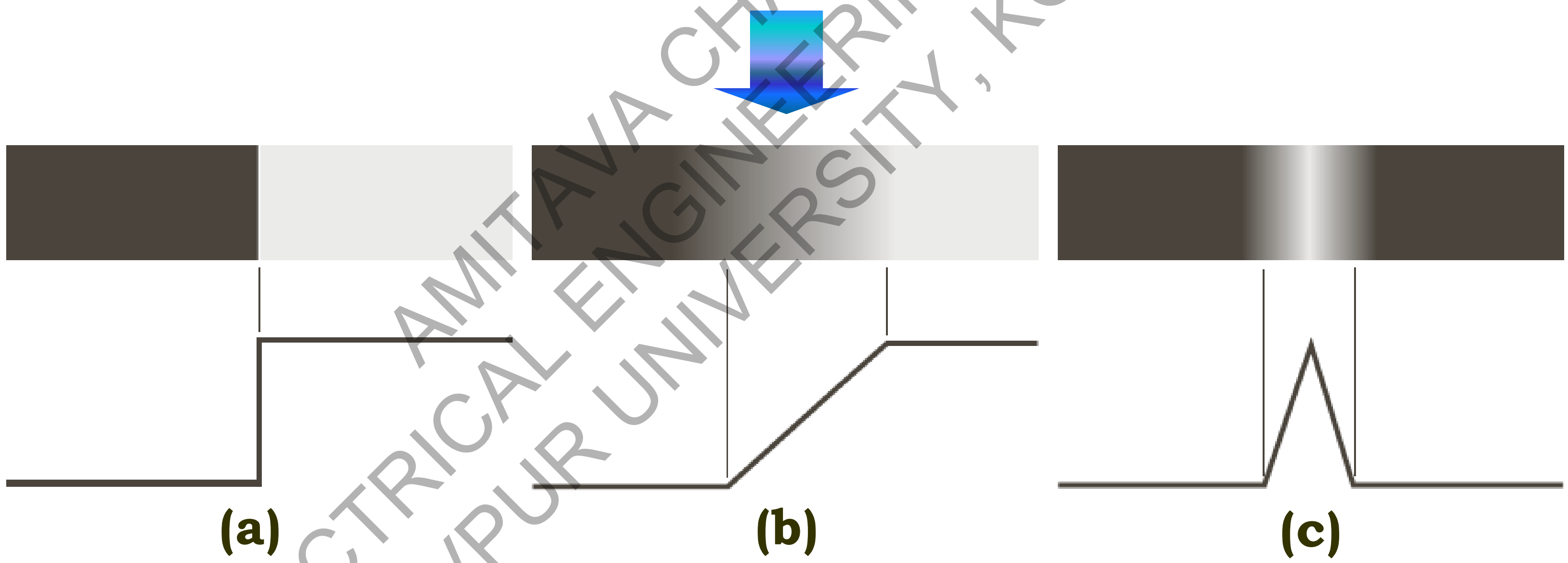
(f) Resulting of Thresholding operation on the Image in (e)

Edge Detection

Edge Detection is the most frequently used technique for segmenting images based on abrupt (local) changes in intensity.

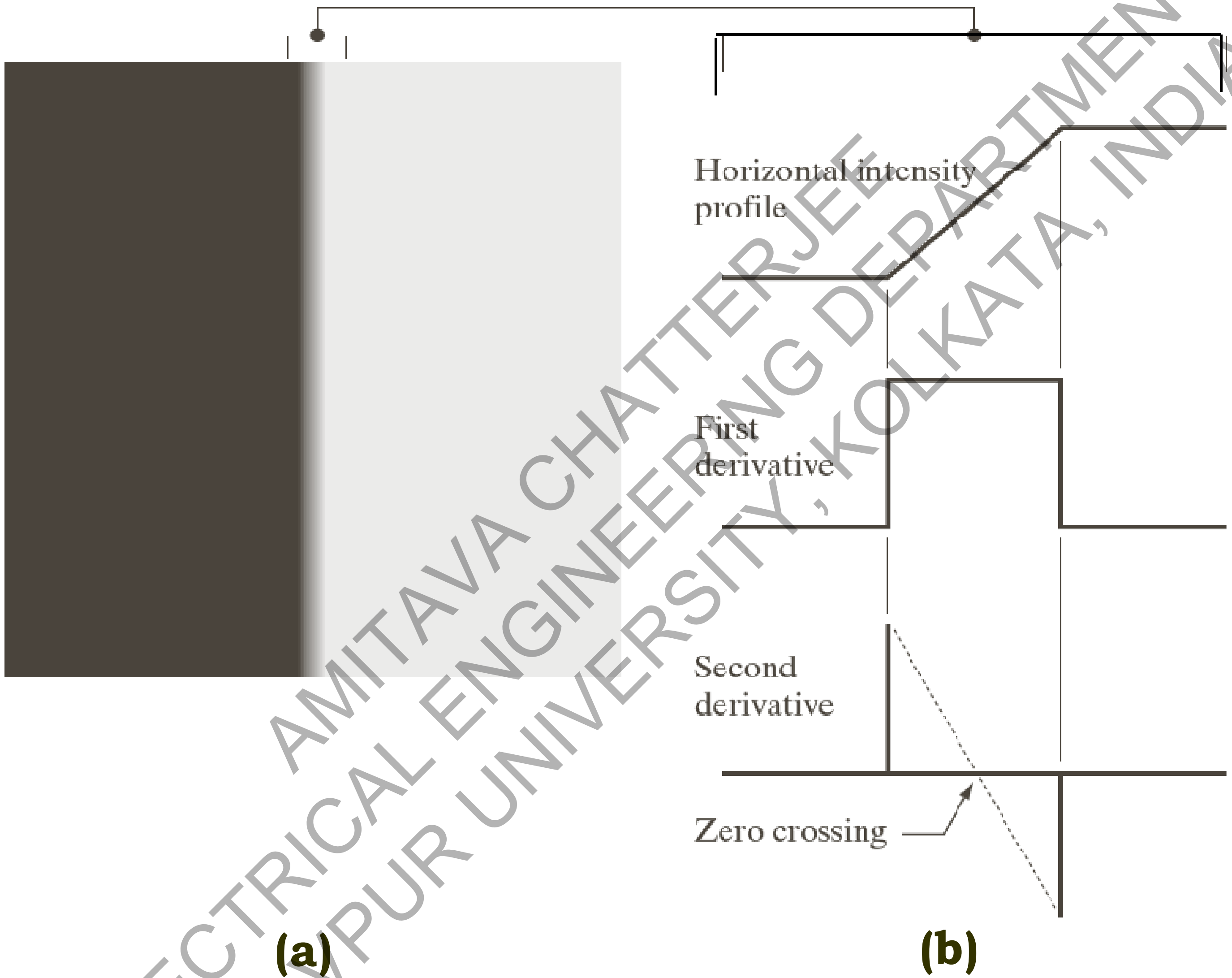
Edge Models

They are classified according to their *intensity profiles*.



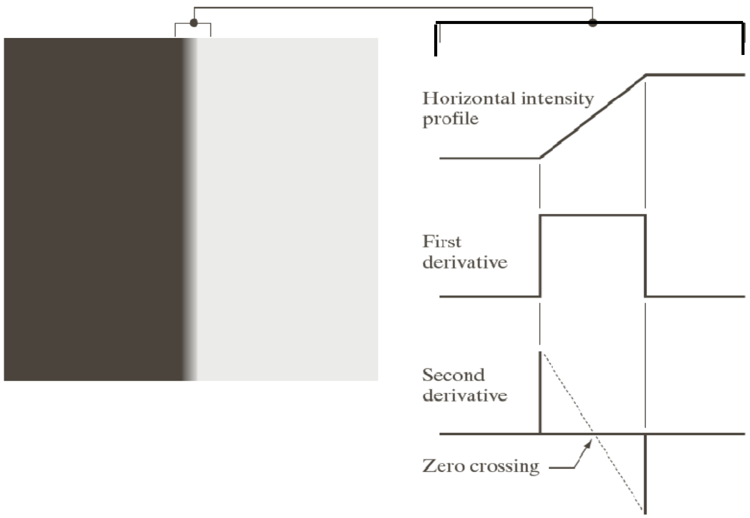
Ideal representations and corresponding intensity profiles of (a) step edge, (b) ramp edge, and (c) roof edge.

Edge Detection



(a) Two regions of constant intensity separated by an ideal vertical ramp edge. (b) Detail near the edge with first and second derivatives.

Edge Detection



Conclusion …

The magnitude of the first derivative can be utilized to detect the presence of an edge at a point in the image.

The sign of the first derivative can be utilized to determine whether an edge pixel lies on the dark or light side of the edge.

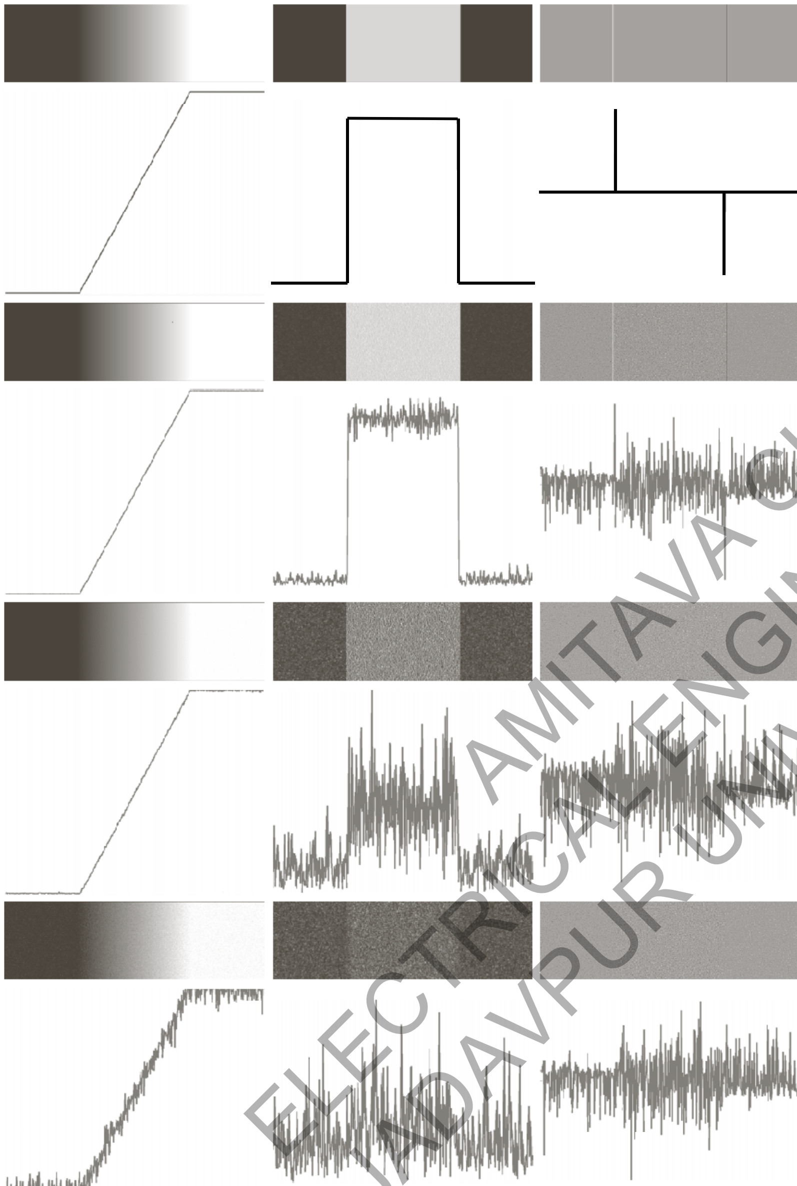
Important Points to be Considered …

The second derivative produces two values for every edge in the image, which is an undesirable feature.

The zero-crossing of the second derivative can be used to locate the centers of thick edges.

Edge Detection

A Comparative Study of 1st and 2nd Derivatives of a Noisy Edge



Column 1. **Images and Intensity Profiles of a Ramp Edge corrupted by Gaussian noise of mean = 0 and S.D. = 0, 0.1, 1.0, and 10.0 intensity levels, respectively.**

Column 2. **First Derivative images and Intensity Profiles.**

Column 3. **Second Derivative images and Intensity Profiles.**

Edge Detection

Three Fundamental Steps Performed in Edge Detection …

Step 1. → **Image Smoothing for Noise Reduction.**

Step 2. → **Detection of Edge Points.**

Step 3. → **Localization of Edge.**

Edge Detection using Image Gradient

The **edge strength** and **direction** at location (x, y) of an image f can be found by using **the gradient** ∇f.

The gradient of a function $f(x, y)$ at coordinates (x, y) is defined as the two-dimensional column vector:

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

The magnitude (length) of vector ∇f:

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

$$M(x,y) \approx |g_x| + |g_y|$$

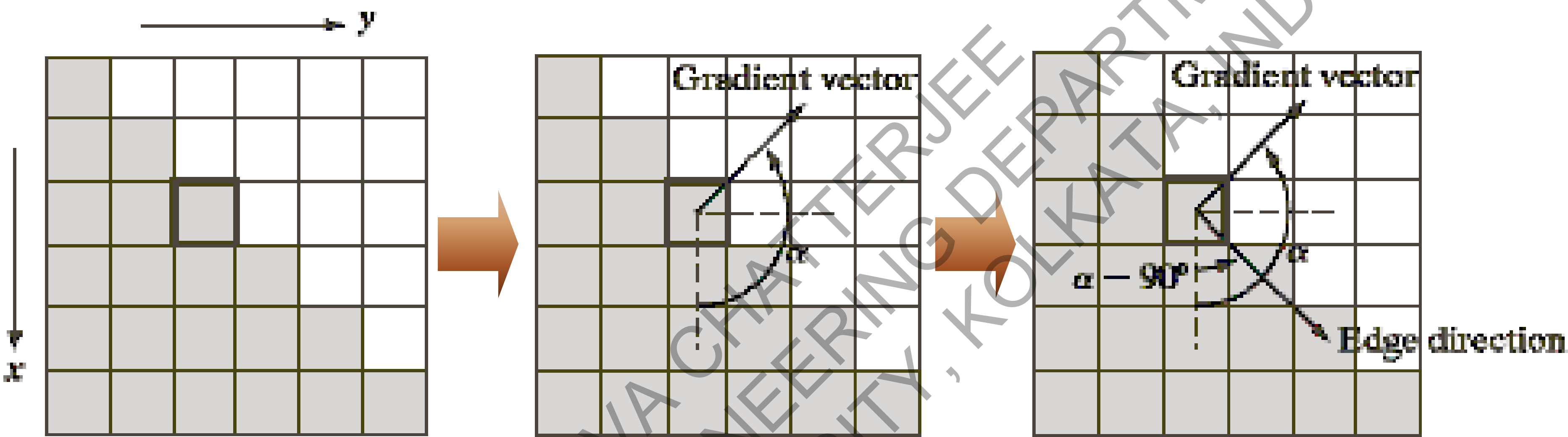
The direction of vector ∇f with respect to the x-axis:

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

g_x, g_y, $M(x, y)$ and $\alpha(x, y)$ are images of the same size as the original. $M(x, y)$ is called the gradient image. The direction of an edge at an arbitrary point (x, y) is *orthogonal* to the direction, $\alpha(x, y)$, of the gradient vector at the point.

Edge Detection using Image Gradient

Determination of Edge Strength and Direction at a Point



(a) (b) (c)

Note. **Each square in the figure represents one pixel. The pixels in gray have value 0 and the pixels in white have value 1.**

At our point of interest:

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

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

$$\alpha(x, y) = \tan^{-1}\begin{bmatrix} \frac{g_y}{g_x} \end{bmatrix} = -45°$$

The direction angle of the edge:

$$\alpha - 90° = 45°$$

The entire edge segment is in the same direction.

Edge Detection using Image Gradient

Gradient Operators

Digital Approximation of First Derivative:

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y)$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y)$$

When diagonal edge directions are important, we need 2-D Masks.

Roberts Cross-Gradient Operators:

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5); \quad g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$

-1	**0**
0	**1**

0	**-1**
1	**0**

-1
1

-1	**1**

One-dimensional masks

Masks that are symmetric about the center point are more popular than masks of even size e.g. 2 × 2 .

Edge Detection using Image Gradient

Gradient Operators

The masks for 1st derivative operators of size 3 × 3:

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

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

Sobel operators

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

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

$$M(x,y) \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

Edge Detection using Image Gradient

Gradient Operators

The masks for 1st derivative operators of size 3 × 3:

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

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

Prewitt operators

$$M(x,y) \approx \left|(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)\right| + \left|(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)\right|$$

Edge Detection using Image Gradient

Gradient Operators

Prewitt and Sobel masks for detecting Diagonal Edges

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

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

Prewitt

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

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

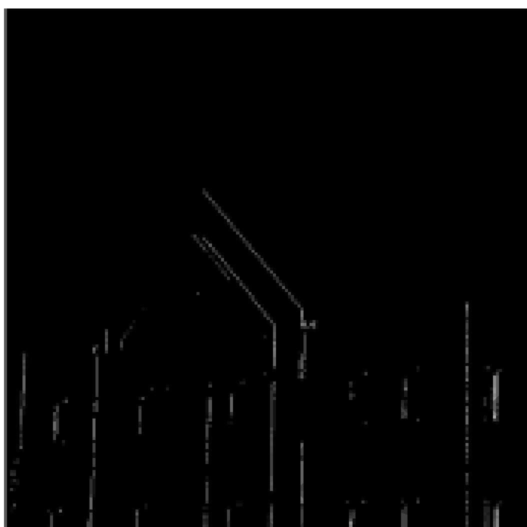
Sobel

Edge Detection using Image Gradient

An Example…

(a) Original Image of a building

(b) Processed Image utilizing the vertical Sobel mask with Automatic Thresholding



(c) Processed Image utilizing the vertical Sobel mask with Specified Threshold

(d) Processed Image utilizing both vertical and horizontal Sobel mask with Specified Threshold

Edge Detection using 2nd Derivative

- ✓ **The Laplacian of a Gaussian (LOG) Edge Detector.**
- ✓ **The Canny Edge Detector.**

Edge Linking and Boundary Detection

Constraints of Edge Detection Algorithms…

The pixels identified on edges seldom completely characterize edges.

Why ??

Because of noise, breaks in the edges due to non-uniform illumination, and other effects that introduce spurious discontinuity in intensity values.

Is there any Solution ??

***Yes*, the Edge Detection should typically be followed by Edge Linking algorithms so that the edge pixels identified are assembled into meaningful edges and/or region boundaries.**

Three fundamental approaches to Edge Linking:

i) Local Processing

ii) Regional Processing

iii) Global Processing

Edge Linking and Boundary Detection

Local Processing

Analyze the characteristics of pixels in a small neighborhood of each edge pixel, identified by any edge detection algorithm, utilizing a predefined criteria.

Link all those points which are found to share some common properties according to the specified criteria.

How to specify the Criteria for Establishing Similarity ??

Utilize the Strength (magnitude) and the Direction of the gradient vector.

Edge Linking and Boundary Detection

Local Processing

Let S_{xy} denote the set of coordinates centered at point (x, y). An edge pixel with coordinates (s, t) in S_{xy} is similar to the pixel at (x, y) if both magnitude criterion and angle criterion are satisfied. Then the pixel at (s, t) can be linked to the pixel at (x, y).

Magnitude Criterion

$$|M(s,t) - M(x,y)| \le E$$

E: a positive threshold

Angle Criterion

$$|\alpha(s,t) - \alpha(x,y)| \le A$$

A: a positive angle threshold

This process is repeated at every location in the image.

However this method is *Computationally Expensive* because all neighbors of every point are needed to be examined.

Edge Linking and Boundary Detection

Local Processing

A Simplified Procedure for Real Time Applications:

Compute Gradient magnitude and angle arrays, $M(x, y)$ and $\alpha(x, y)$ of the input image $f(x, y)$

↓

Form a Binary image g →

$$g(x,y)=\begin{cases}1 & \text{if } M(x,y)>T_M \text{ AND } \alpha(x,y)=A\pm T_A\\ 0 & \text{otherwise}\end{cases}$$

T_M: a threshold; A: specified angle direction;
$\pm T_A$: *band* of acceptable directions about A

↓

Scan rows of g and fill (set to 1) all gaps (sets of 0s) in each row that do not exceed K → **K: a specified length**

↓

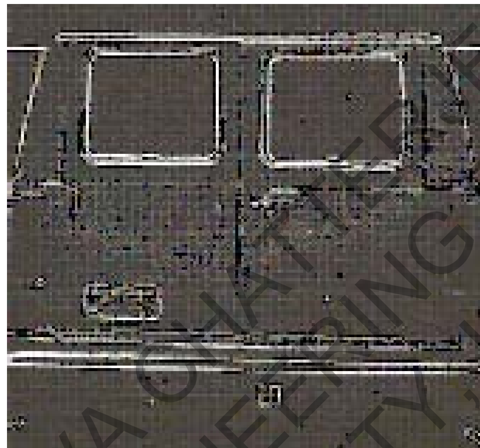
To determine gaps in an arbitrary direction θ, rotate g by θ, apply the previous step and rotate the result back by $-\theta$

Edge Linking and Boundary Detection

Edge Linking by Local Processing

An Example…

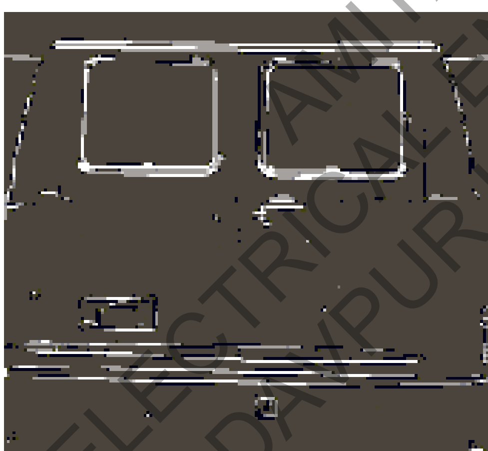
An Image of the Rear of a Vehicle.



Gradient Magnitude Image.

Horizontally connected Edge Pixels.

Vertically connected Edge Pixels.



The Logical OR of the two Preceding Images.

Edge Linking and Boundary Detection

Local Processing

Constraints of Local Processing…

This method can be used in those situations where at least partial knowledge about pixels belonging to individual objects is available.

Will there be any Problem in Practical Situations ??

***Yes*, usually we only have the edge image at our disposal and no knowledge about the locations of objects of interest is available.**

Then, what should be done ??

Then all pixels are candidates for linking and must be accepted or rejected based on predefined global properties.

Edge Linking and Boundary Detection

Global Processing

A Simple Method:

The Problem: **Given n points in an image, we want to find subsets of these points that lie on straight lines.**

The Solution: **Find first all lines determined by every pair of points and then find all subsets of points that are close to particular lines.**

This method requires determination of $n(n-1)/2 \sim n^2$ lines and then $(n)(n(n-1))/2 \sim n^3$ number of comparisons of every point to all lines.

Any Problem ?? → **This method is too computation heavy.**

Any Solution ?? → **Use Hough Transform.**

Edge Linking and Boundary Detection

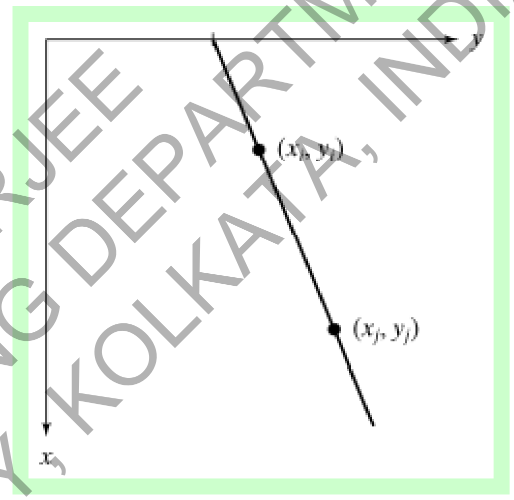
Global Processing by Hough Transform

Line Detection:

Consider a point (x_i, y_i) in the *xy*-plane.

straight line in slope-intercept form

$$y_i = ax_i + b$$

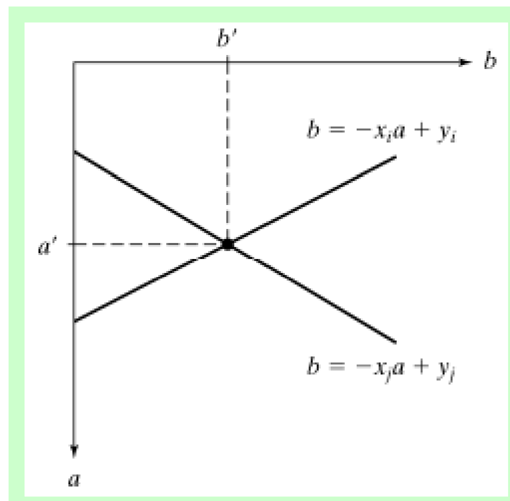


xy-plane

Consider the *ab*-plane or *parameter space*.

straight line in slope-intercept form

$$b = -x_i a + y_i$$



parameter space

Conclusion: **The line containing both (x_i, y_i) and (x_j, y_j) points in the *xy*-plane is the line with a' *slope and* b' *intercept.***

Edge Linking and Boundary Detection

Global Processing by Hough Transform

Line Detection (contd…):

We can plot lines in the *parameter space* for all points (x_k, y_k) in the *xy* - plane and the *principal lines* in that plane can be found by identifying points in the parameter space where large number of *parameter space* lines intersect.

Any Problem with this Method ??

There is a practical difficulty when the line approaches the vertical direction.

Any Solution for this Problem ??

We should use the normal representation of a line.

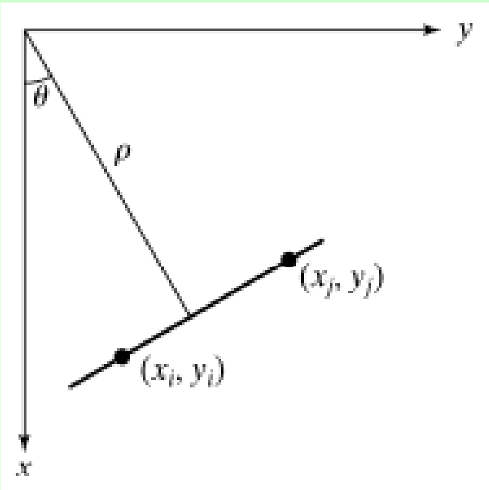
normal representation

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

Edge Linking and Boundary Detection

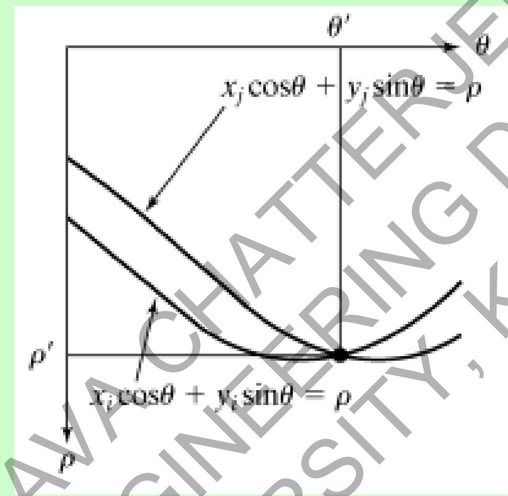
Global Processing by Hough Transform

Line Detection (contd…):

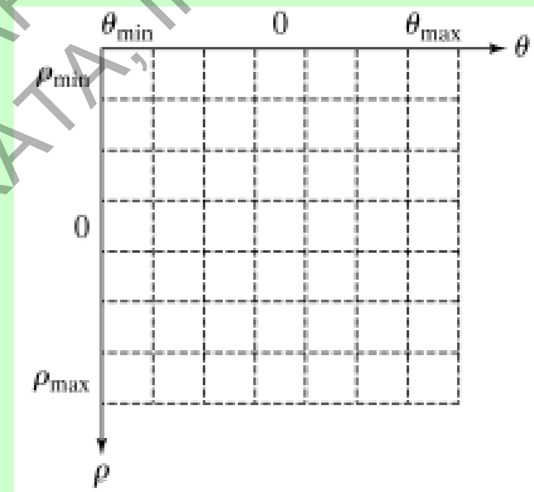


(ρ, θ) parameterization of line in the *xy*-plane.

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



Sinusoidal curves in the $\rho\theta$-plane. (ρ', θ'): point corresponding to the line passing through (x_i, y_i) and (x_j, y_j) in the *xy*-plane.



Division of $\rho\theta$-plane into accumulator cells.

$$-90° \le \theta \le 90°$$

$$-D \le \rho \le D$$

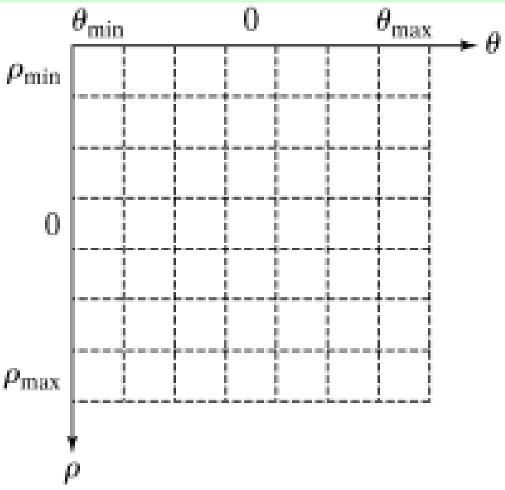
Why is Hough Transform Computationally Attractive ??

Because of the subdivision of the $\rho\theta$-plane into *accumulator cells*.

Edge Linking and Boundary Detection

Global Processing by Hough Transform

Line Detection (contd…):



Division of $\rho\theta$-plane into accumulator cells.

$$-90° \le \theta \le 90°$$

$$-D \le \rho \le D$$

***D*: Maximum distance between opposite corners in an image.**

The cell at coordinates (*i*, *j*) with accumulator value *A*(*i*, *j*) corresponds to the square associated with parameter-space coordinates (ρ_i, θ_j).

Initialization: All cells are initially set to zero.

For every non-background point (x_k, y_k), in the *xy*-plane, θ is made equal to each allowed subdivision on the θ-axis and solve for $\rho = x_k\cos\theta + y_k\sin\theta$.

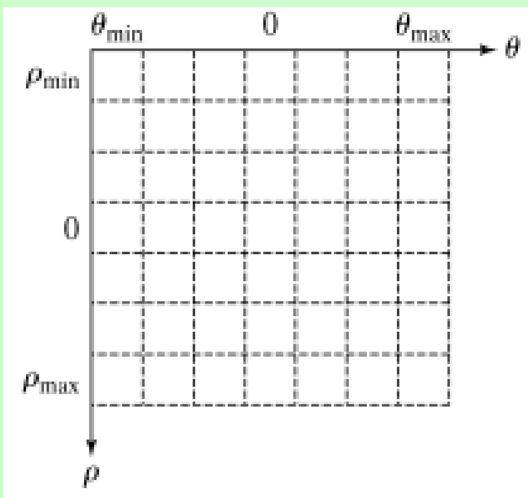
Round-off ρ to nearest allowed cell value along ρ-axis.

Make *A*(*p*, *q*)=*A*(*p*, *q*)+1, if a choice of θ_p results in ρ_q.

Edge Linking and Boundary Detection

Global Processing by Hough Transform

Line Detection (contd…):



Division of $\rho\theta$-plane into accumulator cells.

$$-90° \leq \theta \leq 90°$$

$$-D \leq \rho \leq D$$

***D*: Maximum distance between opposite corners in an image.**

Conclusion at the End of the Previous Procedure…

A value of *P* in *A*(*i*, *j*) means that *P* points in the *xy*-plane lie on the line $x\cos\theta_j + y\sin\theta_j = \rho_i$.

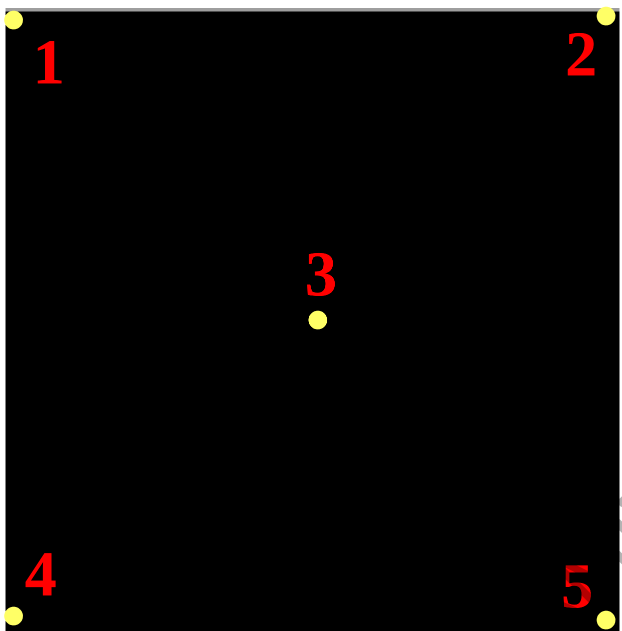
An Important Observation…

The number of subdivision in the $\rho\theta$-plane determines the accuracy of the colinearity of these points.

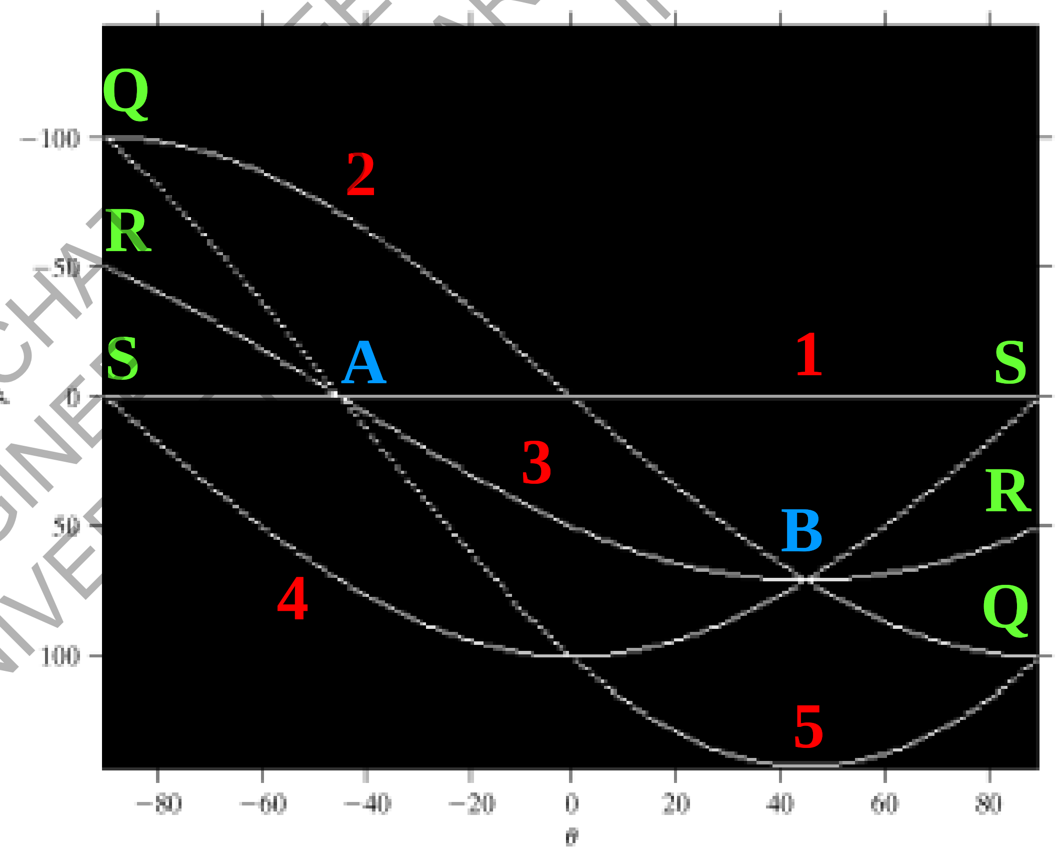
Edge Linking and Boundary Detection

Line Detection by Hough Transform

An Example…



An image containing five points.



Corresponding parameter space.

Edge Linking and Boundary Detection

Hough Transform for the Edge Linking Problem

Obtain a binary edge image using any of the Edge Detection Techniques.

↓

Specify subdivisions in the ρθ-plane.

↓

Examine the counts of the accumulator cells for high pixel concentrations.

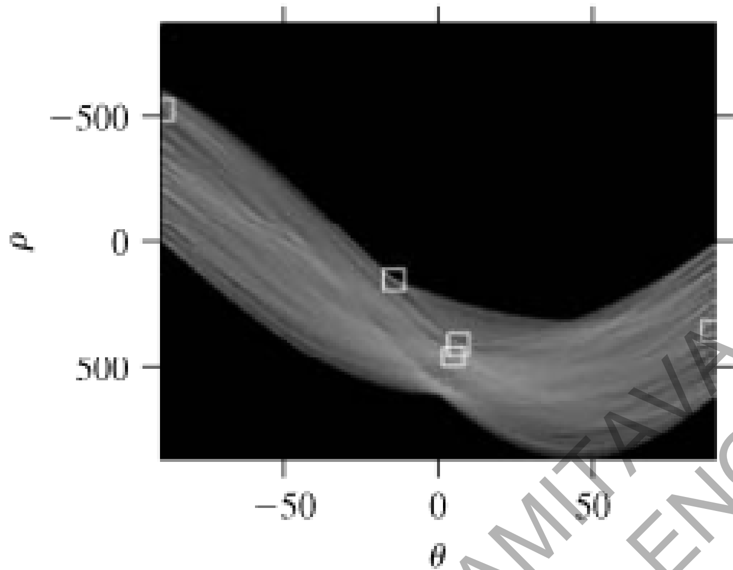
↓

Examine the relationship (principally for continuity) between pixels in a chosen cell.

Edge Linking and Boundary Detection

Line Detection and Linking by Hough Transform

An Example…



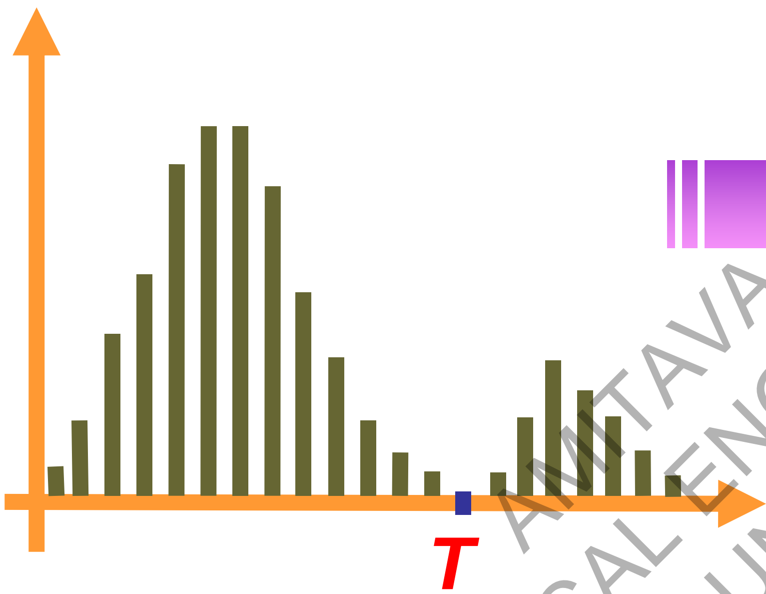
Hough Transform with five
Peak Locations selected.

Line Segments Corresponding to
Hough Transform Peaks.

Similarity based Algorithms

Thresholding

In ***Thresholding***, **we partition images directly into regions based on intensity values.**



Intensity histogram of an image $f(x, y)$, composed of light objects on a dark background.

Segmented Image:

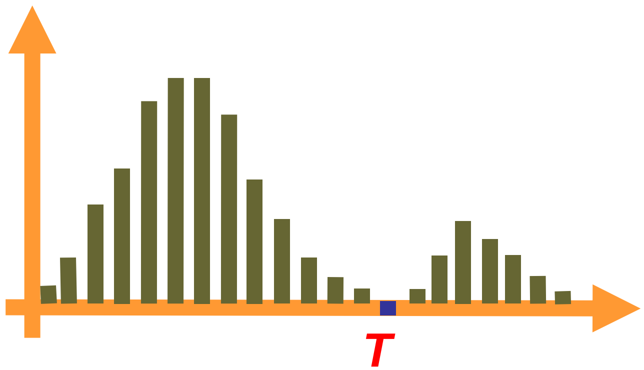
$$g(x,y)=\begin{cases}1 & \text{if } f(x,y)>T\\ 0 & \text{if } f(x,y)\le T\end{cases}$$

A point (x, y) at which $f(x, y) > T$, is called an *object point*.

A point (x, y) at which $f(x, y) \le T$, is called a *background point*.

Similarity based Algorithms

Thresholding



Intensity histogram of an image *f*(*x*, *y*), composed of light objects on a dark background.

Segmented Image:

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

If *T* is a constant applicable over an entire image: → *Global Thresholding*

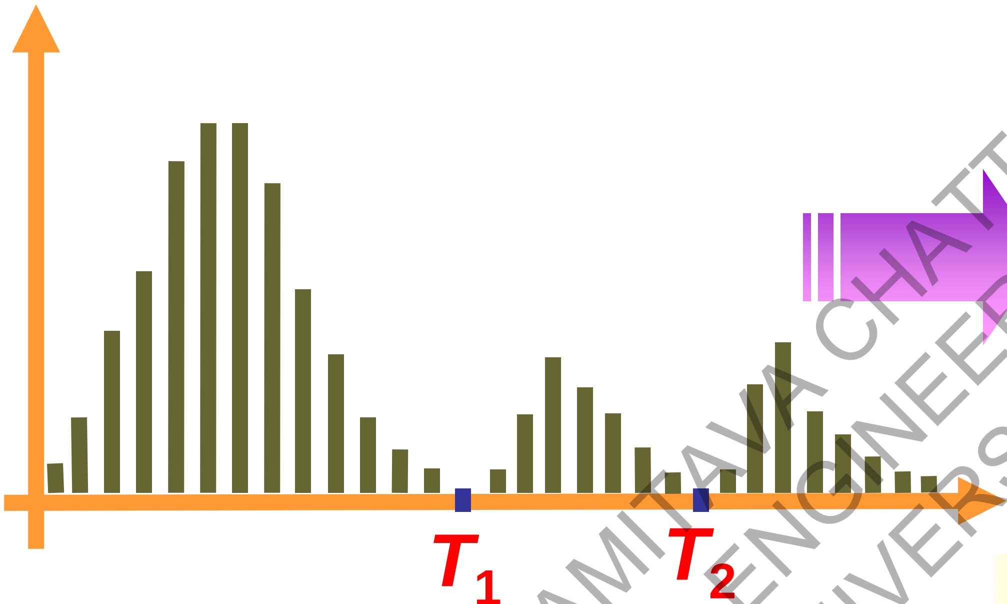
If *T* changes over an image: → *Variable Thresholding*

Variable thresholding where *T* at any point (*x*, *y*) depends on properties of a neighborhood of (*x*, *y*): → *Local Thresholding*

Similarity based Algorithms

Thresholding

A More Difficult Thresholding Problem…



An image intensity histogram with three dominant modes corresponding to e.g. two types of light objects on a dark background.

Segmented Image:

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

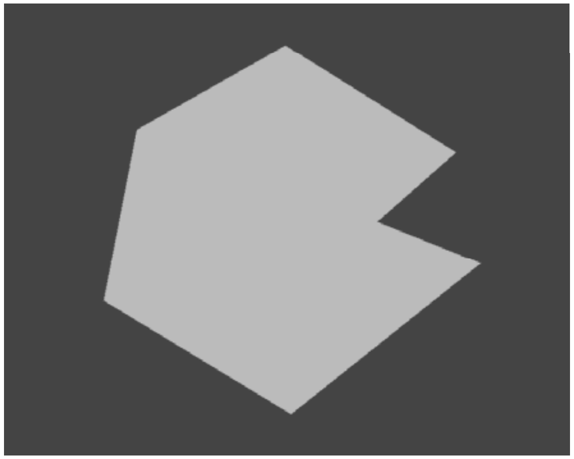
A point (x, y) at which $f(x, y) \le T_1$, is called a ***background point***.

A point (x, y) at which $T_1 < f(x, y) \le T_2$, is called a ***point in one object class***.

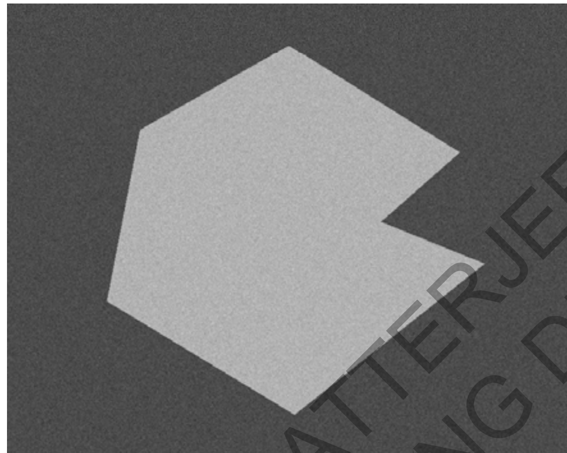
A point (x, y) at which $f(x, y) > T_2$, is called a ***point in the other object class***.

This is called Multiple Thresholding Classification/Segmentation.

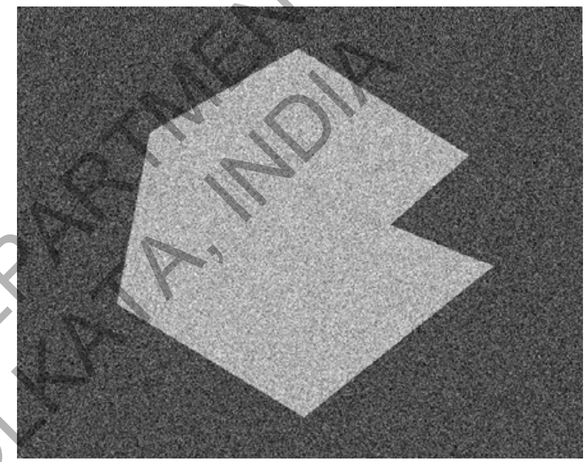
The Role of Noise in Thresholding



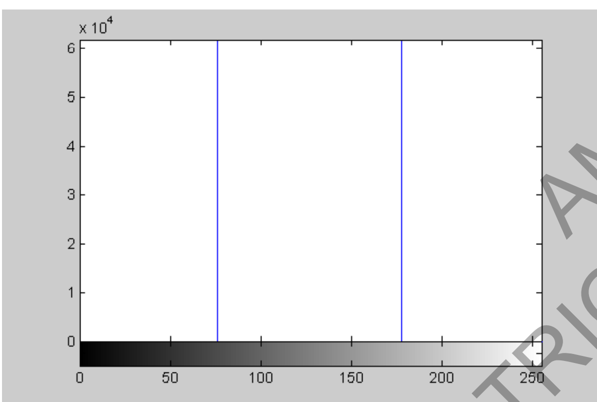
Noiseless 8-bit Septagon Image



Septagon Image with additive Gaussian Noise (Mean: 0, SD: 10 intensity levels)

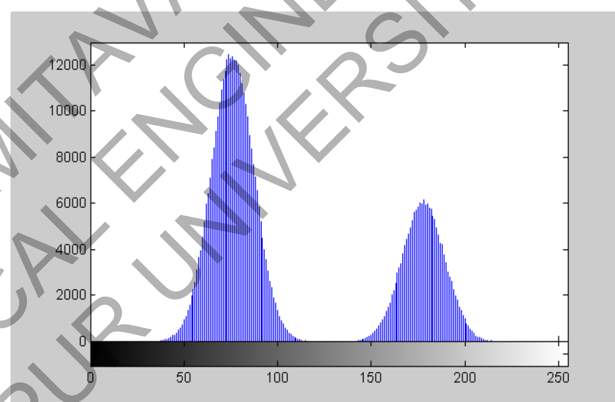


Septagon Image with additive Gaussian Noise (Mean: 0, SD: 50 intensity levels)



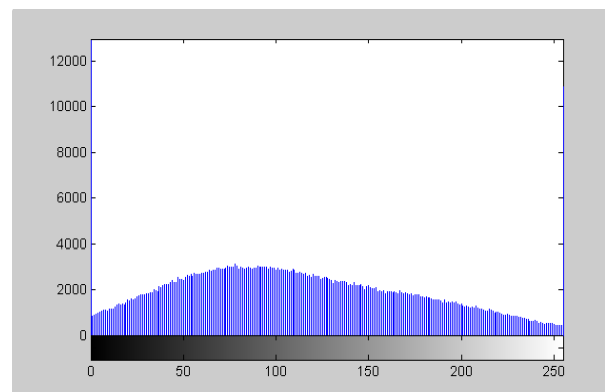
Intensity Histogram

Very easy to segment this image.



Intensity Histogram

Still, it is easy to segment this image.



Intensity Histogram

Now the job gets difficult, because no way we can separate the two modes.

Thresholding

Global Thresholding

An Iterative Algorithm to Perform Basic Global Thresholding…

Step 1: **Select an initial estimate for the global threshold, T.**

Step 2: **Segment the image into two levels using T, producing two groups of pixels: G_1 (*intensity values* $> T$) and G_2 (*intensity values* $\leq T$).**

Step 3: **Compute mean intensity values m_1 and m_2 for the pixels in G_1 and G_2 respectively.**

Step 4: **Compute a new threshold value:** $T = \frac{1}{2}\left(m_1 + m_2\right)$ **.**

Step 5: **Repeat Steps 2 to 4 until the difference between values of T in successive iterations is smaller than a specified ΔT.**

Thresholding

Global Thresholding

The Iterative Algorithm for Basic Global Thresholding…

Conclusions…

This method works well for bi-level thresholding, where the valley between the two histogram modes (i.e. usually between the background and foreground or object under consideration) is reasonably distinct.

What is the Role of Parameter ΔT??

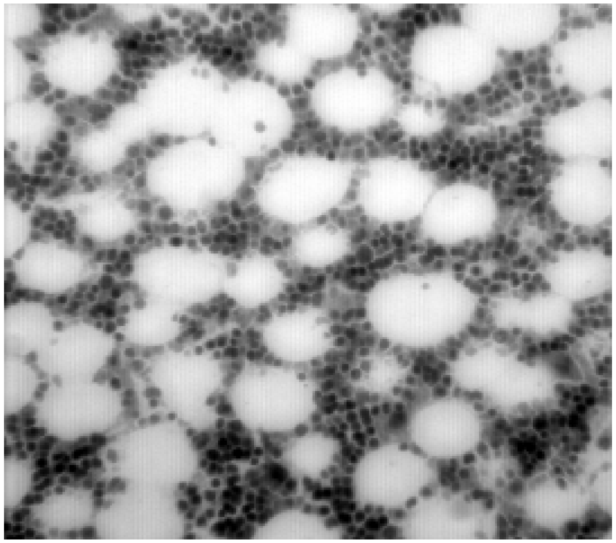
In a situation where the speed of execution is important, this parameter gets influential in controlling the number of iterations.

Larger ΔT → **The algorithm will perform fewer iterations.**

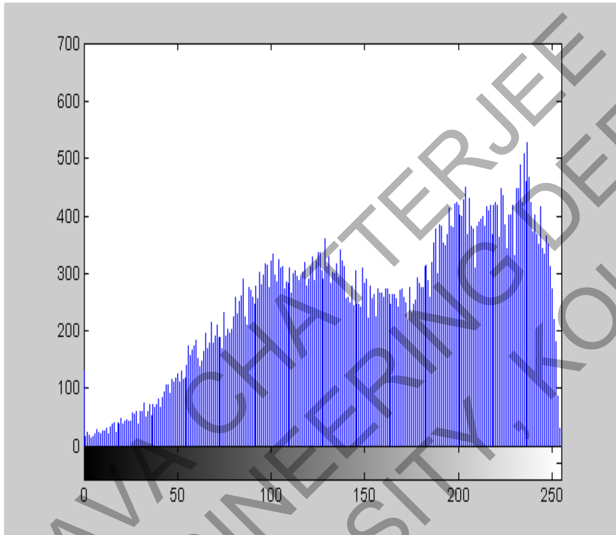
Smaller ΔT → **The algorithm will perform more iterations.**

Global Thresholding

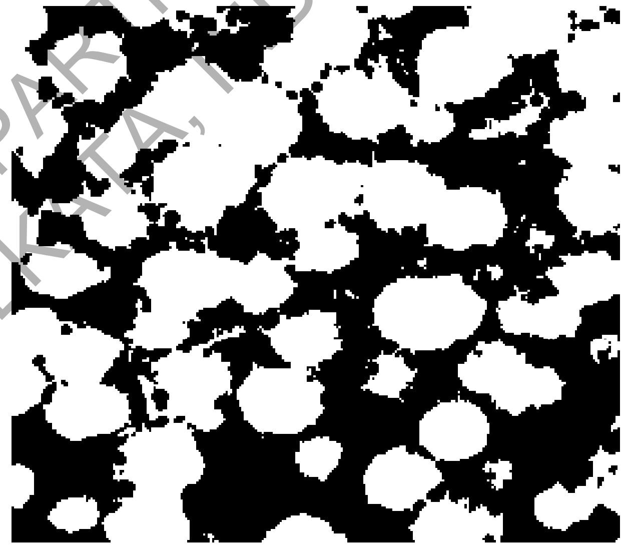
The Iterative Algorithm : An Example…



An original Bone-marrow Image



The Image Intensity histogram



The Segmented Image

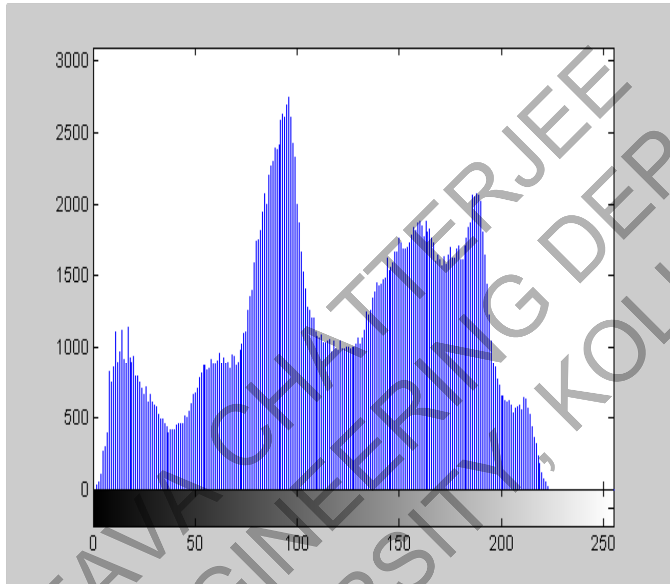
Algorithm Info…

We chose $\Delta T = 0.5$. The algorithm stopped after 5 iterations. The global threshold determined was $T = 152.4113$. We chose the nearest integer for bi-level segmentation of the image i.e. $T = 152$.

Global Thresholding

The Iterative Algorithm : Another Example…

The original Pepper Image



The Image Intensity histogram

The Segmented Image

Algorithm Info…

We chose $\Delta T = 0.1$. The algorithm stopped after 5 iterations. The global threshold determined was $T = 119.3685$. We chose the nearest integer for bi-level segmentation of the image i.e. $T = 119$. With a choice of $\Delta T = 0.01$, the performance of the algorithm did not change.

Image Segmentation

References:

- R. C. Gonzalez and R. E. Woods. Digital Image Processing. Pearson Education Inc., 2008.
- R. C. Gonzalez, R. E. Woods, and S. L. Eddins. Digital Image Processing using MATLAB®. Pearson Education, Inc. 2005.
- S. Annadurai and R. Shanmugalakshmi. Fundamentals of Digital Image Processing. Pearson Education, Inc. 2007.

Thank You…