# Disclaimer

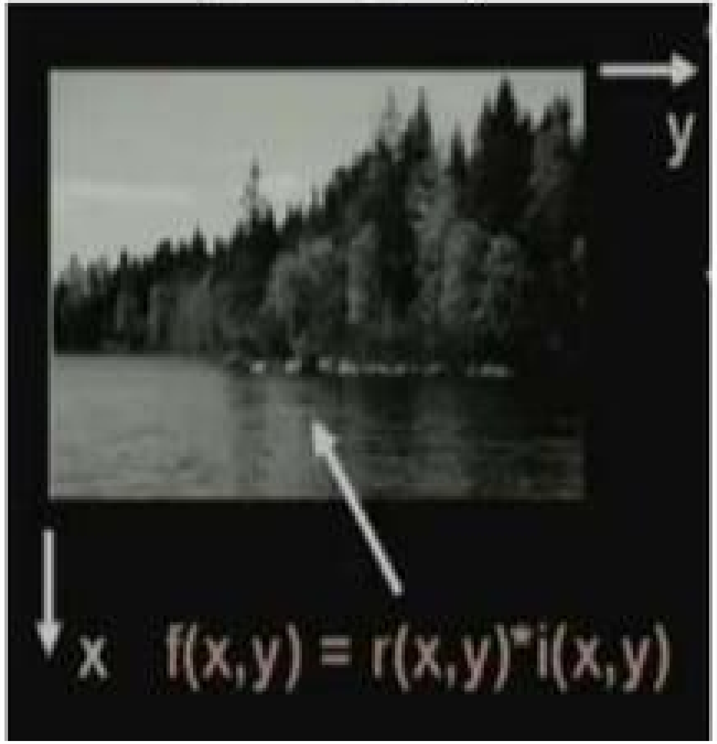
- All images/contents used in this presentation are copyright of original owners.
- The PPT has prepared for academic use only.

# MOTIVATION

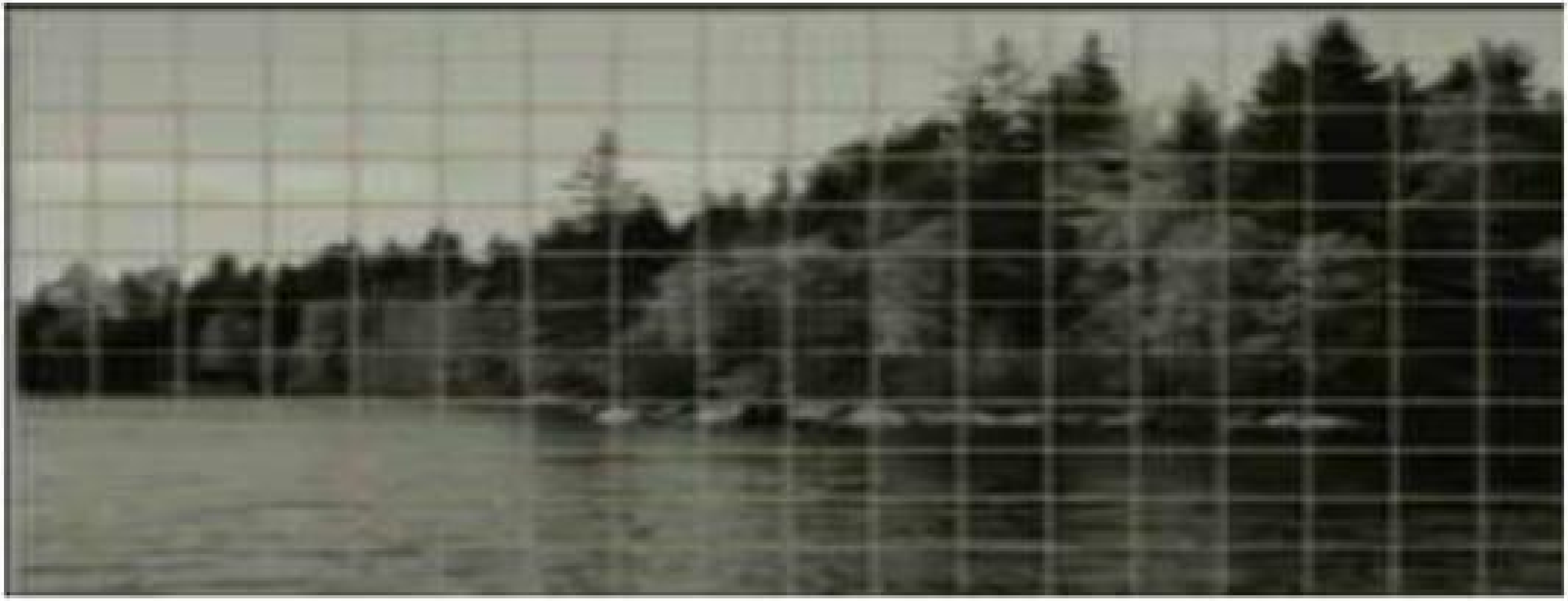
- Segmentation is difficult
- Decades of extensive research, no general "off-the-shelf" solution
- Non- uniform illumination
- No control of the environment
- Inadequate model of the object of interest
- Noise
- Segmentation on trivial images is one of the difficult task in image processing . Still under research
- It has rich mathematical formulations that makes it a worthwhile research topic

# IMAGES:

- Image is replica of object.
- An image defined in the "real world" is considered as a two dimensional function f(x, y) ,where x and y are spatial coordinates and the amplitude of f at any pair of coordinates (x,y) is called the intensity or gray level of the image at that point.
- **TYPES OF IMAGES:**
  - -Gray-tone image
  - -Binary image



# Image representation



- Spatial discretization of grids: to obtain sample values at every point.
- Intensity discretization by a process called quantization:
  representing an image in form of a matrix.

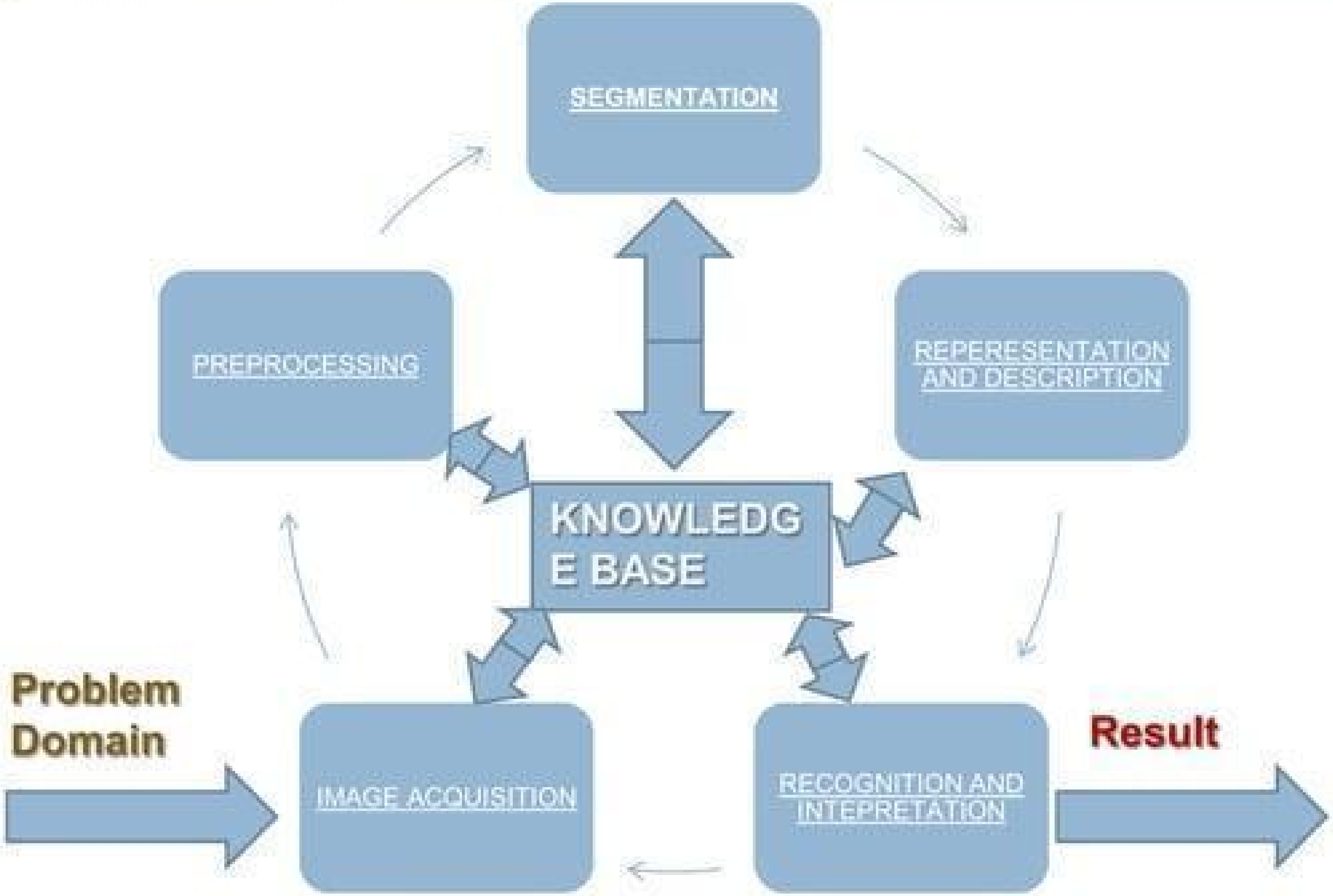
# Image representation

$$I = \begin{bmatrix} f(0,0) & f(0,1) \ldots\ldots\ldots\ldots & f(0,N-1) \\ f(1,0) & f(1,1) \ldots\ldots\ldots\ldots & f(1,N-1) \\ f(2,0) & f(2,1) \ldots\ldots\ldots\ldots & f(2,N-1) \\ \vdots & \vdots & \vdots \\ f(M-1,0) & f(M,1) \ldots & f(M-1,N-1) \end{bmatrix}$$

A matrix of finite dimension , it has m number of rows add n number of columns .
Each of the elements in this matrix representation is called a pixel

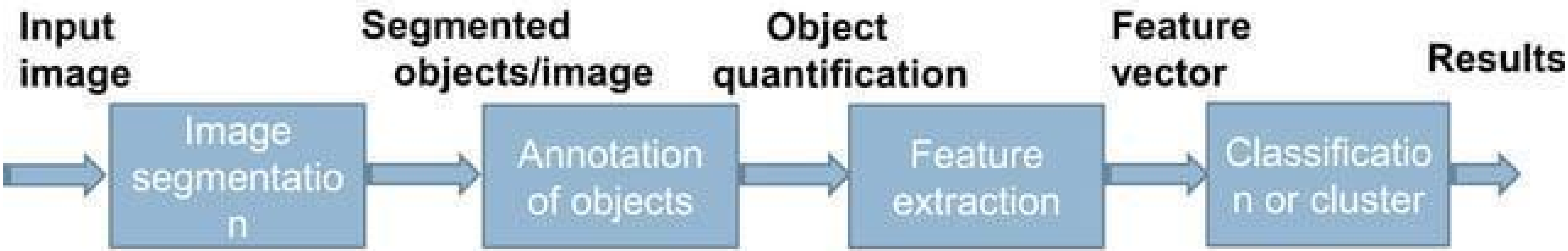
**Image Size :-256 * 256 elements , 512 * 512, 640 * 480 , 1024 * 1024**
**Quantization :- 8 bits**

# Steps in digital image Processing



# What is Image Segmentation ?

- Image segmentation is an aspect of image processing.
- *Image segmentation is a computer vision process.*
- *Image segmentation is the first step in image analysis.*




A typical image analysis pipeline.

# Image Segmentation Defined

There are many definitions:

- In computer vision, Image Segmentation is the process of subdividing a digital image into multiple segments(sets of pixels, also known as superpixels)-Wikipedia,2002
- Segmentation is a process of grouping together pixels that have similar attributes-Efford,2000
- Image Segmentation is the process of partitioning an image into non-intersecting regions such that each region is homogeneous and the union of no two adjacent regions is homogeneous-Pal,1994
- Pixels in a region are similar according to some homogeneity criteria such as colour, intensity or texture so as to locate and identify objects and boundaries (lines,curves,etc) in an image.
- The goal of image segmentation is to simplify/change the representation of an image into something that is more

# Simple example:

segmentation of rice grains

Original image

Segmented (binary) image
Each pixel is assigned a label:
- 0 = not rice grain pixel
- 1 = rice grain pixel

# Why segmentation is useful ?

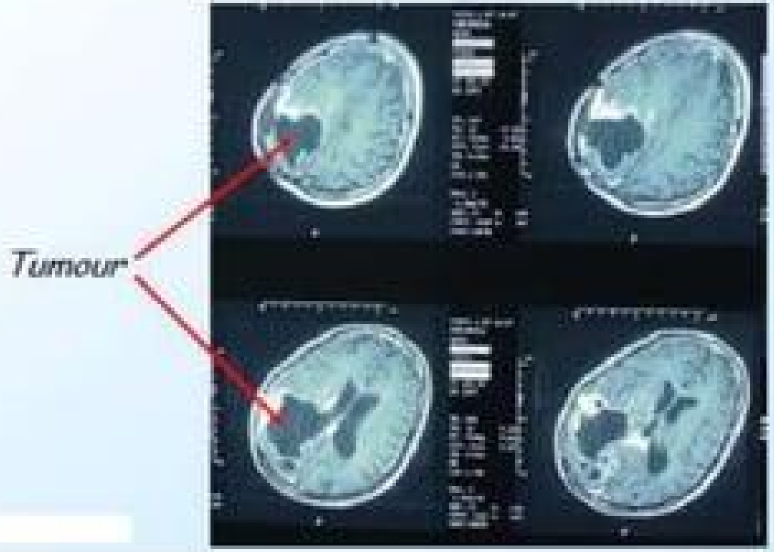
- Segmentation accuracy determines the eventual success or failure of computerized analysis procedure.
- Improvement of pictorial information for human interpretation/perception
- **Mapping and Measurement** - Automatic analysis of remote sensing data from satellites to identify and measure regions of interest. e.g. Petroleum reserves
- It might be possible to analyze the image in the computer and provide cues to the radiologists to help detect important/suspicious structures (e.g.: Computed Aided Diagnosis, CAD)

# Application with examples

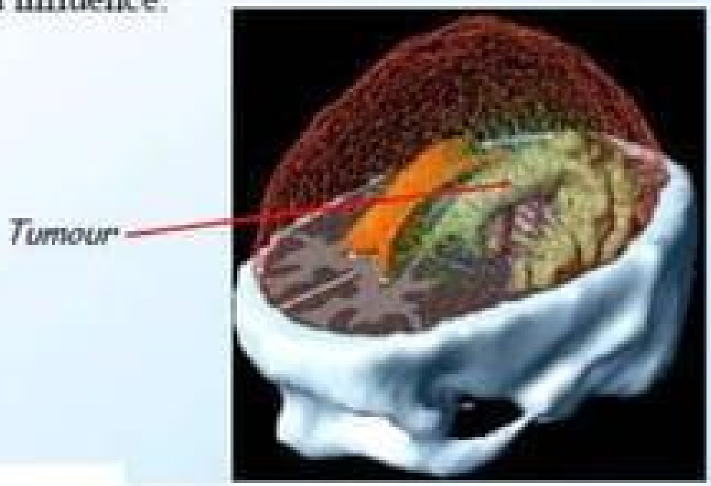
# Medical imaging

- Diagnostic analysis
  - Patient come with headache, visual troubles, and speech difficulties.
  - Diagnosis?

- Diagnostic analysis
  - CT scan of the brain shows a tumour:

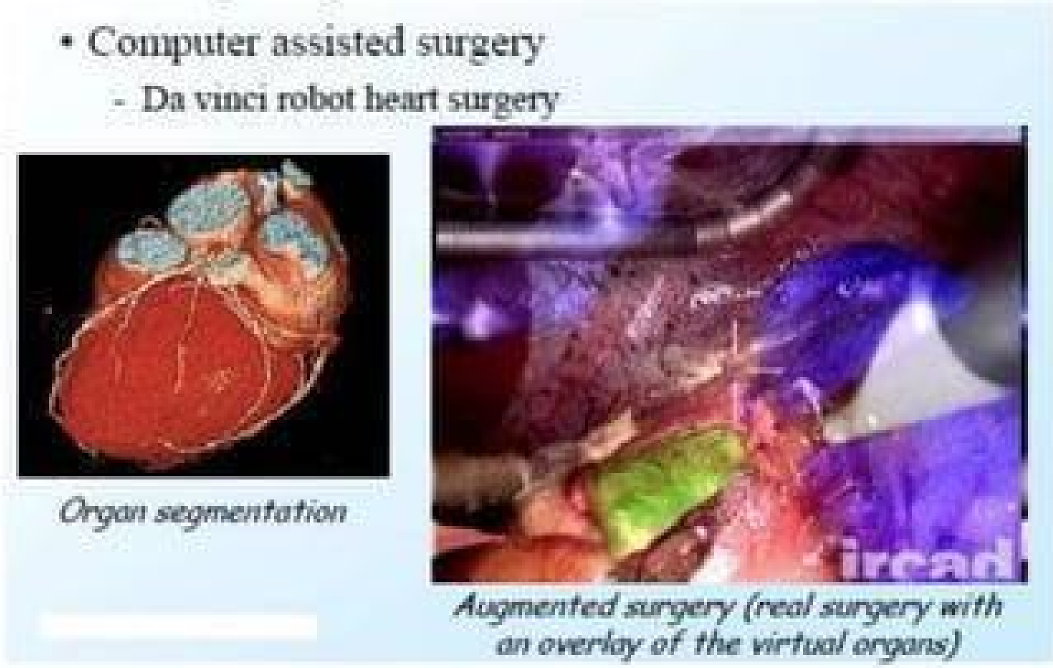



- Diagnostic analysis
  - Segmentation and 3D rendering reveals the tumour size and influence:



# Computer-guided surgery

- Da Vinci robot heart surgery


**VIDEO DEMO**

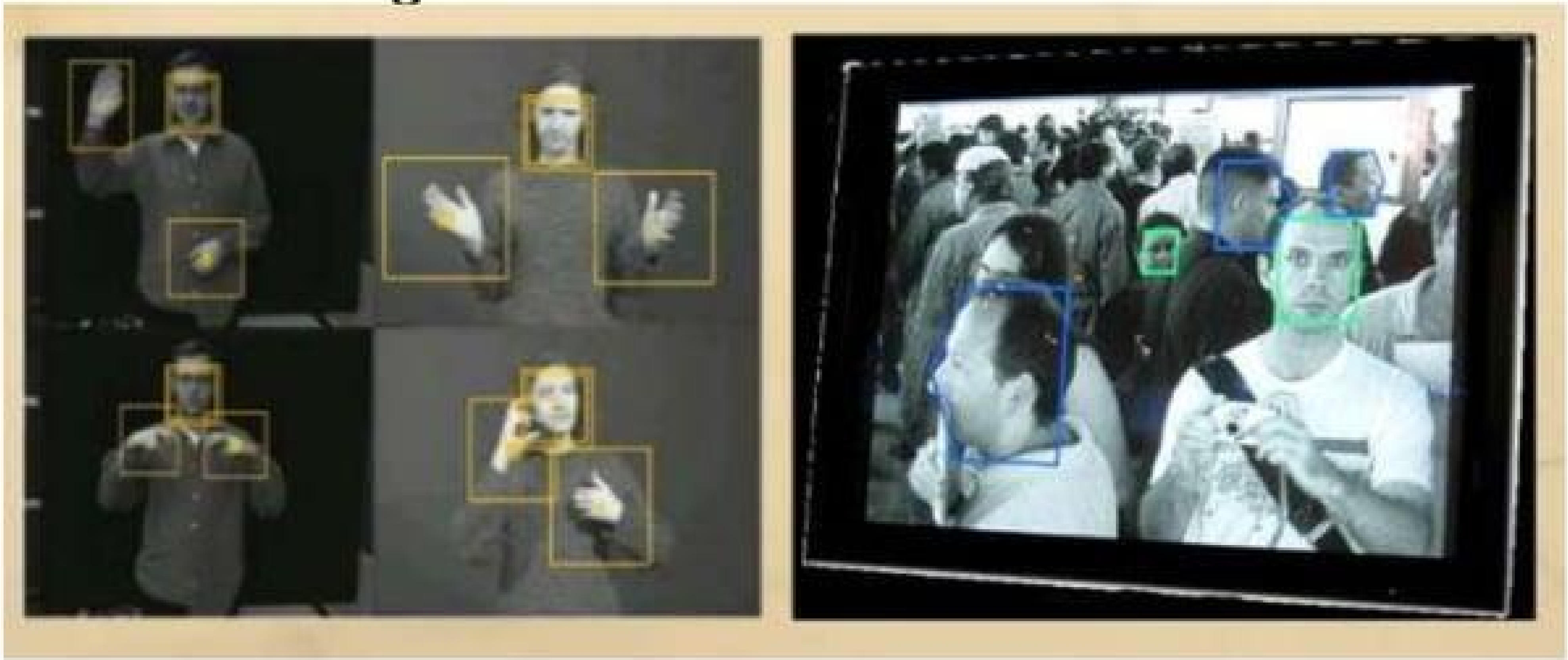
# Object detection

- Pedestrian detection
- Face detection
- Brake light detection
- Locate objects in satellite imagery(roads,buildings,forests,etc)
- Agricultural imaging-crop disease detection

# EKSU on Google Maps

# Recognition tasks

- Face recognition-g+
- Fingerprint recognition
- Iris recognition



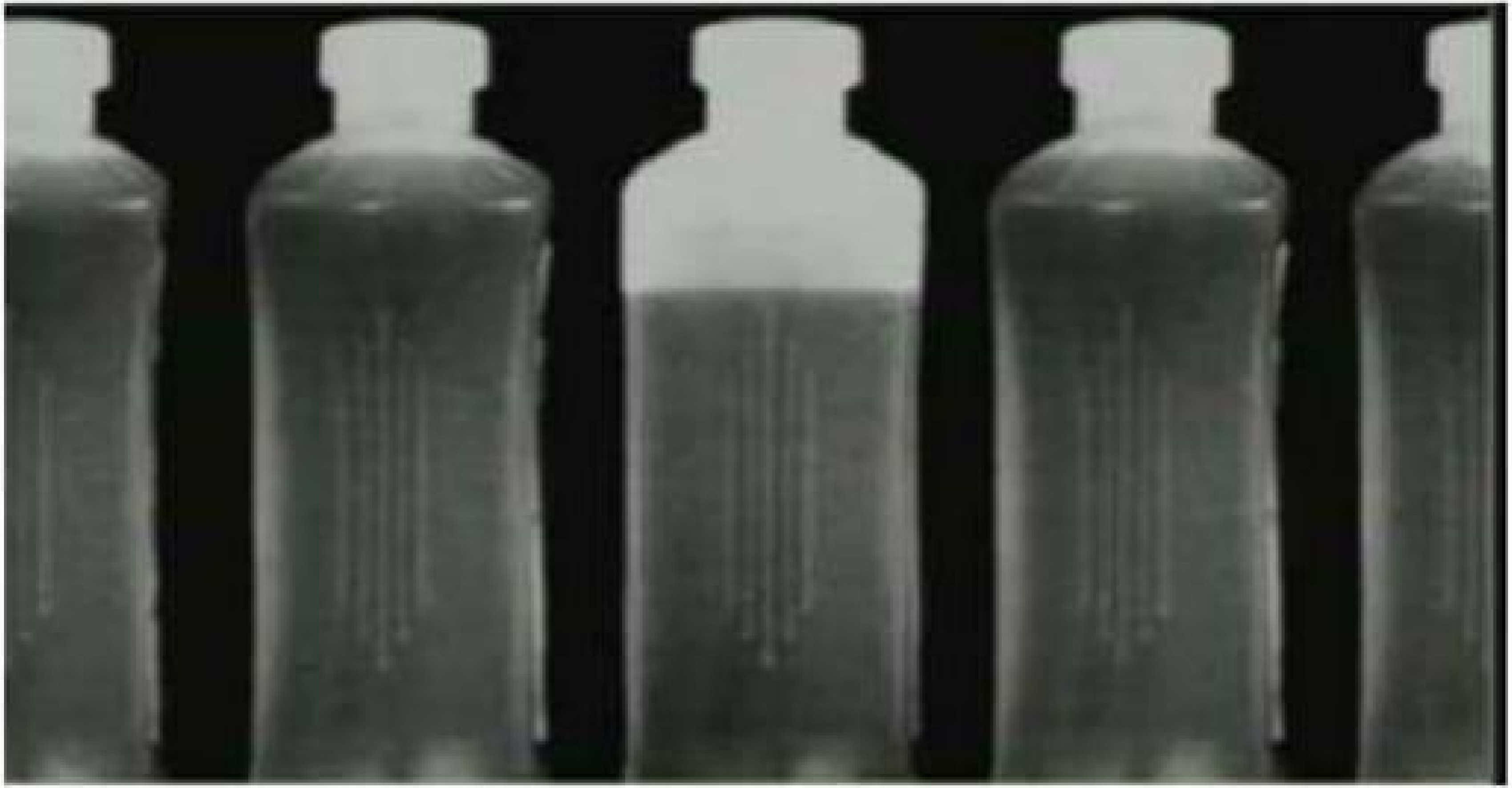
# Machine Vision Application

Here the interest is on procedures for extraction of image information suitable for computer processing

Typical Applications:-

- Industrial Machine Vision for product assembly and inspection.
- Automated Target detection and tracking.
- Machine processing for aerial and satellite imagery for weather prediction and assessment etc.

# Automated inspection



BOTTLING PLANT AUTOMATION

# Other areas of application

- Traffic control systems
- Content-based image retrieval
- Video surveillance
- In sport scenes

# Detection of Discontinuities

- detect the three basic types of gray-level discontinuities
  - points , lines , edges
- the common way is to run a mask through the image

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

**Masking:** A logical operation carried out on an image in order to m
or identify a part of it.

# Point Detection

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

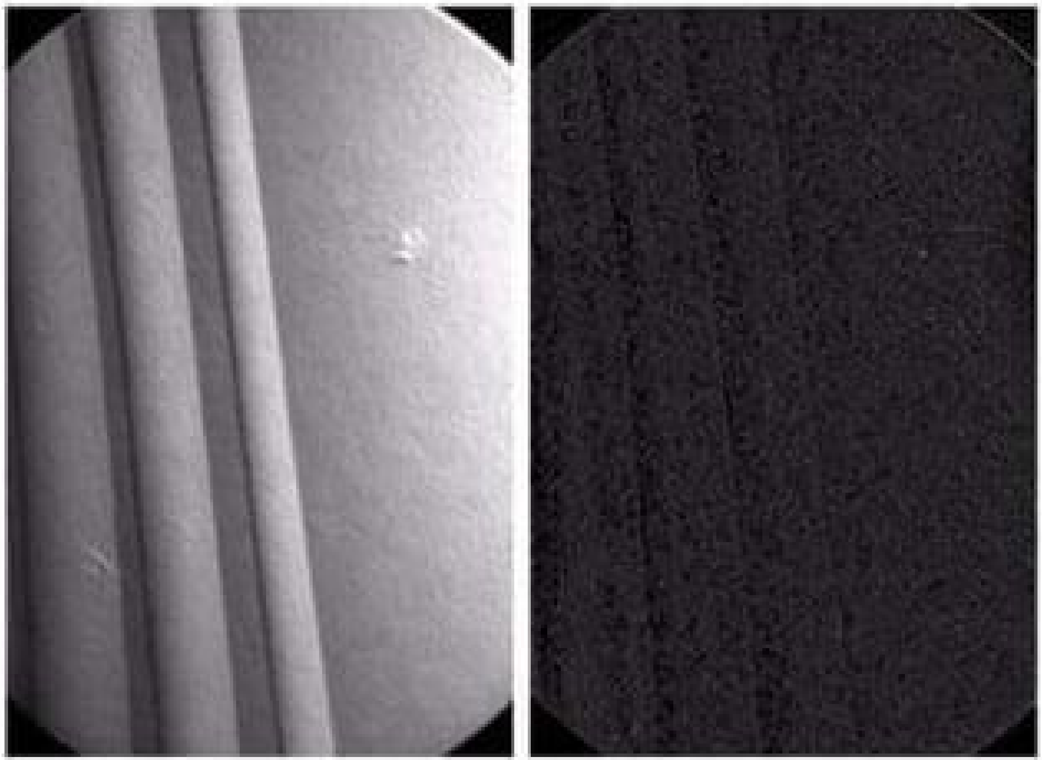
- a point has been detected at the location on which the mark is centered if

$$|R| \geq T$$

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

# Example

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |



a
b c

(a) Point detection mask.
(b) X-ray image of a turbine blade with a porosity.
(c) Result of point detection.
(Original image courtesy of X-TEK Systems Ltd.)

# Line Detection

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 2 | 2 | 2 |
| −1 | −1 | −1 |

Horizontal

| | | |
|---|---|---|
| −1 | −1 | 2 |
| −1 | 2 | −1 |
| 2 | −1 | −1 |

$+45^\circ$

| | | |
|---|---|---|
| −1 | 2 | −1 |
| −1 | 2 | −1 |
| −1 | 2 | −1 |

Vertical

| | | |
|---|---|---|
| 2 | −1 | −1 |
| −1 | 2 | −1 |
| −1 | −1 | 2 |

$-45^\circ$

- Horizontal mask will result with max response when a line passed through the middle row of the mask with a constant background.
- the similar idea is used with other masks.
- note: the preferred direction of each mask is weighted with a larger coefficient (i.e.,2) than other possible directions.

Write different line detection masks in an image.

Z =

| | | |
|---|---|---|
| 10 | 10 | 100 |
| 10 | 100 | 10 |
| 100 | 10 | 10 |

**Different line detection masks in an image**

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 2 | 2 | 2 |
| -1 | -1 | -1 |

**Horizontal**

| | | |
|---|---|---|
| -1 | -1 | 2 |
| -1 | 2 | -1 |
| 2 | -1 | -1 |

**$+45^0$**

| | | |
|---|---|---|
| -1 | 2 | -1 |
| -1 | 2 | -1 |
| -1 | 2 | -1 |

**Vertical**

| | | |
|---|---|---|
| 2 | -1 | -1 |
| -1 | 2 | -1 |
| -1 | -1 | 2 |

**$-45^0$**

Use pixel replication at border to transform 3×3 image to 5×5.

z =

| | | |
|---|---|---|
| 10 | 10 | 100 |
| 10 | 100 | 10 |
| 100 | 10 | 10 |

| | | | | |
|---|---|---|---|---|
| 10 | 10 | 10 | 100 | 100 |
| 10 | 10 | 10 | 100 | 100 |
| 10 | 10 | 100 | 10 | 10 |
| 100 | 100 | 10 | 10 | 10 |
| 100 | 100 | 10 | 10 | 10 |

| | | | | |
|---|---|---|---|---|
| 10 | 10 | 10 | 100 | 100 |
| 10 | -90 | 0 | 90 | 100 |
| 10 | 0 | 0 | 0 | 10 |
| 100 | 90 | 0 | -90 | 10 |
| 100 | 100 | 10 | 10 | 10 |

**Horizontal**

| | | | | |
|---|---|---|---|---|
| 10 | 10 | 10 | 100 | 100 |
| 10 | -90 | 0 | 360 | 100 |
| 10 | 0 | 540 | 0 | 10 |
| 100 | 360 | 0 | -90 | 10 |
| 100 | 100 | 10 | 10 | 10 |

**$+45^0$**

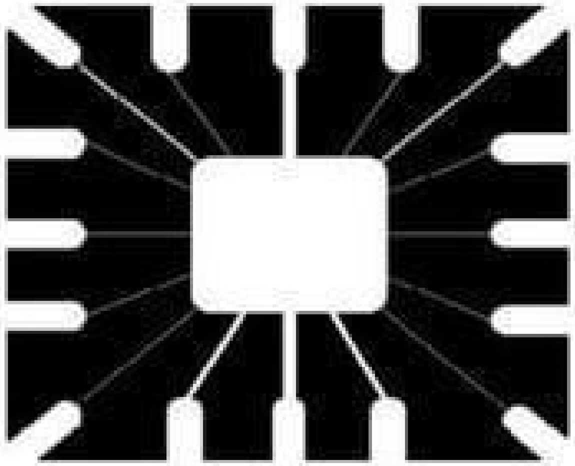
| | | | | |
|---|---|---|---|---|
| 10 | 10 | 10 | 100 | 100 |
| 10 | -90 | 0 | 90 | 100 |
| 10 | 0 | 0 | 0 | 10 |
| 100 | 90 | 0 | -90 | 10 |
| 100 | 100 | 10 | 10 | 10 |

**Vertical**

| | | | | |
|---|---|---|---|---|
| 10 | 10 | 10 | 100 | 100 |
| 10 | 180 | -270 | -180 | 100 |
| 10 | -270 | 0 | -270 | 10 |
| 100 | -180 | -270 | 180 | 10 |
| 100 | 100 | 10 | 10 | 10 |

**$-45^0$**

# Example



(a)

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

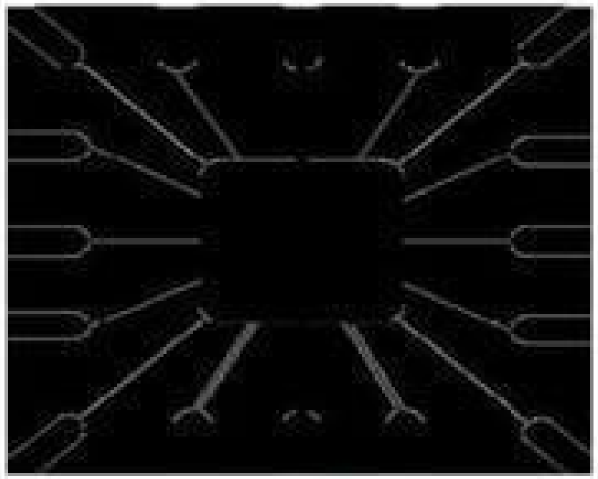
(b)

$$\begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$

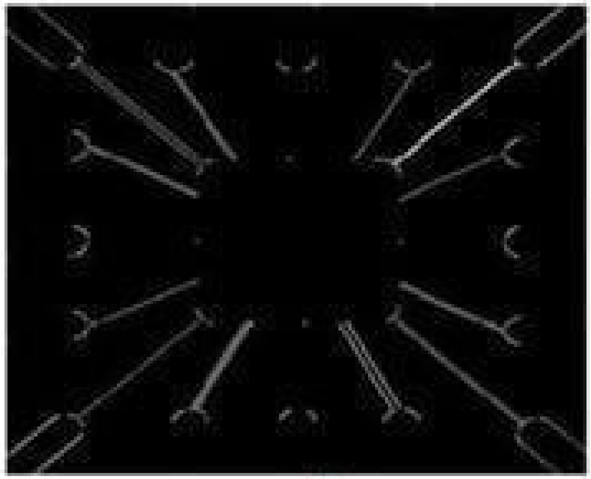
(c)

$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

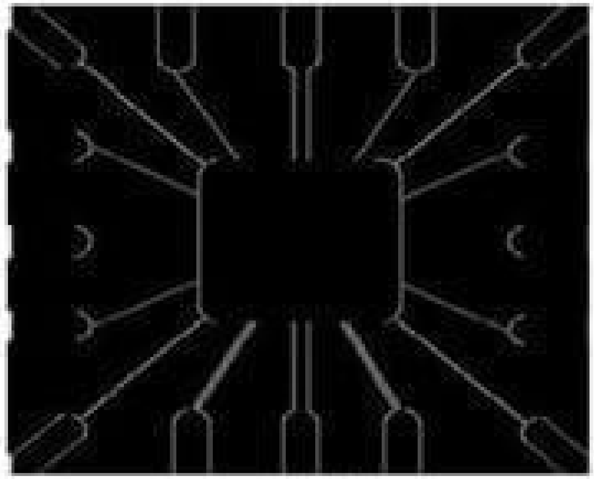
(d)



(e)



(f)



(g)

Line detection. (a) Image of a wire-bond mask; (b) Horizontal line detector mask; (c) +45° line detector mask; (d) Vertical line detector mask; (e) Result of processing with the horizontal line detector mask; (f) Result of processing with the +45° line detector mask; (g) Result of processing with the vertical line detector mask.

# Edge Detection

- Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally has discontinuities.
- Edge detection is used to obtain information from the frames as a precursor step to feature extraction and object recognition.
- This process detects outlines of an object and boundaries between objects and the background in the image
- Approaches for implementing
  - first-order derivative (Gradient operator)
  - second-order derivative (Laplacian operator)

- Edge is the boundary between two homogeneous regions.

# Edge detector

- **Advantages** : easy to implement

  simple to understand

- **Disadvantages**: It is not suitable for very noisy images

  It is not suitable for edgeless images

  It is not suitable for images whose

  boundaries are very smooth

# Edge filter operators/edge detection techniques

- There are seven techniques namely:

  -Sobel operator: most useful and widely available edge filters/gradient masks.

  -Roberts cross edge operator

  -Laplacian operator

  -Prewitt operator

  -Kiresh operator

  -Canny edge detector : most preferred

  -Edge maximization technique(EMT)

# Gradient Operator

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- first derivatives are implemented using the magnitude of the gradient.

$$\nabla f = mag(\nabla \mathrm{f}) = [G_x^2 + G_y^2]^{1/2}$$

$$= \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

commonly approx.

$$\nabla f \approx |G_x| + |G_y|$$

the magnitude becomes nonlinear

# Laplacian

Laplacian operator

(linear operator)

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

$$\begin{aligned}\nabla^2 f = [&f(x+1,y) + f(x-1,y) \\ &+ f(x,y+1) + f(x,y-1) - 4f(x,y)]\end{aligned}$$

Laplacian masks used to implement Eqs.

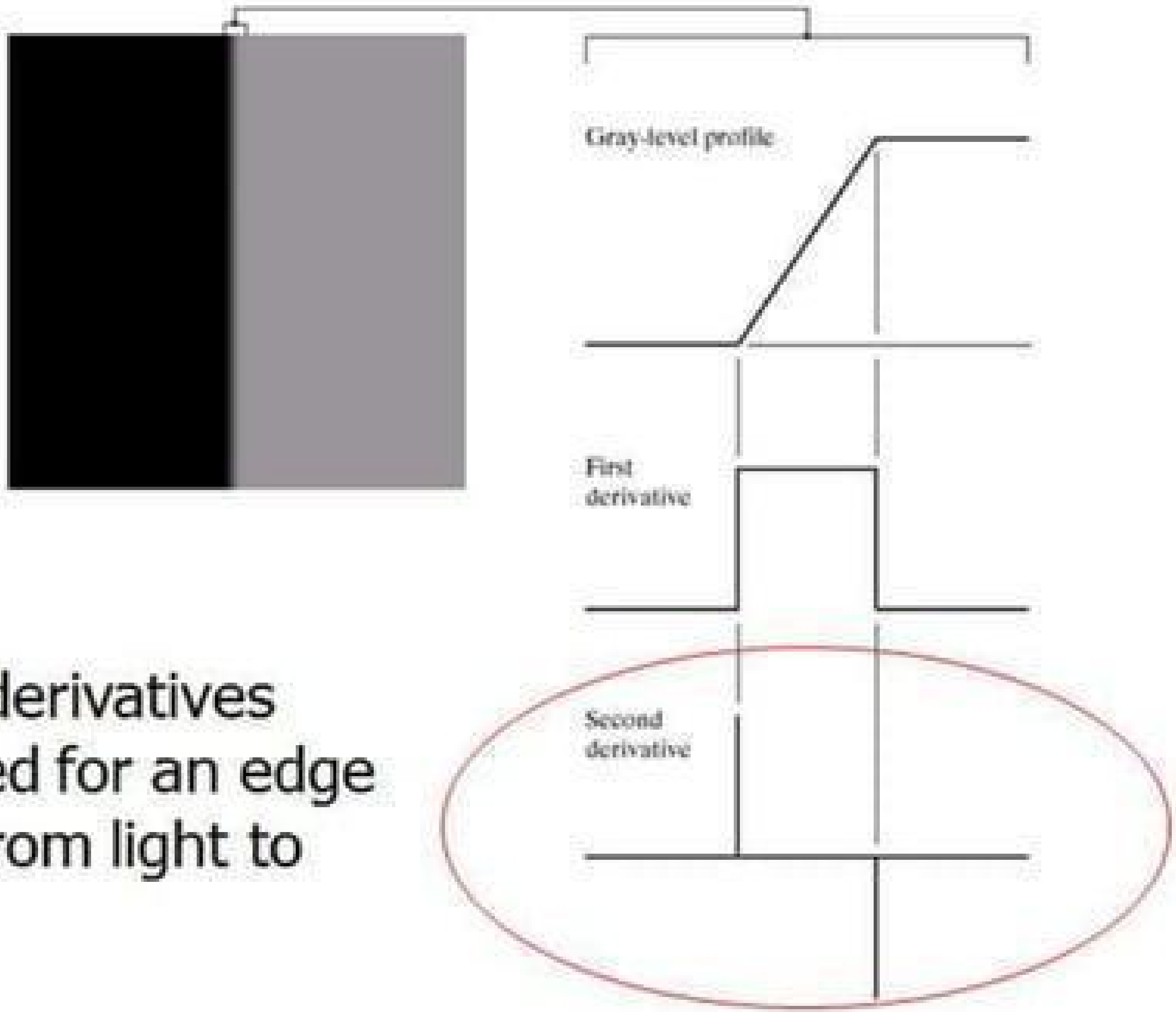
| | | |
|---|---|---|
| 0 | −1 | 0 |
| −1 | 4 | −1 |
| 0 | −1 | 0 |

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

# First and Second derivatives

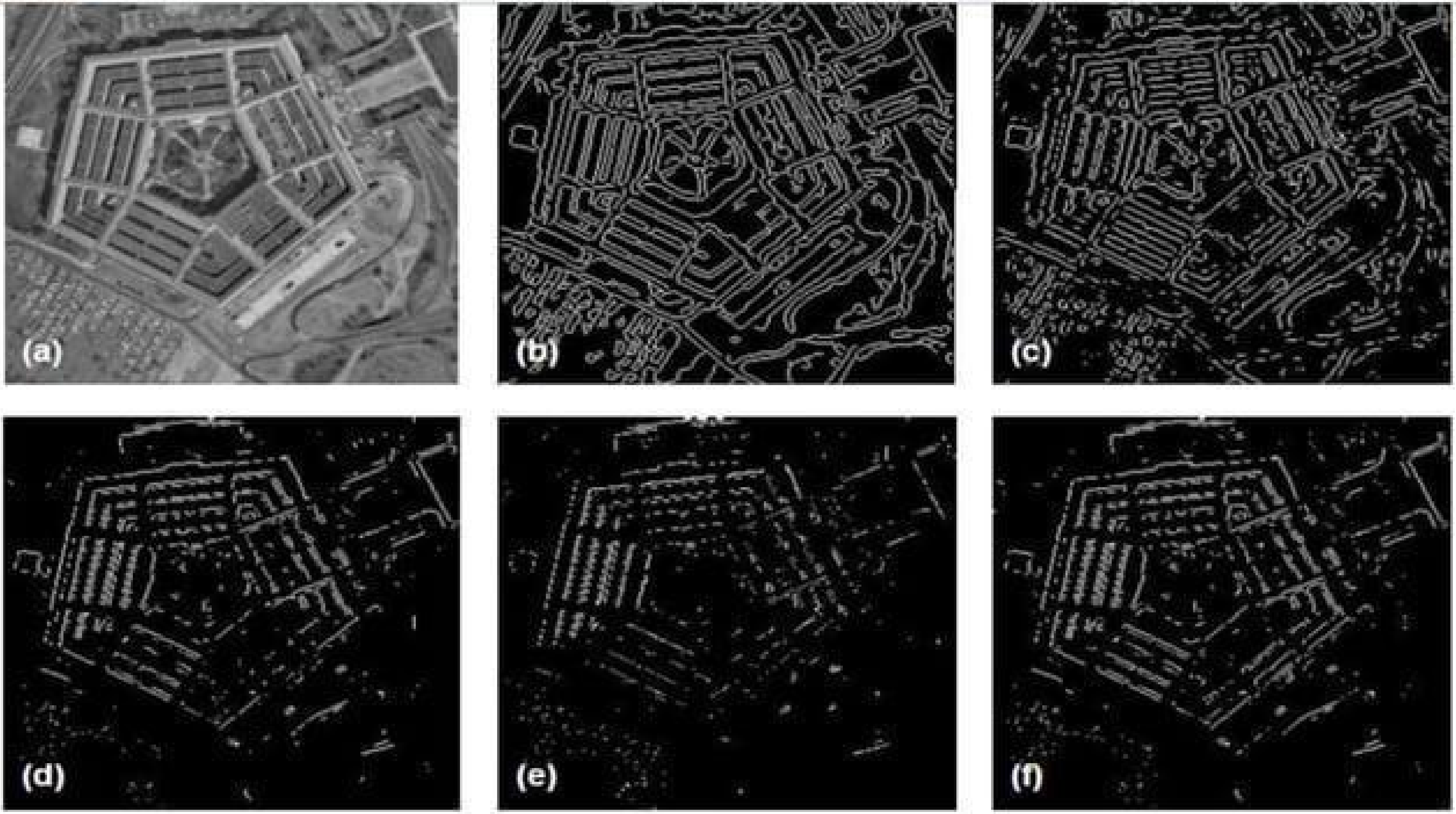
a b

(a) Two regions separated by a vertical edge.
(b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.

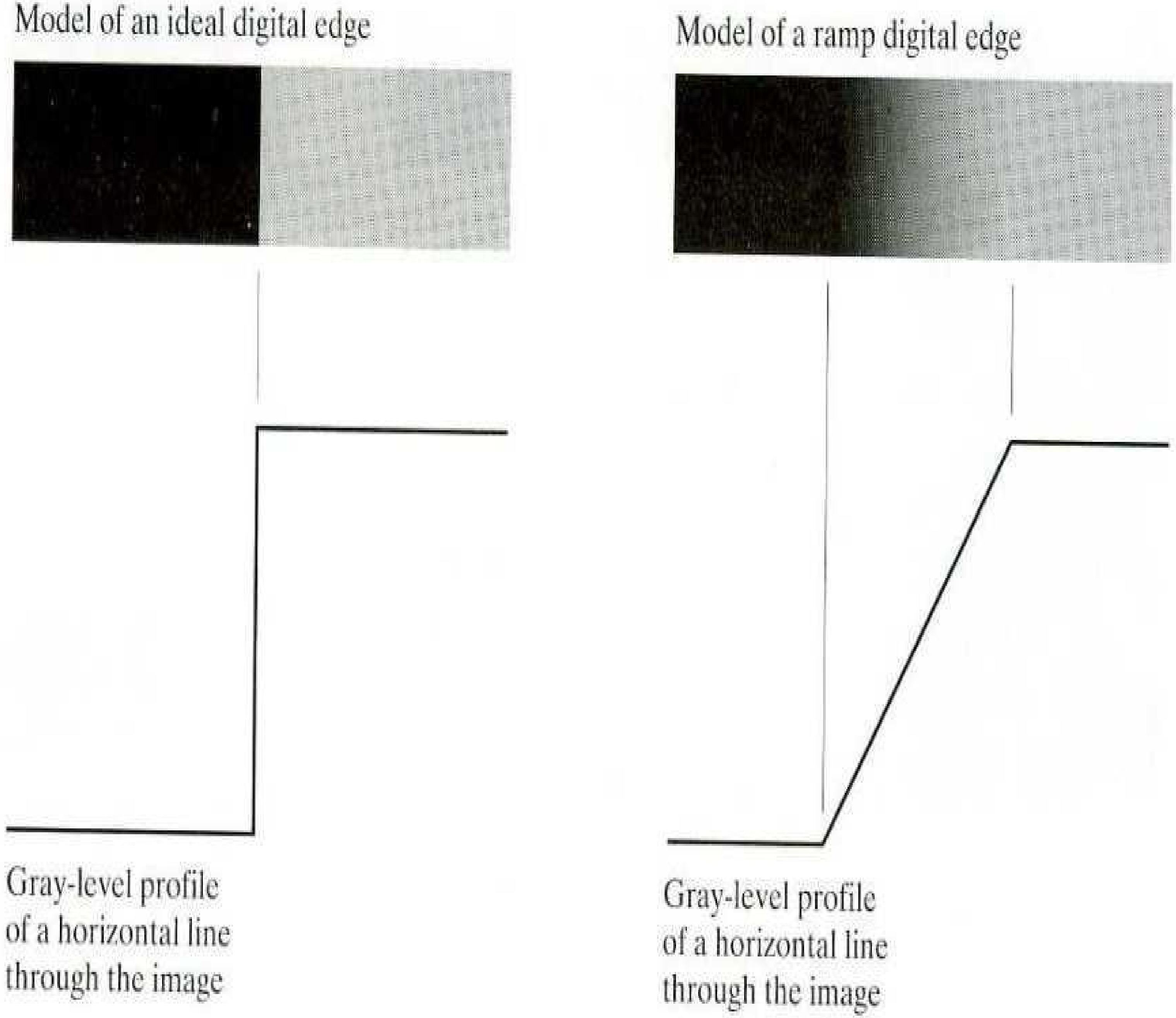



the signs of the derivatives would be reversed for an edge that transitions from light to dark

# Comparing edge filter operators




Examples of edge detectors. (a) A satellite image of Pentagon; (b) Canny filter; (c) Laplacian filter; (d) Prewitt filter; (e) Roberts filter; (f) Sobel filter.

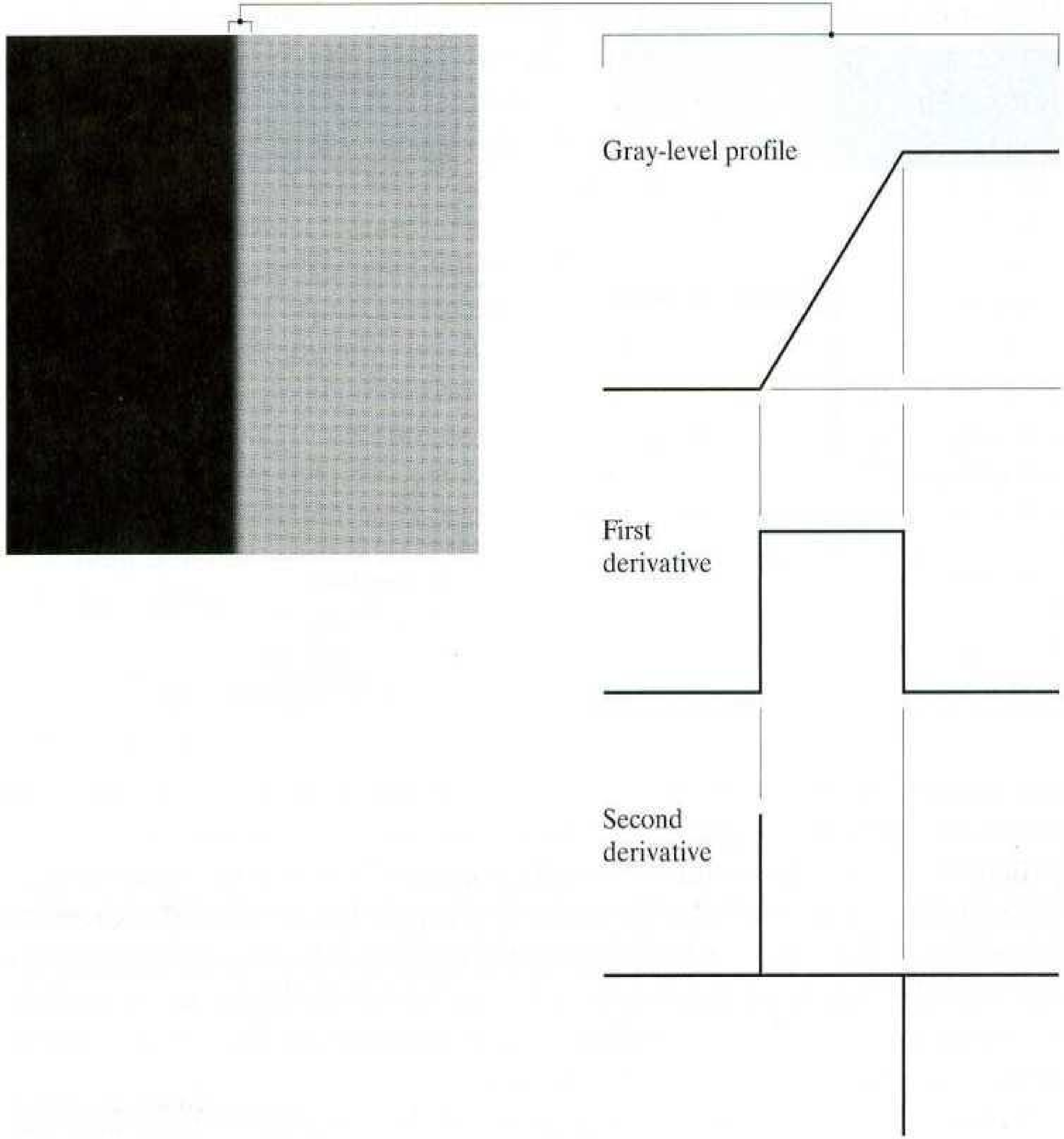


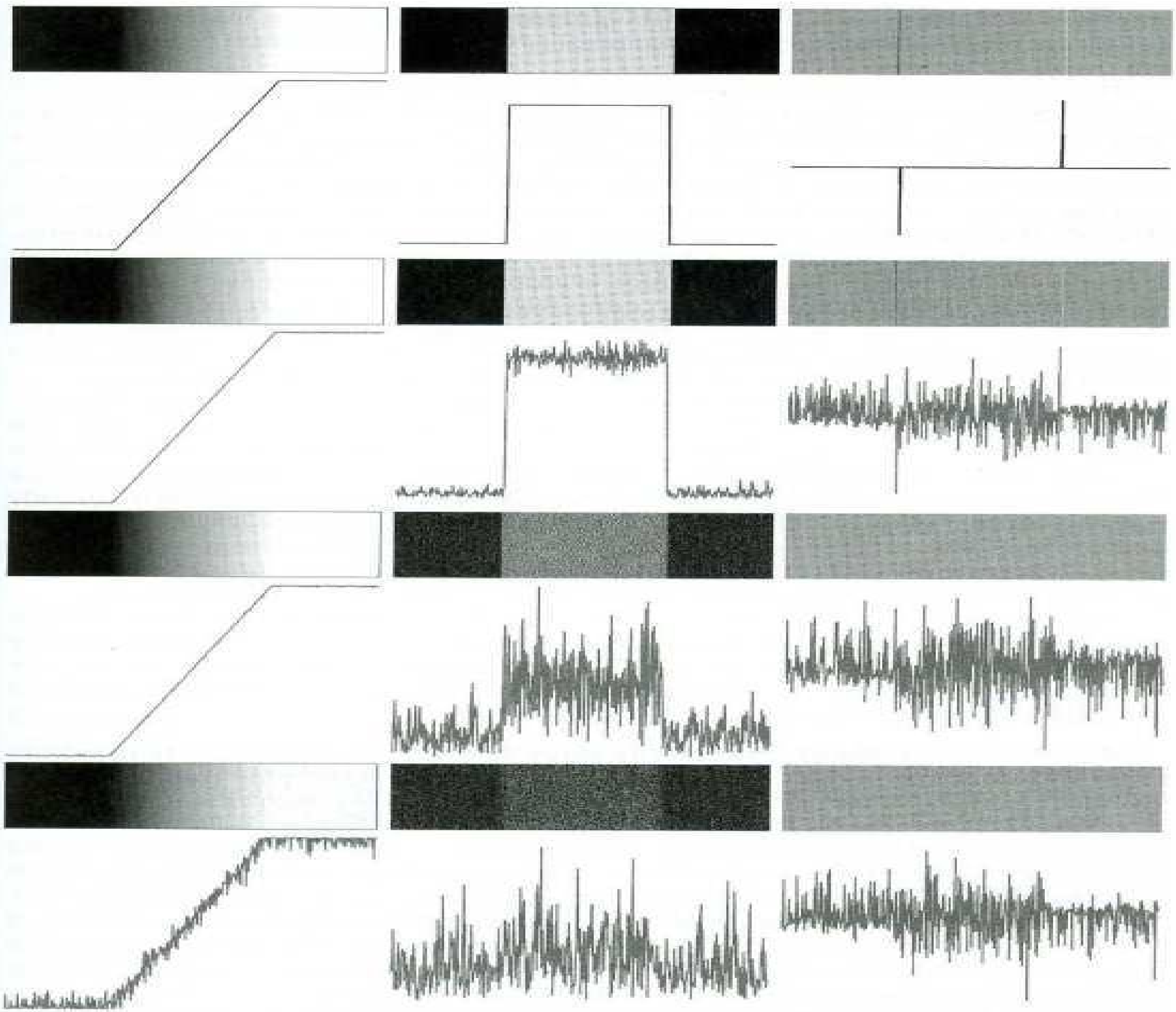

a b

**FIGURE 10.5** (a) Model of an ideal digital edge. (b) Model of a ramp edge. The slope of the ramp is proportional to the degree of blurring in the edge.

a b

**FIGURE 10.6**
(a) Two regions separated by a vertical edge.
(b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.





a
b c
d e
f g

**FIGURE 10.8** A $3 \times 3$ region of an image (the $z$'s are gray-level values) and various masks used to compute the gradient at point labeled $z_5$.

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

| | |
|---|---|
| −1 | 0 |
| 0 | 1 |

| | |
|---|---|
| 0 | −1 |
| 1 | 0 |

Roberts

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

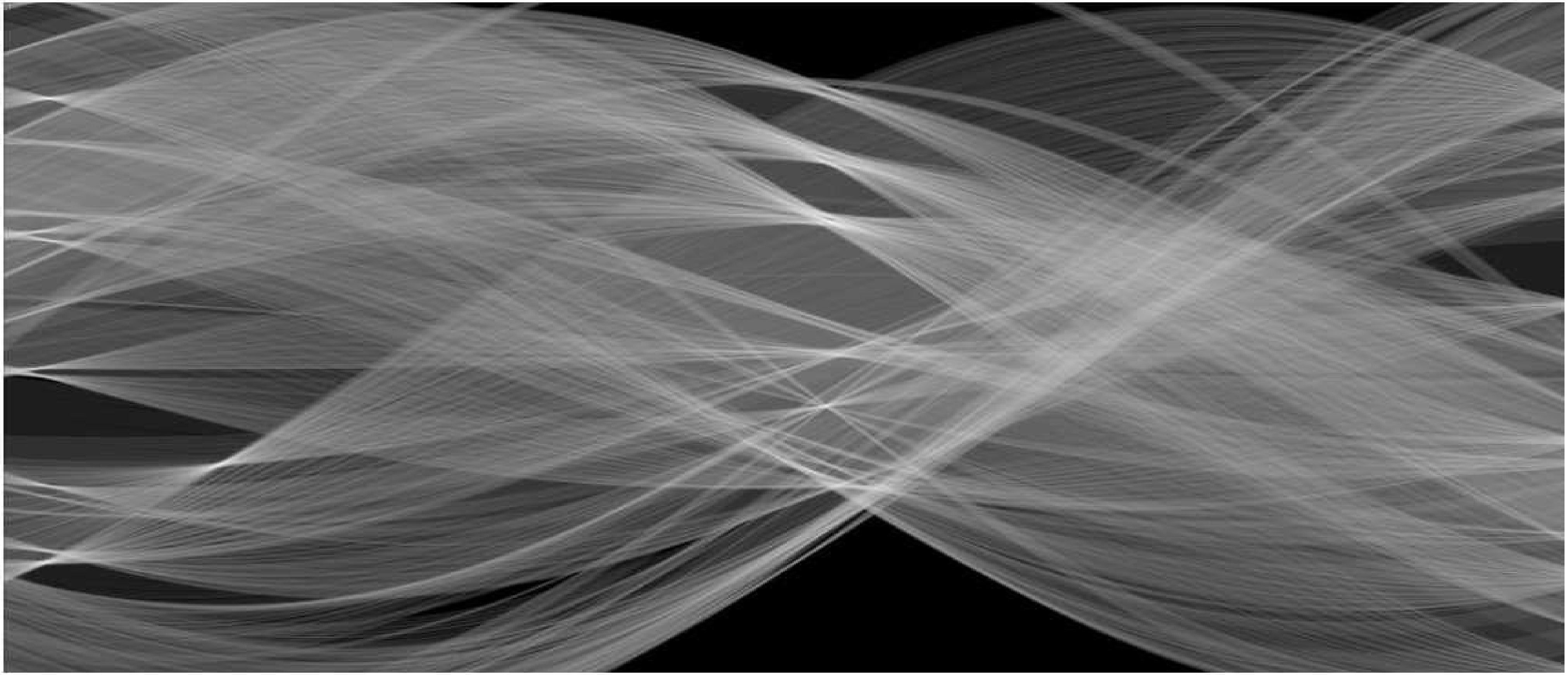
| | | |
|---|---|---|
| −1 | 0 | 1 |
| −1 | 0 | 1 |
| −1 | 0 | 1 |

Prewitt

| | | |
|---|---|---|
| −1 | −2 | −1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −2 | 0 | 2 |
| −1 | 0 | 1 |

Sobel



# Hough Transform

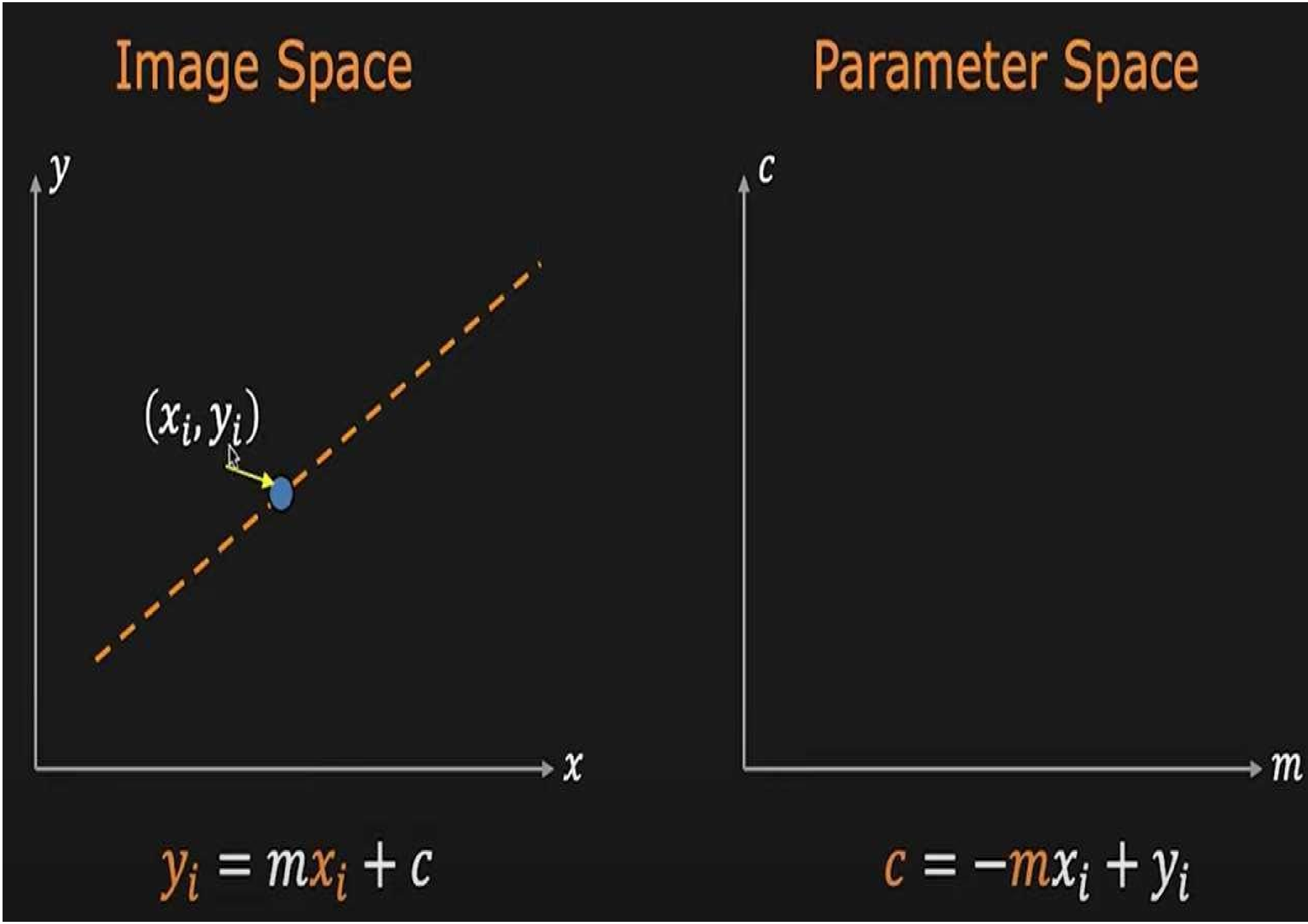

Problem: which edges in an image correspond to the boundary.

Hough Transform can solve this problem using an Algorithmic manner.

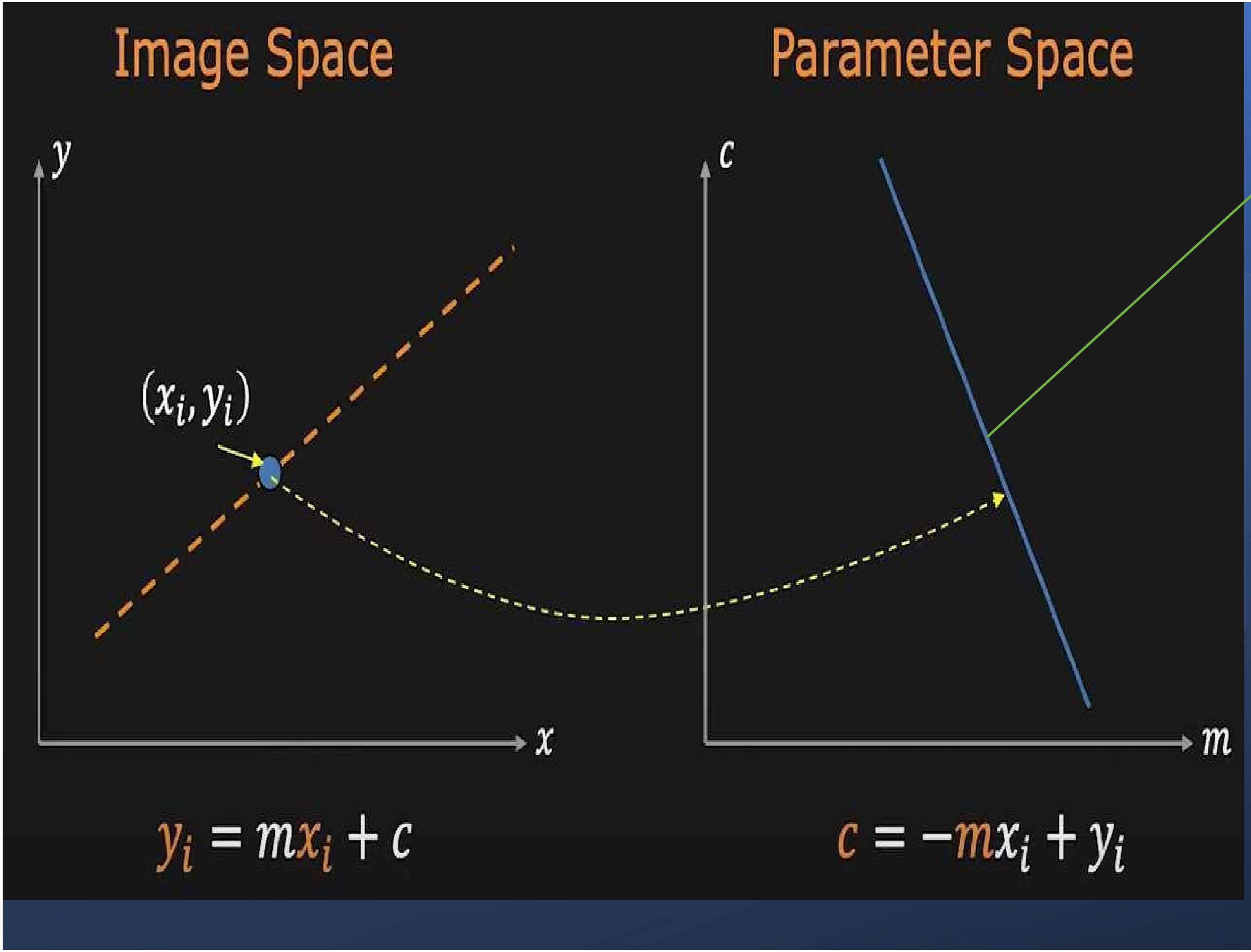
Given that, the boundary can be described using a small number of parameters.

In the Picture:

- ✓ Extraneous data: which point do we fit circles.
- ✓ Incomplete data
- ✓ Noise




The simplest of all shape: straight line.




Imagine all the lines passed through this point

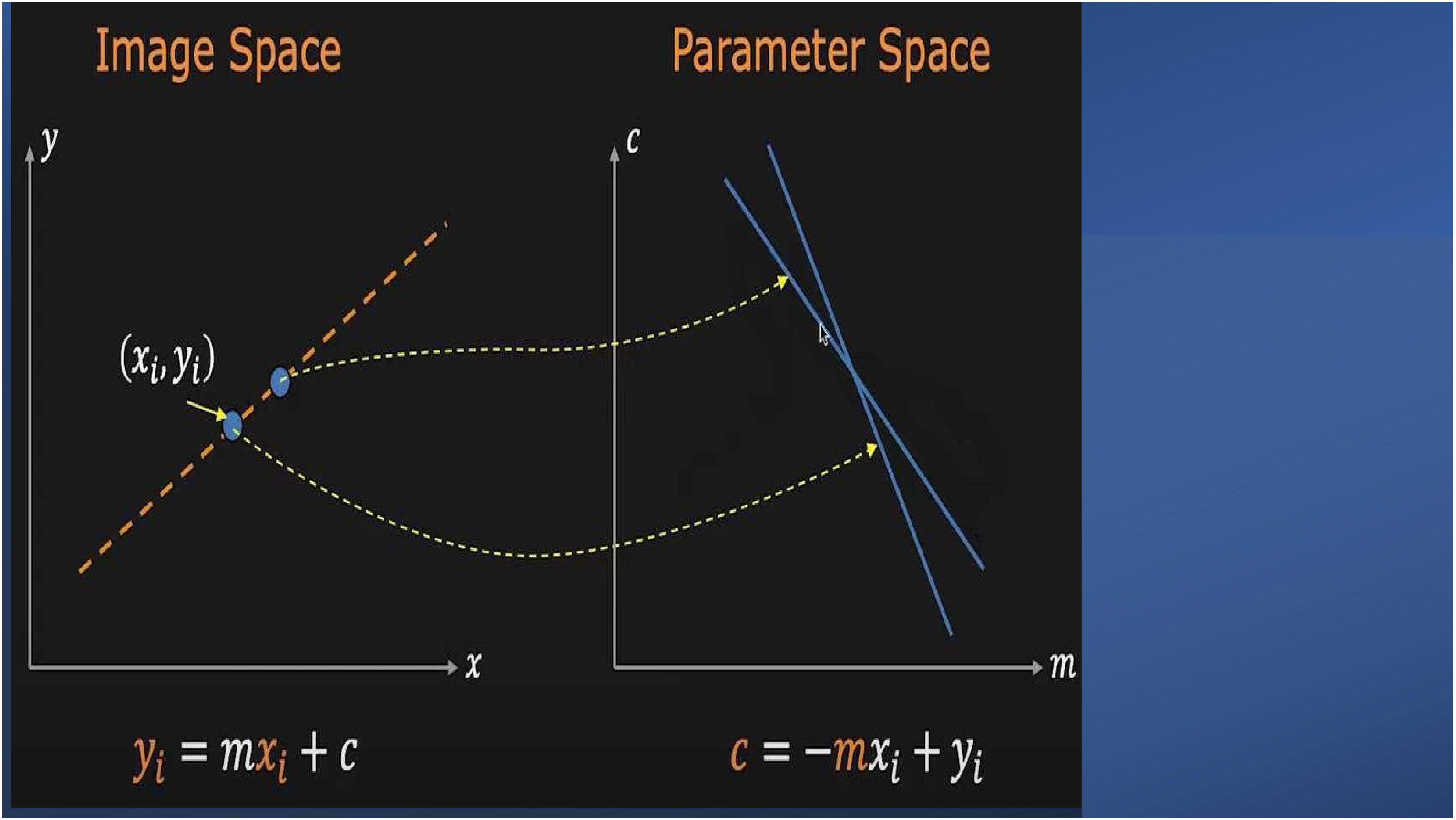


Image Space
Parameter Space
y
x
c
m
$(x_i, y_i)$
$y_i = mx_i + c$
$c = -mx_i + y_i$

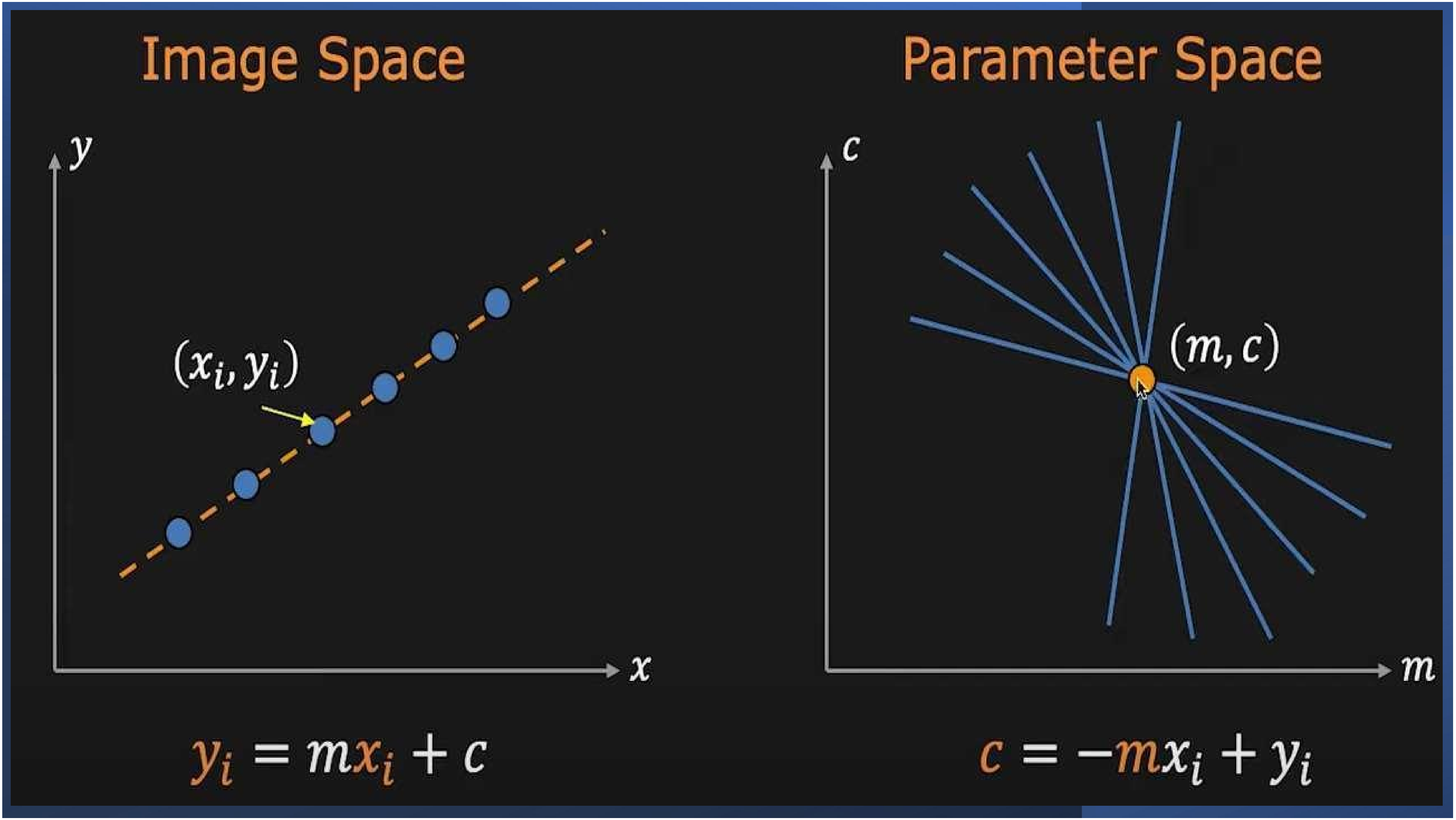


Image Space
Parameter Space
$y$
$x$
$(x_i, y_i)$
$c$
$m$
$(m, c)$
$y_i = mx_i + c$
$c = -mx_i + y_i$

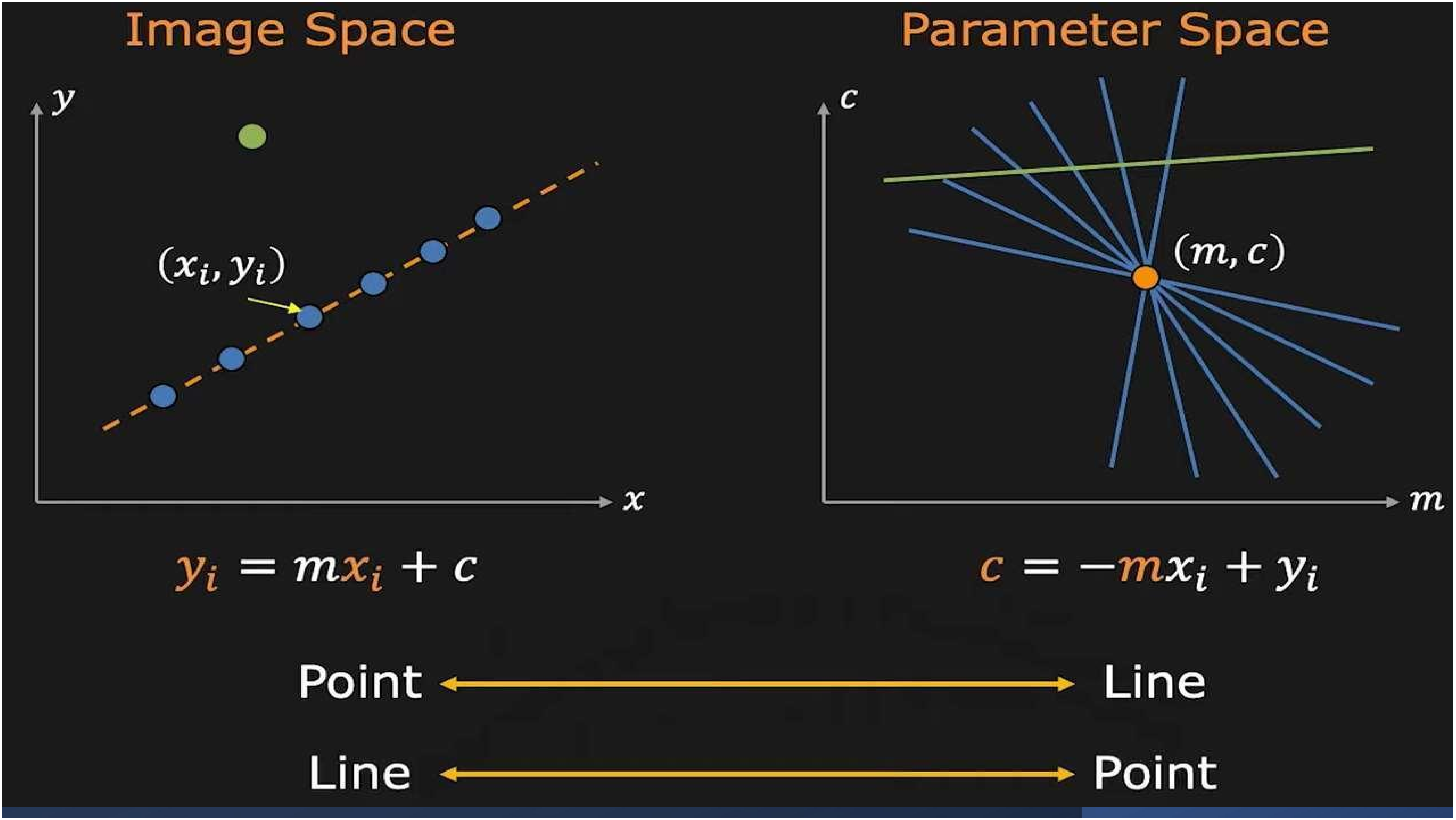
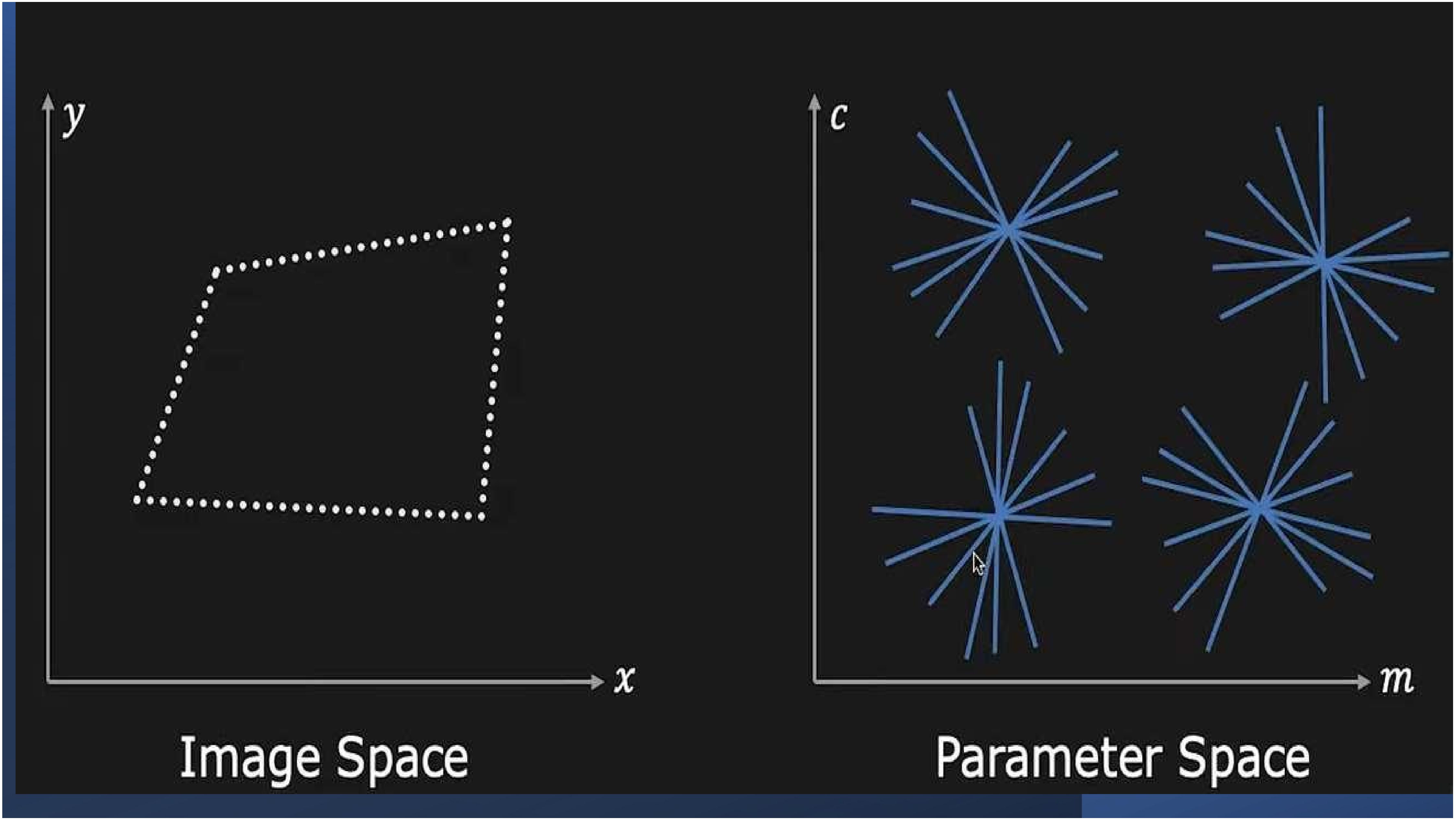
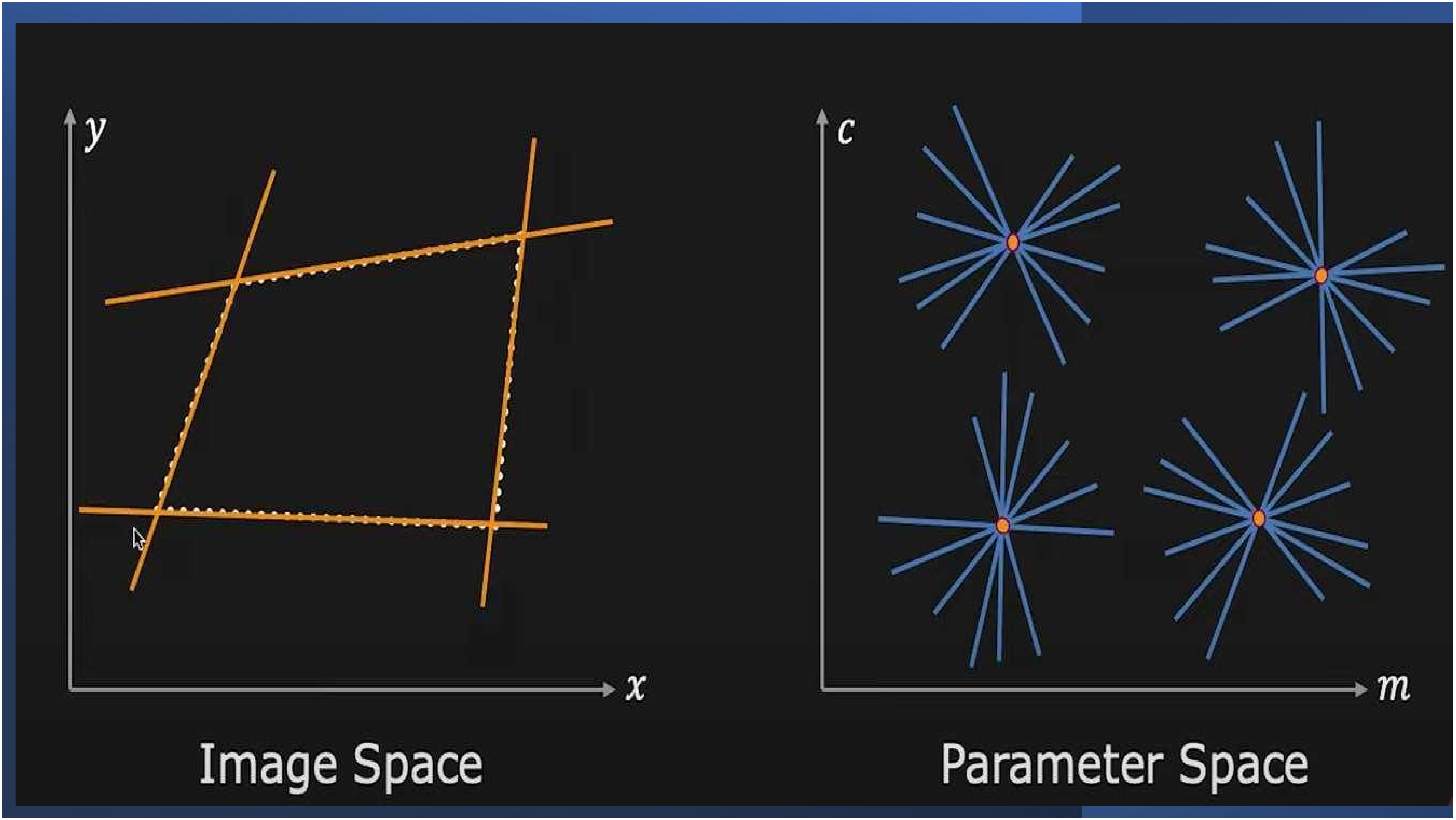


Image Space
Parameter Space
$y$
$x$
$(x_i, y_i)$
$c$
$m$
$(m, c)$
$y_i = mx_i + c$
$c = -mx_i + y_i$
Point
Line
Line
Point


y
x
Image Space
c
m
Parameter Space



y
x
Image Space
c
m
Parameter Space

# Line Detection Algorithm

Step 1. Quantize parameter space $(m, c)$

Step 2. Create accumulator array $A(m, c)$

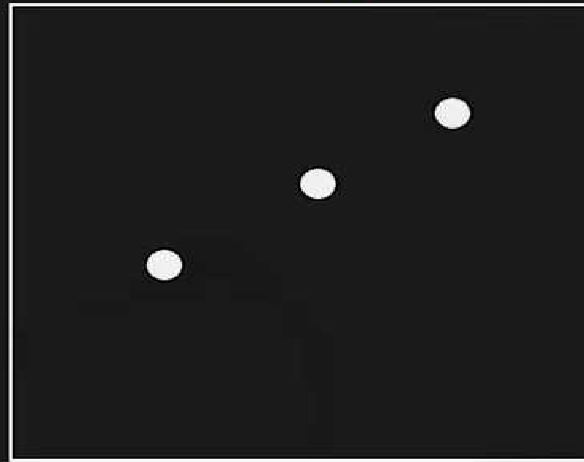
Step 3. Set $A(m, c) = 0$ for all $(m, c)$

Step 4. For each edge point $(x_i, y_i)$,

$$A(m, c) = A(m, c) + 1$$

if $(m, c)$ lies on the line: $c = -mx_i + y_i$

Image



$A(m, c)$

$c$

| | | | | |
|---|---|---|---|---|
| 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 |

Step 1. Quantize parameter space $(m, c)$

Step 2. Create accumulator array $A(m, c)$

Step 3. Set $A(m, c) = 0$ for all $(m, c)$

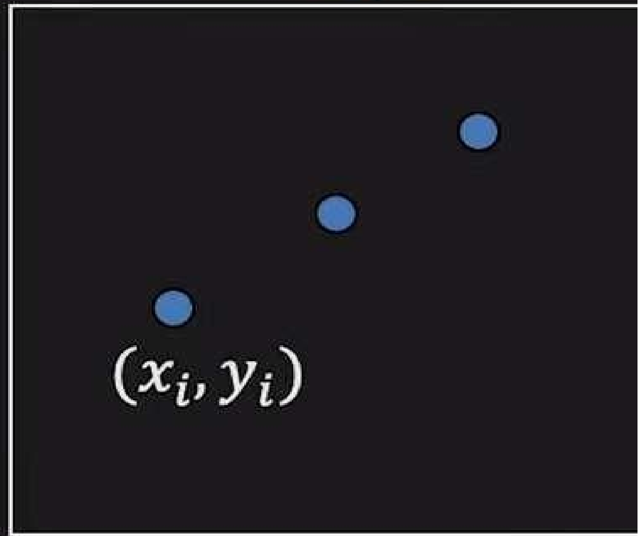
Step 4. For each edge point $(x_i, y_i)$,

$$A(m, c) = A(m, c) + 1$$

if $(m, c)$ lies on the line: $c = -mx_i + y_i$

Step 5. Find local maxima in $A(m, c)$

Image




$A(m, c)$

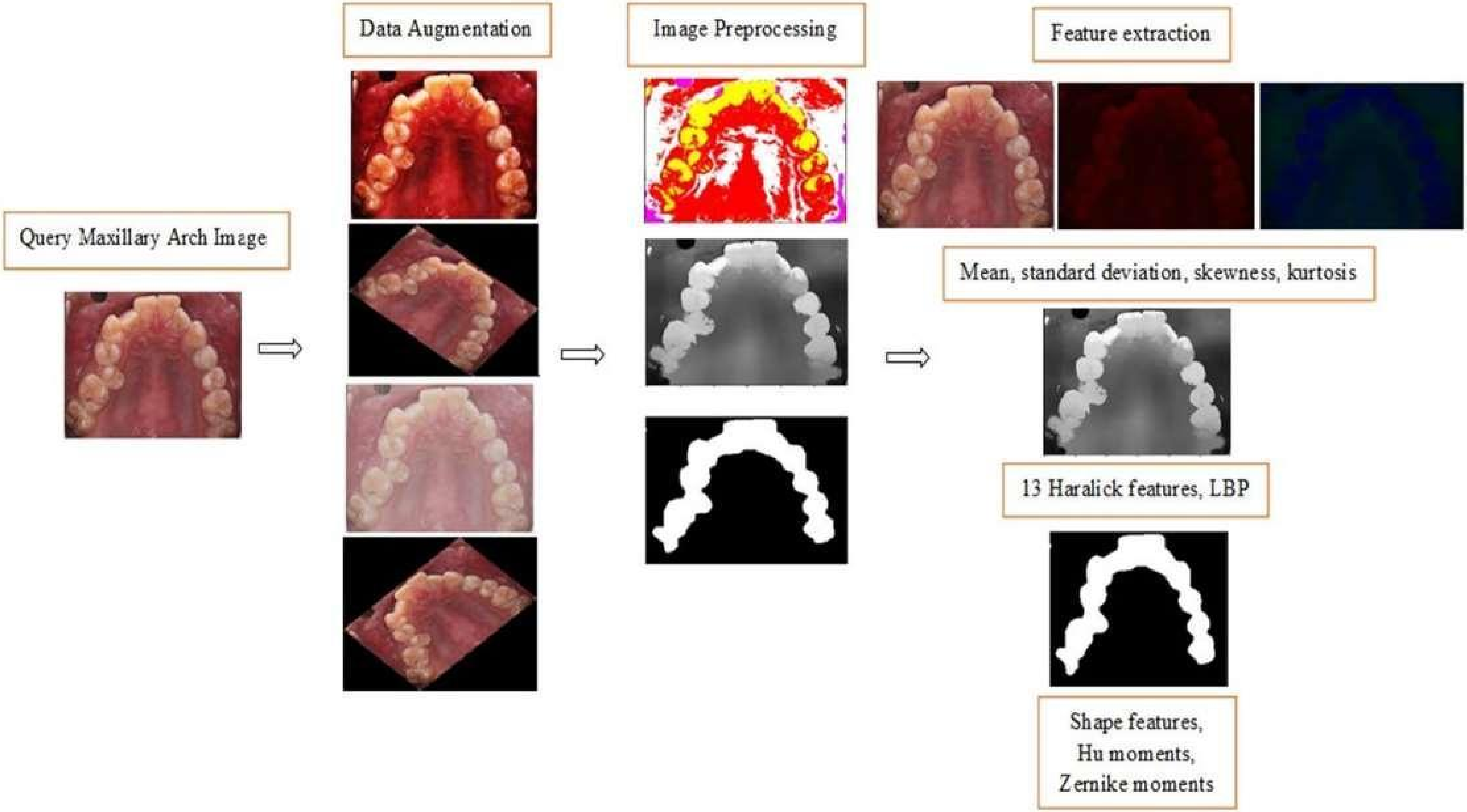
| $c$ | | | | | |
|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | 1 |
| | 0 | 1 | 0 | 1 | 0 |
| | 1 | 1 | 3 | 1 | 1 |
| | 0 | 1 | 0 | 1 | 0 |
| | 1 | 0 | 0 | 0 | 1 |

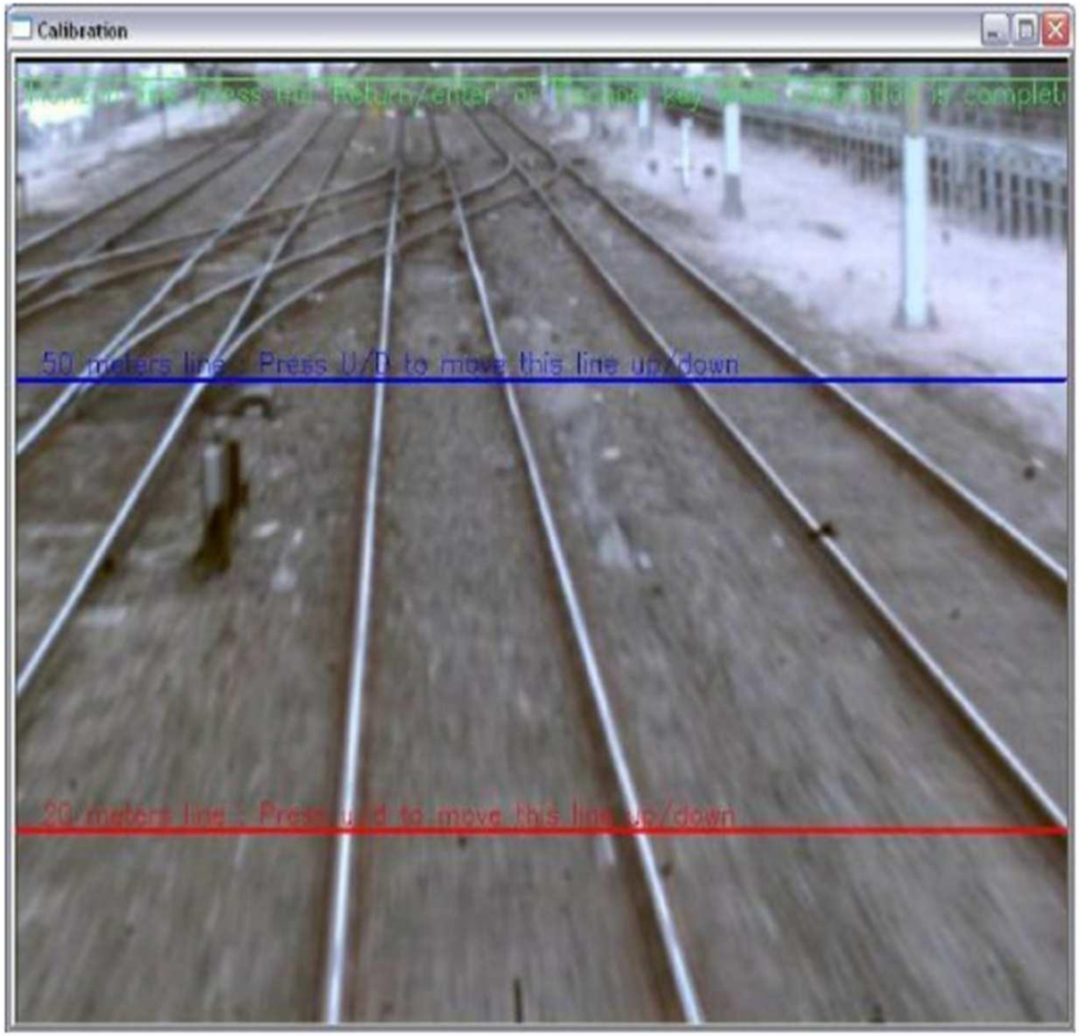
# Hough transform - advantages

- Advantages:
  - Conceptually simple.
  - Easy implementation.
  - Handles missing and occluded data very gracefully.
  - Can be adapted to many types of forms, not just lines.

# Hough transform - disadvantages

- Disadvantages:
  - Computationally complex for objects with many parameters.
  - Looks for only one single type of object.
  - Can be "fooled" by "apparent lines".
  - The length and the position of a line segment cannot be determined.
  - Co-linear line segments cannot be separated.

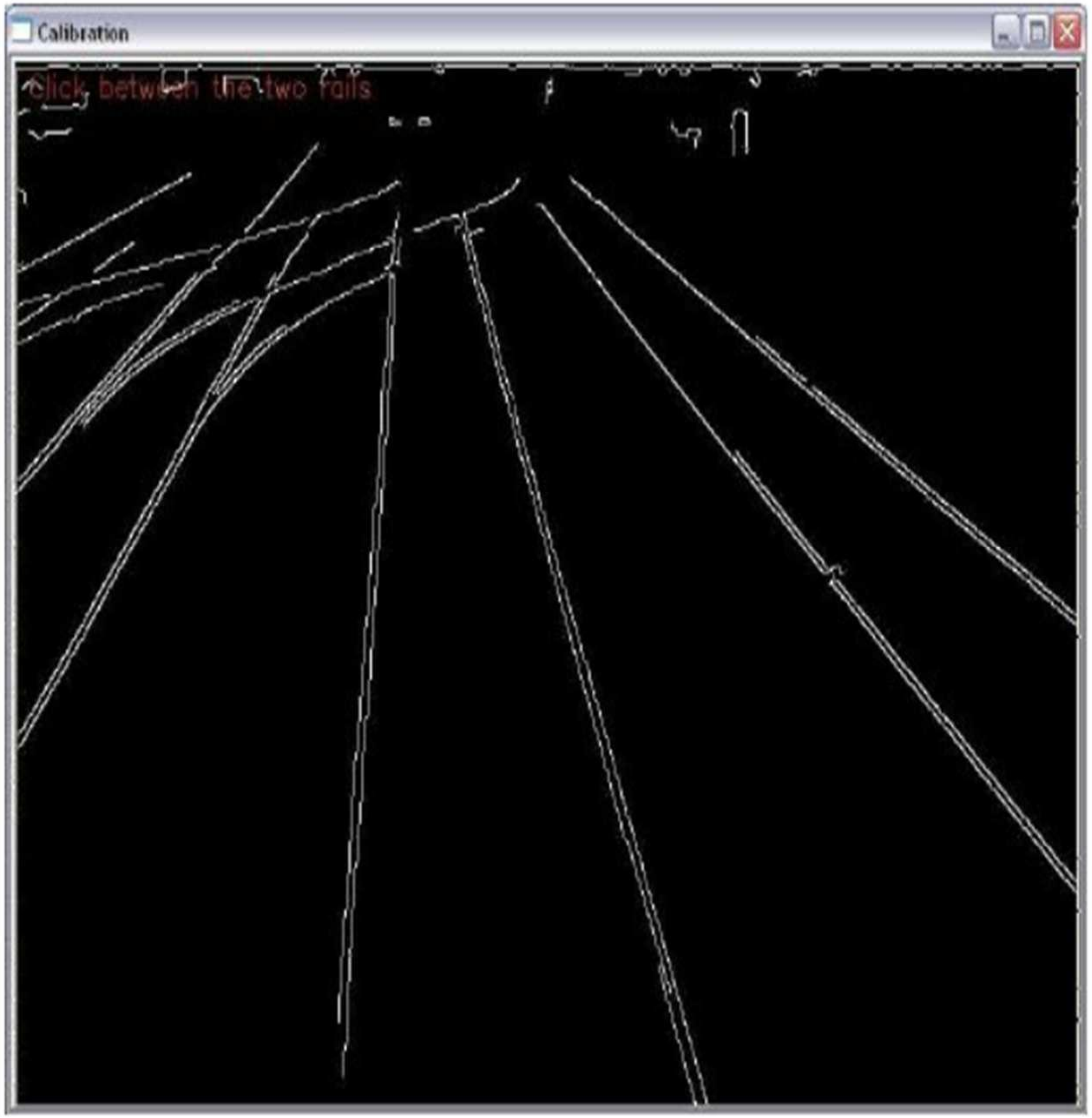

Query Maxillary Arch Image
Data Augmentation
Image Preprocessing
Feature extraction
Mean, standard deviation, skewness, kurtosis
13 Haralick features, LBP
Shape features,
Hu moments,
Zernike moments




# Figure

Caption

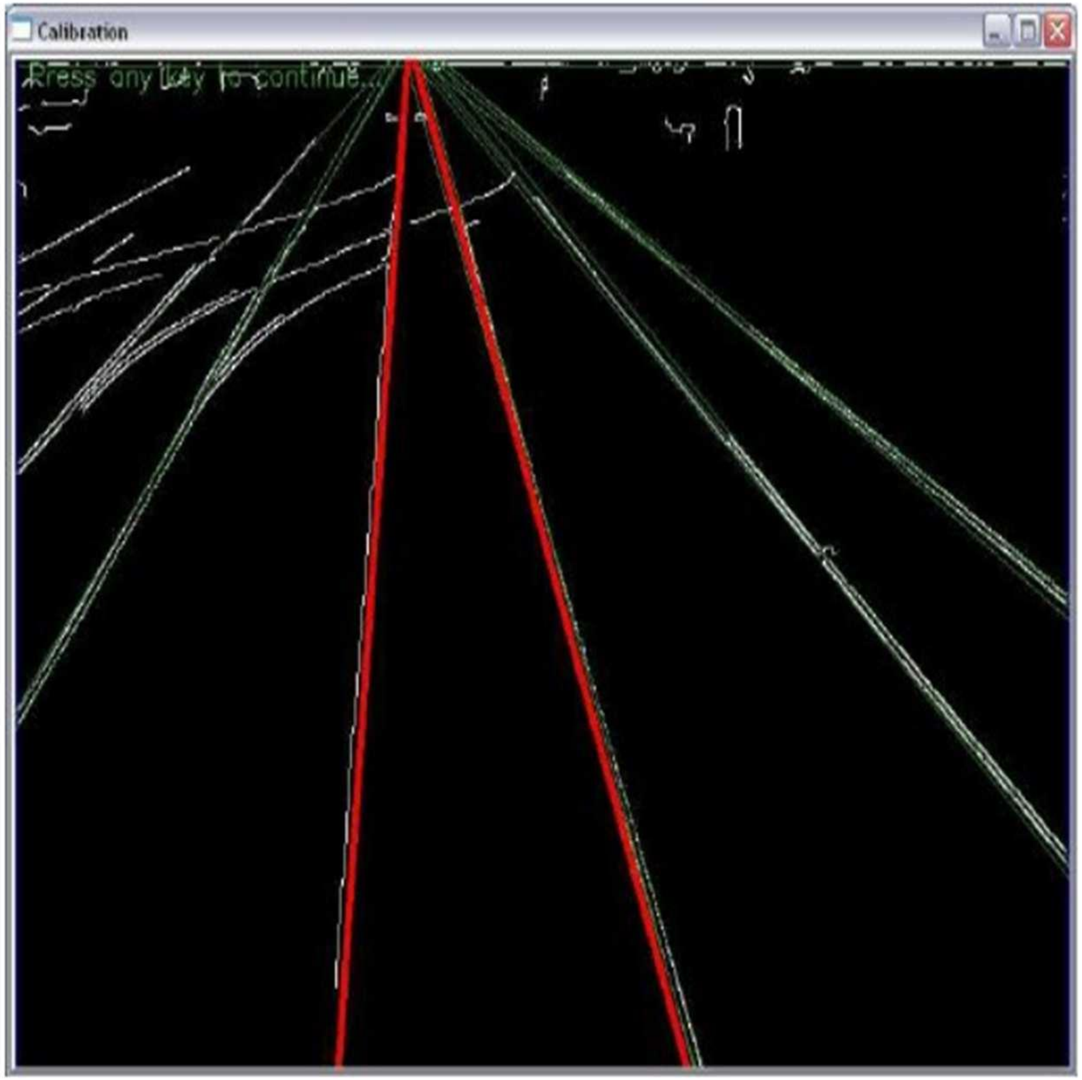
Fig. 4. The calibration trapezoid is specified by the lines at 20 m and 50 m and a pair of rails.




# Figure

Caption

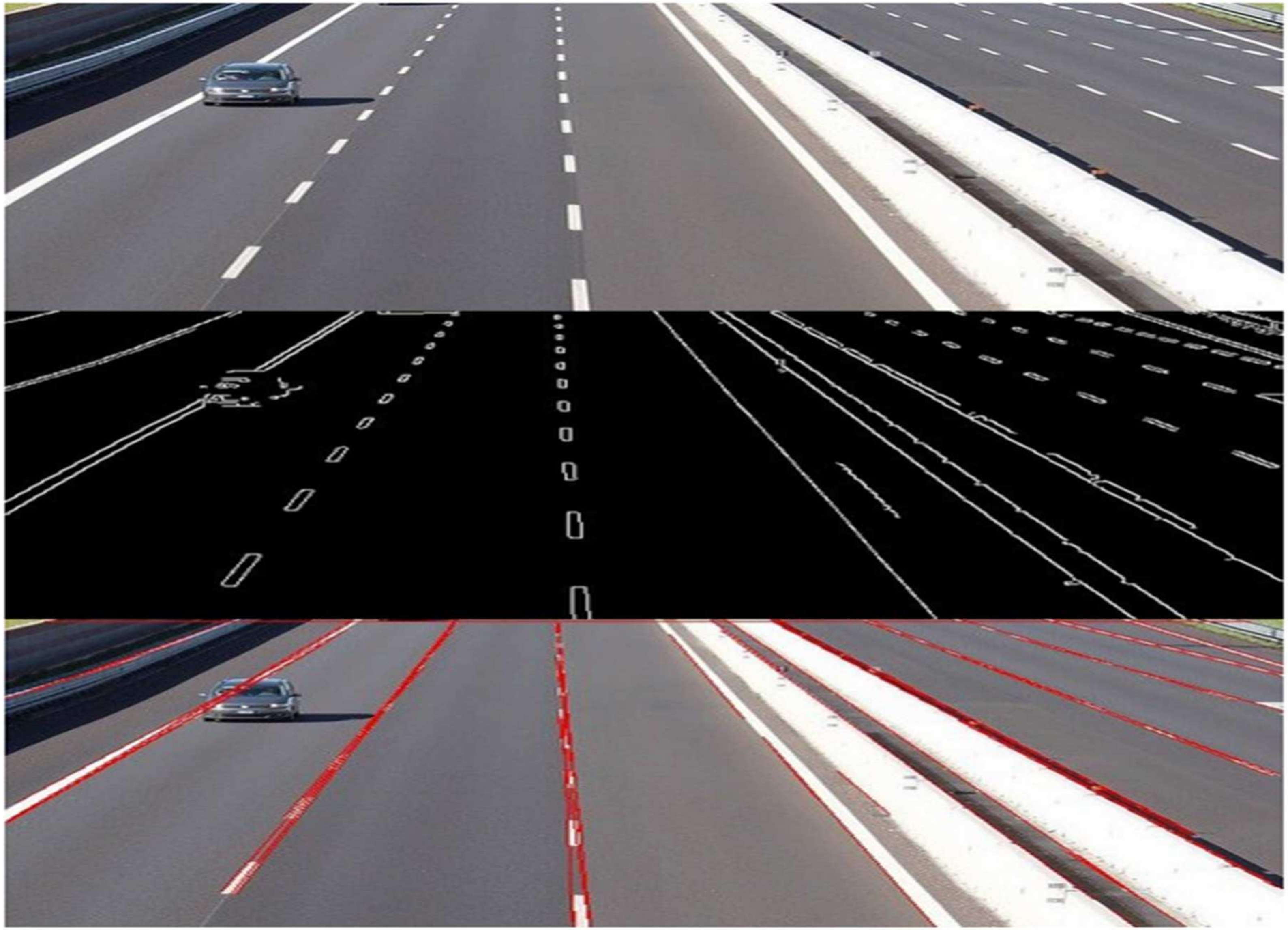
Fig. 5. The Canny edge detection threshold parameters are selected automatically.

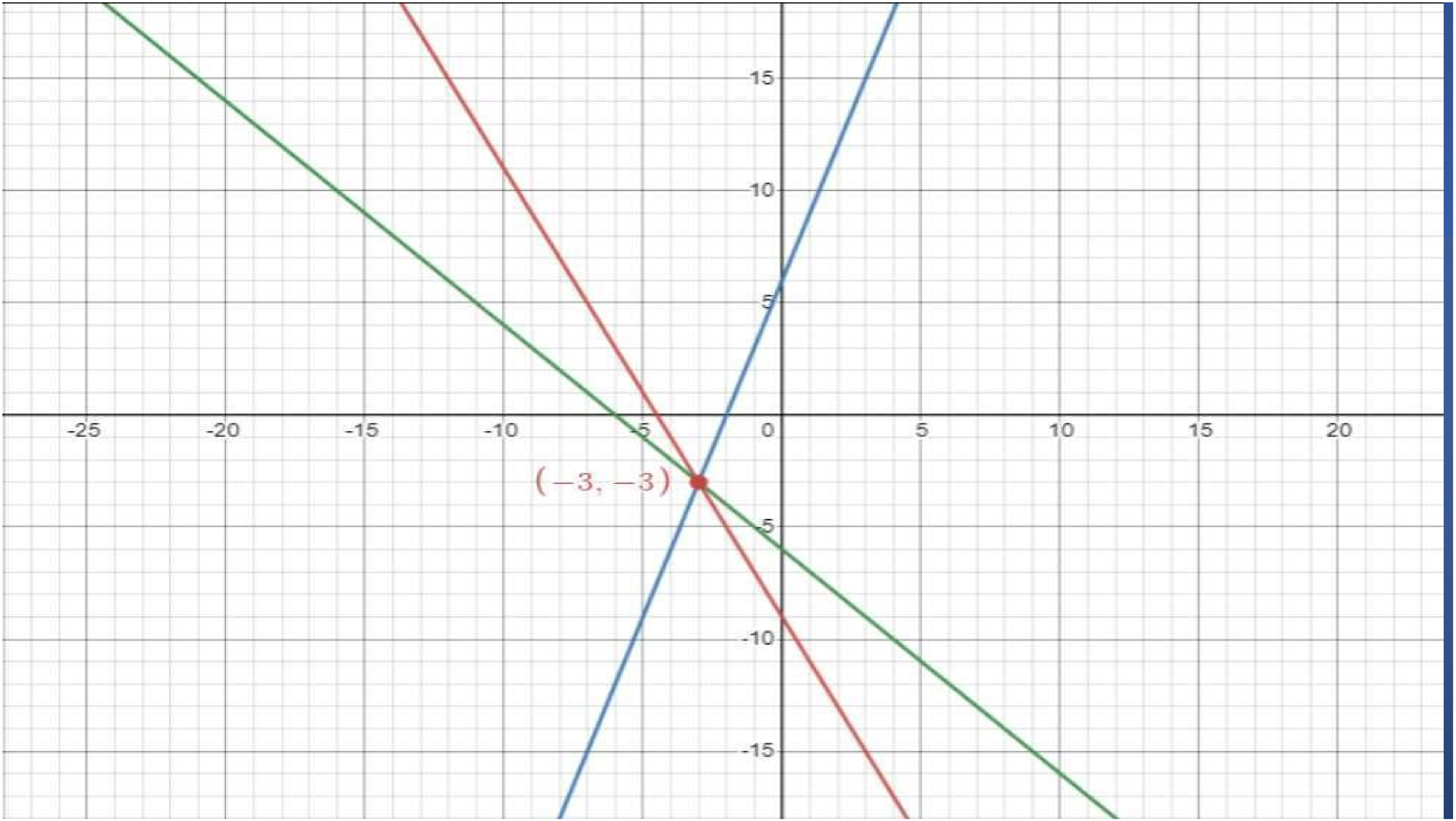



# Figure

Caption

Fig. 6. The rail lines are detected by a Hough transform. The operator clicked in the Hough image between the rails of interest. The identified pair of rails is highlighted in red.





15
10
5
-5
-10
-15
-25
-20
-15
-10
-5
0
5
10
15
20
$(-3, -3)$

# What is Clustering?

- Organizing data into classes such that there is
  - high intra-class similarity
  - low inter-class similarity
- Finding the class labels and the number of classes directly from the data (in contrast to classification).
- More informally, finding natural groupings among objects.

- In machine learning, high intra-class similarity refers to a situation where the data points within the same class or category are very similar to each other. In other words, instances belonging to the same class are more alike and share common characteristics or patterns.

Positive implications:

1. Robustness: High intra-class similarity suggests that the class is well-defined and consistent, making it easier for a classifier to identify and generalize patterns.
2. Generalization: A model trained on data with high intra-class similarity is likely to generalize well to unseen examples of the same class because of the strong similarities.

Negative implications:

1. Overfitting: In cases of extreme intra-class similarity, the model might overfit to specific details of the training data and fail to generalize well to unseen examples.
2. Ambiguity: When intra-class similarity is very high, it becomes challenging for the model to distinguish between different instances of the same class, leading to ambiguous predictions.

Dealing with high intra-class similarity often involves careful preprocessing of the data and thoughtful feature engineering. For example, adding more diverse and discriminative features can help the model better differentiate between similar instances. Additionally, using regularization techniques can mitigate the risk of overfitting in such cases.

On the other hand, if the intra-class similarity is low, it means that data points within the same class are more diverse and dissimilar. In this scenario, the model may have a harder time learning common patterns for each class and might require more complex representations or larger amounts of data to achieve good performance.

When we say "low intra-class similarity," it means that the samples within the same class are relatively dissimilar to each other. In other words, the data points within a particular class are scattered or spread out in the feature space, and they may not form tight or compact clusters.

Various techniques can be employed to address the challenge of low intra-class similarity, such as:

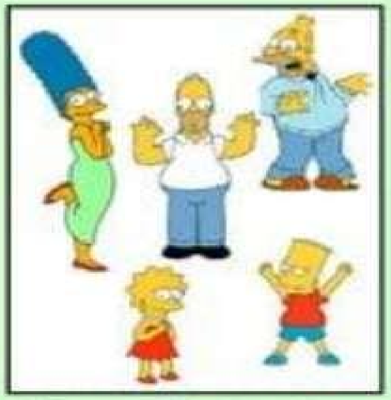
1. Feature Engineering: Creating more informative and discriminative features that help in better class separation.
2. Data Augmentation: Generating additional training samples by applying transformations to existing samples, which can increase the intra-class similarity.
3. Clustering: Performing clustering techniques to group similar samples together and then use the cluster labels as new class labels for training.
4. Prototype Methods: Using prototype-based classification methods that rely on creating representative samples (prototypes) for each class.
5. Generative Models: Using generative models to learn the underlying data distribution and improve the quality of class representations.

The specific approach to address low intra-class similarity depends on the nature of the data and the requirements of the machine learning task at hand.

What is a natural grouping among these objects?

# What is a natural grouping among these objects?

## Clustering is subjective



Simpson's Family

School Employees

Females

Males

# What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

> “It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.”

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

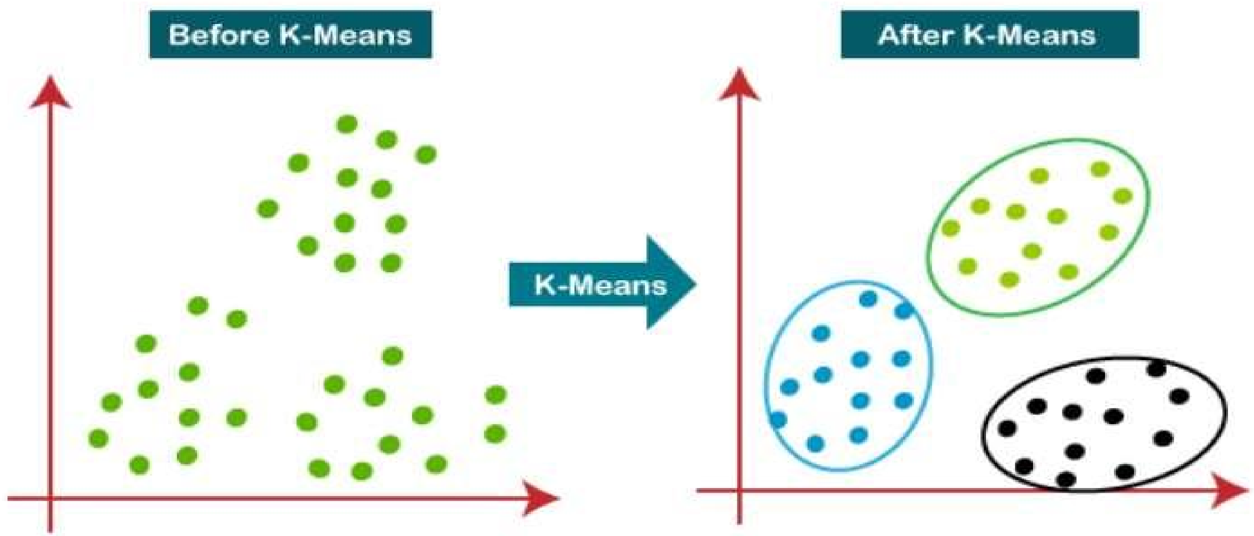
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



# How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

**Step-1:** Select the number K to decide the number of clusters.

**Step-2:** Select random K points or centroids. (It can be other from the input dataset).

**Step-3:** Assign each data point to their closest centroid, which will form the predefined K clusters.

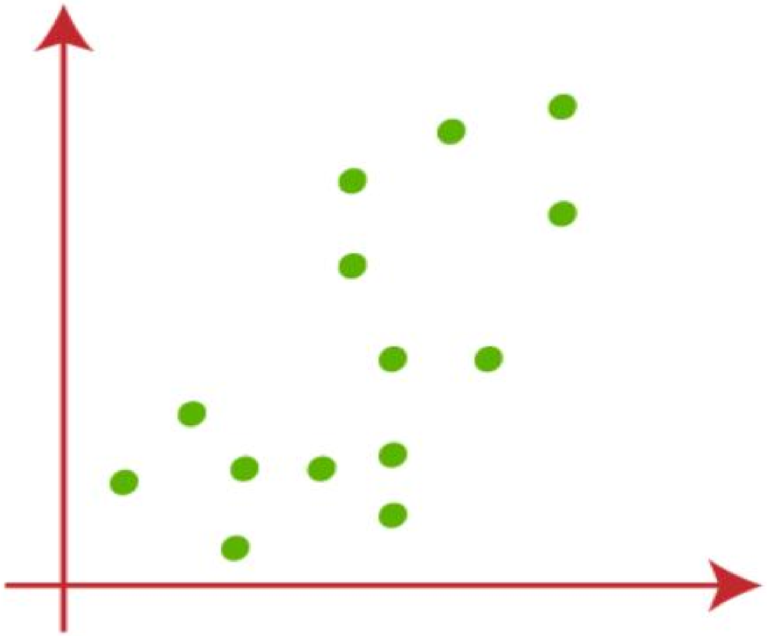
**Step-4:** Calculate the variance and place a new centroid of each cluster.

**Step-5:** Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

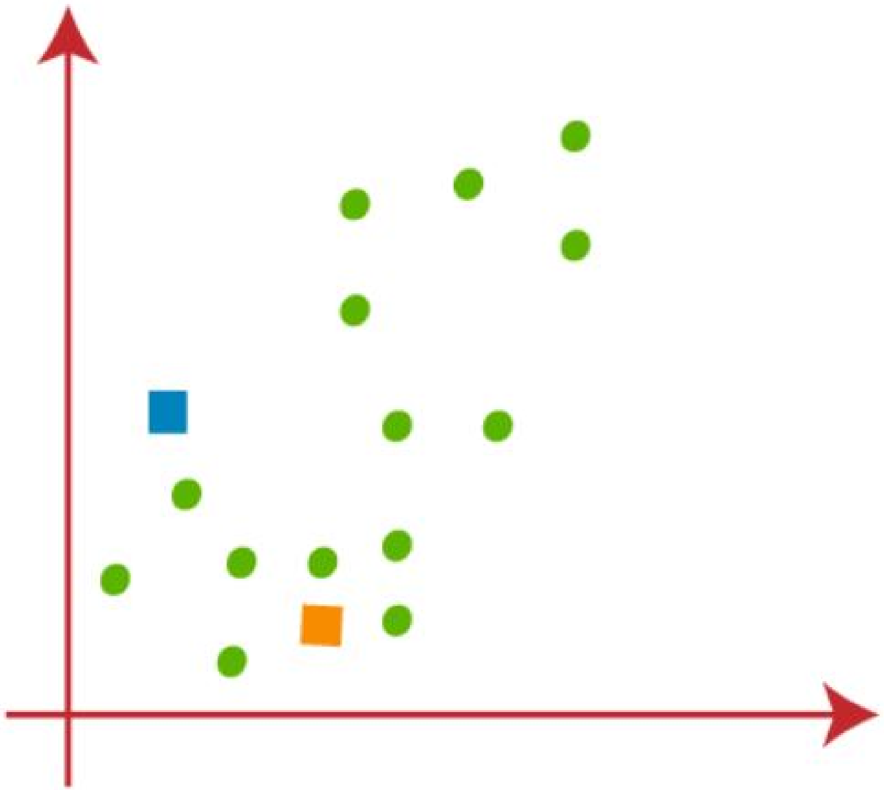
**Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.

**Step-7**: The model is ready.

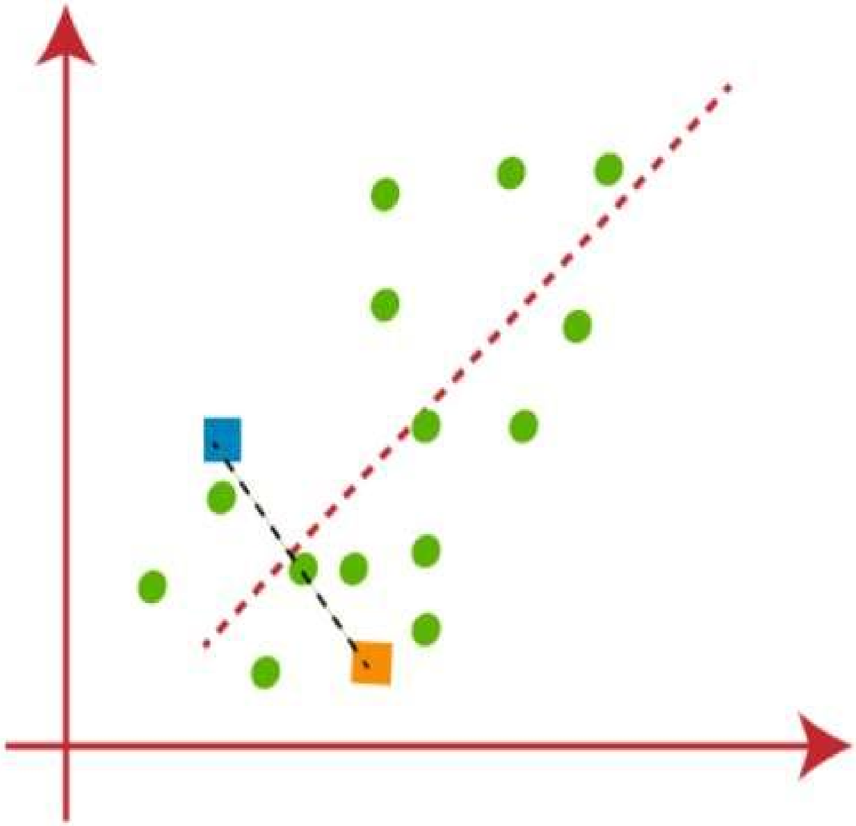
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



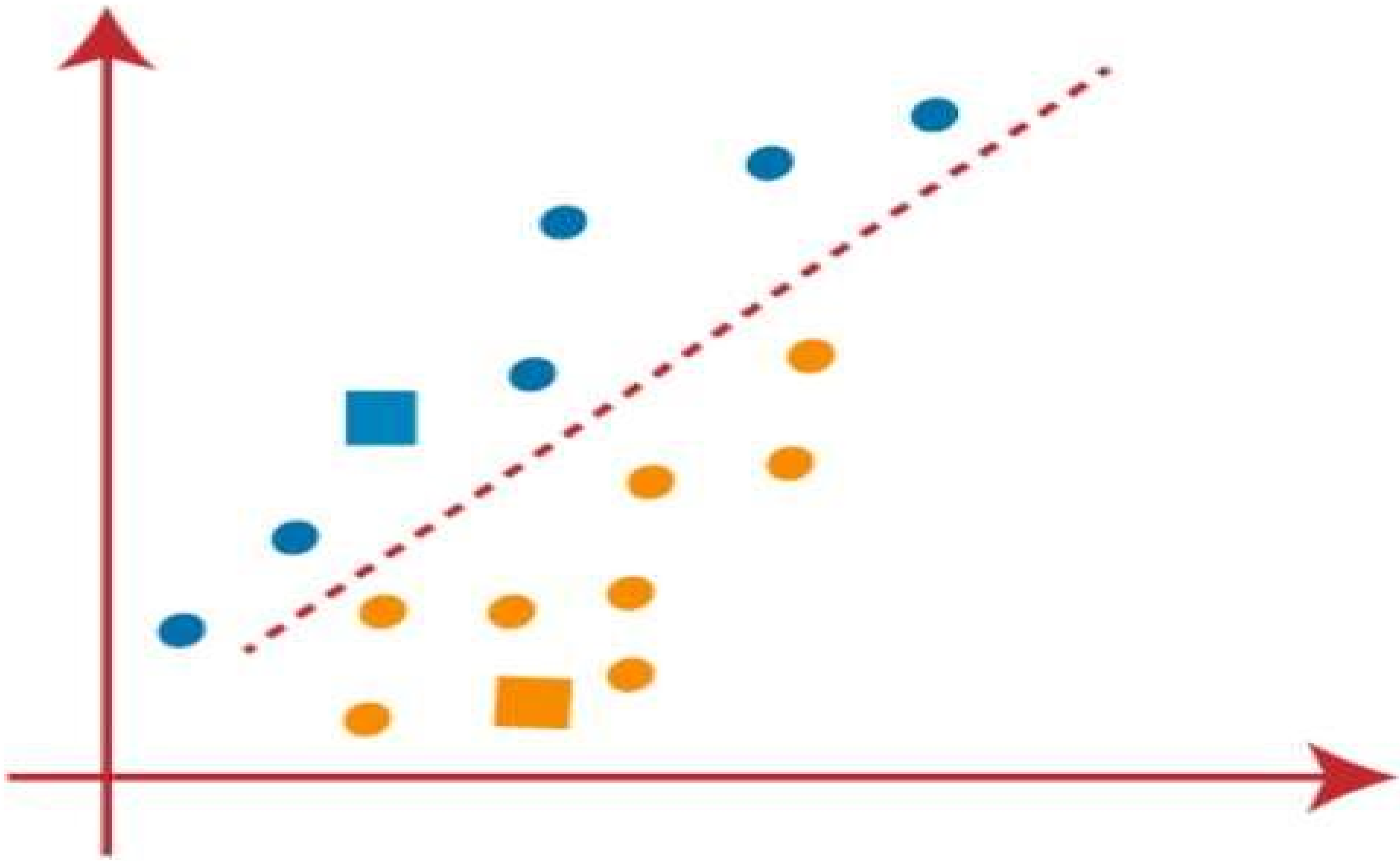
- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image:



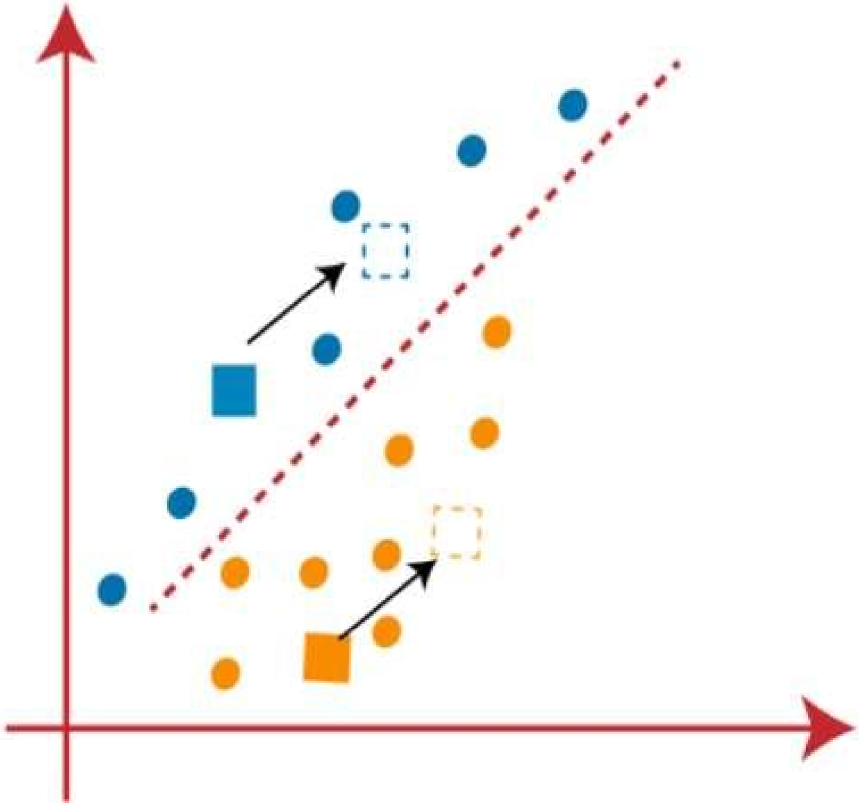
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



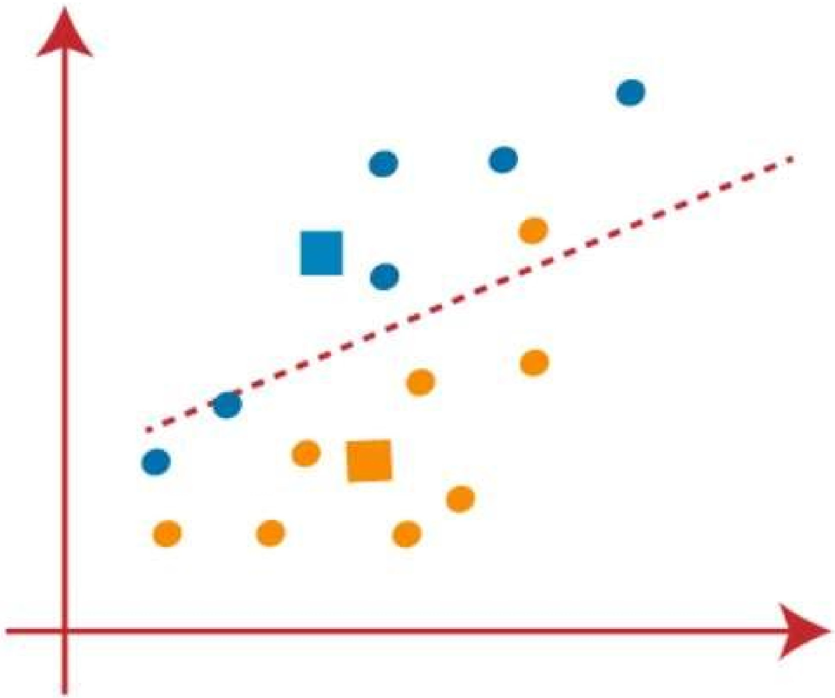
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



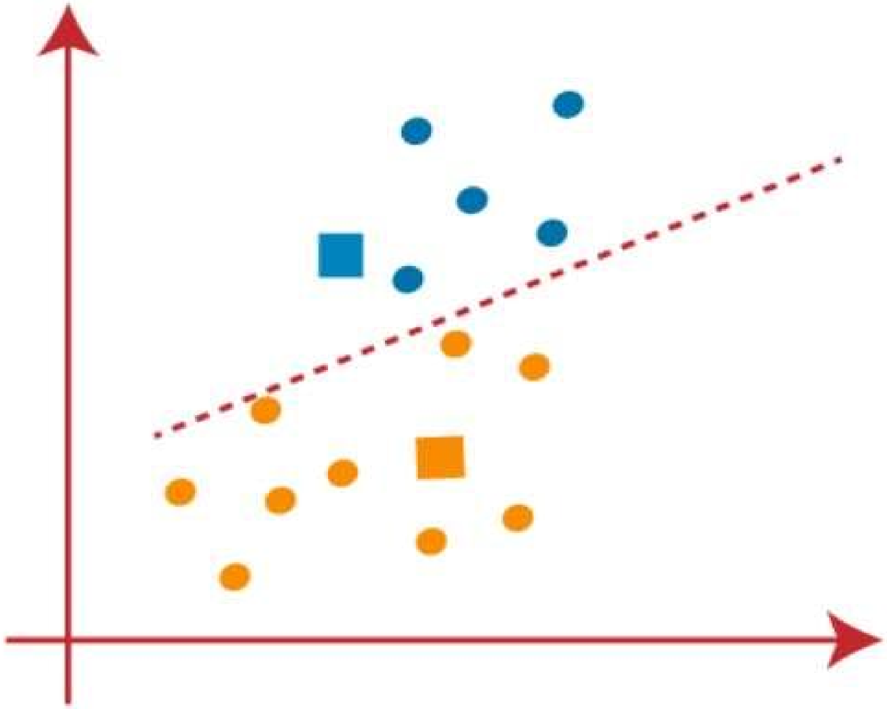
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



- Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:

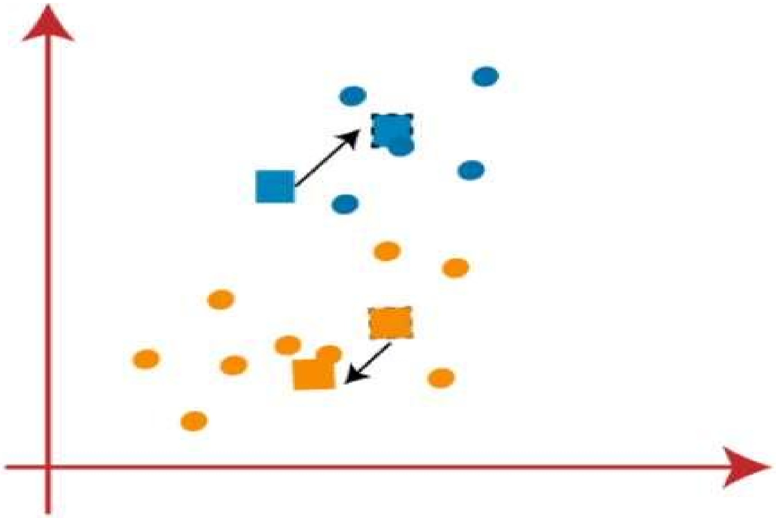


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

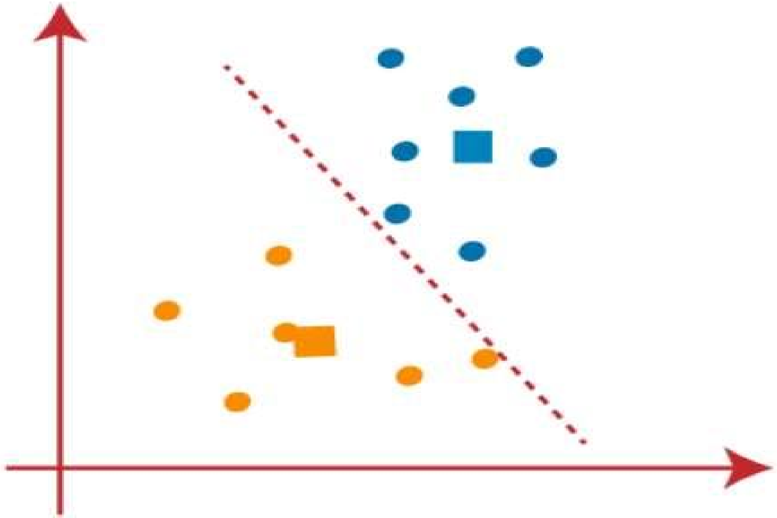


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

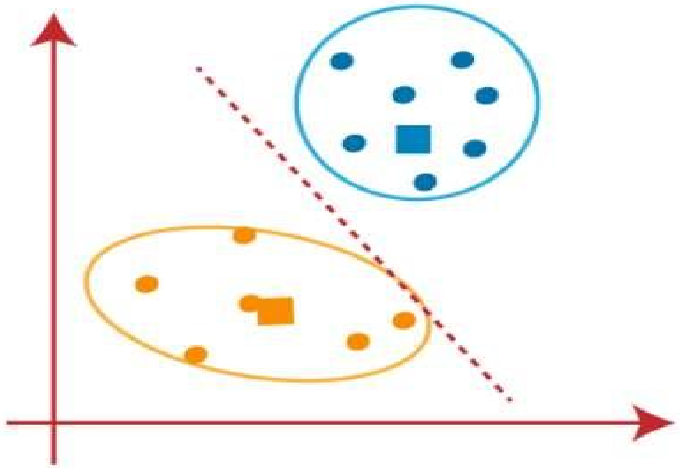
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



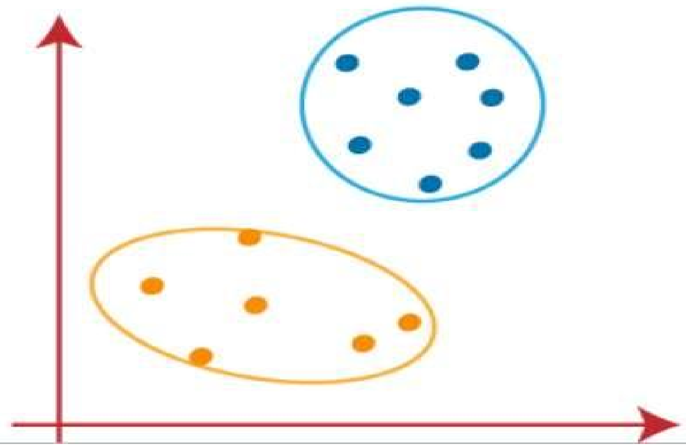
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:



Let's consider a simple example with two-dimensional data points to demonstrate the K-means clustering process. We'll perform K-means clustering with K=2 clusters.

Suppose we have the following data points in a 2D space:

Data points: (2, 3), (3, 5), (3, 4), (5, 1), (6, 2), (8, 3)

Let's assume that we want to find two clusters using the K-means algorithm.

Step 1: Initialization
Randomly select two data points as initial centroids. Let's say we choose (3, 5) and (8, 3) as the initial centroids.

Step 2: Assignment
Calculate the distance between each data point and the centroids. Assign each data point to the nearest centroid. For simplicity, we'll use Euclidean distance as the distance metric.

- Distance from (2, 3) to (3, 5) = sqrt((2-3)^2 + (3-5)^2) = sqrt(2)
- Distance from (2, 3) to (8, 3) = sqrt((2-8)^2 + (3-3)^2) = 6
- Distance from (3, 5) to (3, 5) = 0
- Distance from (3, 5) to (8, 3) = sqrt((3-8)^2 + (5-3)^2) = sqrt(13)
- Distance from (3, 4) to (3, 5) = 1
- Distance from (3, 4) to (8, 3) = sqrt((3-8)^2 + (4-3)^2) = sqrt(10)
- Distance from (5, 1) to (3, 5) = sqrt((5-3)^2 + (1-5)^2) = 4
- Distance from (5, 1) to (8, 3) = sqrt((5-8)^2 + (1-3)^2) = sqrt(5)
- Distance from (6, 2) to (3, 5) = sqrt((6-3)^2 + (2-5)^2) = sqrt(13)
- Distance from (6, 2) to (8, 3) = sqrt((6-8)^2 + (2-3)^2) = sqrt(5)
- Distance from (8, 3) to (3, 5) = sqrt((8-3)^2 + (3-5)^2) = sqrt(13)
- Distance from (8, 3) to (8, 3) = 0

Based on the distances, we can see that (2, 3), (3, 4), and (5, 1) are closer to the first centroid (3, 5), and (6, 2) and (8, 3) are closer to the second centroid (8, 3).

Step 3: Update
Calculate the new centroids by taking the mean of the data points in each cluster.

New centroid 1 = (mean of (2, 3), (3, 4), (5, 1)) = ((2+3+5)/3, (3+4+1)/3) = (3.33, 2.67)

New centroid 2 = (mean of (6, 2), (8, 3)) = ((6+8)/2, (2+3)/2) = (7, 2.5)

Step 4: Repeat
Now, we repeat steps 2 and 3, assigning data points to the nearest centroid and updating the centroids based on the new assignments. We keep iterating until the centroids stabilize.

Let's assume that after one more iteration, the centroids do not change significantly:

Centroid 1 = (3.4, 2.67)
Centroid 2 = (7.25, 2.75)

At this point, the centroids have stabilized, and the K-means algorithm has converged.

The final result is two clusters with the following data points assigned to each cluster:

Cluster 1: (2, 3), (3, 4), (5, 1)
Cluster 2: (6, 2), (8, 3)

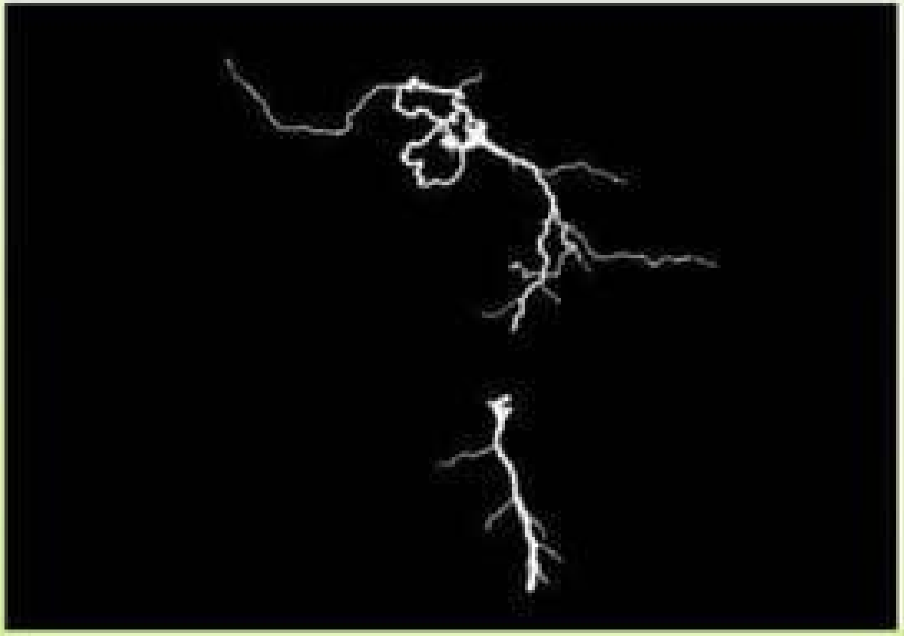
# Example

# Region Growing

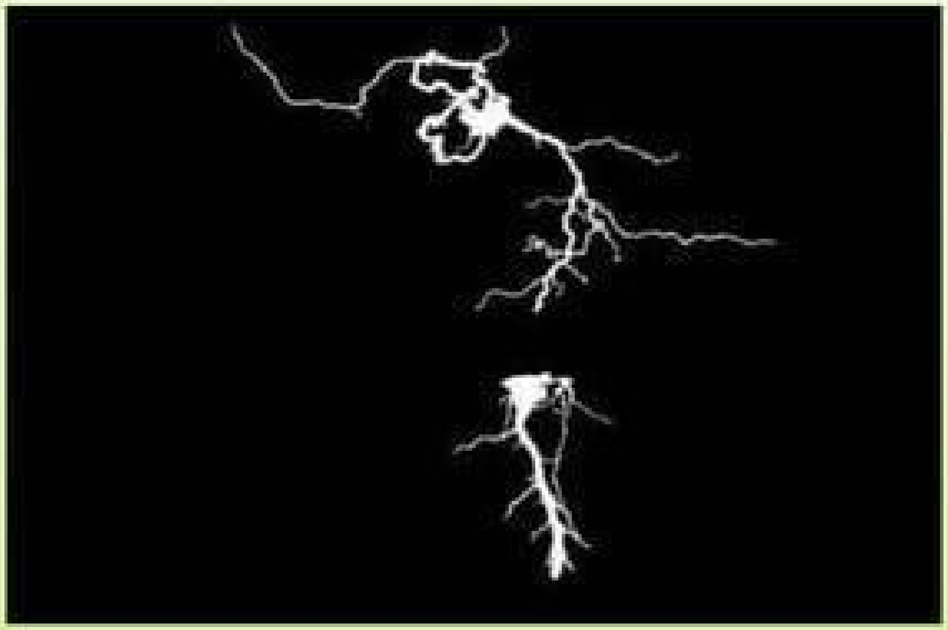
- Region growing is a procedure that groups pixels or sub regions into larger regions.
- The simplest of these approaches is pixel aggregation, which starts with a set of "seed" points and from these grows regions by appending to each seed points those neighboring pixels that have similar properties (such as gray level, texture, color, shape).
- Region growing based techniques are better than the edge-based techniques in noisy images where edges are difficult to detect

Original figure

The Seed Points



Result of region growing



Boundaries of segmented defective welds

# THE ADVANTAGES AND DISADVANTAGES OF REGION GROWING

Advantages

- Region growing methods can correctly separate the regions that have the same properties we define.
- Region growing methods can provide the original images which have clear edges with good segmentation results.
- The concept is simple. We only need a small number of seed points to represent the property we want, then grow the region.

# Disadvantage

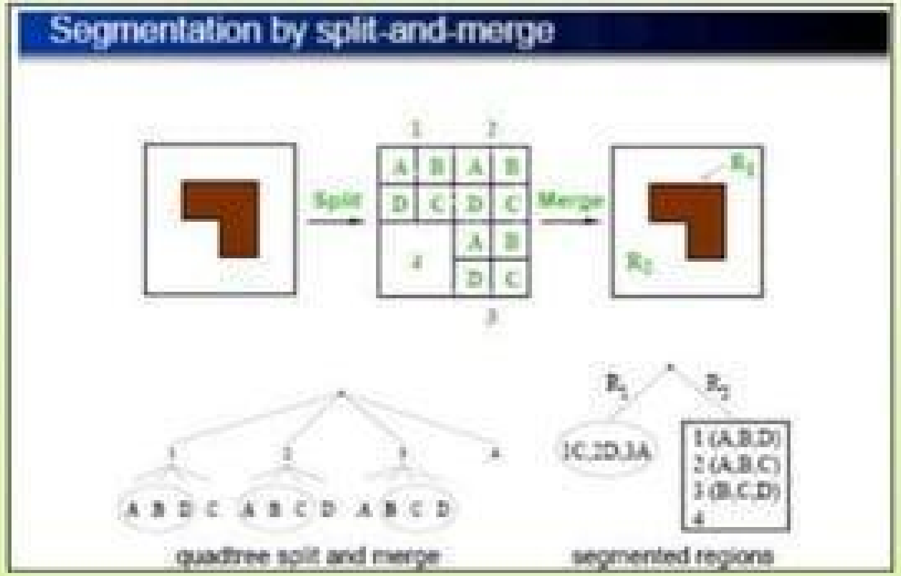
- Computationally expensive
- It is a local method with no global view of the problem.
- Sensitive to noise.
- Unless the image has had a threshold function applied to it, a continuous path of points related to color may exist which connects any two points in the image.

# Split and Merge Approach:

- This is a 2 step procedure:
  - top-down: split image into homogeneous **quadrant regions**
  - bottom-up: merge similar adjacent regions
- The algorithm includes:

  **<u>Top-down</u>**
  - successively subdivide image into quadrant regions *R*
  - stop when all regions are homogeneous: *P(R) = TRUE*) obtain **quadtree structure**

  **<u>Bottom-up</u>**
  - at each level, merge adjacent regions *Rand Rif P(R[ R) = TRUE*
- Iterate until no further splitting/merging is possible



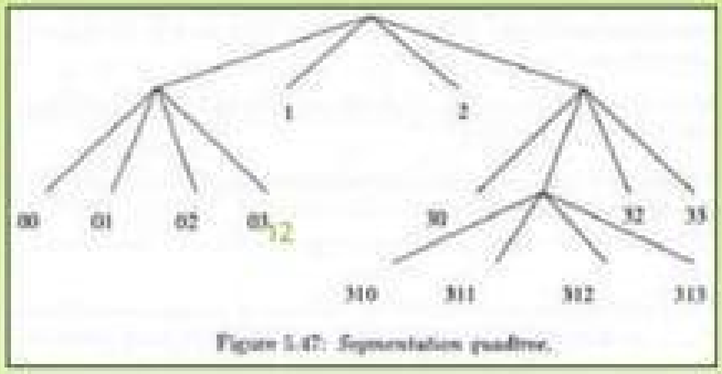
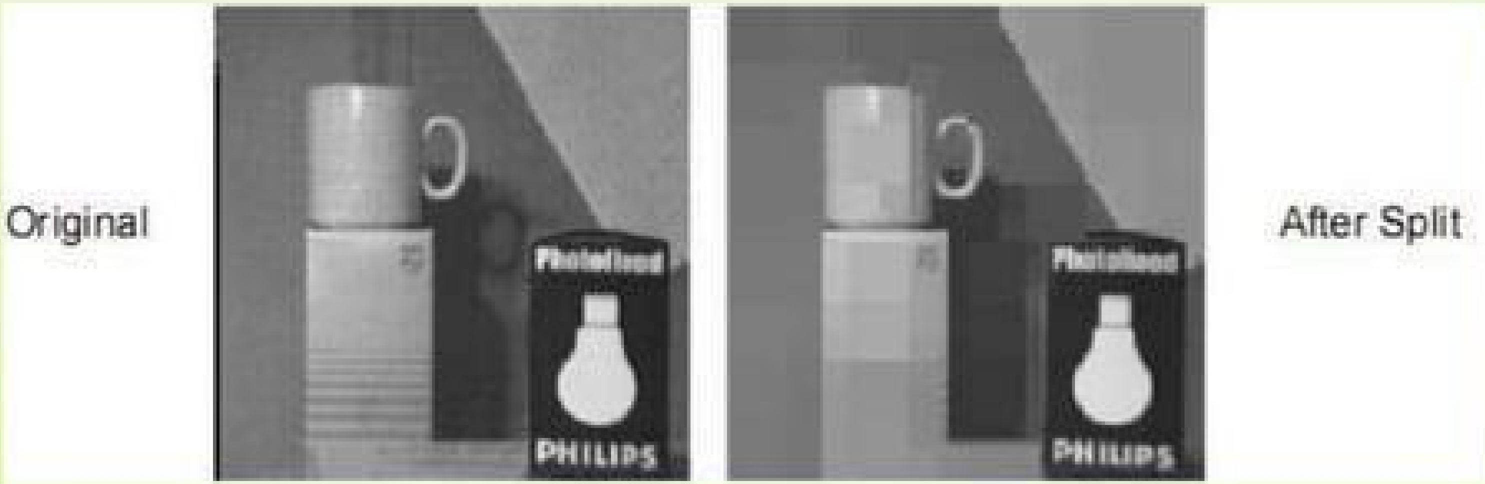
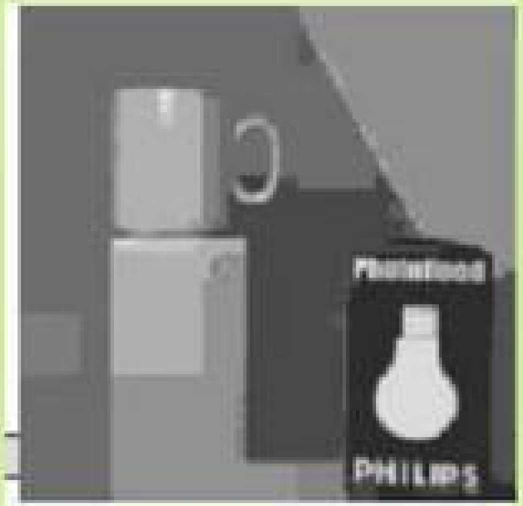



Figure 5.47: Segmentation quadtree.

# EXAMPLE



# The Split-and-Merge Algorithm

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 4 | 9 | 9 | 8 | 1 | 0 |
| 1 | 1 | 8 | 8 | 8 | 4 | 1 | 0 |
| 1 | 1 | 6 | 6 | 6 | 3 | 1 | 0 |
| 1 | 1 | 5 | 6 | 6 | 3 | 1 | 0 |
| 1 | 1 | 5 | 6 | 6 | 2 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

Sample image

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 4 | 9 | 9 | 8 | 1 | 0 |
| 1 | 1 | 8 | 8 | 8 | 4 | 1 | 0 |
| 1 | 1 | 6 | 6 | 6 | 3 | 1 | 0 |
| 1 | 1 | 5 | 6 | 6 | 3 | 1 | 0 |
| 1 | 1 | 5 | 6 | 6 | 2 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

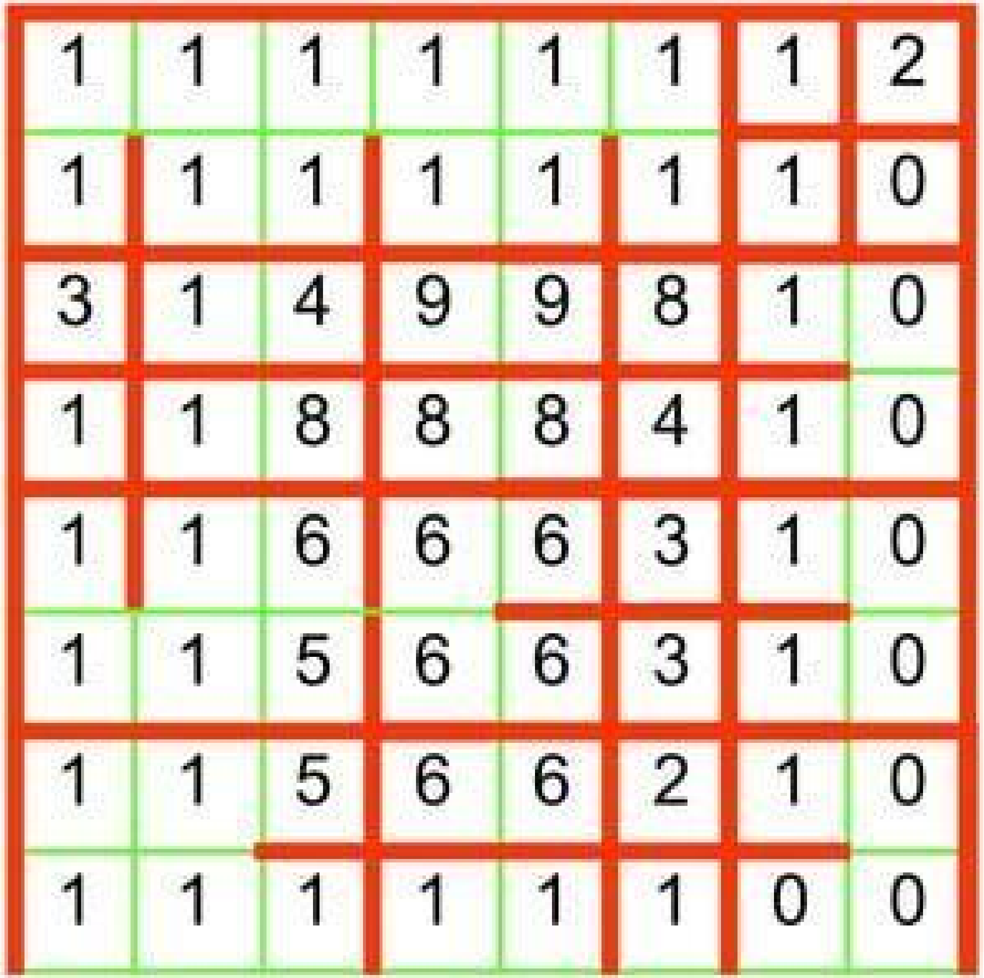
First split

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 4 | 9 | 9 | 8 | 1 | 0 |
| 1 | 1 | 8 | 8 | 8 | 4 | 1 | 0 |
| 1 | 1 | 6 | 6 | 6 | 3 | 1 | 0 |
| 1 | 1 | 5 | 6 | 6 | 3 | 1 | 0 |
| 1 | 1 | 5 | 6 | 6 | 2 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

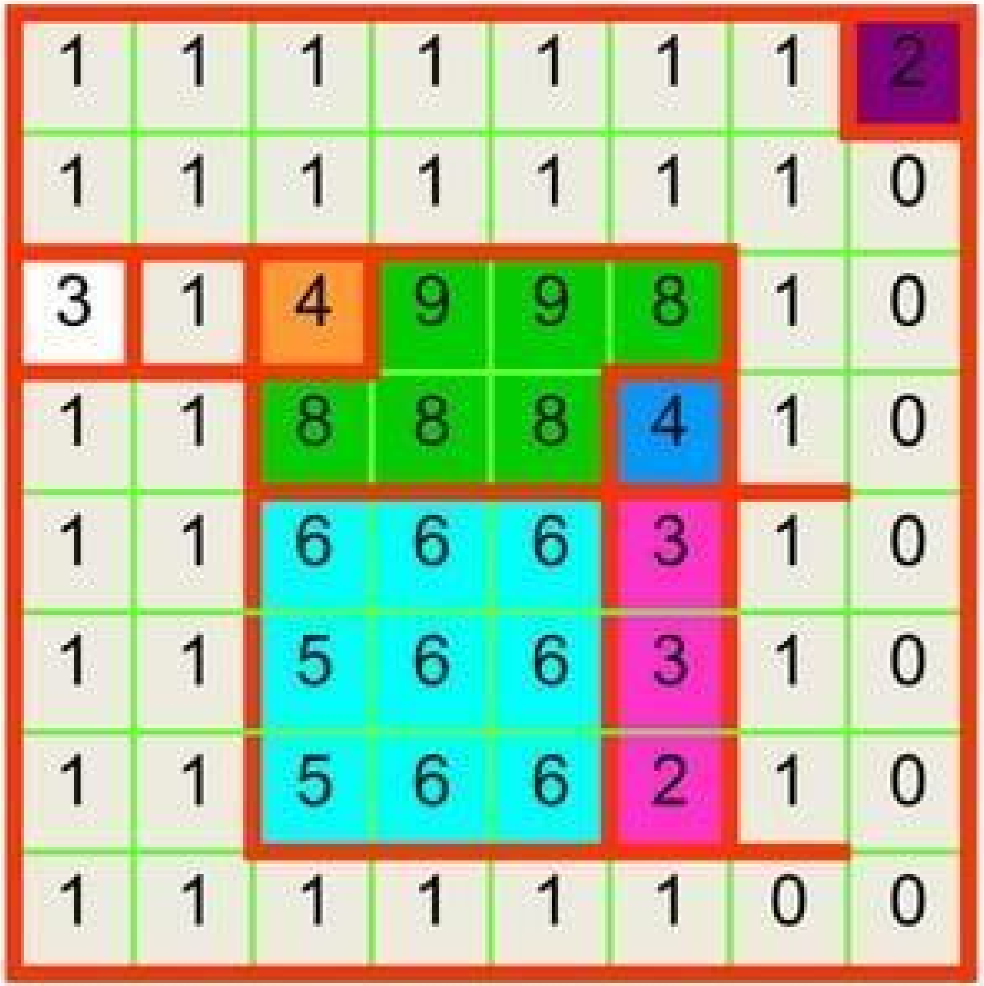
Second split

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 4 | 9 | 9 | 8 | 1 | 0 |
| 1 | 1 | 8 | 8 | 8 | 4 | 1 | 0 |
| 1 | 1 | 6 | 6 | 6 | 3 | 1 | 0 |
| 1 | 1 | 5 | 6 | 6 | 3 | 1 | 0 |
| 1 | 1 | 5 | 6 | 6 | 2 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

Third split




Merge




Final result

# REGION SPLITTING AND MERGING

## Region Splitting

- Region growing starts from a set of seed points.
- An alternative is to start with the whole image as a single region and subdivide the regions that do not satisfy a condition of homogeneity.

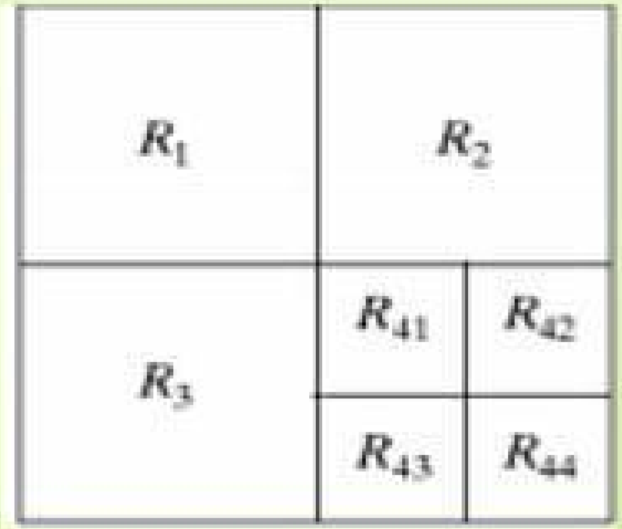
## Region Merging

- Region merging is the opposite of region splitting.
- Start with small regions (e.g. 2x2 or 4x4 regions) and merge the regions that have similar characteristics (such as gray level, variance).
- Typically, splitting and merging approaches are used iteratively

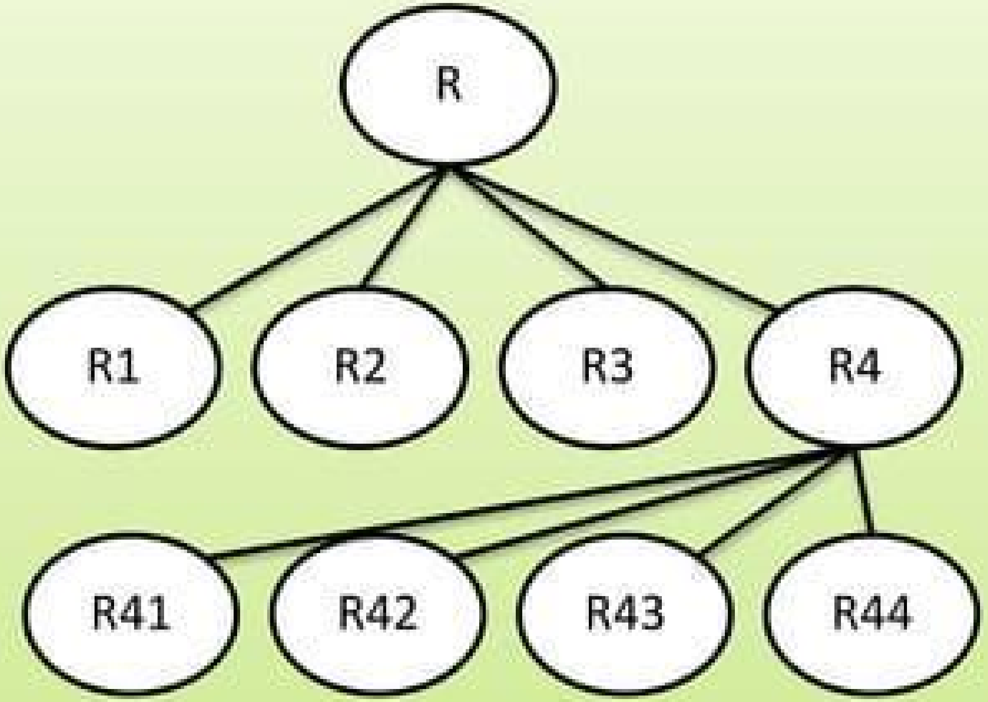
CONTU……….

- Let R represent the entire image region and select a predicate .
- One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that , for $R_i$ , $P(R_i)$ = TRUE.
- If P(R)FALSE divide the image into quadrants .
- If P is FALSE for any quadrant , subdivide that , quadrants and so on.
- This particular splitting technique has a convenient representation in the form called quad tree.

Partitioned image

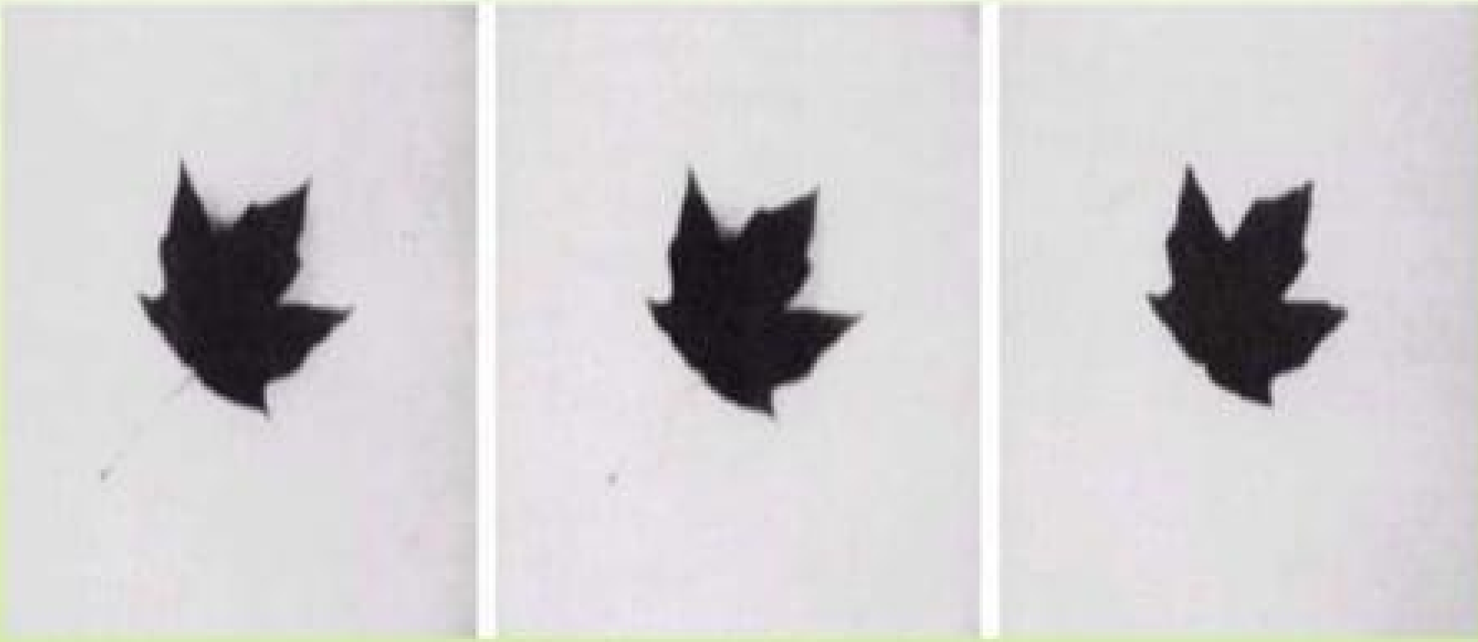



**Corresponding quad tree**



- Split into four disjoint quadrants any region $R_i$ for which $P(R_i)$=FALSE.
- Merge any adjacent regions $R_j$ and $R_k$ for which $P(R_j \cup R_k)$ = TRUE.
- Stop when no further merging or splitting is possible.

# REGION-ORIENTED SEGMENTATION



(a)Original image (b)Result of split and merge procedure (c)Result of thresholding in a

# CONCLUSION

- Region and boundary information for the purpose of segmentation.
- Image segmentation is an essential step in most automatic graphic pattern recognition and scene analysis problems.
- One segmentation technique over another is dictated mostly by the peculiar characteristics of problem being measured.