18CSE357E- BIOMETRICS

Elective
Academic Year 2023-2024 (ODD Semester)

Dr. M.K.VIDHYALAKSHMI
Assistant Professor/CTech

Mobile: 9443223066
Email: vidhyalm1@srmist.edu.in
Room: BEL 404 Lab

Session – 5 & 6

- **Image Processing Basics**
  - What is an Image?
  - Image Acquition
  - Type, Point Operations
  - Geometric Transformations
  - First and Second order derivatives
- **Steps in Edge Detection, Smoothing, Enhancement, Thresholding & Localization**

Image Processing - Basics

Image

- An **image** may be defined as a two-dimensional function, f(x, y), where x and y are ***spatial*** (plane) 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.
- When x, y, and the amplitude values of f are all **finite, discrete quantities**, we call the image a ***digital image***.
- These elements are referred to as *picture elements*, *image elements*, *pels*, and ***pixels***
- An image can be defined by a two-dimensional array specifically arranged in rows and columns.

Image Processing - Basics

An Image is represented as a 2D Matrix

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,N-1) \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & \dots & f(M-1,N-1) \end{bmatrix}$$

Types of Images

Binary Images

- It is the simplest type of image. Also called as **silhouette** image
- It takes only two values i.e, **Black** and **White** or 0 and 1.
- The binary image consists of a 1-bit image and it takes only 1 binary digit to represent a pixel.
- Binary images are mostly used for general shape or outline.
- **For Example:** Optical Character Recognition (OCR).

Types of Images

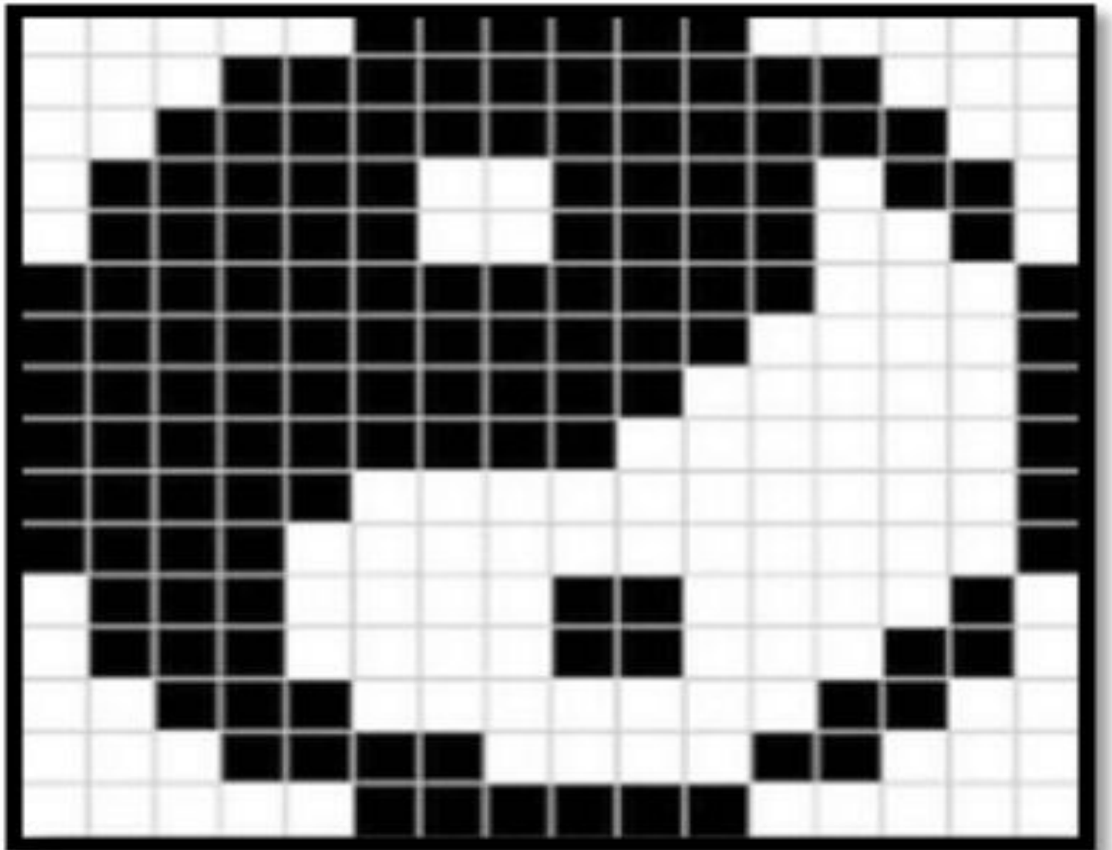


Fig. Binary image

Pros & Cons of Binary Images

SRM INSTITUTE OF SCIENCE & TECHNOLOGY (Deemed to be University u/s 3 of UGC Act, 1956)

Advantages

- **Easy to acquire**: simple digital cameras can be used or low-cost scanners, or thresholding may be applied to grey-level images.
- **Low storage**: no more than 1 bit/pixel
- **Simple processing**: the algorithms are much simpler than those applied to grey-level images.

Disadvantages

- **Limited application**: Application is restricted to tasks where internal detail is not required as a distinguishing characteristic.
- **Does not extend to 3D**:
- **Specialised lighting is required**: it is difficult to obtain reliable binary images without restricting the environment. The simplest example is an overhead projector or light box.

Thresholding

- Binary images are generated using **threshold operation.**
- In the simplest case, an image may consist of a single object or several separated objects of relatively high intensity, viewed against a background of relatively low intensity. This allows **figure/ground** separation by thresholding.
- In order to create the two-valued binary image a simple threshold may be applied so that all the pixels in the image plane are classified into **object** and **background** pixels.
- A binary image function can then be constructed such that pixels above the threshold are foreground (``1'') and below the threshold are background (``0'').

Types of Images

SRM INSTITUTE OF SCIENCE & TECHNOLOGY (Deemed to be University u/s 3 of UGC Act, 1956)

Gray-scale images

- Grayscale images are **monochrome** images, means they have only **one color**.
- Grayscale images do not contain any information about color.
- Each pixel determines available **different grey levels**.
- A normal grayscale image contains **8 bits/pixel** data, which has 256 different grey levels.
- In medical images and astronomy, 12 or 16 bits/pixel images are used.

Types of Images

Gray-scale images

Fig. Gray-scale image

Types of Images

Color images

- Colour images are **three band monochrome** images in which, **each band** contains a **different color** and the actual information is stored in the digital image.
- The color images contain grey level information in each spectral band.
- The images are represented as **red, green and blue (RGB images)**. Each color image has 24 bits/pixel means 8 bits for each of the three color band(RGB).

Types of Images

Color images

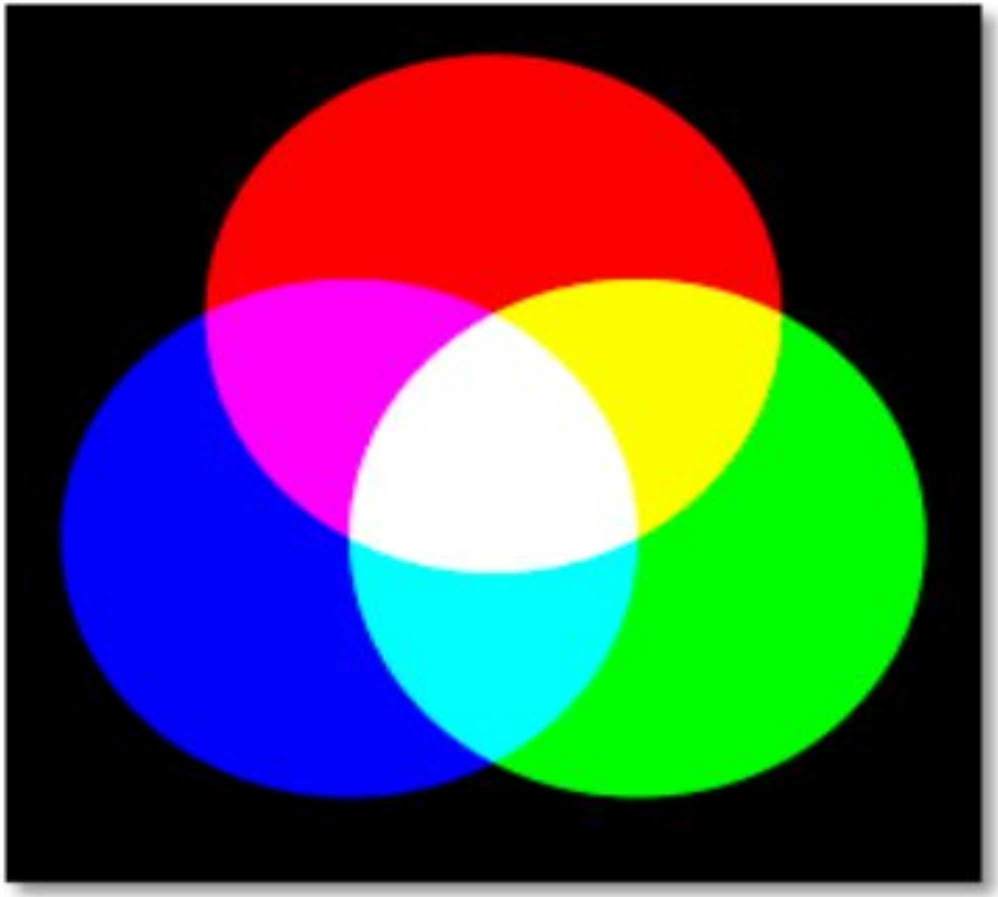


Fig. Colour image

Types of Images

Color images

8-bit color format

- **8-bit** color is used for storing image information in a computer's memory or in a file of an image.
- In this format, **each pixel represents 8 bits**.
- It has 0-255 range of colors, in which **0** is used for **black**, **255** for **white** and **127** for **gray** color.
- The 8-bit color format is also known as a **grayscale image**. Initially, it was used by the UNIX operating system.

Types of Images

Color images

8-bit color format

Types of Images

Color images

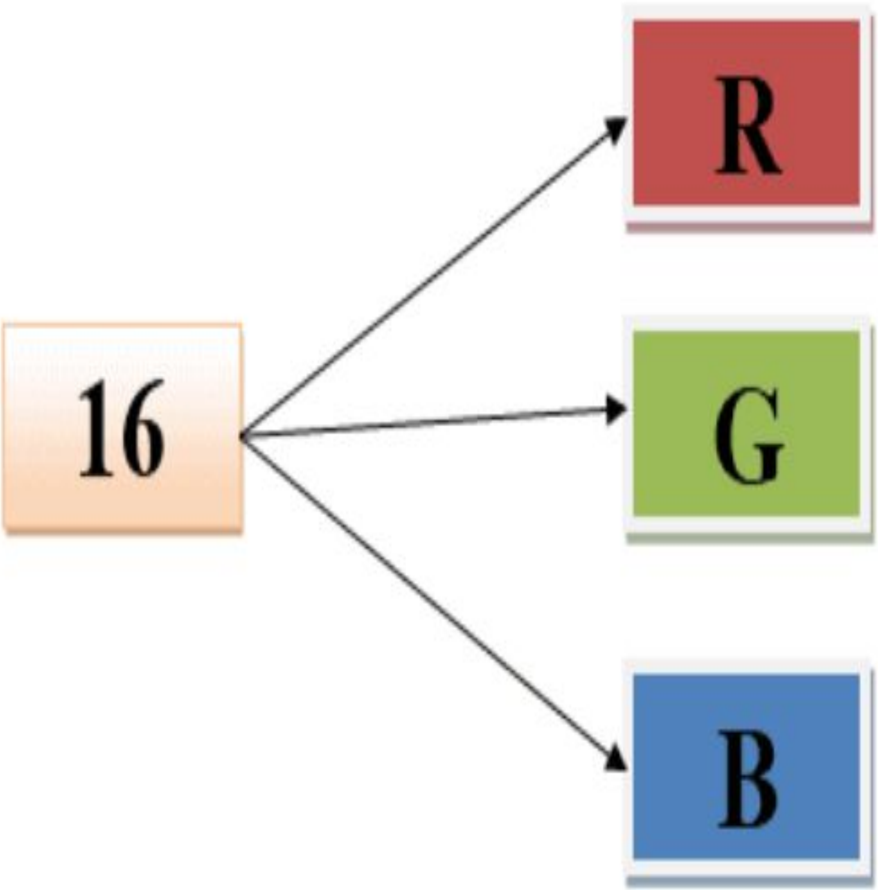
16-bit color format

- The 16-bit color format is also known as **high color** format.
- It has 65,536 different color shades. It is used in the system developed by Microsoft.
- The 16-bit color format is further divided into three formats which are Red, Green, and Blue also known as **RGB format**.
- In RGB format, there are 5 bits for R, 6 bits for G, and 5 bits for B. One additional bit is added in green because in all the 3 colors green color is soothing to eyes.

Types of Images

Color images

16-bit color format



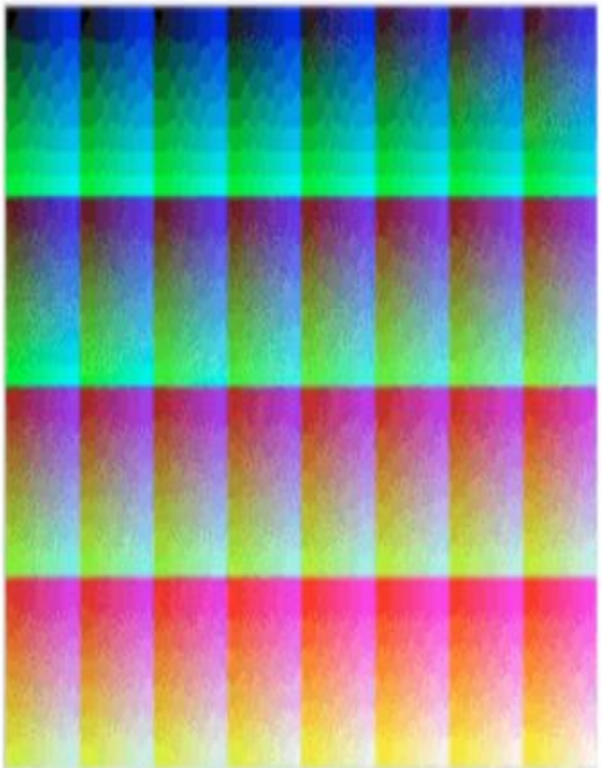


Fig: RGB 16 bits palette

Types of Images

Color images

24-bit color format

- The 24-bit color format is also known as the **true color** format.
- The 24-bit color format is also distributed in Red, Green, and Blue.
- As 24 can be equally divided on 8, so it is distributed equally between 3 different colors like 8 bits for R, 8 bits for G and 8 bits for B.

Types of Images

Color images

24-bit color format

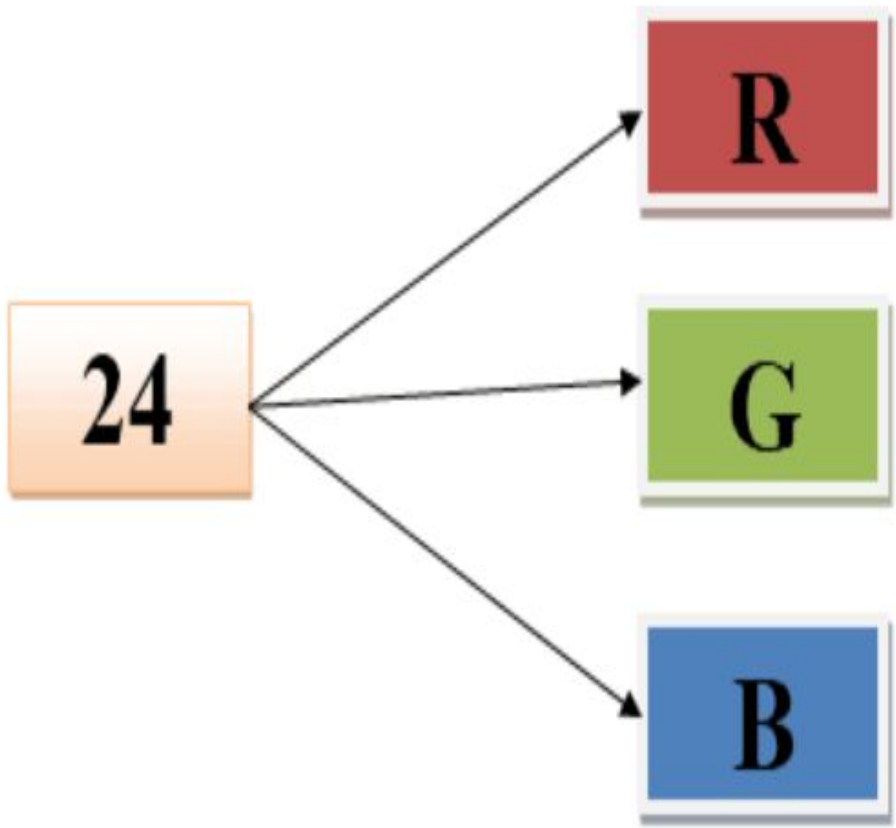


Fig: 16,777,216 colors

Types of Images

Three types of images:

– Binary images

$$g(x,y) \in \{0, 1\}$$

– Gray-scale images

$$g(x,y) \in C$$

typically $c=\{0,\dots,255\}$

– Color Images

three channels:

$$g_R(x,y) \in C \quad g_G(x,y) \in C \quad g_B(x,y) \in C$$

Image Processing - Basics

- Image Processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it.
- PURPOSE OF Image Processing:
- **Visualization** – Observe the objects that are not visible.
- **Image sharpening and restoration** – To create a better image.
- **Image retrieval** – Seek for the image of interest.
- **Measurement of pattern** – Measures various objects in an image.
- **Image Recognition** – Distinguish the objects in an image.

Image Processing - Basics

Fundamental Steps in Image Processing

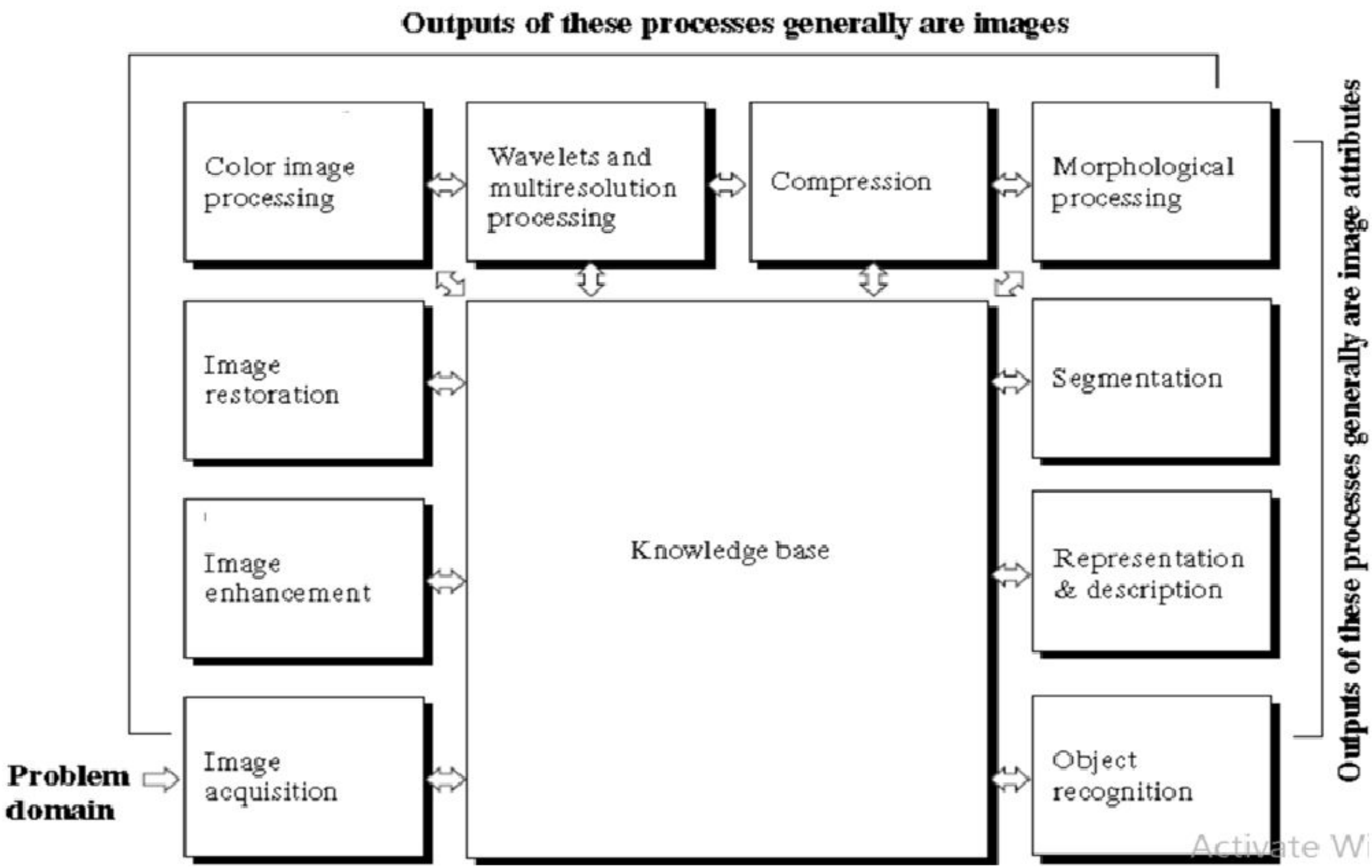


Image Processing - Basics

1.**ACQUISITION**– It could be as simple as being given an image which is in digital form. The main work involves:
a) Scaling
b) Color conversion(RGB to Gray or vice-versa)

2.**IMAGE ENHANCEMENT**– It is amongst the simplest and most appealing in areas of Image Processing it is also used to extract some hidden details from an image and is subjective.

3.**IMAGE RESTORATION**– It also deals with appealing of an image but it is objective(Restoration is based on mathematical or probabilistic model or image degradation).

Image Processing - Basics

4.**COLOR IMAGE PROCESSING**– It deals with pseudocolor and full color image processing color models are applicable to digital image processing.

5.**WAVELETS AND MULTI-RESOLUTION PROCESSING**– It is foundation of representing images in various degrees.

6.**IMAGE COMPRESSION**-It involves in developing some functions to perform this operation. It mainly deals with image size or resolution.

7.**MORPHOLOGICAL PROCESSING**-It deals with tools for extracting image components that are useful in the representation & description of shape.

Image Processing - Basics

8.**SEGMENTATION PROCEDURE**-It includes partitioning an image into its constituent parts or objects. Autonomous segmentation is the most difficult task in Image Processing.

9.**REPRESENTATION & DESCRIPTION**-It follows output of segmentation stage, choosing a representation is only the part of solution for transforming raw data into processed data.

10.**OBJECT DETECTION AND RECOGNITION**-It is a process that assigns a label to an object based on its descriptor

Image Processing - Basics

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

Benefits of Image Processing

- The digital image can be made available in any desired format (improved image, X-Ray, photo negative, etc)
- It helps to improve images for human interpretation
- Information can be processed and extracted from images for machine interpretation
- The pixels in the image can be manipulated to any desired density and contrast
- Images can be stored and retrieved easily
- It allows for easy electronic transmission of images to third-party providers

Point Operations

- Sometimes, pictures taken from the camera may provide poor quality.
- They were the result of the lighting conditions at that moment. So, we need to use image processing techniques to enhance image quality

Some of the image processing techniques are

  – Point Operation
  – Auto-contrast Adjustment
  – Modified Auto-Contrast
  – Histogram Equalization
  – Histogram Specification
  – Etc..

Point Operations

- **Point Operation** is the modification of the pixel value without changing in the size, geometry and local structure of the image.
- The new pixel value depends only on the previous value.
- They are mapped by a **function f(a),** if the function f() not depend on the coordinate, it is called "***global***" or "***homogeneous***" operation.
- Another one is called "***non-homogeneous***" point operation, if it depends on the coordinate.
- Non-homogeneous point operation is used to compensate for uneven lighting during image acquisition.

Point Operations

The common examples of homogeneous operation include:

- **Modifying contrast and brightness**
- **Limiting the Result by Clamping**
- **Inverting Image**
- **Threshold Operation**

Point Operations

- Here, each pixel value is replaced with a new value obtained from the old one.
- If we want to increase the brightness to stretch the contrast, we can simply multiply all pixel values by a scalar, say by 2, to double the range.
- Conversely, to reduce the contrast, we can divide all point values by a scalar.
- If the overall brightness is controlled by a level, l, (e.g., the brightness of global light) and the range is controlled by a gain, k, the brightness of the points in a new picture, N, can be related to the brightness in old picture, O, by

$$\mathbf{N}_{x,y} = k \times \mathbf{O}_{x,y} + l \quad \forall x, y \in 1, N$$

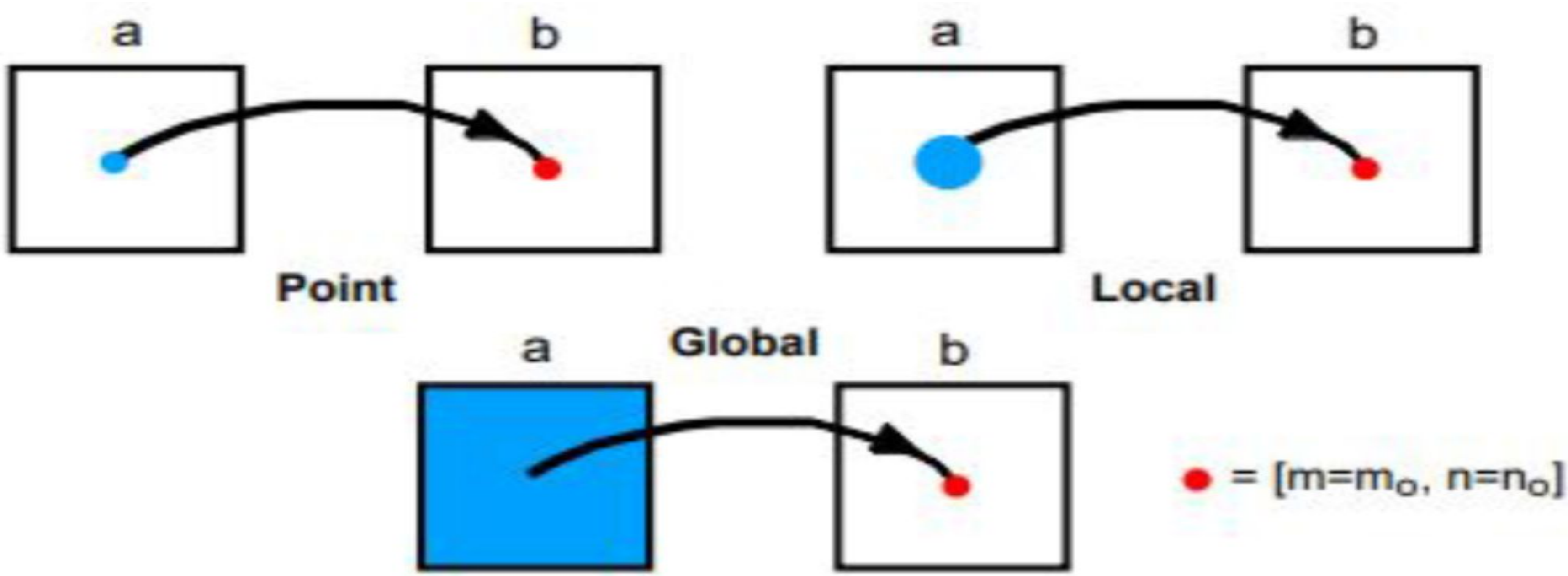
Point Operations

Types of operations

The types of operations that can be applied to digital images to transform an input image $a[m,n]$ into an output image $b[m,n]$ (or another representation) can be classified into three categories as shown in Table 2.

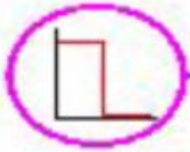
Operation	Characterization	Generic Complexity/Pixel
• *Point*	– the output value at a specific coordinate is dependent only on the input value at that same coordinate.	*constant*
• *Local*	– the output value at a specific coordinate is dependent on the input values in the *neighborhood* of that same coordinate.	P^2
• *Global*	– the output value at a specific coordinate is dependent on all the values in the input image.	N^2

Point Operations



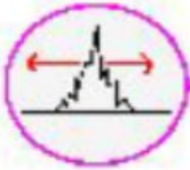
- Point Operation can be defined as follows:
- g(x, y) = T(f(x, y)) where
  - g (x, y) is the output image
  - T is an operator of intensity transformation
  - f (x, y) is the input image

Different Types of Point Operations

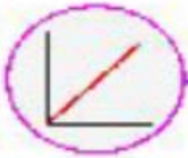


Thresholding - select pixels with given values to produce binary image

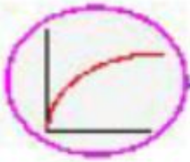
Adaptive Thresholding - like Thresholding except choose values locally



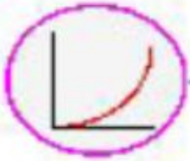
Contrast Stretching - spreading out graylevel distribution



Histogram Equalization - general method of modifying intensity distribution



Logarithm Operator - reducing contrast of brighter regions



Exponential/`Raise to Power' Operator - enhancing contrast of brighter regions

Thresholding

- In many vision applications, it is useful to **separate** out the regions of the image corresponding to **objects** in which we are **interested**, from the background.
- **Thresholding** provides an easy way to perform this **segmentation** on the basis of the different **intensities** or **colors** in the foreground and background regions of an image.
- It is useful to see what areas of an image consist of pixels whose values lie within a **specified range**, or band of **intensities** (or colors).

Point Operation - Thresholding

- It selects pixels that have a particular value or are within a specified range.
- It can be used to find objects within a picture if their brightness level (or range) is known.
- This implies that the object’s brightness must be known as well.
- There are two main forms: **uniform** and **adaptive thresholding**

Point Operation - Thresholding

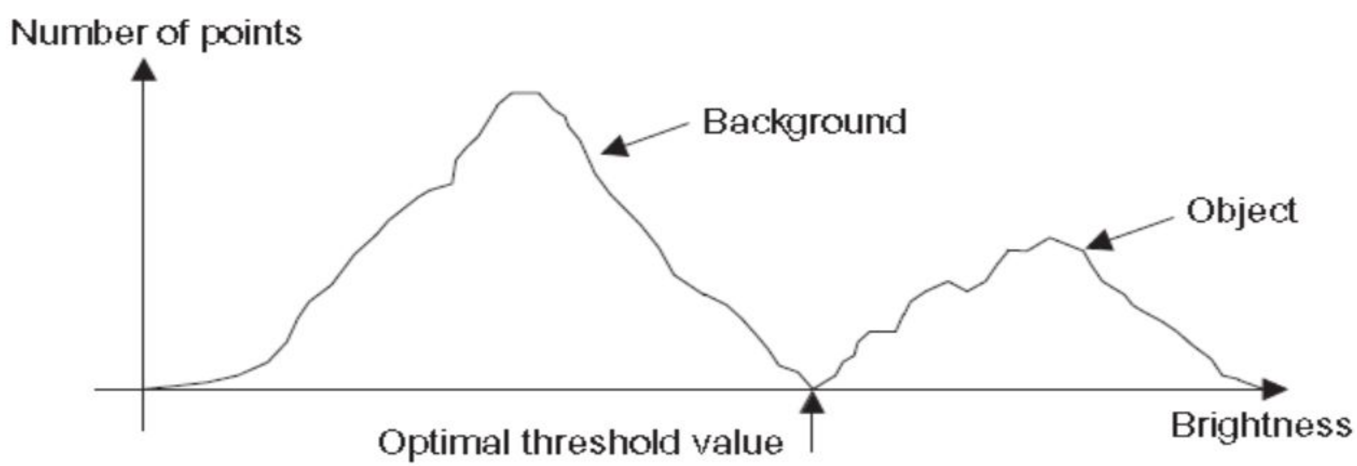
Point Operation - Thresholding

- In **uniform thresholding**, pixels **above** a specified level are set to **white**, those **below** the specified level are set to **black**
- It provides a way of **isolating** points of interest.
- It requires knowledge of the gray level, or the target features might not be selected in the thresholding process.
- If the level is not known, histogram equalization or intensity normalization can be used

Point Operation - Thresholding

Optimal or Adaptive Thresholding

- It seeks to select a value for the threshold that separates an object from its background.
- This suggests that the object has a different range of intensities to the background, in order that an appropriate threshold can be chosen



Interpolation

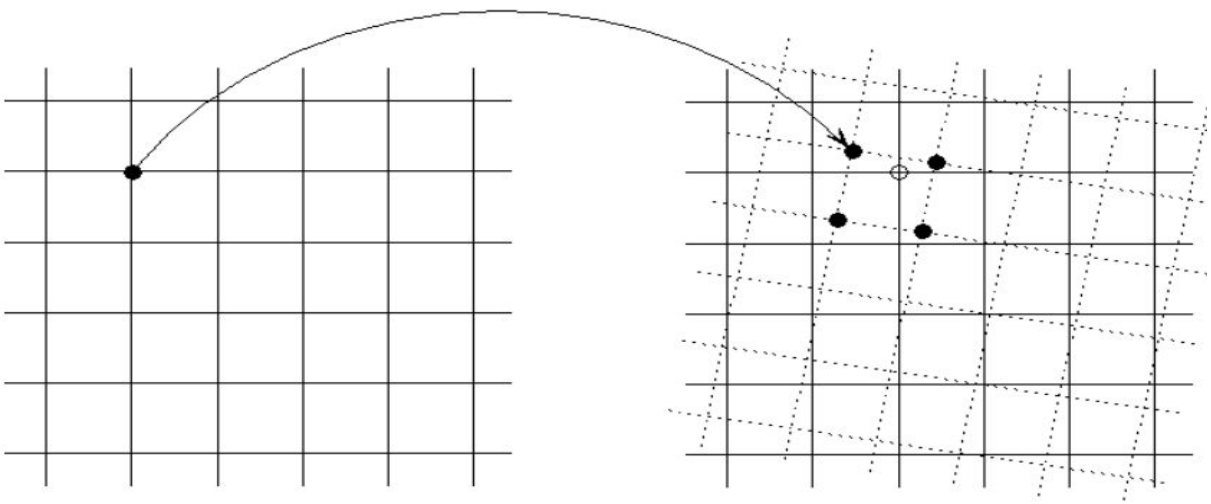
- Image interpolation occurs in all digital photos at some stage
- It happens anytime you resize or remap (distort) your image from one pixel grid to another.
- Image resizing is necessary when you need to increase or decrease the total number of pixels
- whereas remapping can occur under a wider variety of scenarios: correcting for lens distortion, changing perspective, and rotating an image.

Interpolation

- Even if the same image resize or remap is performed, the results can vary significantly depending on the interpolation algorithm.
- It is only an approximation, therefore an image will always lose some quality each time interpolation is performed.

Interpolation

- Interpolation is needed to find the value of the image at the **grid** points in the target coordinate system.
- The mapping T locates the grid points of A in the coordinate system of B, but those grid points are not on the grid of B.



Interpolation

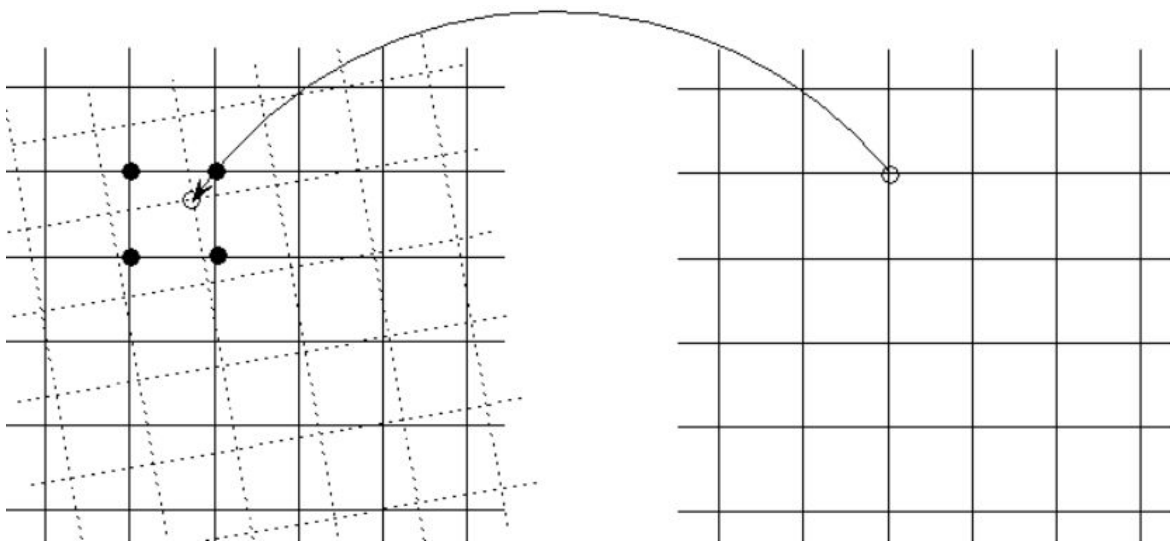
- To find the values on the grid points of B we need to interpolate from the values at the projected locations.
- Finding the closest projected points to a given grid point can be computationally expensive.
- Projecting the grid of B into the coordinate system of A maintains the known image values on a regular grid. This makes it simple to find the nearest points for each interpolation calculation

Interpolation

- Let Qg be the homogeneous grid coordinates of B and let H be the transformation from A to B. Then

$\mathbf{P} = \mathbf{H}^{-1}\mathbf{Q}_g$ represents the projection from B to A.

- We want to find the value at each point P given from the values on Pg, the homogeneous grid coordinates of A.



Histogram

- An **image histogram** is a type of **histogram** that acts as a graphical representation of the tonal distribution in a digital **image**. It plots the number of pixels for each tonal value. ... Thus, the **histogram** for a very dark **image** will have most of its data points on the left side and center of the graph.
- In an image processing context, the histogram of an image normally refers to a histogram of the pixel intensity values.
- This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image.

Histogram

Histogram of the above scenery

Histogram

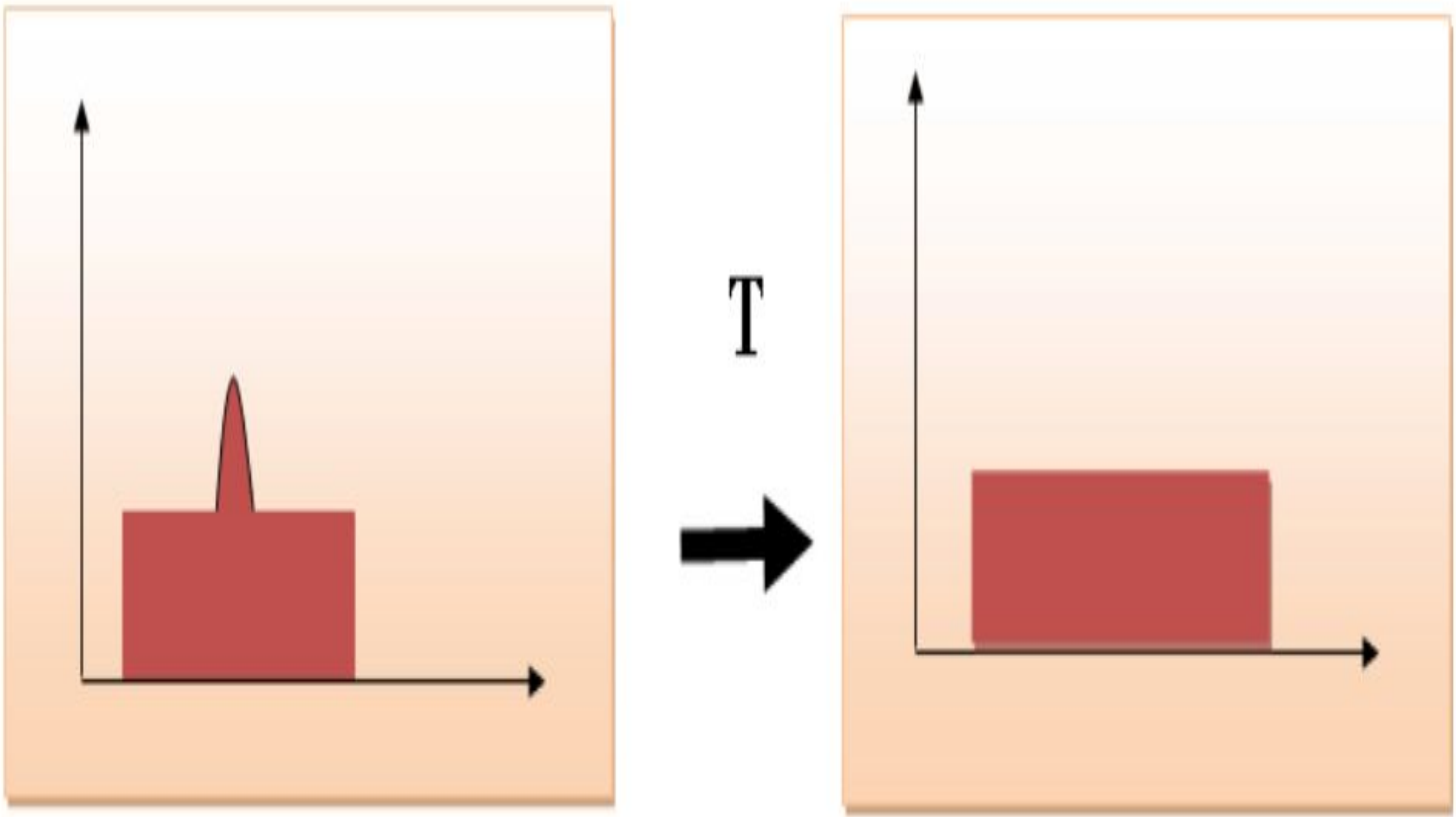
Histogram : is the discrete function $h(r_k)=n_k$, where r_k is the k^{th} gray level in the range of [0, L-1] and n_k is the number of pixels having gray level r_k.

Normalized histogram : is $p(r_k)=n_k/n$, for $k=0,1,\ldots,L-1$ and $p(r_k)$ can be considered to give an estimate of the probability of occurrence of gray level r_k.

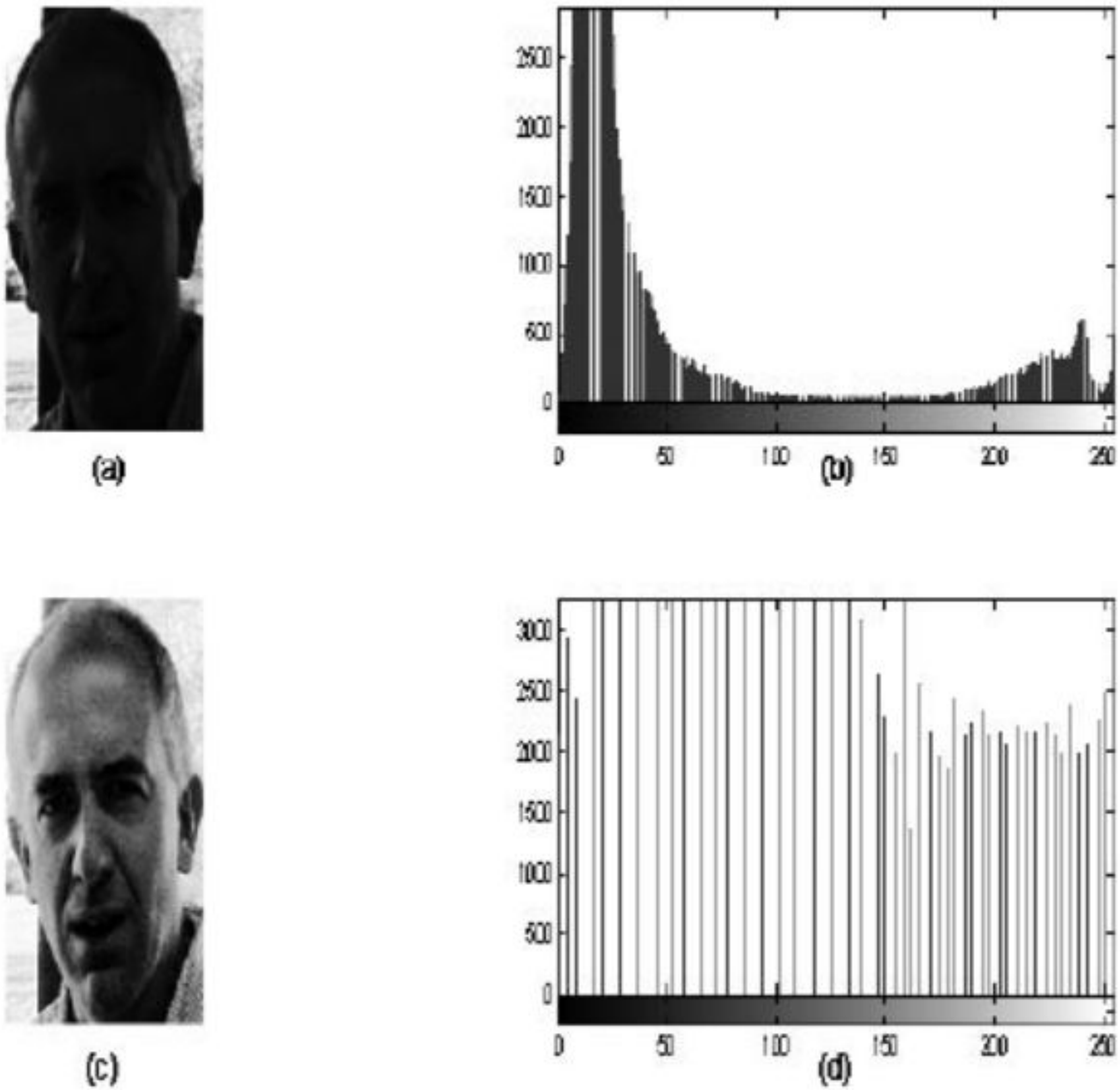
Histogram Equalization

- Histogram equalization is used for equalizing all the pixel values of an image.
- Transformation is done in such a way that uniform flattened histogram is produced.
- Histogram equalization increases the dynamic range of pixel values and makes an equal count of pixels at each level which produces a flat histogram with high contrast image.

Histogram Equalization



Histogram Equalization



(a) A face image from the CALTECH face database, (b) its histogram, (c) the equalized face image using HE, (d) and its respective histogram.

Applications of Histogram

- In digital image processing, histograms are used for simple calculations in software.
- It is used to analyze an image. Properties of an image can be predicted by the detailed study of the histogram.
- The brightness of the image can be adjusted by having the details of its histogram.
- The contrast of the image can be adjusted according to the need by having details of the x-axis of a histogram.
- It is used for image equalization. Gray level intensities are expanded along the x-axis to produce a high contrast image.
- Histograms are used in thresholding as it improves the appearance of the image

Sampling & Quantization

- In order to become suitable for digital processing, an image function f(x,y) must be digitized both spatially and in amplitude. Converting analog Image to Digital Image needs
- Sampling(digitizing of coordinates)
- Quantization(Digitizing of Amplitude)
- Image has 1.Coordinates 2.Amplitude/intensity
- The sampling rate determines the spatial resolution of the digitized image, while the quantization level determines the number of grey levels in the digitized image. A magnitude of the sampled image is expressed as a digital value in image processing. The transition between continuous values of the image function and its digital equivalent is called quantization.

Types of Neighbourhood

- Neighborhood operations play a key role in modern digital image processing. It is therefore important to understand how images can be sampled and how that relates to the various neighborhoods that can be used to process an image.
- Rectangular sampling – In most cases, images are sampled by laying a rectangular grid over an image as illustrated in Figure 1. This results in the type of sampling shown in Figure 3ab.
- • Hexagonal sampling – An alternative sampling scheme is shown in Figure 3c and is termed hexagonal sampling.

Types of Neighbourhood

- **Connectivity between pixels**
- It is an important concept in digital image processing.
- It is used for establishing boundaries of objects and components of regions in an image.
- Two pixels are said to be connected:
- if they are adjacent in some sense(neighbour pixels,4/8/m-adjacency)
- if their gray levels satisfy a specified criterion of similarity(equal intensity level)
- There are three types of connectivity on the basis of adjacency. They are:
- **a) 4-connectivity:** Two or more pixels are said to be 4-connected if they are 4-adjacent with each others.
- **b) 8-connectivity:** Two or more pixels are said to be 8-connected if they are 8-adjacent with each others.
- **c) m-connectivity:** Two or more pixels are said to be m-connected if they are m-adjacent with each others.

Types of Neighbourhood

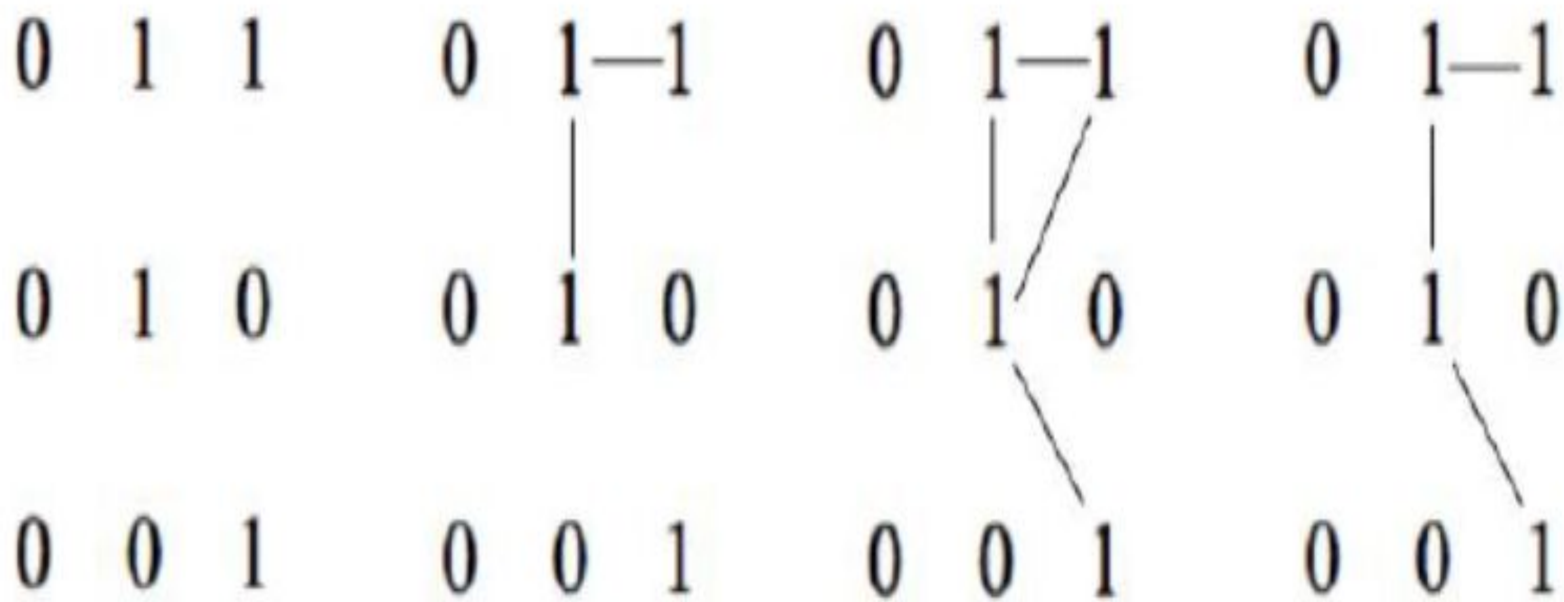


Fig: An arrangement of pixels

Fig: 4-connectivity of pixels

Fig: 8-connectivity of pixels

Fig: m-connectivity of pixels

Sampling Techniques

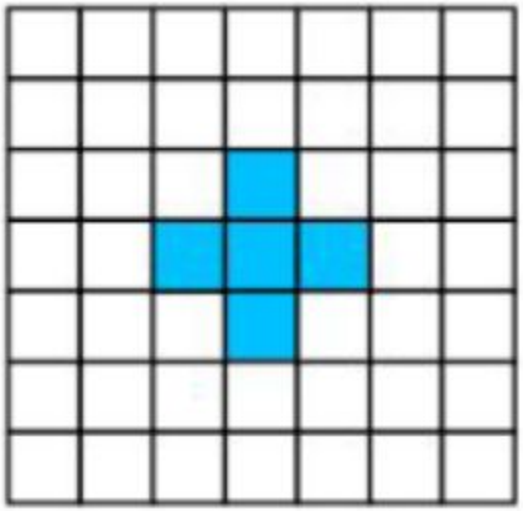


Figure 3a
Rectangular sampling
4-connected

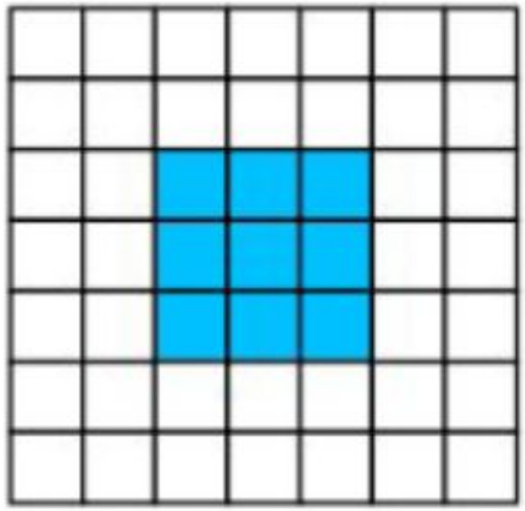


Figure 3b
Rectangular sampling
8-connected

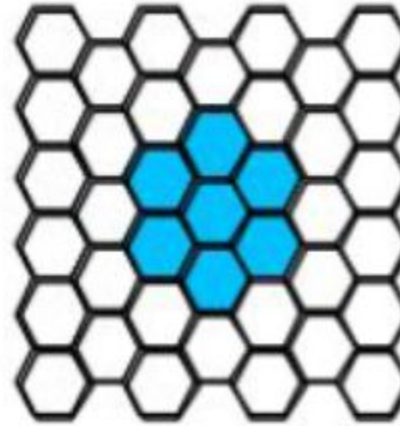
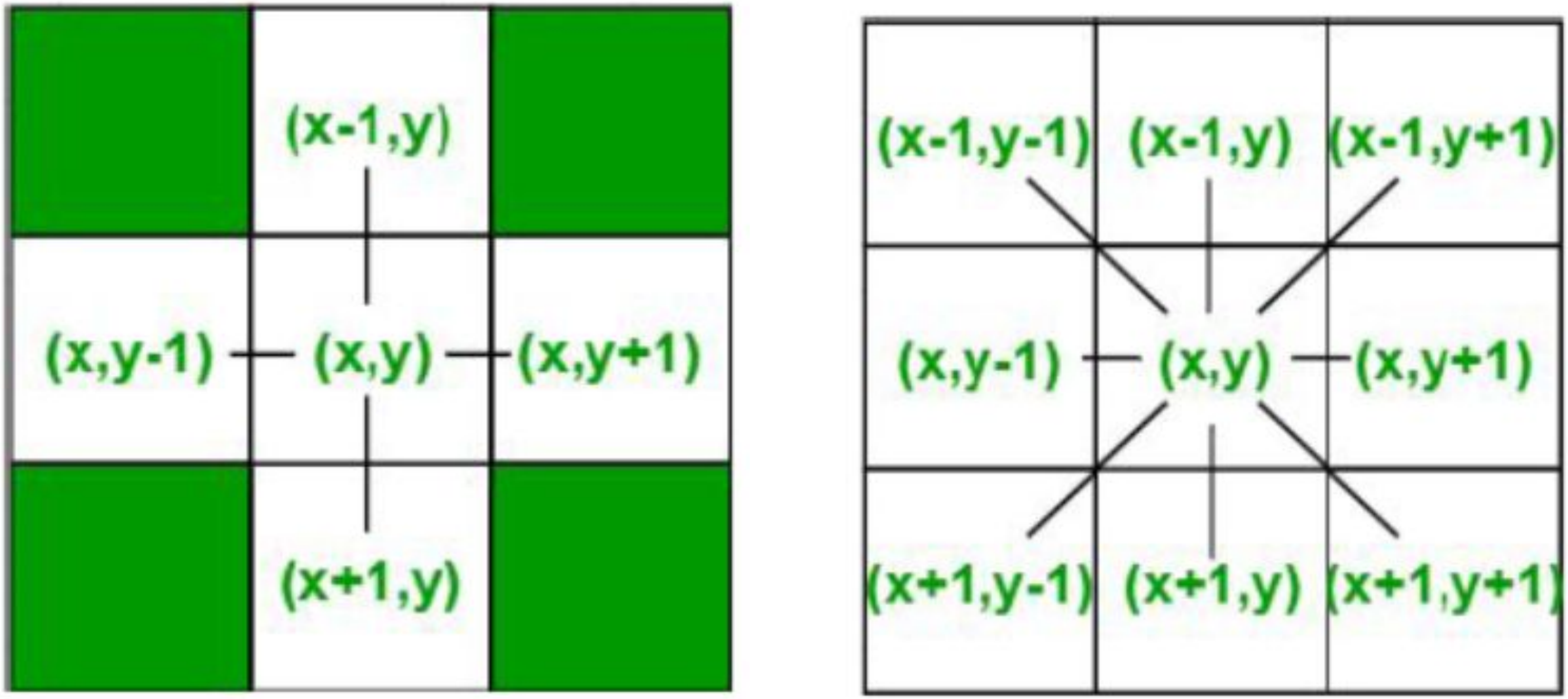


Figure 3c
Hexagonal sampling
6-connected

4 – Connected Vs 8 - Connected



Geometric Transformations

A **geometric operation** maps pixel information (i.e. the intensity values at each pixel location) in an input image to another location in an output image.

- Scale - change image content size
- Rotate - change image content orientation
- Reflect - flip over image contents
- Translate - change image content position
- Affine Transformation Affine transformation is **a linear mapping method that preserves points, straight lines, and planes**.

Geometric Transformations

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

- A set of image transformations where the geometry of image is changed without altering its actual pixel values are commonly referred to as "Geometric" transformation. In general, you can apply multiple operations on it, but, the actual pixel values will remain unchanged.

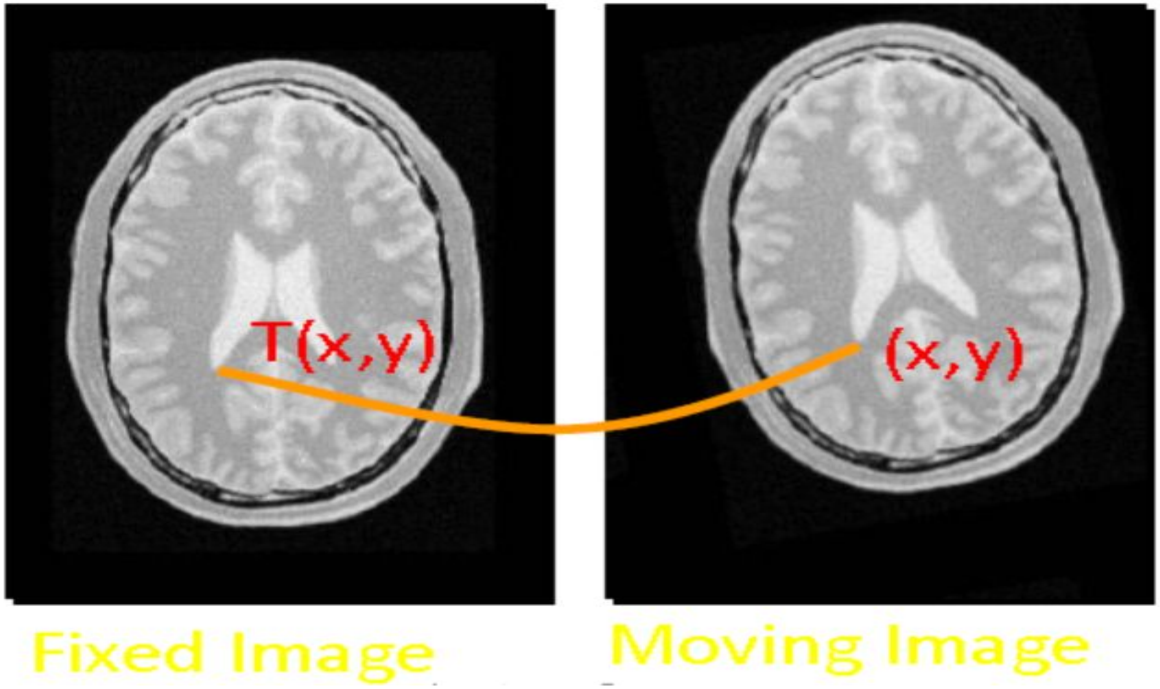
- In these transformations, pixel values are not changed, the positions of pixel values are changed.

- Geometric transformations are common in computer graphics, and are often used in image analysis.
- Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured.
- If one attempts to match two different images of the same object, a geometric transformation may be needed.

Geometric Transformations

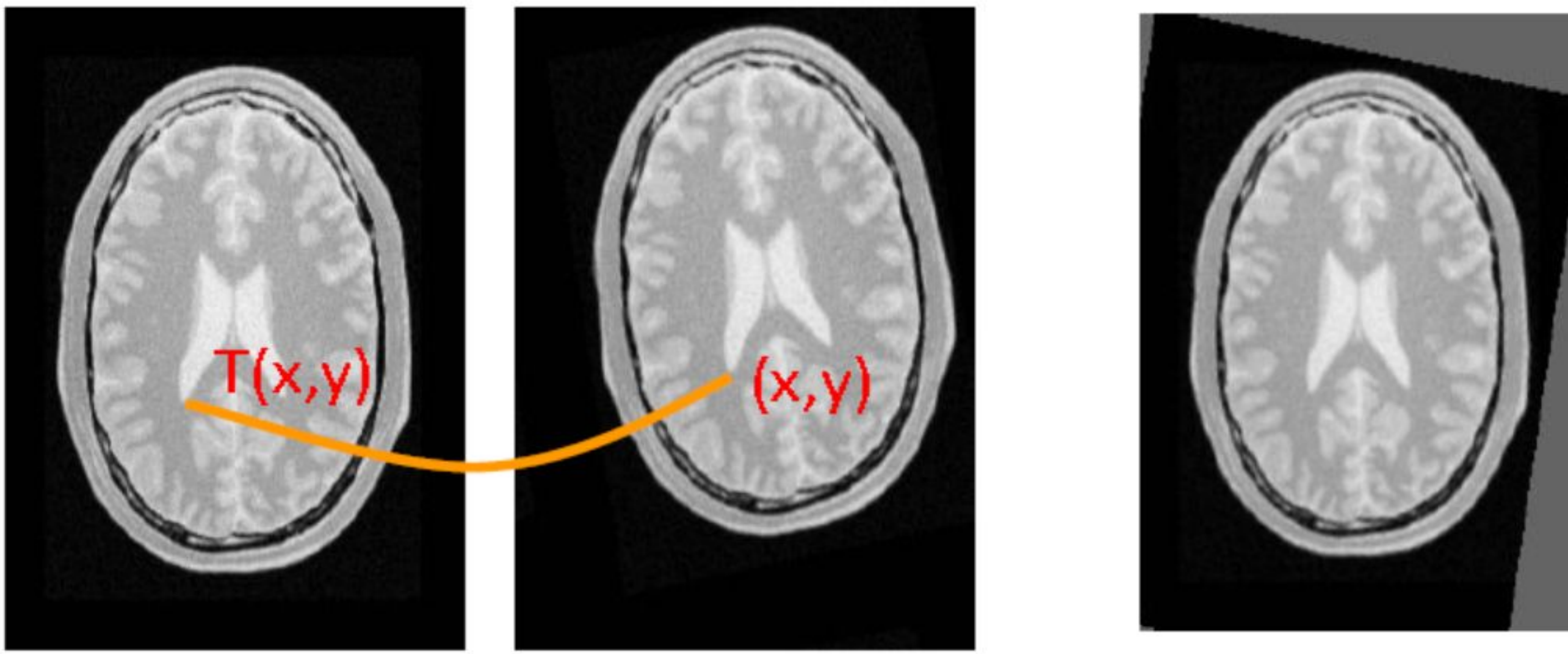
A geometric transform consists of two basic steps ...

– Step1: determining the pixel co-ordinate transformation
    - mapping of the co-ordinates of the moving image pixel to the point in the fixed image.



Geometric Transformations

– Step2: determining the brightness of the points in the digital grid of the transformed image.
    - brightness is usually computed as an interpolation of the brightnesses of several points in the neighborhood



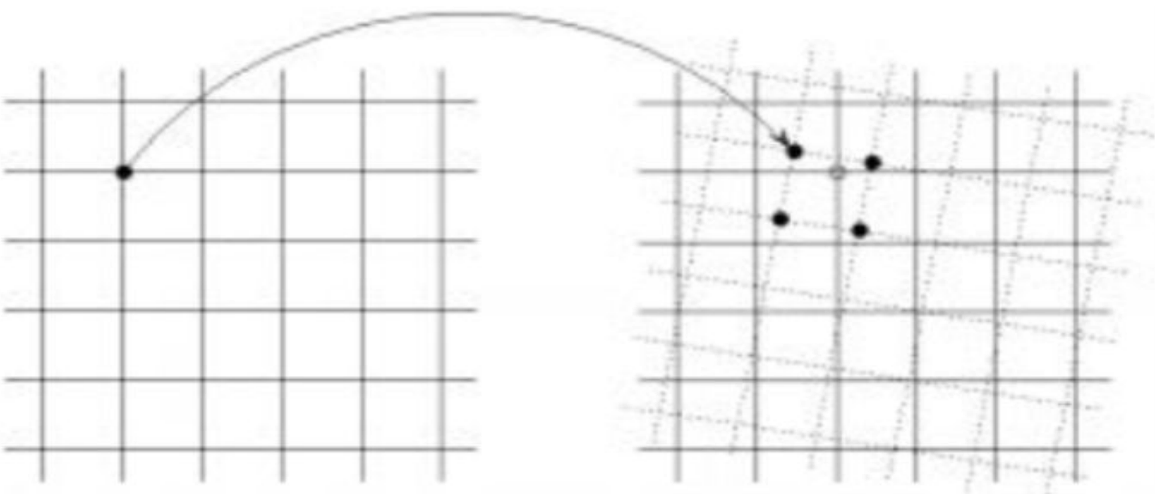
Fixed Image   Moving Image   xformed Moving Image

Geometric Transformations

Spatial Transformation

In a spatial transformation each point (x, y) of image A is mapped to a point (u, v) in a new coordinate system.

$$u = f_1(x, y)$$
$$v = f_2(x, y)$$



Mapping from (x, y) to (u, v) coordinates. A digital image array has an implicit grid that is mapped to discrete points in the new domain. These points may not fall on grid points in the new domain.

Major Geometric Transformations

Translation

- **Let us now look at the transformations in Fig.** and define their concrete mapping equations. Translation is simply a matter of shifting the image horizontally and vertically with a given off-set (measured in pixels) denoted Ax and Ay. For translation the mapping is thus defined as

$$x' = x + \Delta x \qquad \Rightarrow \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$
$$y' = y + \Delta y$$

- So if Ax = 100 and Ay = 100 then each pixel is shifted 100 pixels in both the x- and y-direction

Major Geometric Transformations

Scaling

- **When scaling an image,** it is made smaller or bigger in the x- and/or y-direction.
- Say we have an image of size 300 x 200 and we wish to transform it into a 600 x 100 image. The x-direction is then scaled by: 600/300 = 2.
- We denote this the x-scale factor and write it as Sx = 2. Similarly Sy = 100/200 = 1/2. Together this means that the pixel in the image f(x, y) at position (x, y) = (100,100) is mapped to anew position in the image g(x',y'), namely (x ' ,y') = (100 · 2,100 · 1/2) = (200, 50).
- In general, scaling is expressed as

$$\begin{aligned} x' &= x \cdot S_x \\ y' &= y \cdot S_y \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Major Geometric Transformations

Rotation

- When rotating an image, as illustrated in Fig. we need to define the amount of rotation in terms of an angle.
- We denote this angle θ meaning that each pixel in f(x, y) is rotated θ degrees.
- The transformation is defined as

$$x' = x \cdot \cos\theta - y \cdot \sin\theta$$
$$y' = x \cdot \sin\theta + y \cdot \cos\theta$$

$$\Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Major Geometric Transformations

Shearing

- To shear an image means to shift pixels either horizontally, Bx, or vertically, By.
- The difference from translation is that the shifting is not done by the same amount, but depends on wherein the image a pixel is.
- The transformation is defined as

$$\begin{aligned} x' &= x + y \cdot B_x \\ y' &= x \cdot B_y + y \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & B_x \\ B_y & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

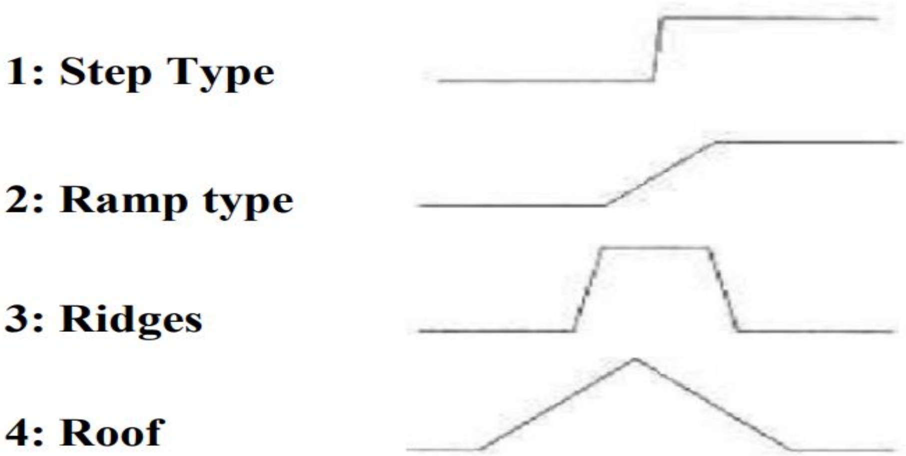
Edge Detection

- **Edges**: are abrupt changes in intensity, discontinuity in image brightness or contrast; usually edges occur on the boundary of two regions.
- **Edge detection**: is an image processing technique for finding the **boundaries of objects within images**.
- It works by detecting discontinuities in brightness.
- Edge detection is used for **image segmentation** and **data extraction** in areas such as image processing, computer vision, and machine vision.

Edge Detection

Why we use edge detection?

- Reduce unnecessary information in the image while preserving the structure of the image.
- Extract important features of an image such as corners, lines, and curves.
- Edges provide strong visual clues that can help the recognition process.
- Type of edges



Edge Detection

Edge Detection Steps

- Smoothing: Noise Reduction.
- Enhancement: Edge sharpening.
- Detection: Which to discard and which to maintain.
  - Thresholding.
- Localization: determine the exact location of an edge.
  - Edge thinning and linking are usually required in this step.

Edge Detection Methods

There are various methods, and the following are some of the most commonly used methods-

- Roberts edge detection
- Prewitt edge detection
- Sobel edge detection
- Laplacian edge detection
- Canny edge detection

First Order & Second Order Derivatives

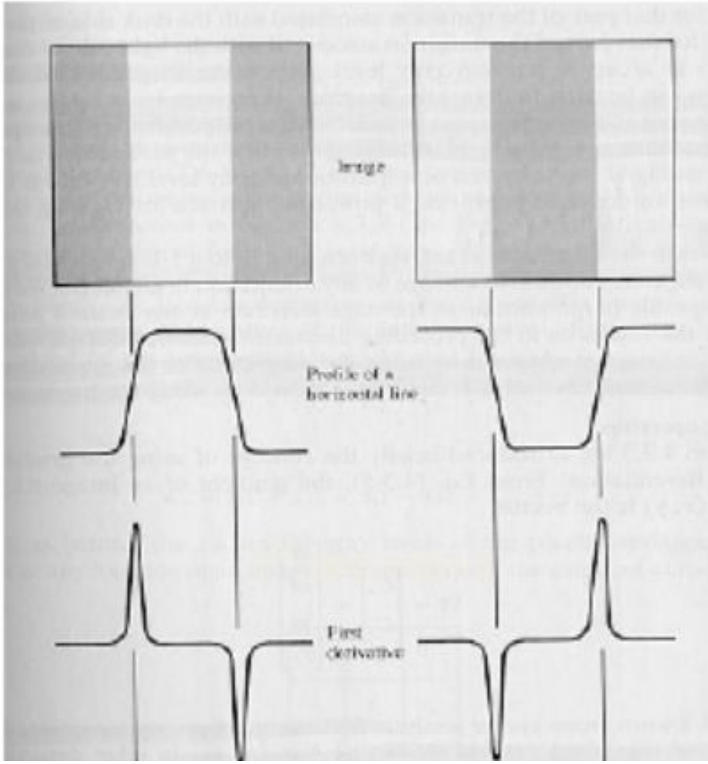
- This method we take the 1 st **derivative** of the **intensity value** across the image and **find points** where the derivative is **maximum** then the edge could be **located**.
- The **gradient** is a vector, whose components **measure** how rapid **pixel value are changing** with distance in the x and y direction.
- **Magnitude** of Gradient vector is used for **implementation** of **first order derivative** in image processing
- **Laplacian** is for **second order** implementation in image processing.

Edge Detection using Derivatives

- Often, points that lie on an edge are detected by:

(1) Detecting the local **maxima** or **minima** of the first derivative.

(2) Detecting the **zero-crossings** of the second derivative.



1st & 2nd Order Derivatives

Gradient Estimation

Estimation of the intensity gradient at a pixel in the x and y direction, for an image f, is given by:

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

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

We can introduce noise smoothing by convoluting with a low pass filter (e.g. mean, Gaussian, etc)

The gradient calculation (g_x,g_y) can be expressed as:

$$g_x = h_x * f(x, y)$$

$$g_y = h_y * f(x, y)$$

Edge Detection Using First Derivative (Gradient)

The first derivate of an image can be computed using the gradient:

$$\nabla f \quad grad(f) = \begin{pmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{pmatrix}$$

The gradient is a vector which has magnitude and direction:

Edge Detection Using First Derivative (Gradient)

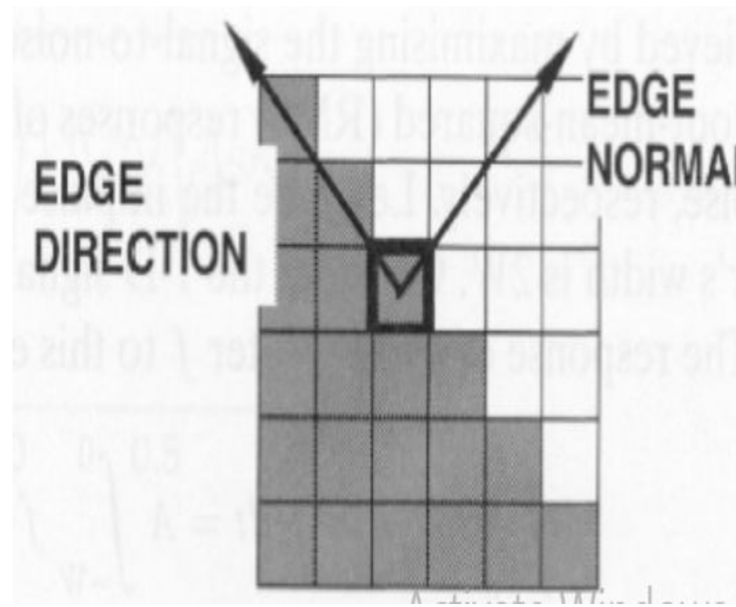
Magnitude: indicates edge strength.

Direction: indicates edge direction.

– i.e., perpendicular to edge direction

$$magnitude(grad(f)) = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}$$

$$direction(grad(f)) = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$



Session 7

- *Robert's method,*
- *Sobel's method,*
- *Perwitts*

Roberts Edge Detection

- The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image.
- It highlights regions of high spatial frequency which often correspond to edges.
- The input to the operator is a **grayscale image**, as is the output.
- **Pixel values** at each point in the output represent the **estimated absolute magnitude** of the **spatial gradient** of the input image at that point

Roberts Edge Detection

- **Robert Operator**: It is a **gradient-based** operator computes the **sum of squares** of the **differences** between **diagonally adjacent pixels** in an image through **discrete differentiation**.
- Then the gradient approximation is made. It uses the following 2 x 2 kernels or masks –

+1	0
0	-1

Gx

0	+1
-1	0

Gy

The Roberts cross operator

+1	0
0	-1

Gx

0	+1
-1	0

Gy

Following is the gradient magnitude:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

As it is much faster to compute An approximate magnitude is computed:

$$|G| = |Gx| + |Gy|$$

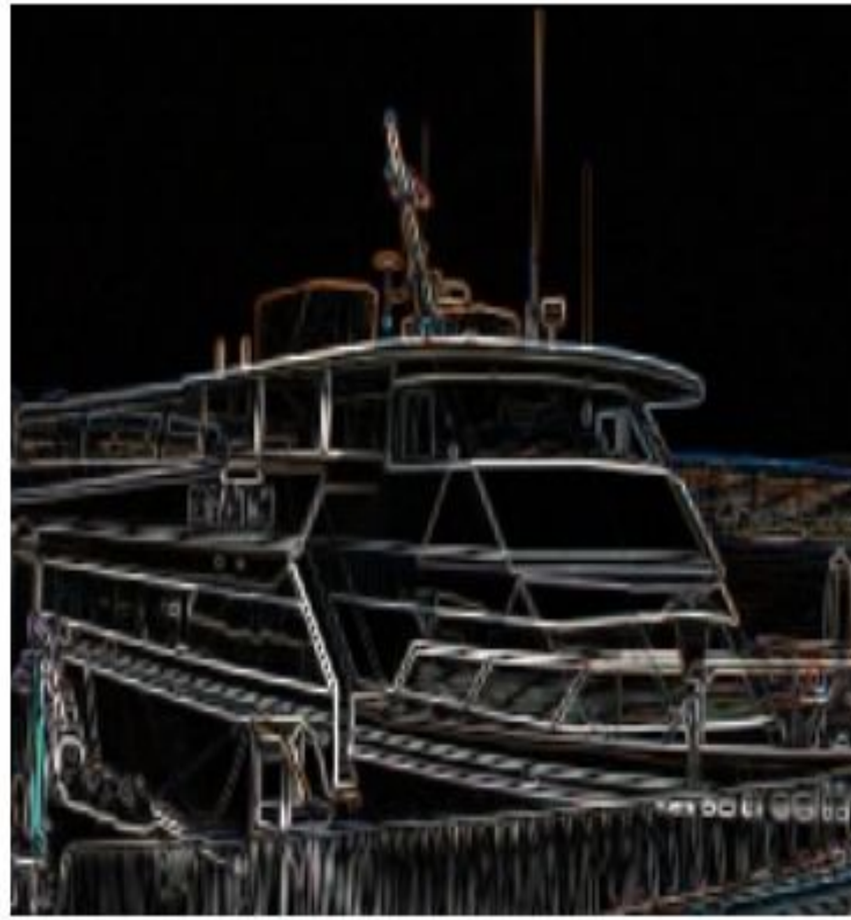
Roberts Edge Detection

1: Input – Read an image
2: Convert the true-color RGB image to the grayscale image
3: Convert the image to double4: Pre-allocate the filtered_image matrix with zeros
5: Define Robert Operator Mask
6: Edge Detection Process (Compute Gradient approximation and magnitude of vector)
7: Display the filtered image
8: Thresholding on the filtered image
9: Display the edge-detected image

Roberts Edge Detection

Source

Roberts Edge



Roberts Edge Detection

Advantages:

1. Detection of edges and orientation are very easy
2. Diagonal direction points are preserved

Limitations:

1. Very sensitive to noise
2. Not very accurate in edge detection

Sobel Edge Detection Method

- The Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial frequency that correspond to edges.
- It is used to find the **approximate absolute gradient magnitude** at each point in an input grayscale image.
- The operator consists of a pair of **3×3 convolution** kernels. One kernel is simply the other rotated by 90°.
- This is very similar to the Roberts Cross operator.

Sobel Edge Detection Method

- The Sobel Operator

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

Gx

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

Gy

Sobel Edge Detection Method

To compute G_x and G_y we move the appropriate kernel (window) over the input image, computing the value for one pixel and then shifting one pixel to the right. Once the end of the row is reached, we move down to the beginning of the next row.

The example below shows the calculation of a value of G_x:

a11	a12	a13	…	
a21	a22	a23	…	
a31	a32	a33	…	
…	…	…	…	

Input image

kernel =

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

b11	b12	b13	…	
b21	b22	b23	…	
b31	b32	b33	…	
…	…	…	…	

Output image (G_x)

$$b_{22} = a_{13} - a_{11} + 2a_{23} - 2a_{21} + a_{33} - a_{31}$$

Sobel Edge Detection Method

SRM INSTITUTE OF SCIENCE & TECHNOLOGY (Deemed to be University u/s 3 of UGC Act, 1956)

At each pixel in the image, the gradient approximations given by G_x and G_y are combined to give the gradient magnitude, using:

$$G = \sqrt{G_x^2 + G_y^2}$$

The gradient's direction is calculated using:

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

A Θ value of 0 would indicate a vertical edge that is darker on the left side.

Sobel Edge Detection Method

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

- It works by calculating the **gradient of image intensity** at each pixel within the image.
- It finds the direction of the **largest increase** from light to dark and the rate of change in that direction.
- The result shows how abruptly or smoothly the image changes at each pixel, and therefore how likely it is that that pixel represents an edge. It also shows how that edge is likely to be oriented.
- The result of applying the filter to a pixel in a region of constant intensity is a zero vector.
- The result of applying it to a pixel on an edge is a vector that points across the edge from darker to brighter values.

Sobel Edge Detection Method

Source Image

Sobel

The Prewitt operator

- Prewitt operator is a **differentiation operator**.
- Prewitt operator is used for calculating the approximate gradient of the image intensity function.
- In an image, at each point, the Prewitt operator results in gradient vector or normal vector.
- In Prewitt operator, an image is convolved in the horizontal and vertical direction with small, separable and integer-valued filter. It is inexpensive in terms of computations.

$$h_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \qquad h_2 = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

Templates of Prewitt operator

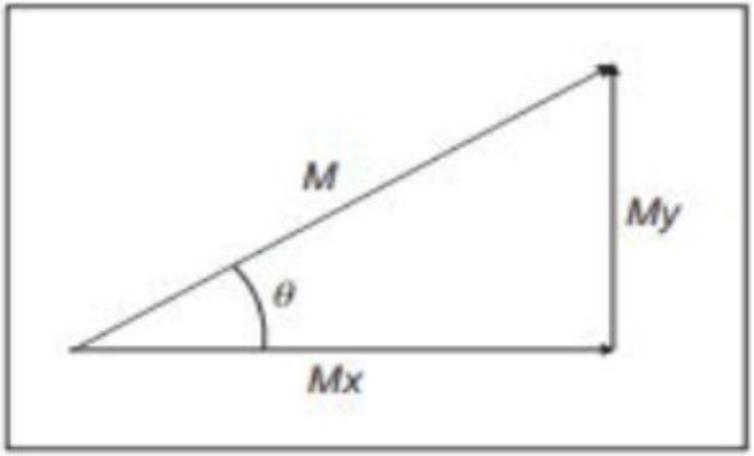


FIGURE 4.7

Edge detection in vectorial format.

1	0	–1
1	0	–1
1	0	–1

(a) *Mx*

1	1	1
0	0	0
–1	–1	–1

(b) *My*

FIGURE 4.8

Templates for Prewitt operator.

Summary

- **First Order Derivative Edge Detection.** Generally, the first order derivative operators are very sensitive to noise and produce thicker edges.
- a.1) **Roberts** filtering: diagonal edge gradients, susceptible to fluctations. Gives no information about edge orientation and works best with binary images.
- a.2) **Prewitt** filter: The Prewitt operator is a discrete differentiation operator which functions similar to the Sobel operator, by computing the gradient for the image intensity function. Makes use of the maximum directional gradient. As compared to Sobel, the Prewitt masks are simpler to implement but are very sensitive to noise.
- a.3) **Sobel** filter: Detects edges are where the gradient magnitude is high.This makes the Sobel edge detector more sensitive to diagonal edge than horizontal and vertical edges.
- Sobel and Prewitt methods are very effectively providing good edge maps

Second Order Derivatives

- *Laplacian of Gaussian,*
- *Zero crossing*

Laplacian of Gaussian

- The Laplacian is a 2-D **isotropic** measure of the 2nd spatial derivative of an image.
- The Laplacian of an image highlights **regions of rapid intensity change** and is often used for edge detection.
- The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian smoothing filter in order to reduce its sensitivity to noise, and hence the two variants will be described together here
- The operator normally takes a single graylevel image as input and produces another graylevel image as output.

Second Order Derivatives

- A very popular second order operator is the Laplacian operator.
- The Laplacian of a function f(x,y), is defined by:

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

- The Template for LT is

0	1	0
1	-4	1
0	1	0

- In Laplacian, the input image is represented as a set of discrete pixels. So, **discrete convolution kernel** which can approximate second derivatives in the definition is found.

Second Order Derivatives

Because derivatives of any order are linear operations, the Laplacian is a linear operator.

To express this equation in discrete form, in the *x-direction, we have*

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

and, similarly, in the *y-direction we have*

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

Therefore, discrete Laplacian of two variables is

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

Templates for LT Operator

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

-1	2	-1
2	-4	2
-1	2	-1

There are **disadvantages** to the use of second order derivatives.

- Second derivatives will exaggerated noise twice as much.
- No directional information about the edge is given.
  - The problems that the presence of noise causes when using edge detectors means we should try to reduce the noise in an image prior to or in conjunction with the edge detection process.

Gaussian

- Another smoothing method is ***Gaussian smoothing***
- Gaussian smoothing is performed by convolving an image with a Gaussian operator which is defined below.
- By using Gaussian smoothing in conjunction with the Laplacian operator, or another Gaussian operator, it is possible to detect edges.
- Lets look at the Gaussian smoothing process first.
- The ***Gaussian distribution*** function in two variables, *g*(*x*,*y*), is defined by

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

- where sigma is the standard deviation representing the width of the Gaussian distribution

Gaussian

Just like in the case of the Sobel Operator, we cannot calculate the second derivative directly because pixels in an image are discrete. We need to approximate it using the convolution operator. The two most common kernels are:

0	-1	0
-1	4	-1
0	-1	0

LOG

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

Detection of Discontinuities Gradient Operators (laplacian of guassian: LOG)

* Consider the function:

A Gaussian function

$$G(r) = -e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad \sigma : \text{the standard deviation}$$

* The Laplacian of G is

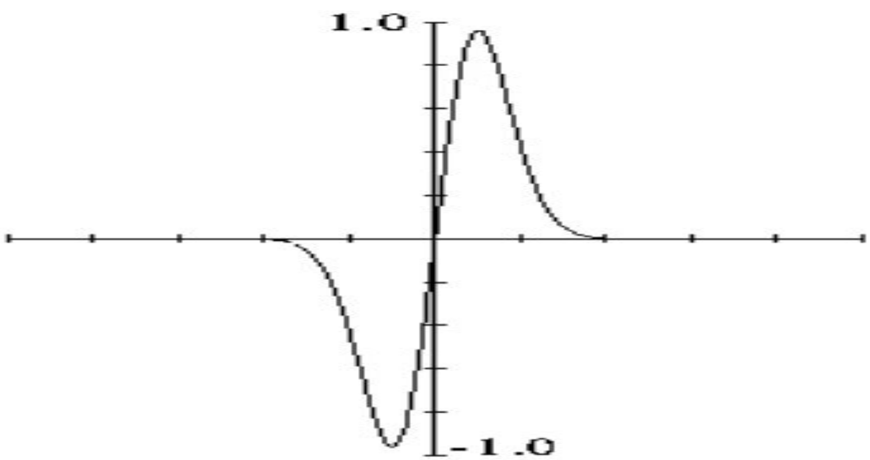
$$\nabla^2 G(r) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

The Laplacian of a Gaussian (LoG)

* The Laplacian of a Gaussian sometimes is called the **Mexican hat function.** It also can be computed by **smoothing the image with the Gaussian smoothing mask,**

The LOG operator

- This method of edge detection was first proposed by Marr and Hildreth at MIT who introduced the principle of the ***zero-crossing*** method.
- The basic principle of this method is to find the position in an image where the **second derivatives become zero**.
- These positions correspond to edge positions as shown in



(C) LOG of f(x)

LOG

-1	-1	-1
-1	8	-1
-1	-1	-1

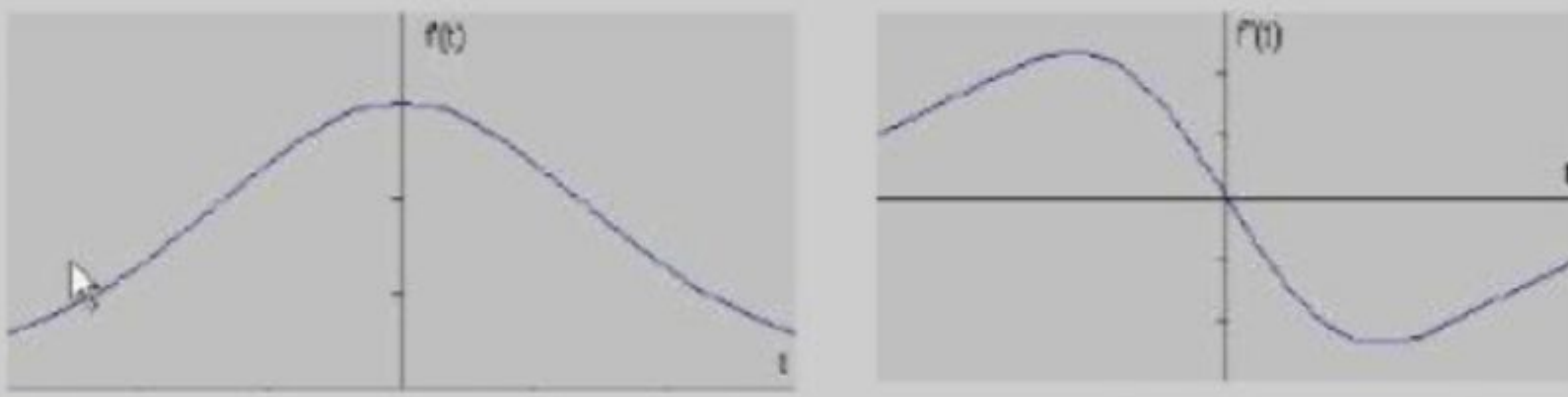
Calculating just the Laplacian will result in a lot of noise, so we need to convolve a Gaussian smoothing filter with the Laplacian filter to reduce noise prior to computing the second derivatives. The equation that combines both of these filters is called the Laplacian of Gaussian and is as follows:

$$LoG = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Zero Crossing

Zero Crossing based Edge Detection

- Indicates the presence of a maxima.
- Pixel value passes through zero (changes its sign).



The LOG operator

- Zero Crossing based Edge Detection

Defined as:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- The Gaussian function firstly smooths or blurs any step edges.
- The second derivative of the blurred image is taken; it has a zero-crossing at the edge.

NOTE: Blurring is advantageous here:

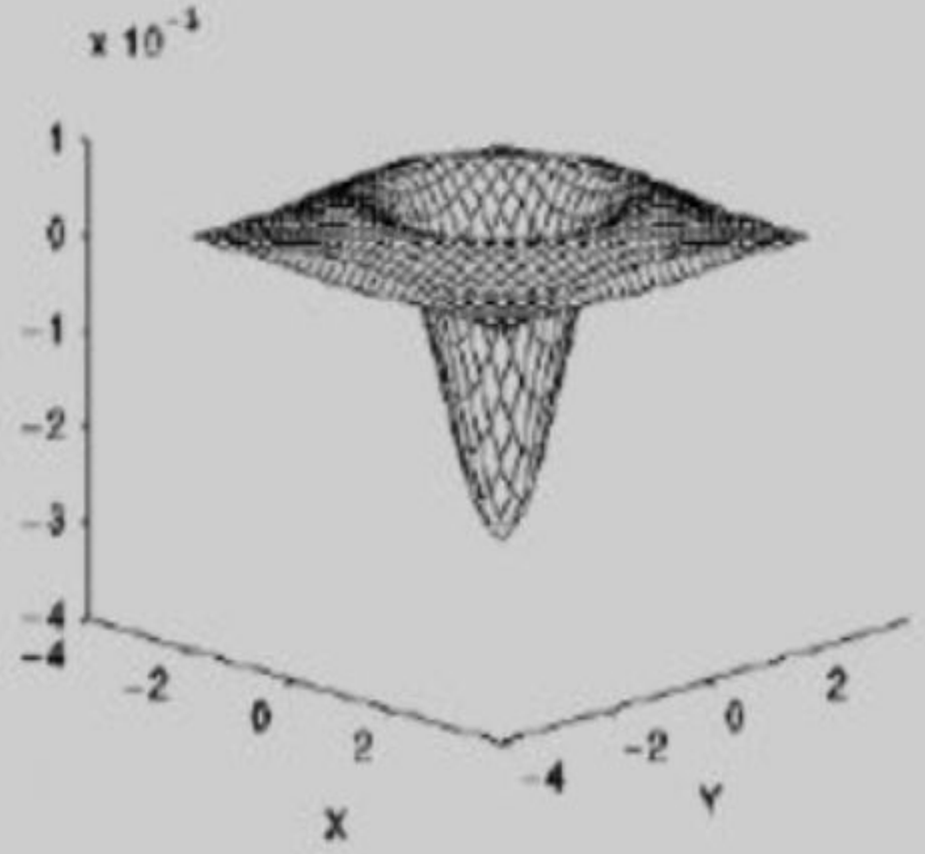
- Laplacian would be infinity at (unsmoothed) step edge.
- Edge position still preserved

The LOG based Edge Detection

- Laplacian of Gaussian

1	1	1
1	8	1
1	1	1

-1	2	-1
2	4	2
-1	2	-1



The LOG based Edge Detection

- LOG operator is still susceptible to noise, but the effects of noise can be reduced by ignoring zero-crossings produced by small changes in image intensity.
- LOG operator gives edge direction information as well as edge points - determined from the direction of the

Steps:

- Smoothing: Gaussian filter
- Enhance edges: Laplacian operator
- Zero crossings denote the edge location
- Use linear interpolation to determine the sub-pixel location of the edge

The LOG based Edge Detection

Computationally cheaper to implement since we can combine the two filters into one filter but it.

Doesn't provide information about the direction of the edge.

Probability of false and missing edges remain.

Localization is better than Gradient Operators

Session 8 and 9

- Low Level Features Extraction
- Describing Image Motion
- High Level Features Extraction
- Template Matching

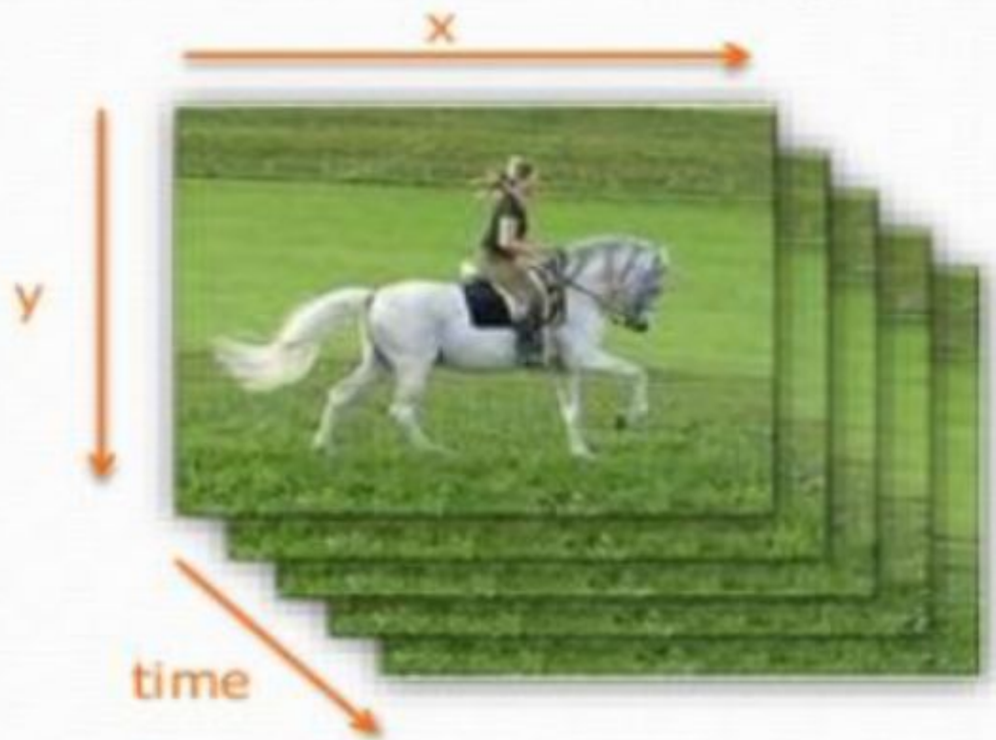
- Hough Transform for Lines
- Hough Transforms for Circles and Elipses

Low Level Feature Extraction

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

Describing image motion

- Motion detection based on comparing of the current video frame with one from the previous frames.



Low Level Feature Extraction

- The simplest way we can detect *motion* is by image differencing.

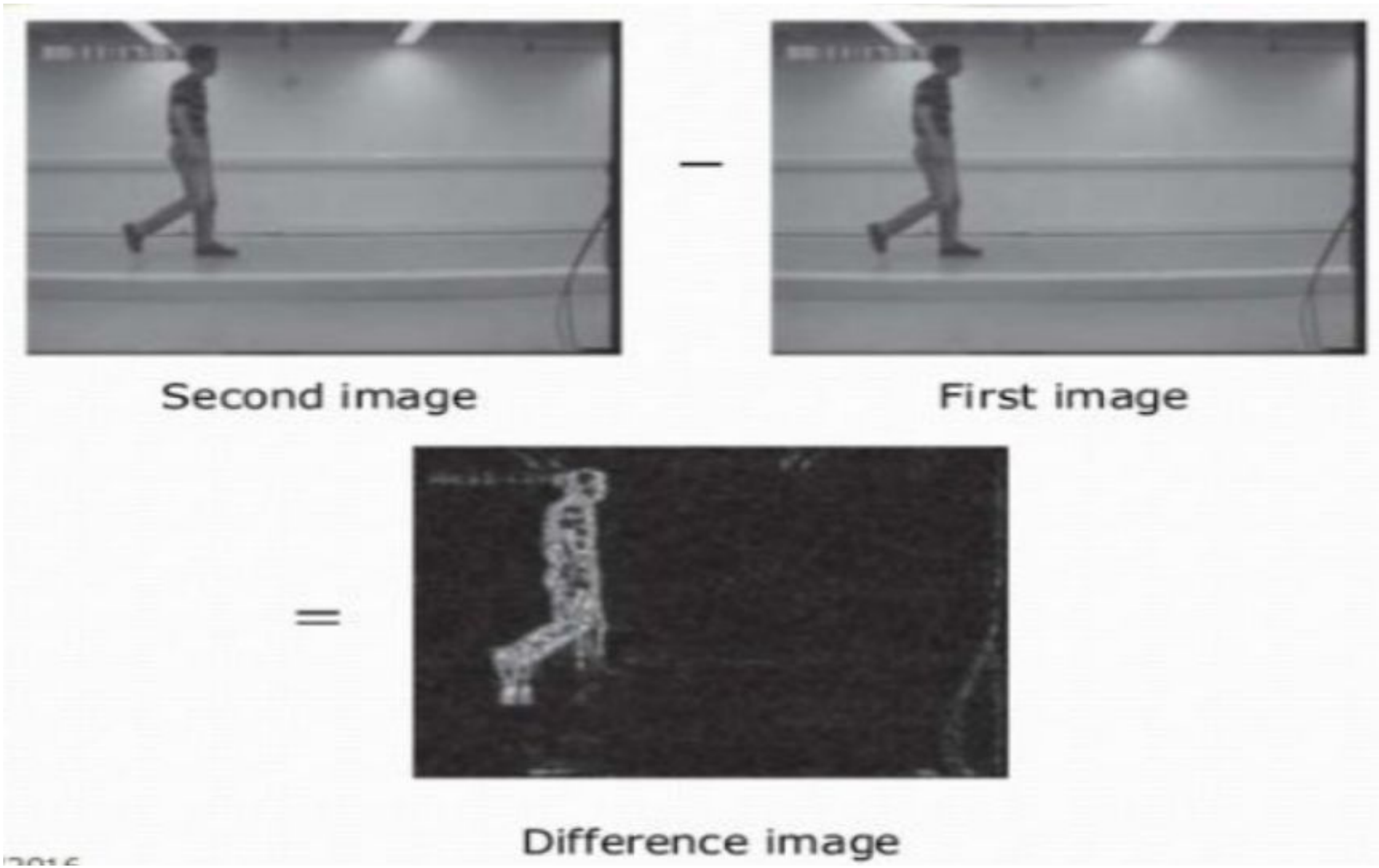
$$D(t) = P(t) - P(t-1)$$

$D(t) = 0$
no motion

$D(t) \neq 0$
Pixel intensity changes
there is motion

Low Level Feature Extraction

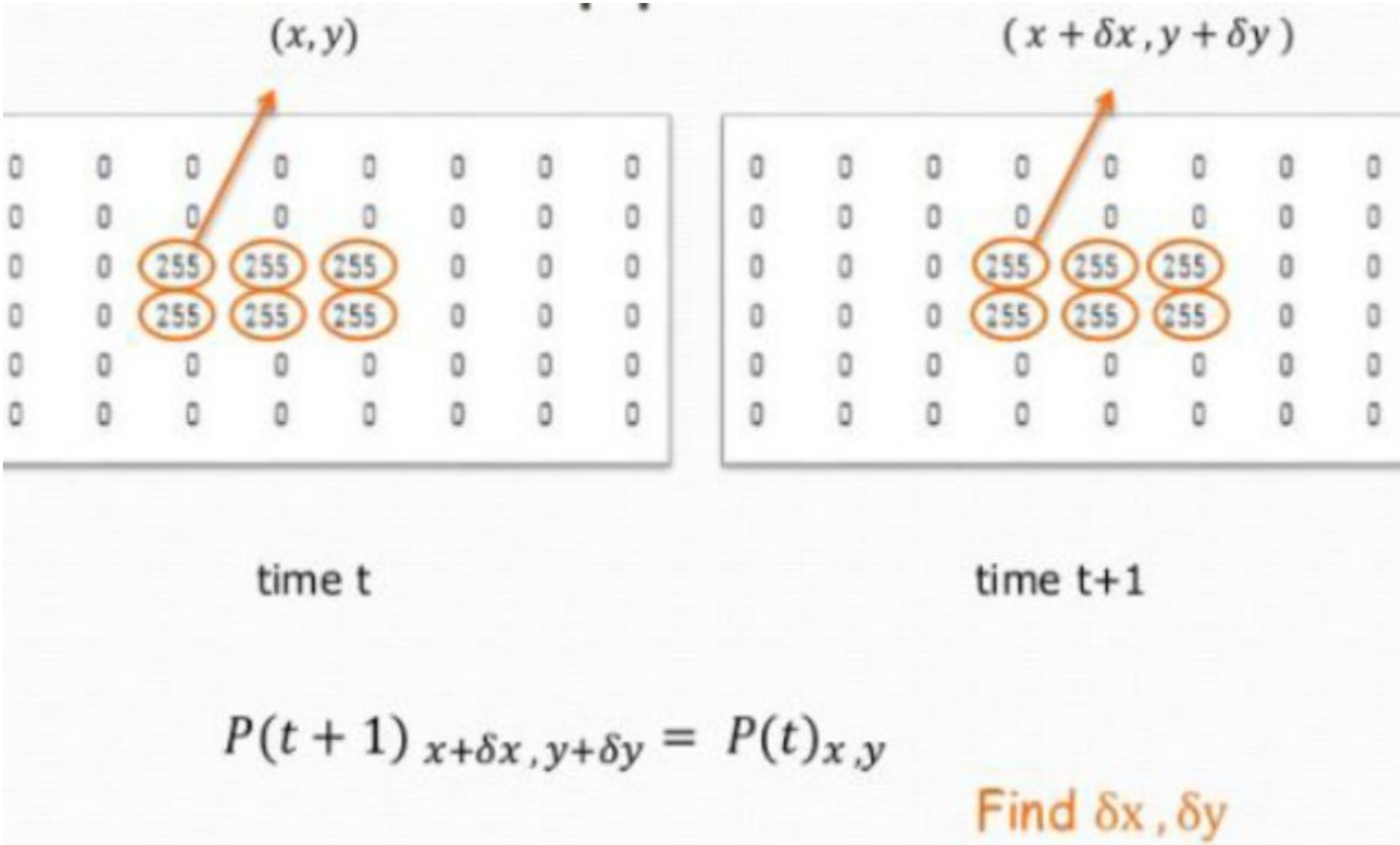
- Difference Image



Approaches to Detect Motions

- There are many approaches for detecting the motion in an image
  - Area Based Approach
  - Differential Approach

Area Based Approach



Area Based Approach

- Consider neighborhood pixels.

$$e_{x,y} = \sum_{(x',y')\in W} \left(P(t+1)_{x'+\delta x\,,\,y'+\delta y} - P(t)_{x',y'} \right)^2$$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	255	255	255	0	0	0
0	0	255	255	255	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	255	255	255	0	0
0	0	0	255	255	255	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Area Based Approach

Assume that:

- The *brightness* at the point in the *new* position should be the same as the brightness at the *old* position.
- *The neighboring* points move with *similar* velocity.

Differential Approach

Optical Flow:

- The velocity field in the image which transforms one image into the next image in a sequence.

- Component of optical flow detect the motion
  - u:rate of chang in X
  - v: rate of change in Y

Differential Approach

- Calculate the u, v for each pixel.
- Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects.

Differential Approach

- The pair of equations gives iterative means for calculating the optical flow of images based on differentials.

$$u_{x,y}^{n+1} = \bar{u}_{x,y}^{n} - \lambda \left(\frac{\nabla x_{x,y} \bar{u}_{x,y} + \nabla y_{x,y} \bar{v}_{x,y} + \nabla t_{x,y}}{1 + \lambda (\nabla x_{x,y}^{2} + \nabla x_{x,y}^{2})} \right) (\nabla x_{x,y})$$

$$v_{x,y}^{n+1} = \bar{v}_{x,y}^{n} - \lambda \left(\frac{\nabla x_{x,y} \bar{u}_{x,y} + \nabla y_{x,y} \bar{v}_{x,y} + \nabla t_{x,y}}{1 + \lambda (\nabla x_{x,y}^{2} + \nabla x_{x,y}^{2})} \right) (\nabla y_{x,y})$$

Difference between AB and DB Approaches

Area-based	Differential
• Slow. ◦ high computation.	• Faster than area-based.
• Flow is clear.	• Uncertain flow.
• Not concerned with rotation.	

Difference between AB and DB Approaches

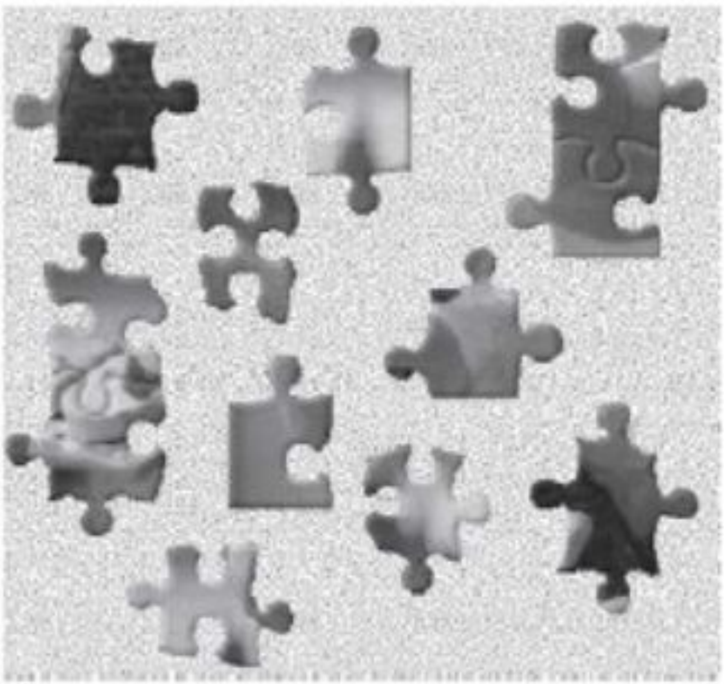
High Level Features

- High level feature extraction concerns finding shapes and objects in computer images.
- Shape extraction by matching is the concept of template matching
- Template matching is model based approach in which the shape is extracted by searching for the best correlation between a known model and pixels in a image.

Template Matching

- Template matching is conceptually a simple process. We need to match a template to an image, where the template is a sub-image that contains the shape we are trying to find.
- we center the template on an image point and count up how many points in the template matched those in the image. The procedure is repeated for the entire image and the point which led to the best match, the maximum count, is deemed to be the point where the shape (given by the template) lies within the image.

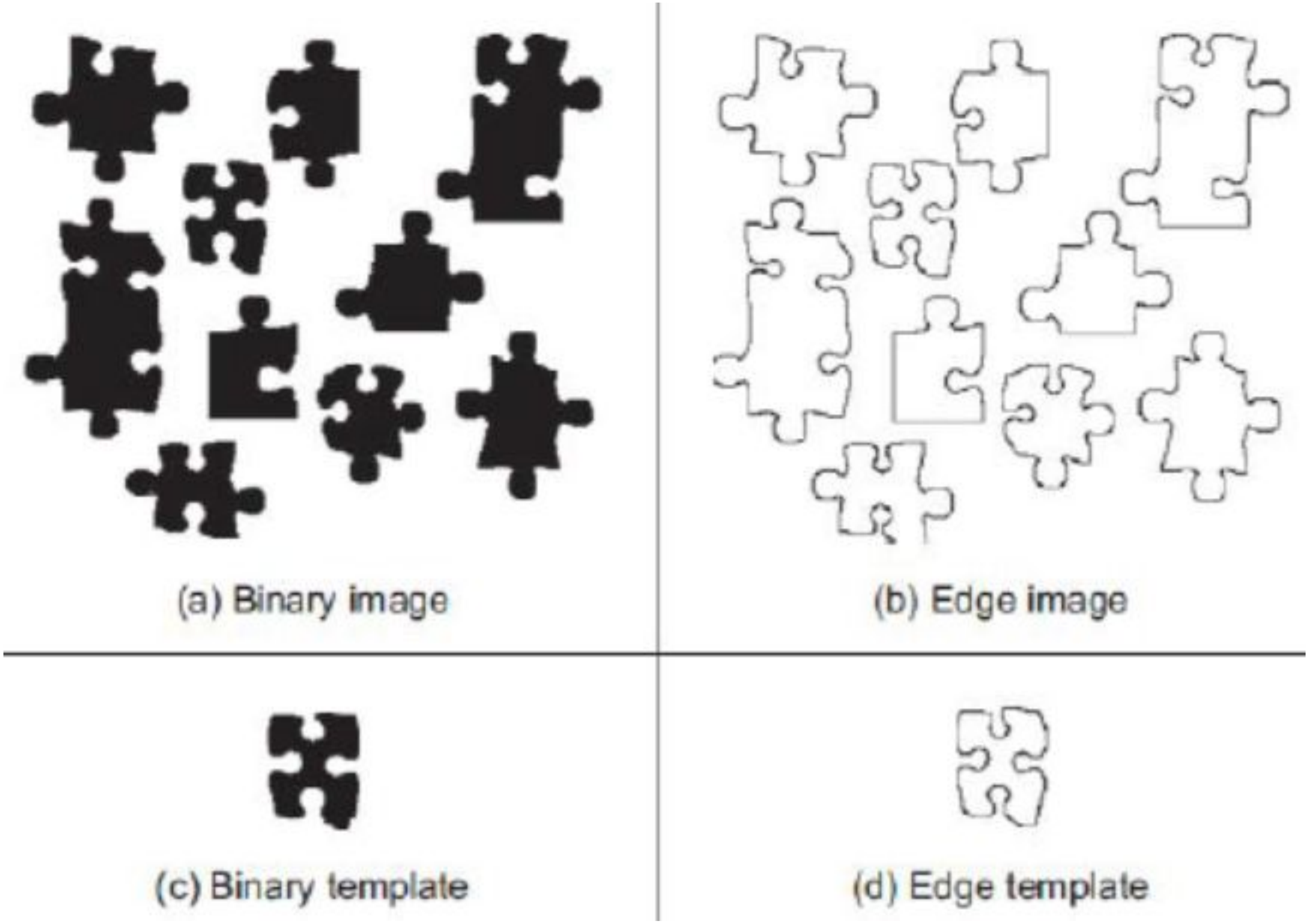
Template Matching



(a) Image containing shapes

(b) Template of target shape

Template Matching

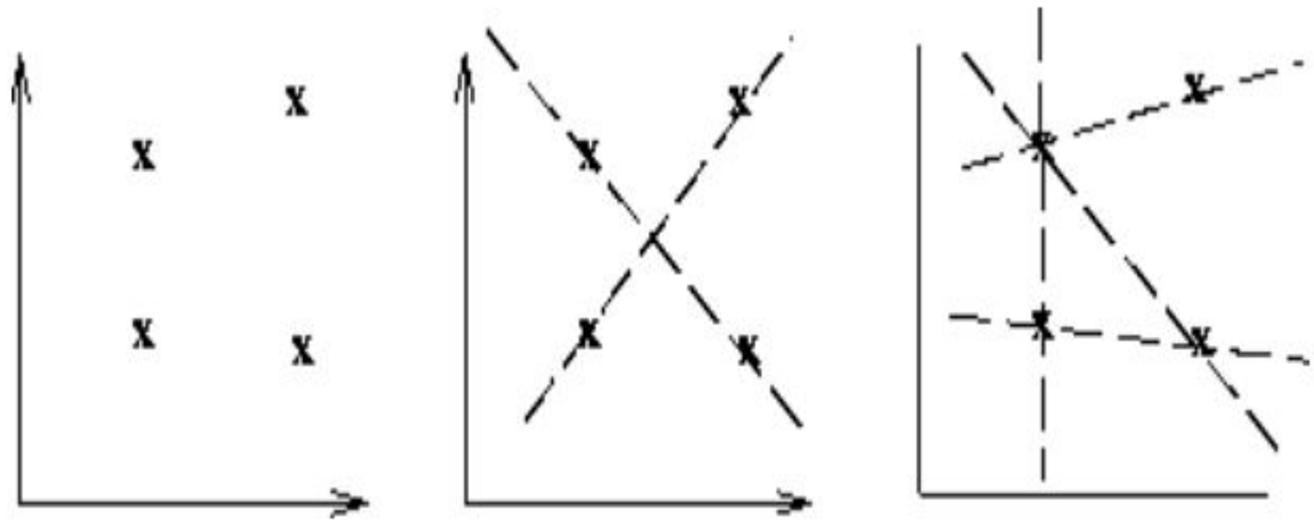


Hough Transform

- The Hough transform is a technique which can be used to **isolate features** of a **particular shape** within an image.
- Because it requires that the desired features be specified in some **parametric form**, the *classical* Hough transform is most commonly used for the detection of regular **curves** such as **lines, circles, ellipses**, etc.
- The idea behind the Hough technique for **line detection** is that each input measurement (*e.g.* **coordinate point**) indicates its contribution to a **globally consistent solution** (*e.g.* the physical line which gave rise to that image point).

Hough Transform

- As an example, consider the common problem of **fitting** a set of line segments to a **set of discrete image points** (*e.g.* pixel locations output from an edge detector).
- Figure shows some possible solutions to this problem.
- Here the lack of ***a priori*** **knowledge** about the number of **desired** line segments render this problem under-constrained.



Hough Transform

- The **Hough Space** is a **2D plane** that has a **horizontal axis** representing the **slope** and the **vertical axis** representing the **intercept** of a **line** on the **edge image**.
- A line on an edge image is represented in the form of $\boldsymbol{y = ax + b}$
- **One line** on the edge image produces **a point** on the Hough Space since a line is characterized by its slope a and intercept b.
- On the other hand, **an edge point** (x_i, y_i) on the edge image can have an **infinite number of lines** pass through it.
- Therefore, an edge point **produces a line** in the Hough Space in the form of $\boldsymbol{b = ax_i + y_i}$

Hough Transform

- In the Hough Transform algorithm, the Hough Space is used to determine whether **a line exists** in the edge image.
- There is **one flaw** with representing lines in the form of $y = ax + b$ and the Hough Space with the slope and intercept.
- In this form, the algorithm won't be able to detect **vertical lines** because the **slope *a* is undefined/ infinity** for vertical lines

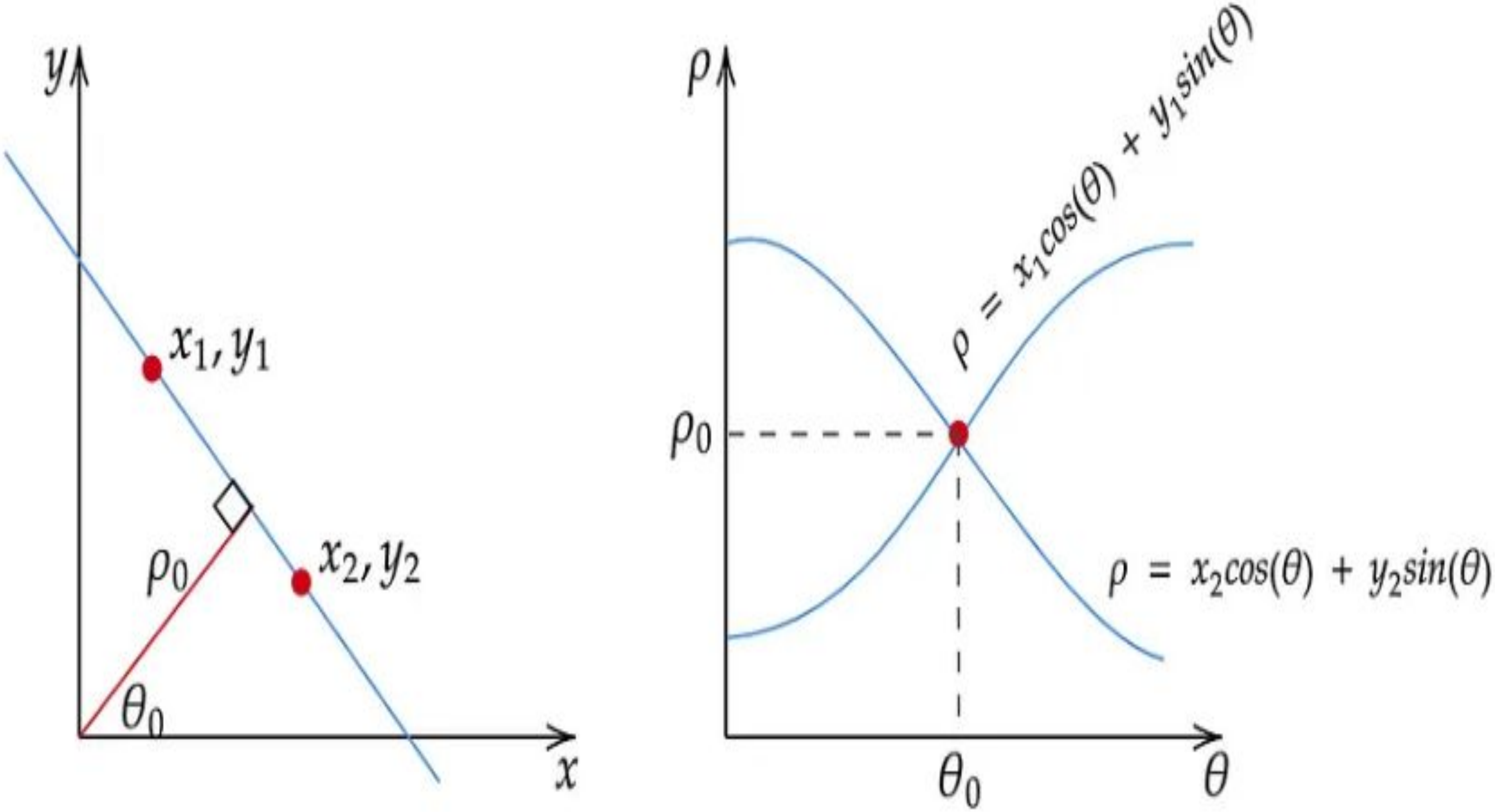
Hough Transform

- This means that a computer would need an **infinite amount of memory** to represent all possible values of *a*.
- To avoid this issue, a **straight line** is instead represented by a **line** called the **normal line** that passes through the **origin** and **perpendicular** to that straight line.
- The form of the normal line is $\boldsymbol{\rho = x \cos(\theta) + y \sin(\theta)}$ where **ρ** is the length of the **normal line** and **θ** is the **angle** between the normal line and the x axis.

Hough Transform

- So, a convenient equation for describing a set of lines uses parametric or normal notion: $\boldsymbol{\rho = x \cos(\theta) + y \sin(\theta)}$
- where **ρ** is the length of a **normal** from the origin to this line and theta (**θ)** is the orientation of **ρ** with respect to the X-axis.
- For any point (x,y) on this line, **ρ** and **θ** are constant.
- Instead of representing the Hough Space with the slope *a* and intercept *b*, it is now represented with ‘**ρ’** and ‘θ’ where the **horizontal axis** is for the **θ** values and the **vertical axis** are for the **ρ** values.
- The ***mapping*** of **edge points** onto the **Hough Space** works in a similar manner except that an edge point (x_i, y_i) now generates a **cosine curve** in the Hough Space instead of a straight line

Hough Transform



Hough Transform

- As mentioned, an edge point **produces** a **cosine curve** in the Hough Space.
- From this, if we were to **map all** the edge points from an edge image onto the Hough Space, it will generate **a lot of cosine curves**.
- If two edge points **lay** on the **same line**, their corresponding cosine curves will **intersect each other** on a specific (ρ, θ) pair.
- Thus, the Hough Transform algorithm detects lines by finding the (ρ, θ) pairs that have a **number of intersections** larger than a **certain threshold**.

Hough Transform – Algorithm

1. **Determine the range of ρ and θ**. Typically, the range of **θ is [0, 180]** degrees and ρ is **[-d, d],** where **d** is the diagonal length of the edge. It is important to quantize the range of ρ and θ, which means there should only be a **finite number** of possible values.

2. **Create a 2D array called the "accumulator"** with the dimensions (**num_rhos, num_thetas**) to represent the Hough Space and set all its values to **zero**.

3. **Perform edge detection on the original image** (ED). You can do this with any ED technique you like.

Hough Transform – Algorithm

4. For **every pixel** on the **edge** image, **check whether the pixel is an edge pixel**. If it is an **edge pixel**, **loop over** all possible values of θ, **compute** the corresponding **ρ**, **find** the θ and ρ **index** in the accumulator, then **increase** the accumulator base on those index pairs.

5. **Iterate over the accumulator's values**. If the value is **larger** than a certain **threshold**, get the ρ and θ index, get the value of ρ and θ from the **index pair** which can then be **converted back** to the form of $y = ax + b$.

Hough Transform – Algorithm

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

Advantages

- The HT benefits from not requiring all pixels on a single line to be contiguous. As a result, it can be quite effective when identifying lines with small gaps due to noise or when objects are partially occluded.

Disadvantages

- It can produce deceptive results when objects align by accident;
- Rather than finite lines with definite ends, detected lines are infinite lines defined by their (m,c) values.

Circles Detection using HT

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

The HT can be extended by replacing the equation of the curve in the detection process. The equation of the curve can be given in **explicit** or **parametric** form. In explicit form, the HT can be defined by considering the equation for a circle given by

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{5.30}$$

This equation defines a locus of points (x,y) centered on an origin (x_0,y_0) and with radius r. This equation can again be visualized in two dual ways: as a locus of points (x,y) in an image and as a locus of points (x_0,y_0) centered on (x,y) with radius r.

Circles

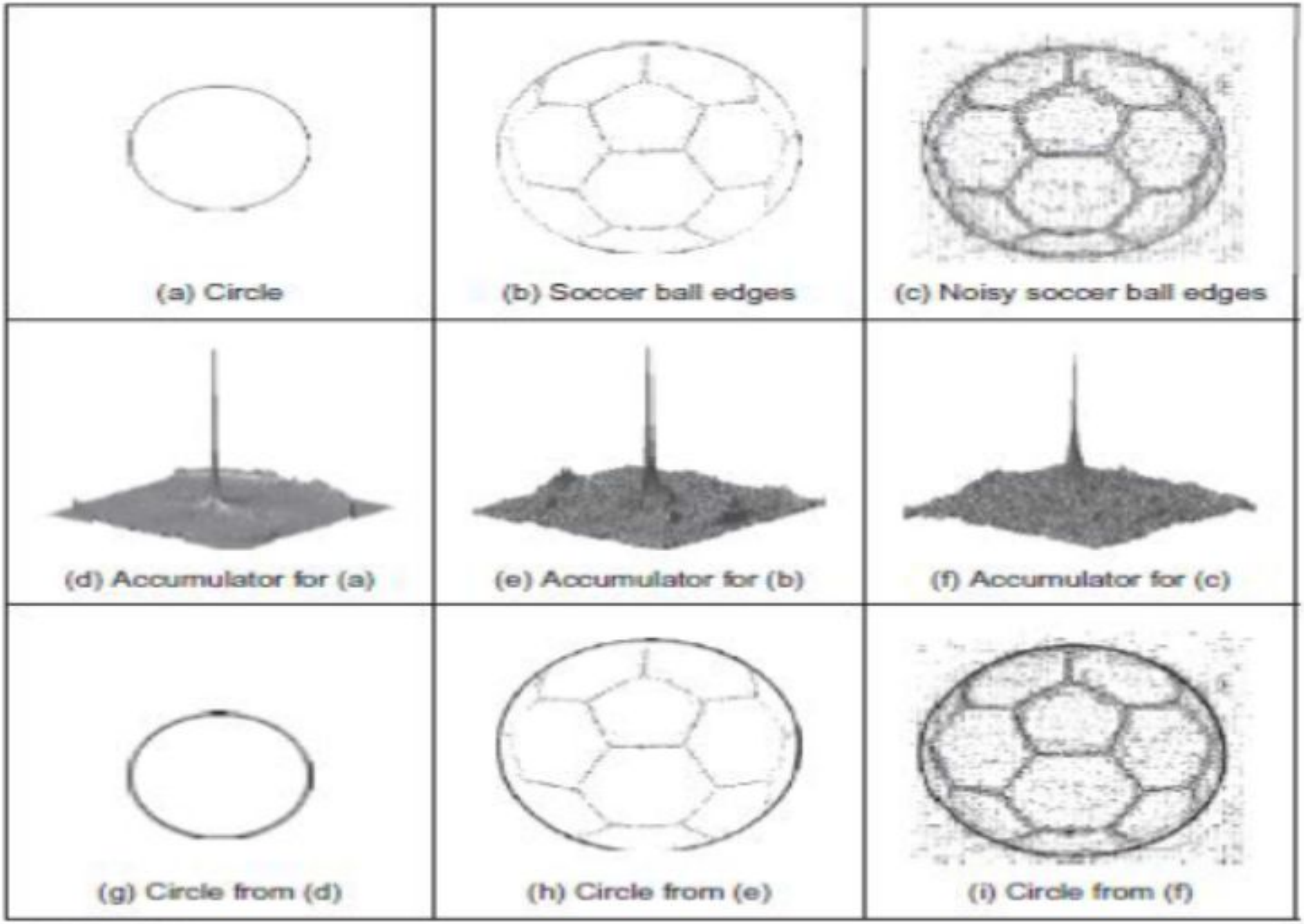
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

The advantage of this representation is that it allows us to solve for the parameters. Thus, the HT mapping is defined by

$$x_0 = x - r\cos(\theta); \quad y_0 = y - r\sin(\theta) \tag{5.32}$$

These equations define the points in the accumulator space (Figure 5.19(b)) dependent on the radius r. Note that θ is not a free parameter but defines the trace of the curve. The trace of the curve (or surface) is commonly referred to as the point spread function.

Circles



(a) Circle
(b) Soccer ball edges
(c) Noisy soccer ball edges
(d) Accumulator for (a)
(e) Accumulator for (b)
(f) Accumulator for (c)
(g) Circle from (d)
(h) Circle from (e)
(i) Circle from (f)

Circles

Algorithm

- Initialize the accumulator (H[a,b,r]) to all zeros
- Find the edge image using any edge detector
- For r= 0 to diagonal image length
    - For each edge pixel (x,y) in the image
        - For Θ = 0 to 360
            - $a = x – r*\cos\Theta$
            - $b = y – r*\sin\Theta$
            - H[a,b,r] = H[a,b,r] +1
- Find the [a,b,r] value(s), where H[a,b,r] is above a suitable threshold value

Ellipse Detection using HT

Circles are very important in shape detection since many objects have a circular shape. However, because of the camera's viewpoint, circles do not always look like circles in images. Images are formed by mapping a shape in 3D space into a plane (the image plane). This mapping performs a perspective transformation. In this process, a circle is deformed to look like an ellipse. We can define the mapping between the circle and an ellipse by a similarity transformation. That is,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\rho) & \sin(\rho) \\ -\sin(\rho) & \cos(\rho) \end{bmatrix} \begin{bmatrix} S_x \\ S_y \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{5.33}$$

where (x', y') define the coordinates of the circle in Eq. (5.31), ρ represents the orientation, (S_x, S_y) a scale factor and (t_x, t_y) a translation. If we define

$$\begin{matrix} a_0 = t_x & a_x = S_x \cos(\rho) & b_x = S_y \sin(\rho) \\ b_0 = t_y & a_y = -S_x \sin(\rho) & b_y = S_y \cos(\rho) \end{matrix} \tag{5.34}$$

then the circle is deformed into

$$\begin{aligned} x &= a_0 + a_x \cos(\theta) + b_x \sin(\theta) \\ y &= b_0 + a_y \cos(\theta) + b_y \sin(\theta) \end{aligned} \tag{5.35}$$

This equation corresponds to the polar representation of an ellipse. This polar form contains six parameters $(a_0, b_0, a_x, b_x, a_y, b_y)$ that characterize the shape of the ellipse. θ is not a free parameter and it only addresses a particular point in the locus of the ellipse (just as it was used to trace the circle in Eq. (5.32)). However,

Ellipse Detection using HT

three axis parameters must jointly describe size and rotation. In fact, the axis parameters can be related to the orientation and the length along the axes by

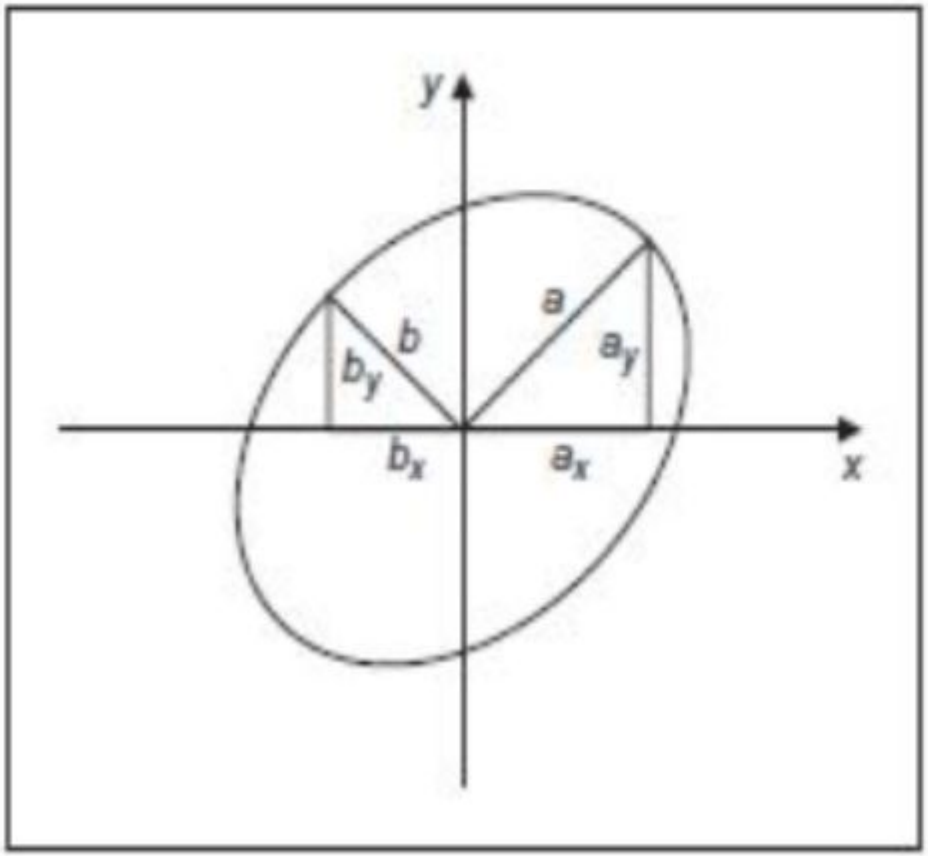
$$\tan(\rho) = \frac{a_y}{a_x} \quad a = \sqrt{a_x^2 + a_y^2} \quad b = \sqrt{b_x^2 + b_y^2} \tag{5.36}$$

where (a,b) are the axes of the ellipse, as illustrated in Figure 5.22.

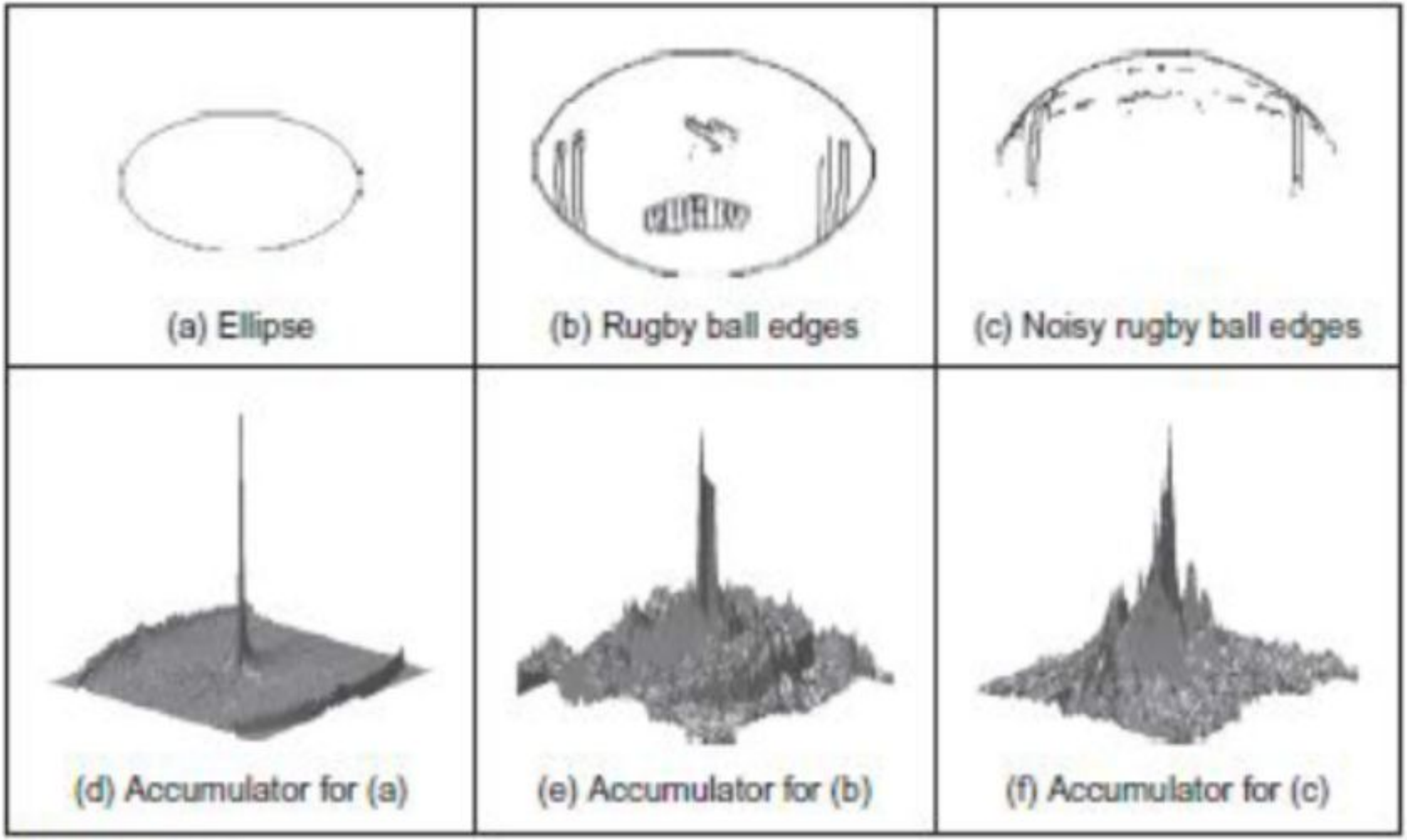
In a similar way to Eq. (5.31), Eq. (5.35) can be used to generate the mapping function in the HT. In this case, the location of the center of the ellipse is given by

$$\begin{aligned} a_0 &= x - a_x\cos(\theta) + b_x\sin(\theta) \\ b_0 &= y - a_y\cos(\theta) + b_y\sin(\theta) \end{aligned} \tag{5.37}$$

Ellipses Axes



HT for Ellipses



HT for Ellipses

```
%Hough Transform for Ellipses

function HTEllipse(inputimage,a,b)

%image size
[rows,columns]=size(inputimage);

%accumulator
acc=zeros(rows,columns);

%image
for x=1:columns
  for y=1:rows
    if(inputimage(y,x)==0)
      for ang=0:360
        t=(ang*pi)/180;
        x0=round(x-a*cos(t));
        y0=round(y-b*sin(t));
        if(x0<columns & x0>0 & y0<rows & y0>0)
          acc(y0,x0)=acc(y0,x0)+1;
        end
      end
    end
  end
end
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

End of UNIT 1 (Part 2)