IMAGE PROCESSING

UNIT 1 TOPICS…..

- *Image processing basics*
- *what is image, acquisition, type, point operations, Geometric transformations*
- *First and second derivative*
- *steps in edge detection, smoothening, enhancement, thresholding, localization,*
- *Robert's method, Sobel's method, Perwitts*
- *Laplacian of Gaussian, Zero crossing*
- *Low level feature extraction, Describing image motion*
- *High level feature extraction ,Template matching*
- *Hough transform for lines*
- *Hough transform for circles and ellipses*

image – digital image

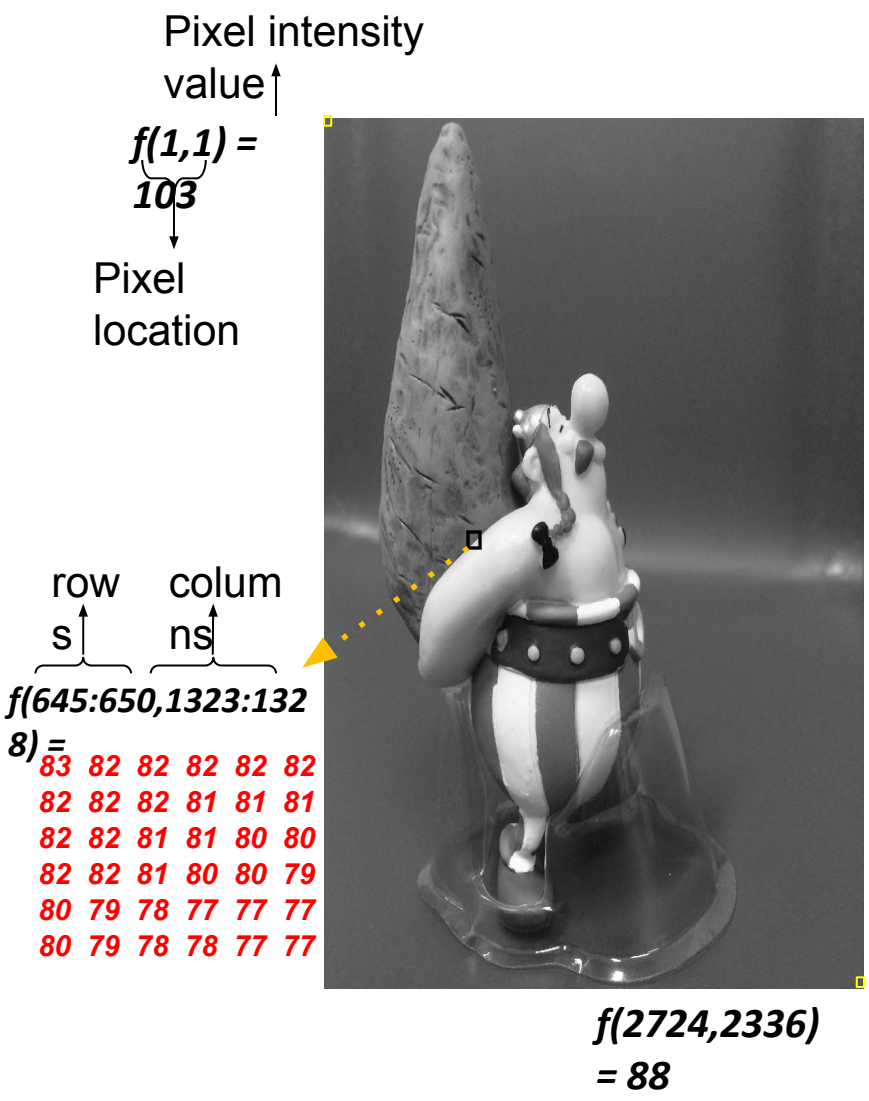
An image is a two-dimensional function **f(x,y)**, where x and y are the **spatial** (plane) coordinates, and the amplitude of **f** at any pair of coordinates (x,y) is called the intensity of the image at that level.

If **x,y** and the **amplitude** values of **f** are **finite** and **discrete quantities**, we call the image a **digital image**. A digital image is composed of a finite number of elements called **pixels**, each of which has a particular location and value.

Image

- **What is an image?**
- An image is defined 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** of that image at that point. When x,y, and amplitude values of **F** are finite, we call it a **digital image**.
- In other words, an image can be defined by a two-dimensional array specifically arranged in rows and columns.
- Digital Image is composed of a finite number of elements, each of which elements have a particular value at a particular location. These elements are referred to as *picture elements, image elements, and pixels*.
- A *Pixel* is most widely used to denote the elements of a Digital Image.

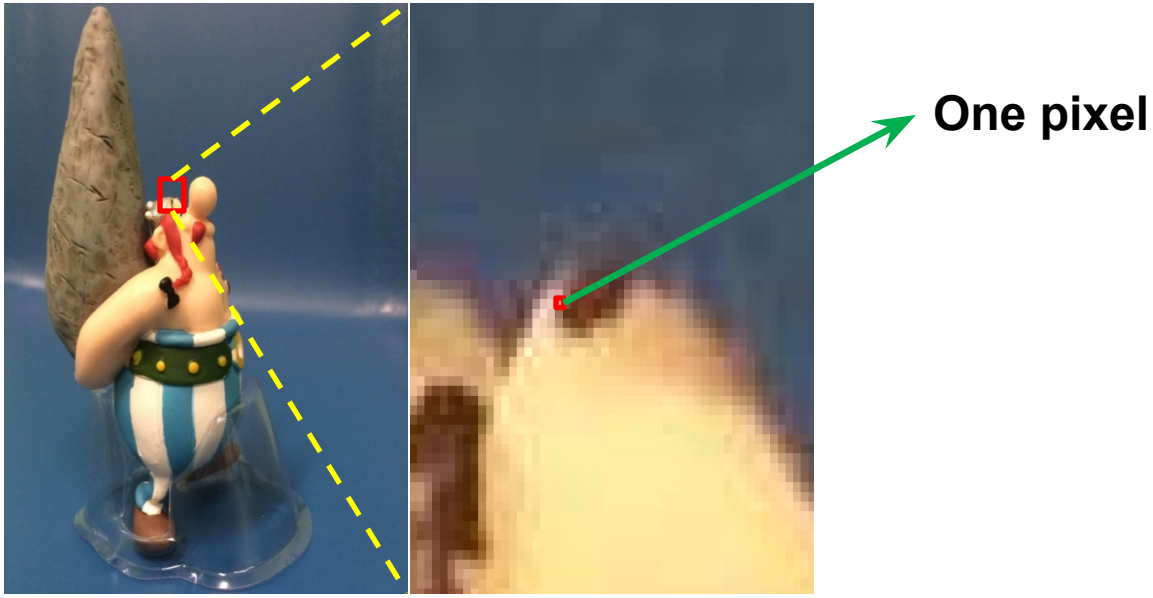
First Digital Photograph



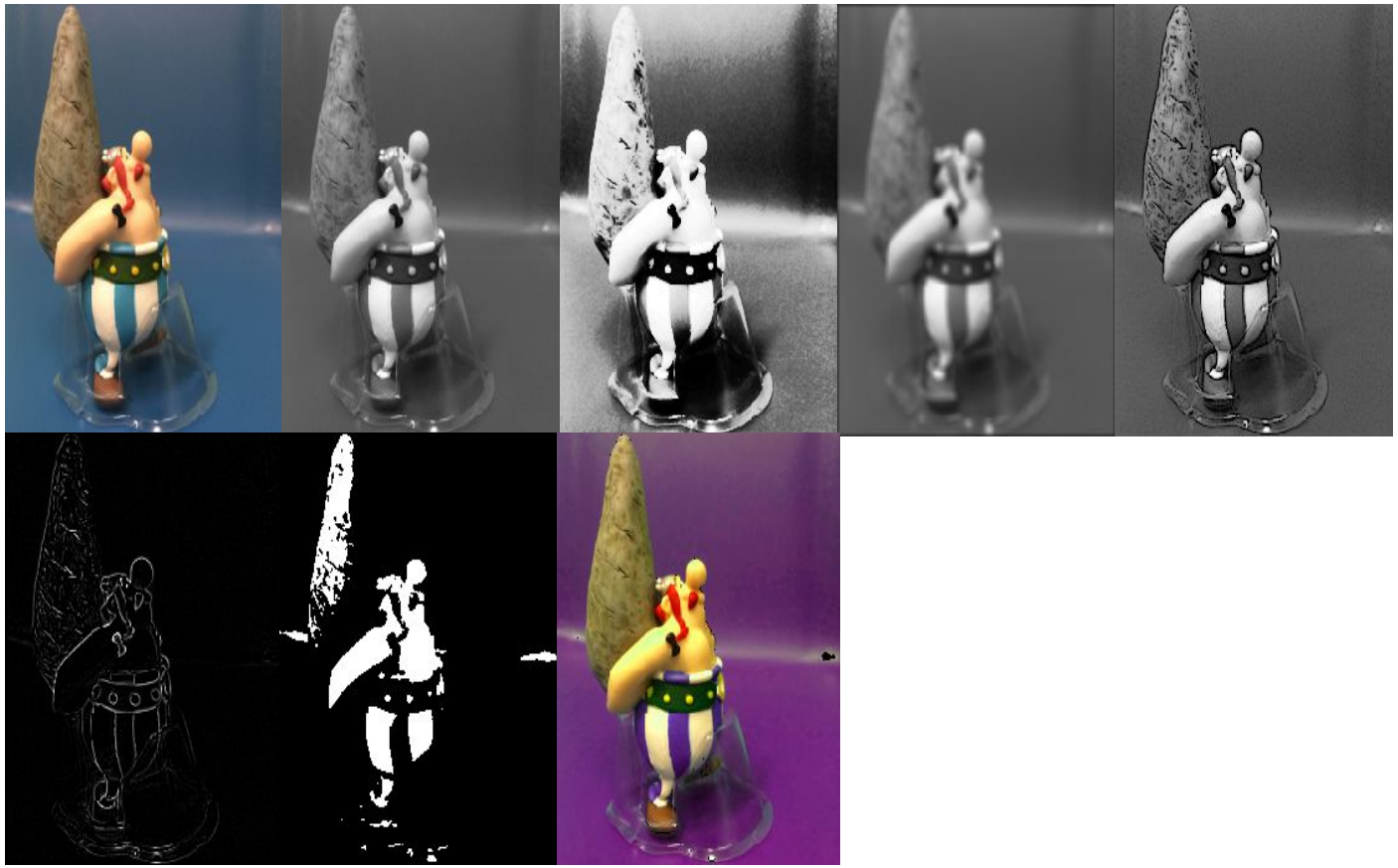
Consider the following image (2724x2336 pixels) to be 2D function or a **matrix** with **rows** and **columns**

In 8-bit representation Pixel intensity values change between 0 (Black) and 255 (White)

Remember *digitization* implies that a digital image is an *approximation* of a real scene



Digital Image Processing



Types of Digital Images

The images types we will consider are:

1) binary

2) gray-scale,

3) color,

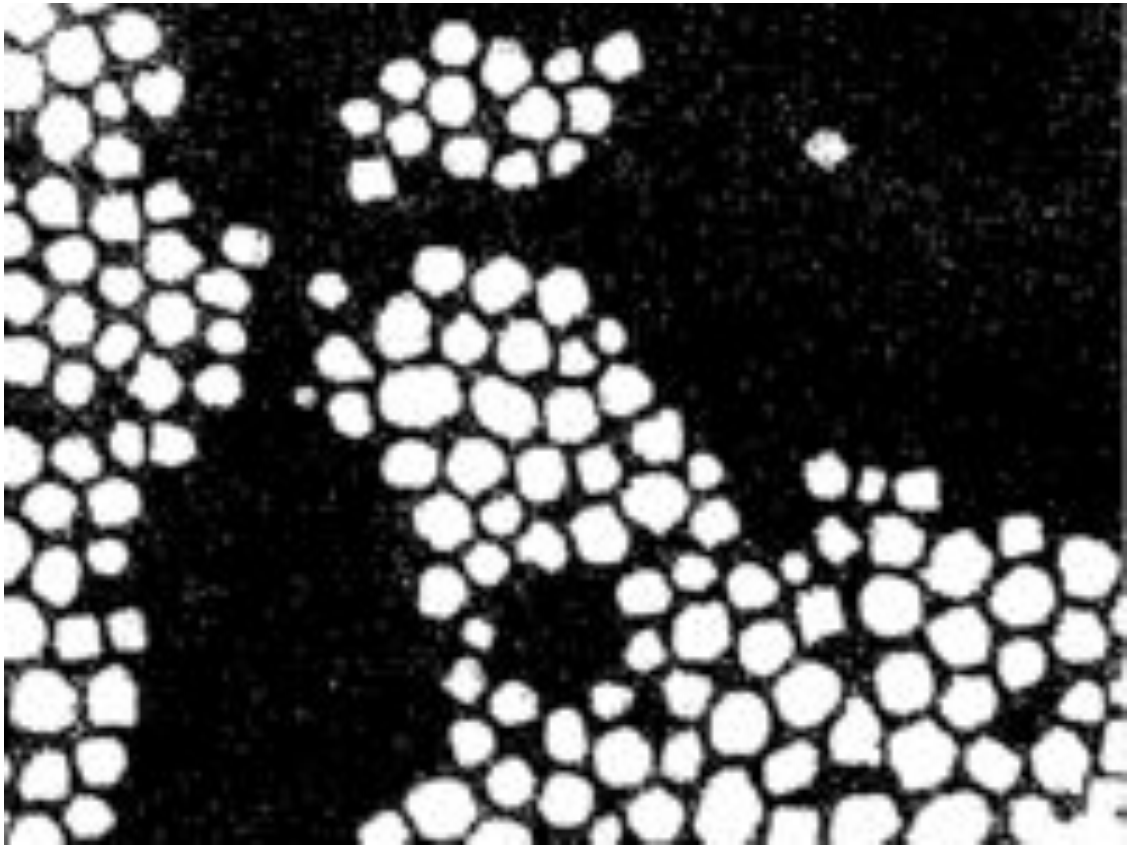
Binary Images

- Binary images are the simplest type of images and can take on two values, typically black and white, or 0 and 1.
- A binary image is referred to as a 1-bit image because it takes only 1 binary digit to represent each pixel.
- These types of images are frequently used in applications where the only information required is general shape or outline, for example optical character recognition (OCR).

- Binary images are often created from the gray-scale images via a threshold operation, where every pixel above the threshold value is turned white (‘1’), and those below it are turned black (‘0’).

(a) Object outline. (b) Page of text used in OCR application

BINARY IMAGE



Gray-scale images

- Gray-scale images are referred to as monochrome (one-color) images.
- They contain gray-level information, no color information.
- The number of bits used for each pixel determines the number of different gray levels available.
- The typical gray-scale image contains 8bits/pixel data, which allows us to have 256 different gray levels. The figure below shows examples of gray-scale images.

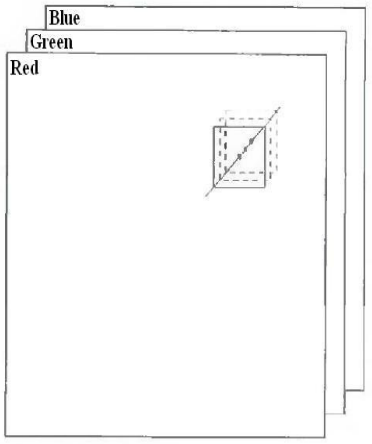
Example

Examples of gray-scale images

Color images

- Color images can be modeled as three-band monochrome image data, where each band of data corresponds to a different color.
- The actual information stored in the digital image data is the gray-level information in each spectral band.
- Typical color images are represented as red, green, and blue (RGB images). Using the 8-bit monochrome standard as a model, the corresponding color image would have 24-bits/pixel (8-bits for each of the three color bands red, green, and blue).

- The figure below illustrates a representation of a typical RGB color image.



Representation of a typical RGB color image

COLOR IMAGE

Image Types

Three types of images:

- Binary images

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

- Gray-scale images

  $g(x,y) \in C$

  typically c={0,…,255}

- Color Images

  three channels:

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

Gray Scale Image

x =	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
y =															
41	210	209	204	202	197	247	143	71	64	80	84	54	54	57	58
42	206	196	203	197	195	210	207	56	63	58	53	53	61	62	51
43	201	207	192	201	198	213	156	69	65	57	55	52	53	60	50
44	216	206	211	193	202	207	208	57	69	60	55	77	49	62	61
45	221	206	211	194	196	197	220	56	63	60	55	46	97	58	106
46	209	214	224	199	194	193	204	173	64	60	59	51	62	56	48
47	204	212	213	208	191	190	191	214	60	62	66	76	51	49	55
48	214	215	215	207	208	180	172	188	69	72	55	49	56	52	56
49	209	205	214	205	204	196	187	196	86	62	66	87	57	60	48
50	208	209	205	203	202	186	174	185	149	71	63	55	55	45	56
51	207	210	211	199	217	194	183	177	209	90	62	64	52	93	52
52	208	205	209	209	197	194	183	187	187	239	58	68	61	51	56
53	204	206	203	209	195	203	188	185	183	221	75	61	58	60	60
54	200	203	199	236	188	197	183	190	183	196	122	63	58	64	66
55	205	210	202	203	199	197	196	181	173	186	105	62	57	64	63

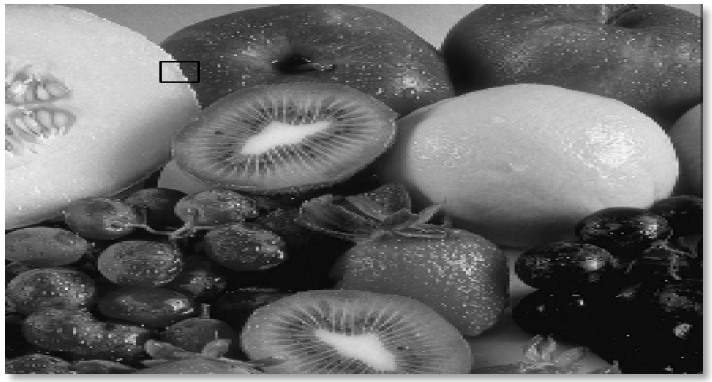
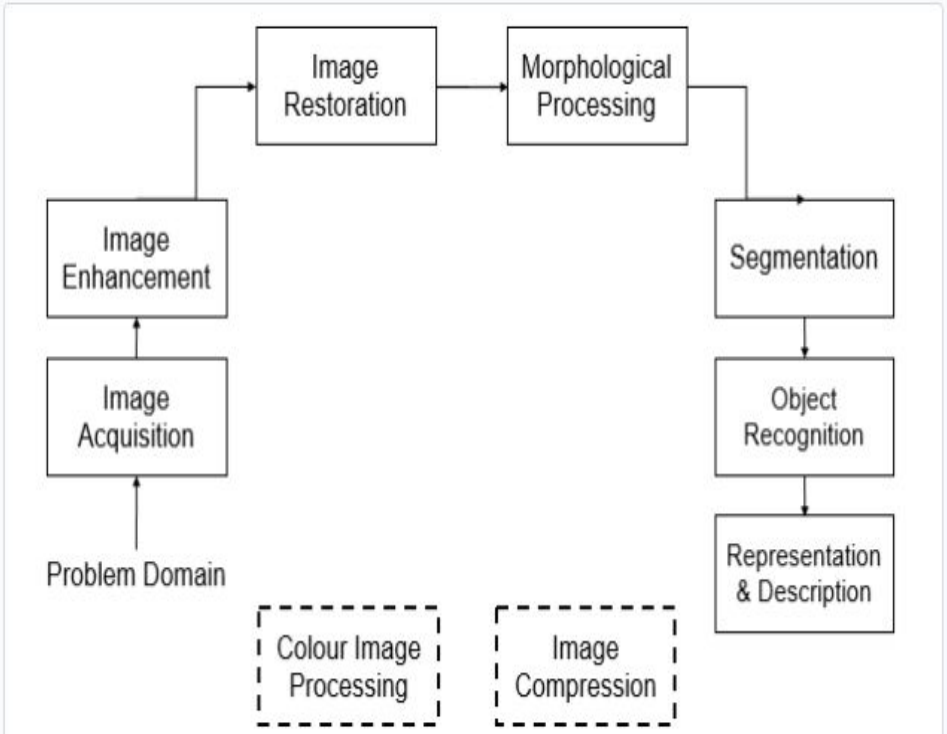


IMAGE PROCESSING

- 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 STEPS

This is the **Image Processing Block Diagram** image step by step as follow:



Block Diagram of Digital Image Processing System

Image acquisition

- Image acquisition by single sensor: Photodiode is an example of single sensor, it is constructed of silicon material whose output voltage is proportional to light.
- Image acquisition using sensor strips: A geometry that is used more frequently than the single sensors consists of an in line arrangement of sensors in the form of sensor strip.
- Image acquisition by sensor arrays: Individual sensors are arranged in the form of sensor 2D arrays

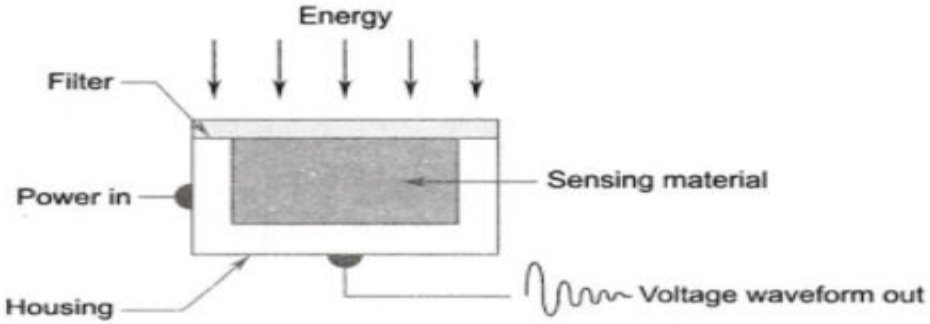


Fig: Single image sensor

Fig: Line sensor

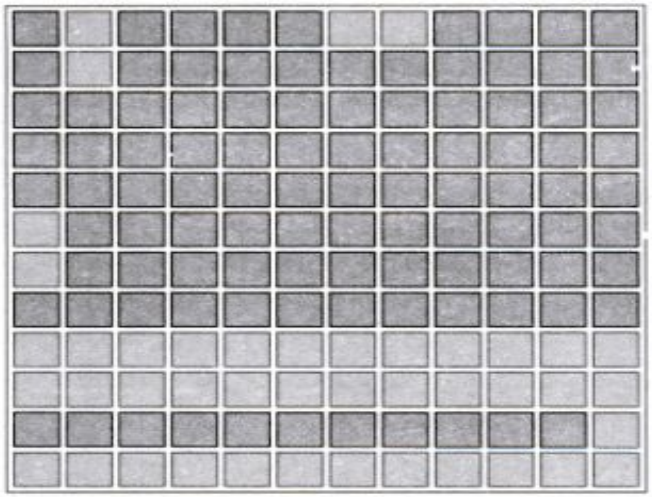
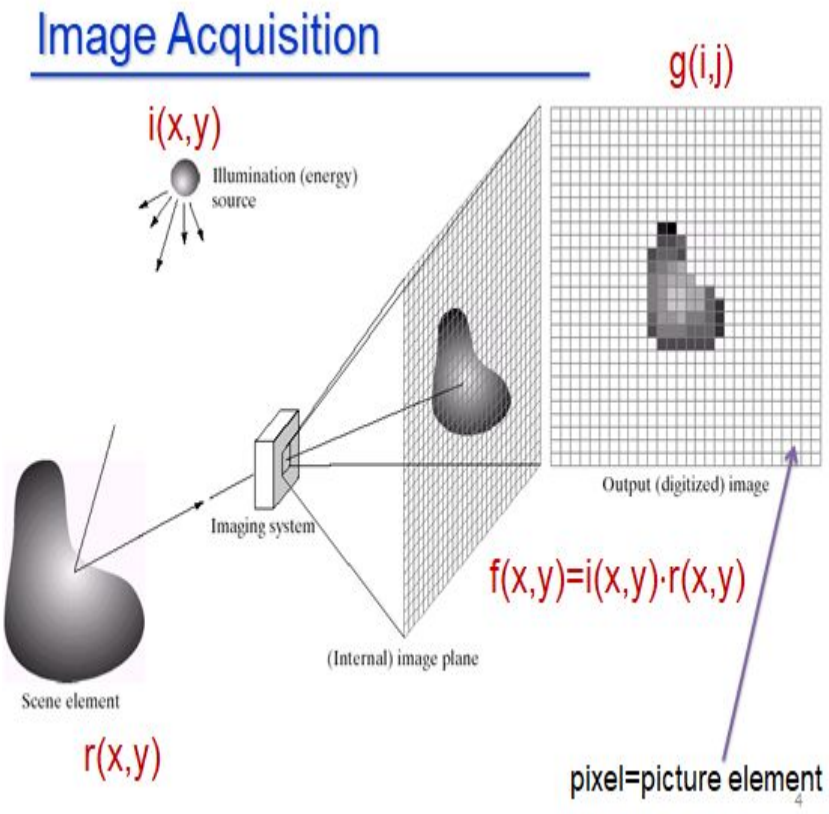


Fig: Array sensor

IMAGE ACQUISITION



POINT OPERATIONS

2.2.1 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

Table 2: Types of image operations. Image size = $N \times N$; neighborhood size = $P \times P$. Note that the complexity is specified in operations *per pixel*.

POINT OPERATION DIAGRAM

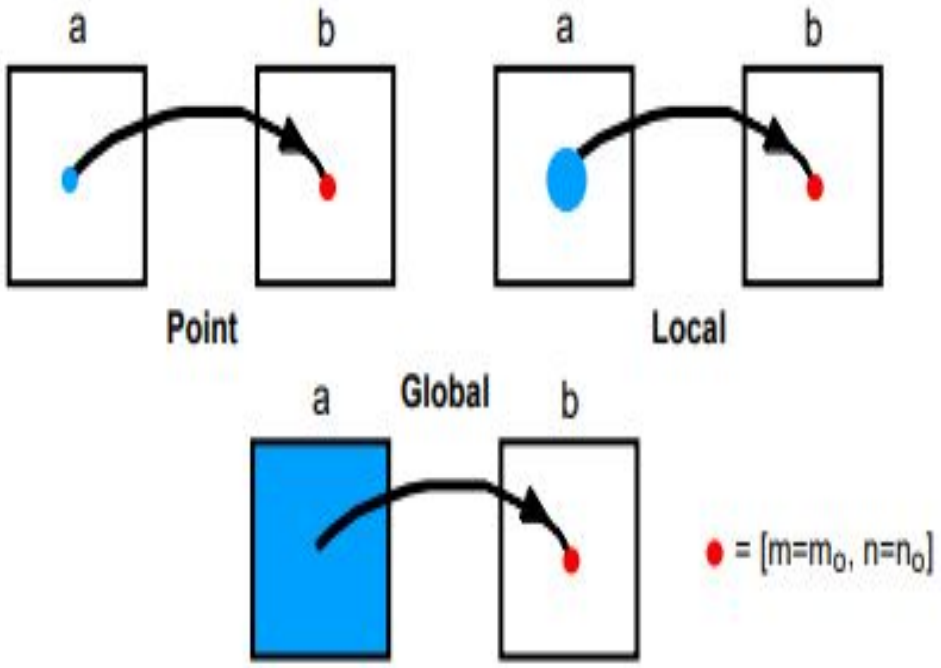
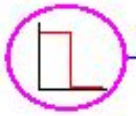
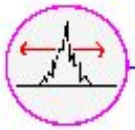


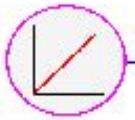
Figure 2: Illustration of various types of image operations

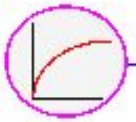
DIFFERENT POINT OPERATIONS

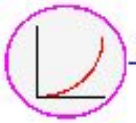

Thresholding - select pixels with given values to produce binary image


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

Logarithm Operator

- The dynamic range of an image can be compressed by replacing each pixel value with its logarithm. This has the effect that low intensity pixel values are enhanced. Applying a pixel logarithm operator to an image can be useful in applications where the dynamic range may too large to be displayed on a screen (or to be recorded on a film in the first place).
- **Exponential/`Raise to Power' Operator**
- The exponential and `raise to power' operators are two anamorphosis operators which can be applied to grayscale images. Like the logarithmic transform, they are used to change the dynamic range of an image. However, in contrast to the logarithmic operator, they enhance high intensity pixel values.

- Logarithmic - Useful for enhancing details in the darker regions of the image at the expense of detail in the brighter regions.

  Exponential - The effect is the reverse of that obtained with logarithmic mapping.

Thresholding

- **Thresholding**. Definition: An image processing method that creates a bitonal also known as binary image based on setting a threshold value on the pixel intensity of the original image. While most commonly applied to grayscale images, it can also be applied to color images.

Thresholding

```
[a,b]=size(I3);
Threshlold1 = median(median(I3));
for i = 1:a
    for j = 1:b
        c= I3(i,j);
        if c>=Threshlold1
        IB1(i,j) = c;
        else
        IB2(i,j) = c;
        end
    end
end
figure
imshow(IB1)
figure
imshow(IB2)
```

Thresholding output

Basic Arithmetic Operations

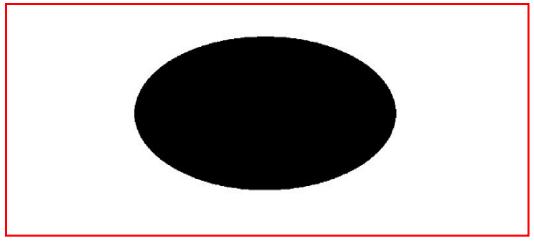
Operation	Definition
ADD	c = a + b
SUB	c = a - b
MUL	c = a * b
DIV	c = a / b
LOG	c = log(a)
EXP	c = exp(a)
SQRT	c = sqrt(a)
TRIG.	c = sin/cos/tan(a)
INVERT	$c = (2^B - 1) - a$

Basic Logic Operations

Operation	Definition
NOT	
OR	
AND	
XOR	

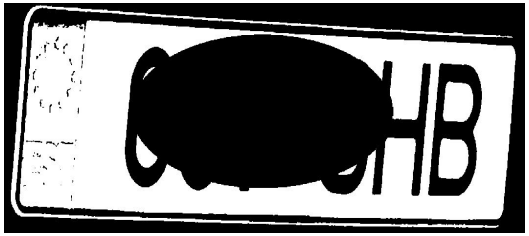
Arithmetic and Logic Operations

a



b

NOT(a)



a . b

a + b

Addition Operation

Subtraction Operation

Multiplication by 2

ARITHMETIC OPERATIONS
(IMAGE ENHANCEMENT)

1. ADDITION OPERATION

$I = I_1 + I_2$

$$\begin{bmatrix} 10 & 10 & 10 \\ 11 & 11 & 10 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 11 \\ 0 & 0 & 10 \end{bmatrix} = \begin{bmatrix} & & \\ & & \end{bmatrix}$$

- Superimposing one image on another

USES → (i) Changing image background
(ii) Water mark Images
(iii) Putting Logos etc

2 SUBTRACTION OPERATION

$I = I_1 - I_2$

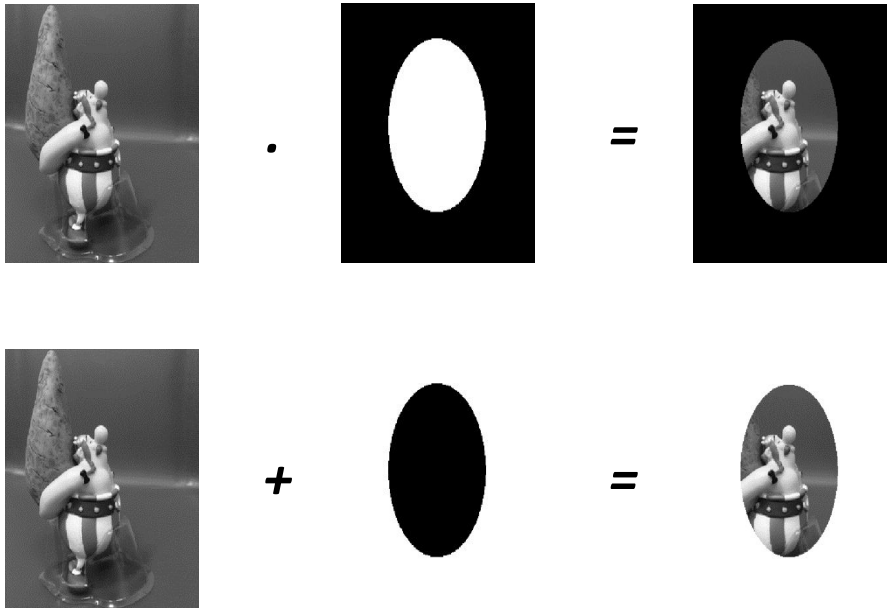
$$\begin{bmatrix} 10 & 10 & 10 \\ 11 & 11 & 10 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 11 \\ 0 & 10 & 11 \end{bmatrix}$$

$$= \begin{bmatrix} & & \\ & & \end{bmatrix}$$

- Common part get diminished.

USES → (i) Change detection
(ii) Surveillance
(iii) Deforestation detection
(iv) Medical Application (X-Ray)

Arithmetic and Logic Operations



LOGICAL OPERATIONS (IMAGE ENHANCEMENT)

AND

lly used to mask out part of an image.

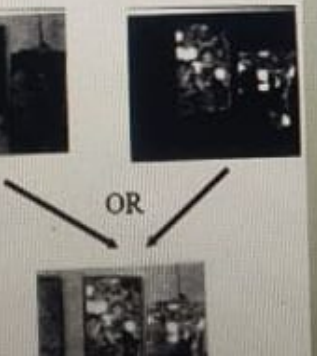
Image 2 Result

OR

an be added with a logical OR operator.

Image 2 Result

AND Result of OR



OR

1 AND operations

$I = I_1$ and I_2

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \text{ AND } \begin{matrix} 0\ 1\ 0 \\ 1\ 0\ 0 \\ 1\ 1\ 1 \end{matrix}$$

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

Resultant Image

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

2. OR operation

$I = I_1$ OR I_2

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \text{ OR } \begin{matrix} 1\ 0\ 1 \\ 0\ 1\ 1 \\ 1\ 1\ 1 \\ 0\ 0\ 0 \end{matrix}$$

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

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

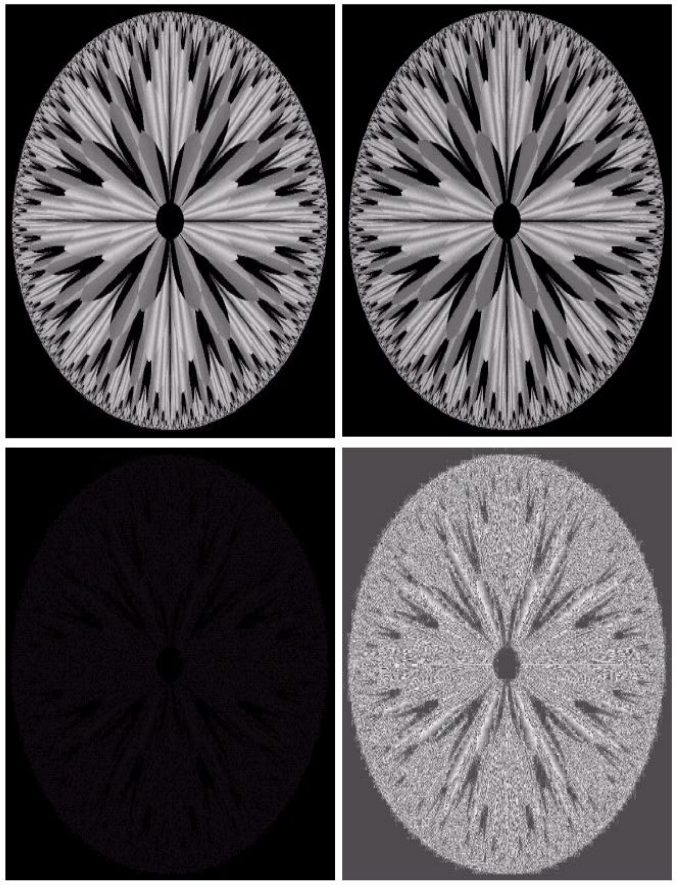
3. NOT op

$I = \text{NOT}$

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

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

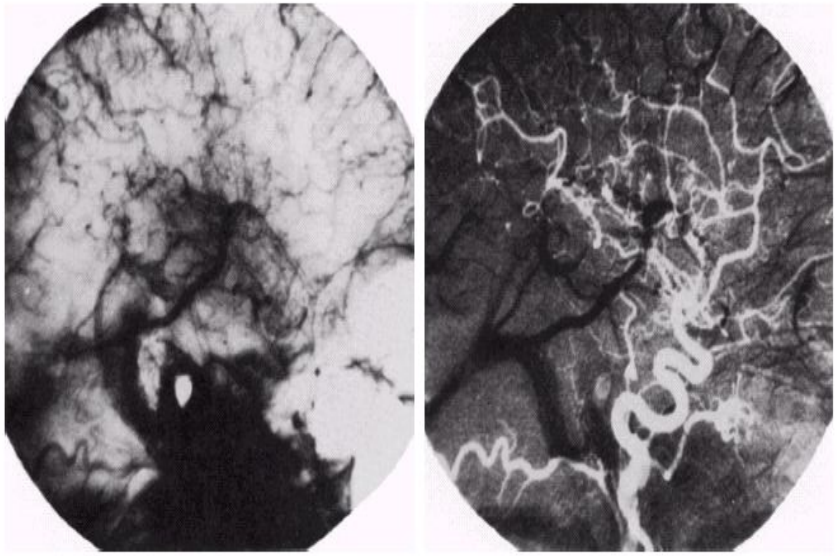
Image Subtraction



a b
c d

(a) original fractal image. (b) Result of setting the four lower-order bit planes to zero. (c) Difference between (a) and (b) . (d) Histogram equalized difference image.

Image Subtraction



a b

Enhancement by image subtraction. (a) Mask image. (b) An image with mask subtracted out.

Histogram Processing

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,…,L-1 and $p(r_k)$ can be considered to give an estimate of the probability of occurrence of gray level r_k.

HISTOGRAM

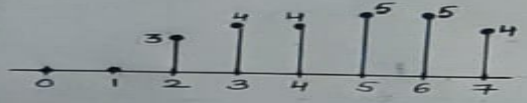
- An **image histogram** is a type of **histogram** that acts as a graphical representation of the tonal distribution in a digital **image**. It plots the number of pixels for each tonal value. ... Thus, the **histogram** for a very dark **image** will have most of its data points on the left side and 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 EQUALIZATION → Image Enhancement.

→ Graphical Representation → Data.
→ Image Processing → Data related to the Digital Image.
→ Representation → Frequency of occurance of various gray level's.

6	6	7	7	6
5	2	2	3	4
3	3	4	4	5
5	7	3	6	2
7	6	5	5	4

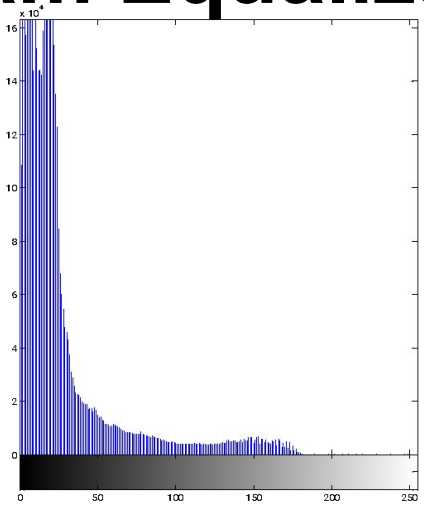
0 → 0
1 → 0
2 → 3
3 → 4
4 → 4
5 → 5
6 → 5
7 → 4

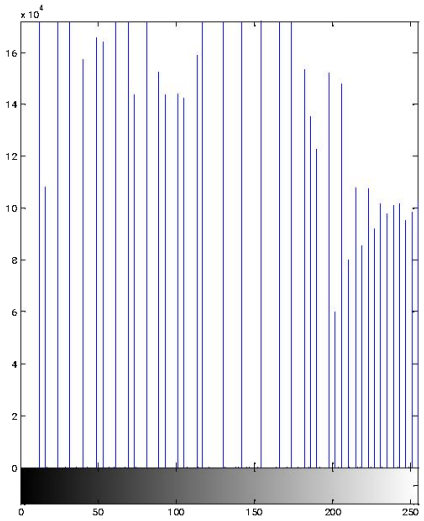


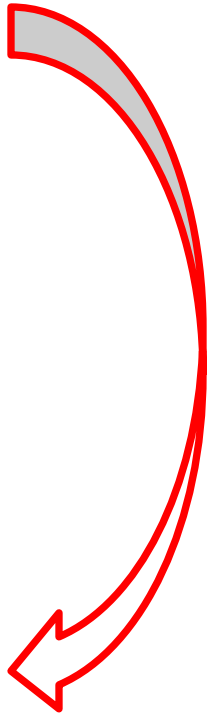
→ Use → Manipulating Contrast & Brightness.
→ Q ity → Normalizing → Histogram → Flat Profile

EC Academy

Histogram Equalization



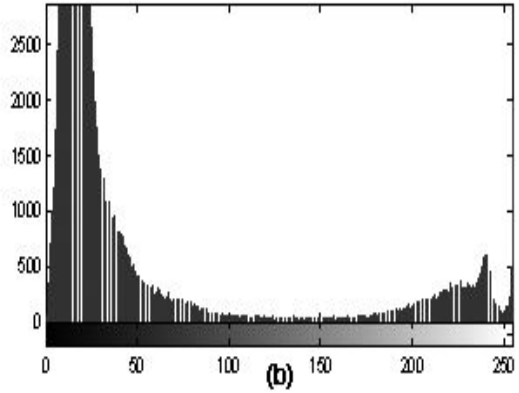




Histogram Equalization

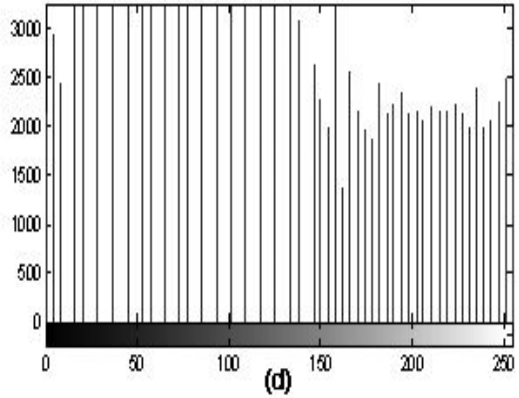
Histogram Equalization

(a)



(b)

(c)



(d)

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

Histogram Equalization

Histogram equalization : is a method which increases the dynamic range of the gray-levels in a low-contrast image to cover full range of gray-levels.

How-to-Do: is achieved by having a transformation function which is the Cumulative Distribution Function (CDF) of a given PDF of gray-levels in a given image.

Histogram Equalization

Histogram equalization : the new intensity value of pixel x is calculated by:

$$I(x) = \text{round}\left(\frac{cdf(x) - \min cdf}{1 - \min cdf} \times (L - 1) \right)$$

MATLAB CODE

- imhist(I1)
- I2=histeq(I1)

Histogram Equalization

Histogram equalization : the probability function of the output levels is uniform.

Note : the transformation function is simply the CDF.

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 NEIGHBORHOOD

- 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.

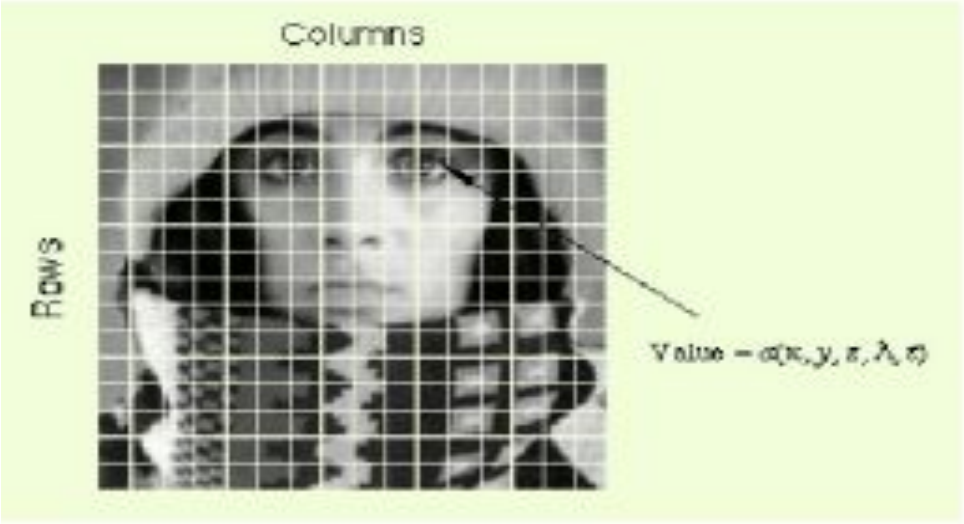


Figure 1: Digitization of a continuous image.

- **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.

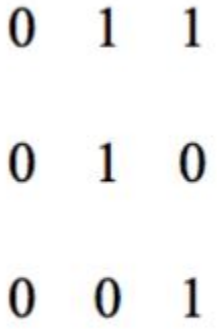


Fig: An arrangement of pixels

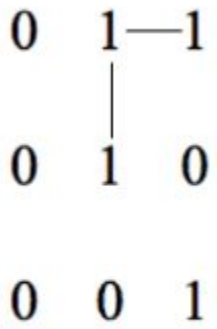


Fig: 4-connectivity of pixels

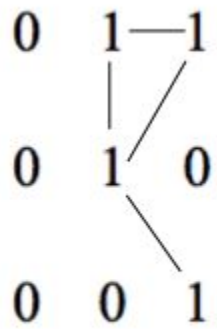


Fig: 8-connectivity of pixels

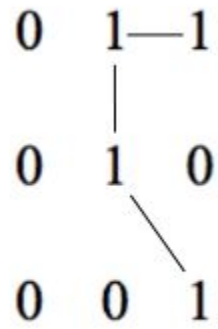


Fig: m-connectivity of pixels

SAMPLING TECHNIQUES

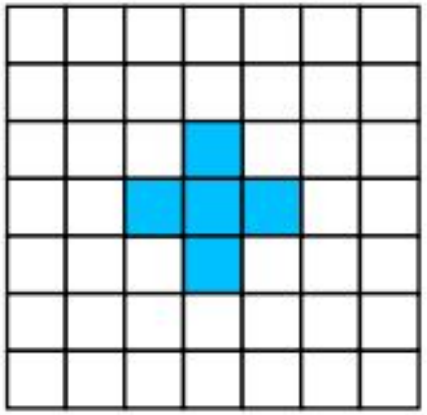


Figure 3a
Rectangular sampling
4-connected

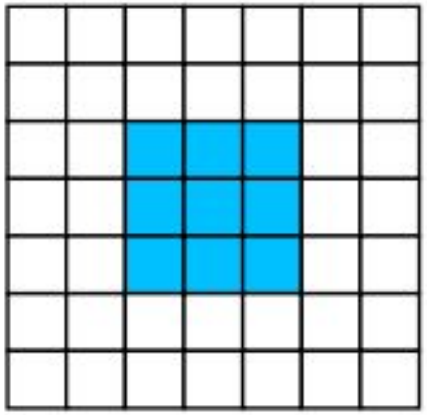
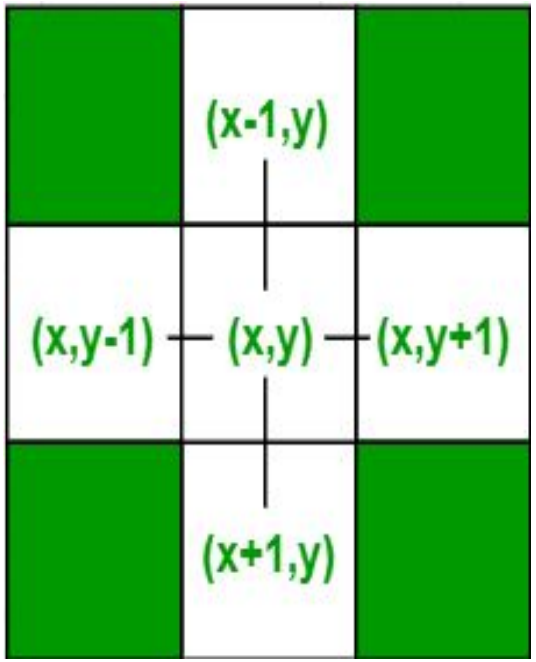


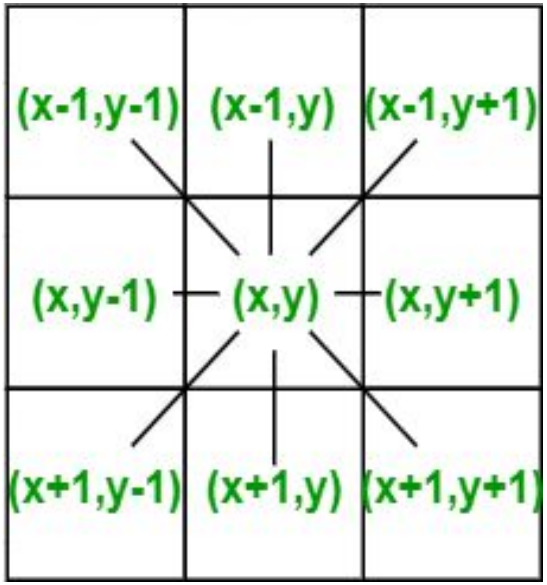
Figure 3b
Rectangular sampling
8-connected

Figure 3c
Hexagonal sampling
6-connected

4-Connected

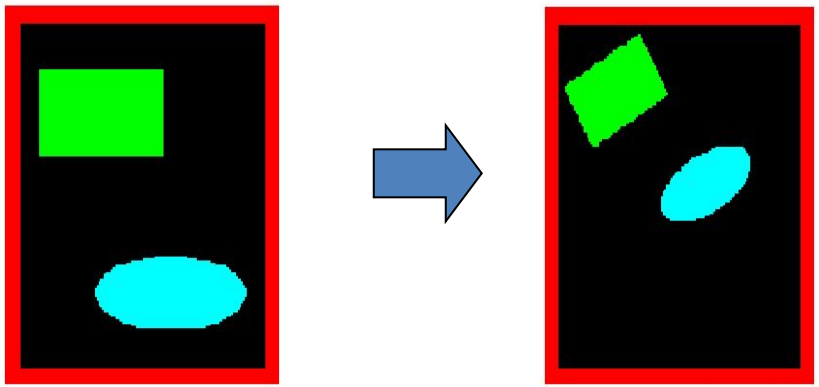


8-Connected



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

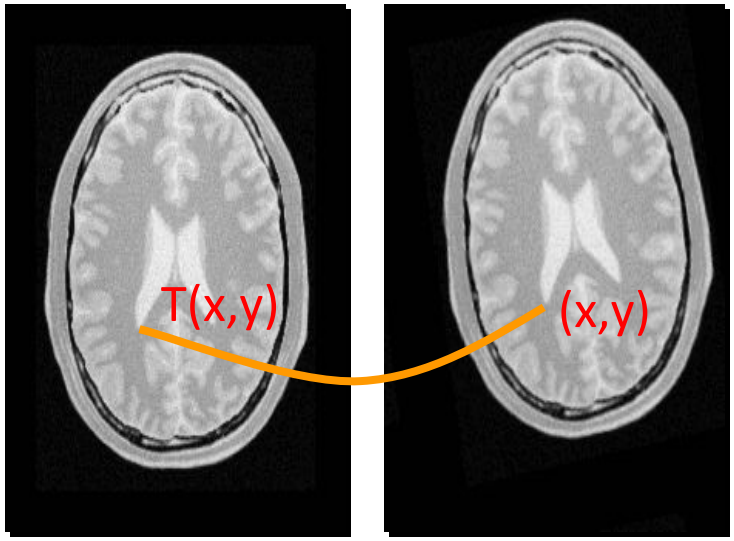
- A set of image transformations where the geometry of image is changed without altering its actual pixel values are commonly referred to as “Geometric” transformation. In general, you can apply multiple operations on it, but, the actual pixel values will remain unchanged.

- 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.
- Examples?

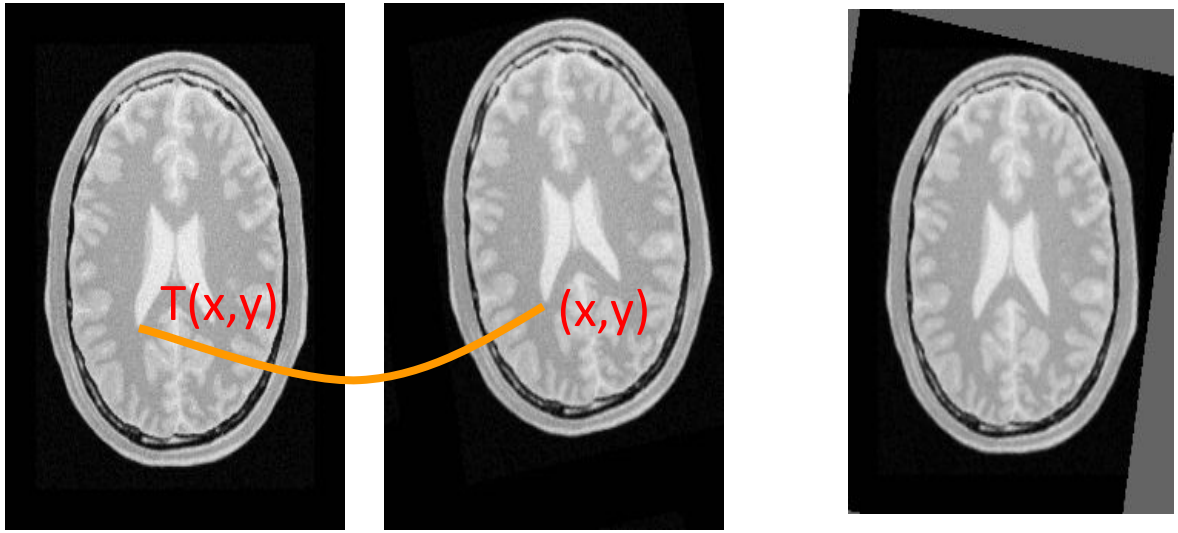
Geometric Transformations

- 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.



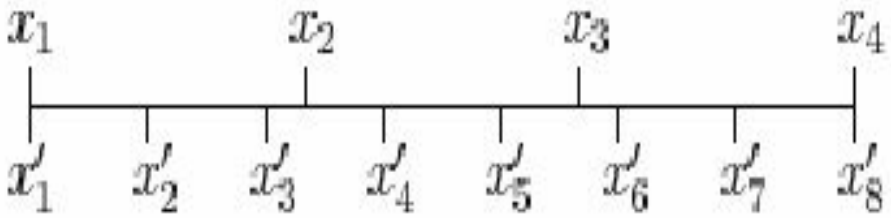
Fixed Image Moving Image xformed Moving Image

To follow the order in the text book, we'll discuss step 2 first.

Interpolation of Data: The primary use of interpolation is to help users, be they scientists, photographers, engineers or mathematicians, determine what data might exist outside of their collected data. Outside the domain of mathematics, interpolation is frequently used to scale images and to convert the sampling rate of digital signals.

$f(x)$

- Given a function at 4 points, how to "guess" values at other points?



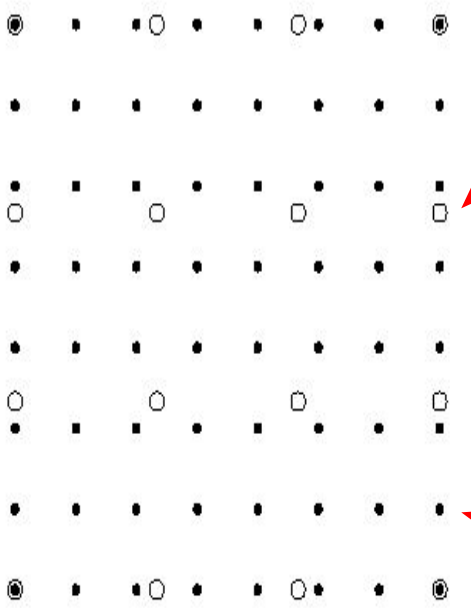
x_1, x_2, x_3, x_4 are original points

X'_i are new points

- Guessing at the function values within the known range is called interpolation.
- Interpolation has great significance in general image/video processing.

Another Example

Interpolation on an image (4x4 8x8) after scaling



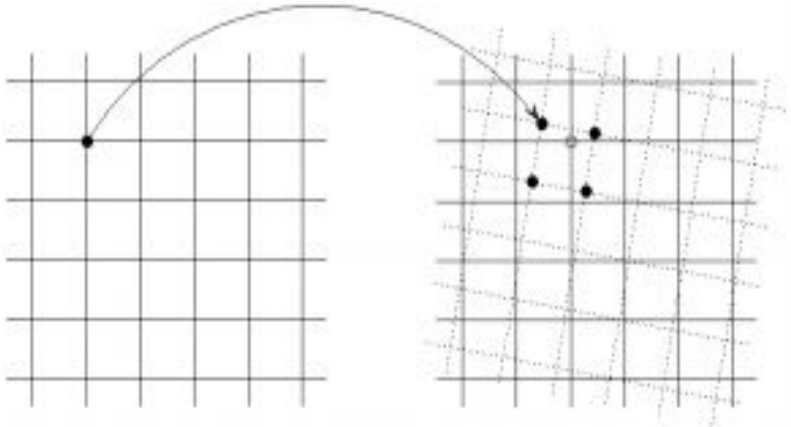
Open circle:
Original image pixel

Closed circle:
New pixels

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.

Linear interpolation

Linear interpolation

- explores four points neighboring the point (x,y), and assumes that the brightness function is linear in this neighborhood.

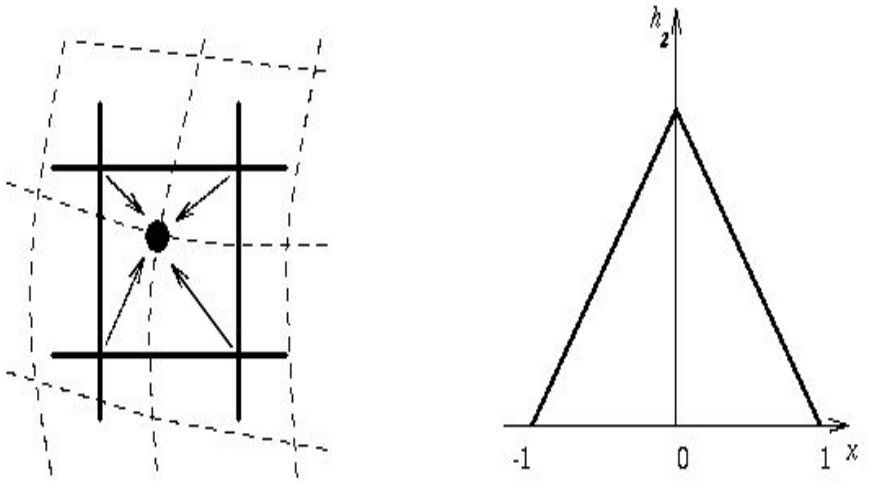


Figure 4.8 Linear interpolation.

Nearest neighbour

Nearest neighbor interpolation

- assigns to the point (x,y) the brightness value of the nearest point g in the discrete raster

$$f_1(x,y) = g_s(round\,(x),\ round\,(y)) \qquad (4.21)$$

- The right side of Figure shows how the new brightness is assigned.
- Dashed lines show how the inverse planar transformation maps the raster of the output image into the input image - full lines show the raster of the input image.

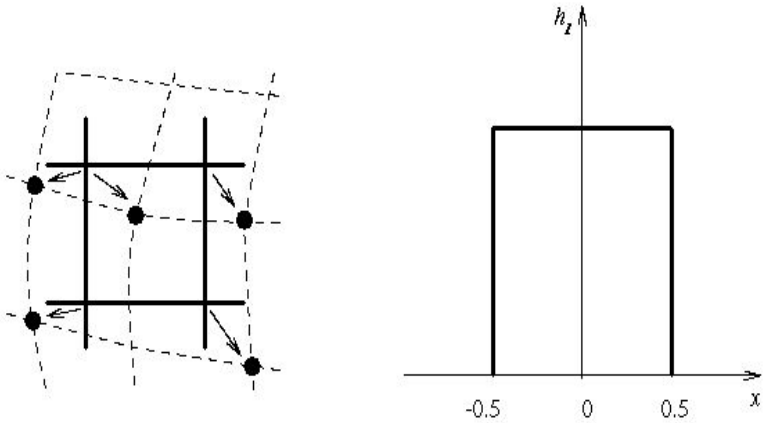


Figure 4.7 Nearest neighbourhood interpolation.

Important geometric transformation

- Rotation - by the angle phi about the origin

$$\begin{aligned} x' &= x\cos\phi + y\sin\phi \\ y' &= -x\sin\phi + y\cos\phi \\ J &= 1 \end{aligned}$$

- Change of scale - a in the x axis and b in the y axis

$$\begin{aligned} x' &= ax \\ y' &= bx \\ J &= ab \end{aligned}$$

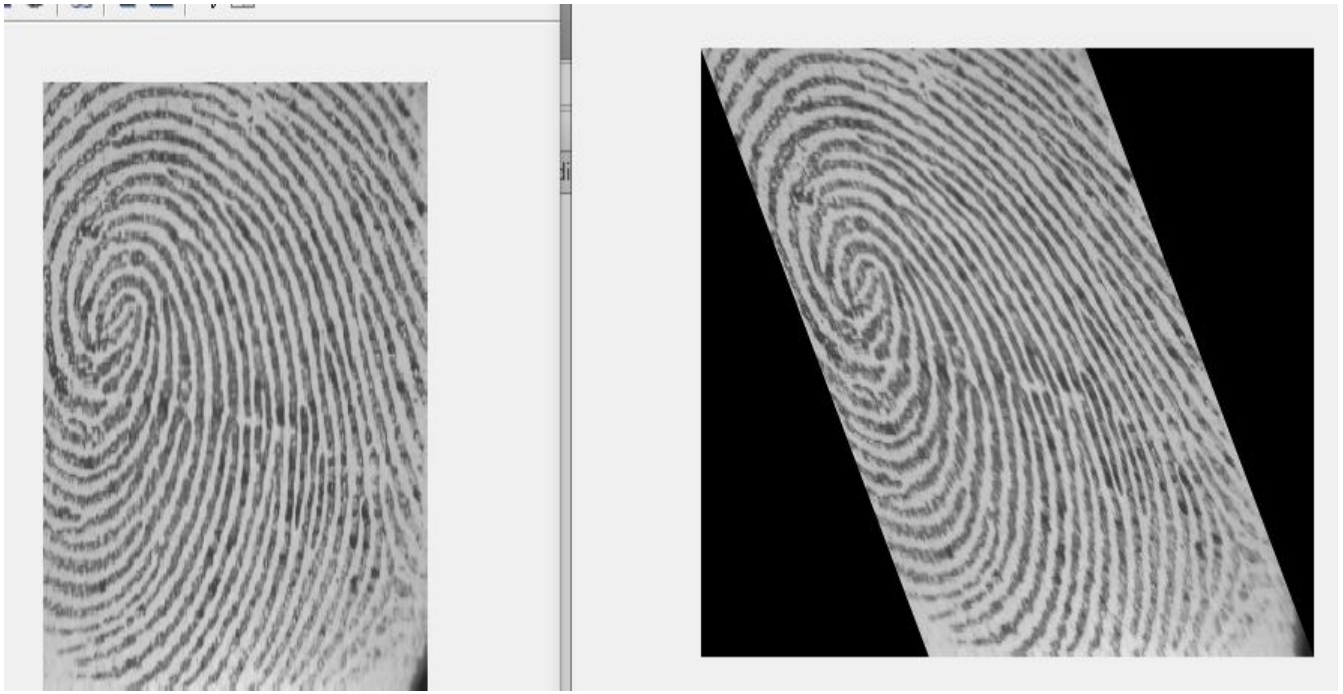
- Skewing by the angle phi

$$\begin{aligned} x' &= x + y\tan\phi \\ y' &= y \\ J &= 1 \end{aligned}$$

Sample code

- I3=imread("D:\sample\1_1.bmp")
- figure
- imshow(I3)
- tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
- J = imtransform(I3,tform);
- imshow(I3)
- figure
- imshow(J)

output



EDGES

Edges

- Abrupt change in the intensity of pixels.
- Discontinuity in image brightness or contrast.
- Usually edges occur on the boundary of two regions .

EDGE DETECTION

Edge Detection

- Process of identifying edges in an image to be used as a fundamental asset in image analysis.
- Locating areas with strong intensity contrasts.

EDGE DETECTION STEPS

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.

Methods of edge detection

Methods of Edge Detection

- Gradient methods (First Order Derivative)
  - local maxima and minima using first derivative in an image.
  - Compute Gradient magnitude horizontally and vertically.
- Zero-crossing methods (Second Order Derivative)
  - locate zeros in the second derivative of an image.
  - Laplacian of an Image.

FIRST DERIVATIVE

- 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.

TYPES

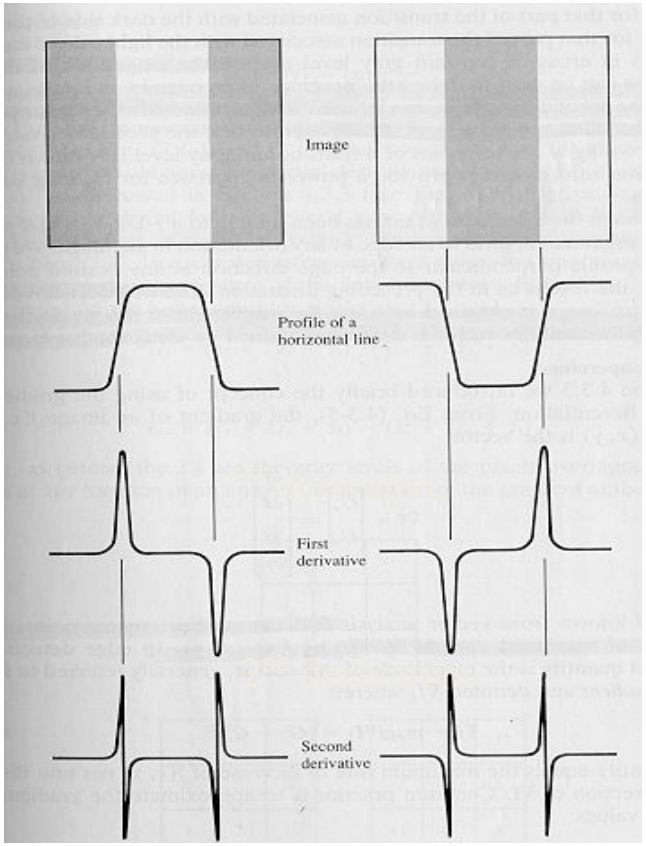
- Magnitude of **Gradient** vector is used for implementation of first derivative in image processing,
- **Laplacian** is for second order implementation in image processing.

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.



FIRST ORDER DERIVATIVE

- A basic definition of the first-order derivative of a one-dimensional function $f(x)$ is the difference

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

- Second-order derivative of $f(x)$ as the difference

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

First-Order Derivatives (Nonlinear) Image Sharpening:

First derivatives in image processing are implemented using the magnitude of the gradient. For a function $f(x, y)$, the gradient of f *at coordinates* (x, y) *is defined* as the two-dimensional column *vector*

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

This vector has the important geometrical property that it points in the direction of the greatest rate of change of *f at location (x, y).*

FIRST ORDER

First-Order Derivatives (Nonlinear) Image Sharpening:

The *magnitude (length) of vector denoted as M(x, y), where*

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

is the *value at (x, y) of the rate of change in the direction of the gradient vector.* Note that *M(x, y) is an image of the same size as the original, created when x and y are allowed to vary over all pixel locations in f. It is common practice* to refer to this image as the *gradient image (or simply as the gradient when the* meaning is clear).

Edge Detection Using First Derivative (Gradient)

2D functions:

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

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

Gradient Representation

- The gradient is a vector which has magnitude and direction:

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

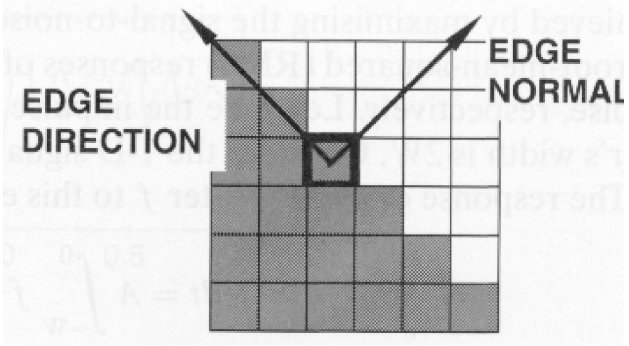
$$direction(grad(f)) = \tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$$

or

(approximation)

Magnitude: indicates edge strength.

- **Direction:** indicates edge direction.
  - i.e., perpendicular to edge direction



FIRST ORDER DERIVATIVE

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

LOW LEVEL FEATURE DETECTION

- Low level features are basic features that can be extracted automatically from an image without any shape information.(Information about spatial relationship)
- Thresholding is actually a form of low level feature extraction performed as point operation

Low level feature detection

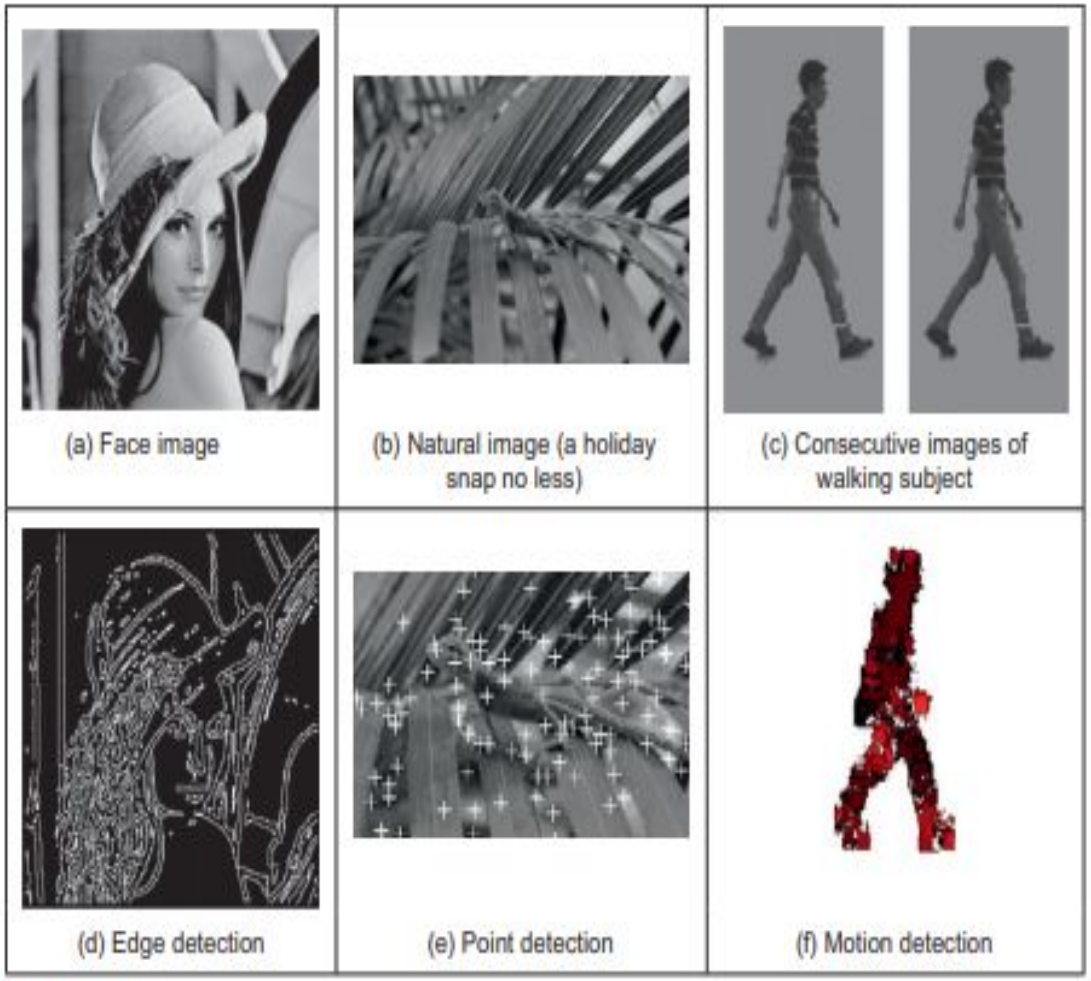


FIGURE 4.1

Low-level feature detection.

First order edge detection operators

- The boundary of an object is a step change in the intensity levels. The edge is at the position of the step change.
- To detect the edge position we can use first-order differentiation since this emphasizes changefirst-order differentiation gives no response when applied to signals that do not change

First order edge detection operators

- change in intensity can be revealed by differencing adjacent points.
- Differencing horizontally adjacent points will detect vertical changes in intensity and is often called a horizontal edge detector by virtue of its action.
- A horizontal operator will not show up horizontal changes in intensity since the difference is zero.

HORIZONTAL EDGE DETECTOR

- When applied to an image P the action of the horizontal edge detector forms the difference between two horizontally adjacent points, as such d

$$Ex_{x,y} = |P_{x,y} - P_{x+1,y}| \quad \forall x \in 1, N-1;\ y \in 1, N$$

VERTICAL EDGE DETECTOR

- In order to detect horizontal edges, we need a vertical edge detector which differences vertically adjacent points.
- This will determine horizontal intensity changes but not vertical ones, so the vertical edge detector detects the horizontal edges, Ey,

$$\mathrm{Ey}_{x,y} = |\mathrm{P}_{x,y} - \mathrm{P}_{x,y+1}| \quad \forall x \in 1, N;\ y \in 1, N-1$$

FIRST ORDER EDGE DETECTION

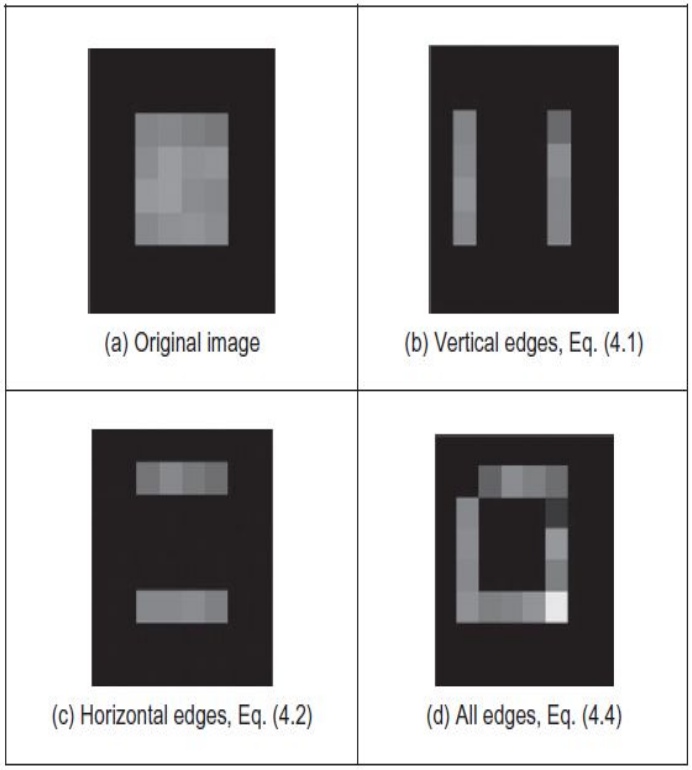


FIGURE 4.2

First-order edge detection.

BOTH EDGES

Combining the two gives an operator E that can detect vertical and horizontal edges **together**, that is,

$$\mathbf{E}_{x,y} = |\mathbf{P}_{x,y} - \mathbf{P}_{x+1,y} + \mathbf{P}_{x,y} - \mathbf{P}_{x,y+1}| \quad \forall x, y \in 1, N-1 \tag{4.3}$$

which gives:

$$\mathbf{E}_{x,y} = |2 \times \mathbf{P}_{x,y} - \mathbf{P}_{x+1,y} - \mathbf{P}_{x,y+1}| \quad \forall x, y \in 1, N-1 \tag{4.4}$$

Equation (4.4) gives the coefficients of a differencing template which can be convolved with an image to detect all the edge points, such as those shown in Figure 4.2(d). As in the previous chapter, the current point of operation (the posi-

ROBERT

- The Roberts cross operator (Roberts, 1965) was one of the earliest edge detection operators.
- It implements a version of basic first-order edge detection and uses two templates that differentiate pixel values in a diagonal manner, as *Mx My*

In implementation, the maximum value delivered by application of these templates is stored as the value of the edge at that point. The edge point $\mathbf{E}_{x,y}$ is then the maximum of the two values derived by convolving the two templates at an image point $\mathbf{P}_{x,y}$:

$$\mathbf{E}_{x,y} = \max\{|M^{+} * \mathbf{P}_{x,y}|, |M^{-} * \mathbf{P}_{x,y}|\} \quad \forall x, y \in 1, N-1 \qquad (4.11)$$

ROBERT CROSS OPERATOR TEMPLATE

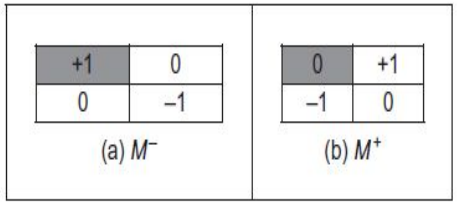


FIGURE 4.5

Templates for Roberts cross operator.

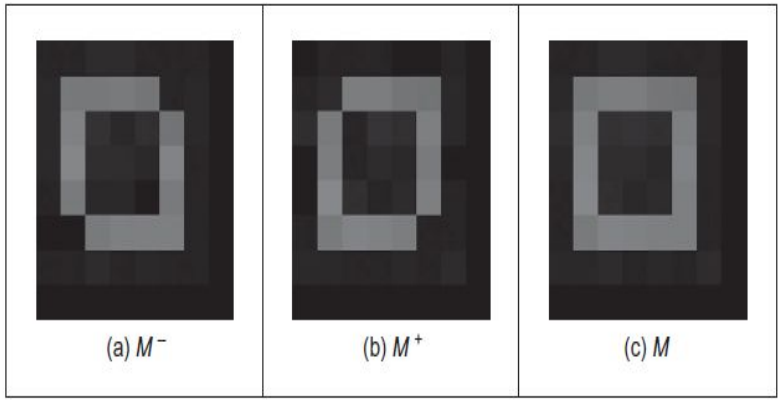


FIGURE 4.6

Applying the Roberts cross operator.

Robert edge detection steps

step 1: Input – Read an image
Step 2: Convert the true-color RGB image to the grayscale image
Step 3: Convert the image to double
Step 4: Pre-allocate the filtered_image matrix with zeros
Step 5: Define Robert Operator Mask
Step 6: Edge Detection Process (Compute Gradient approximation and magnitude of vector)
Step 7: Display the filtered image
Step 8: Thresholding on the filtered image
Step 9: Display the edge-detected image

Robert code

```
input_image = imread('[name of input image file].[file format]');
 % Displaying Input Image
input_image = uint8(input_image);
figure, imshow(input_image); title('Input Image');
% Convert the truecolor RGB image to the grayscale image
input_image = rgb2gray(input_image);
% Convert the image to double
input_image = double(input_image);
% Pre-allocate the filtered_image matrix with zeros
filtered_image = zeros(size(input_image));
% Robert Operator Mask
Mx = [1 0; 0 -1];
My = [0 1; -1 0];
```

```
    % Edge Detection Process
    % When i = 1 and j = 1, then filtered_image pixel
    % position will be filtered_image(1, 1)
    % The mask is of 2x2, so we need to traverse
    % to filtered_image(size(input_image, 1) - 1
    %, size(input_image, 2) - 1)
for i = 1:size(input_image, 1) - 1
   for j = 1:size(input_image, 2) - 1
      % Gradient approximations
      Gx = sum(sum(Mx.*input_image(i:i+1, j:j+1)));
      Gy = sum(sum(My.*input_image(i:i+1, j:j+1)));
      % Calculate magnitude of vector
      filtered_image(i, j) = sqrt(Gx.^2 + Gy.^2);
   end
end
```

- % Displaying Filtered Image

```
filtered_image = uint8(filtered_image);
figure, imshow(filtered_image); title('Filtered Image');
```

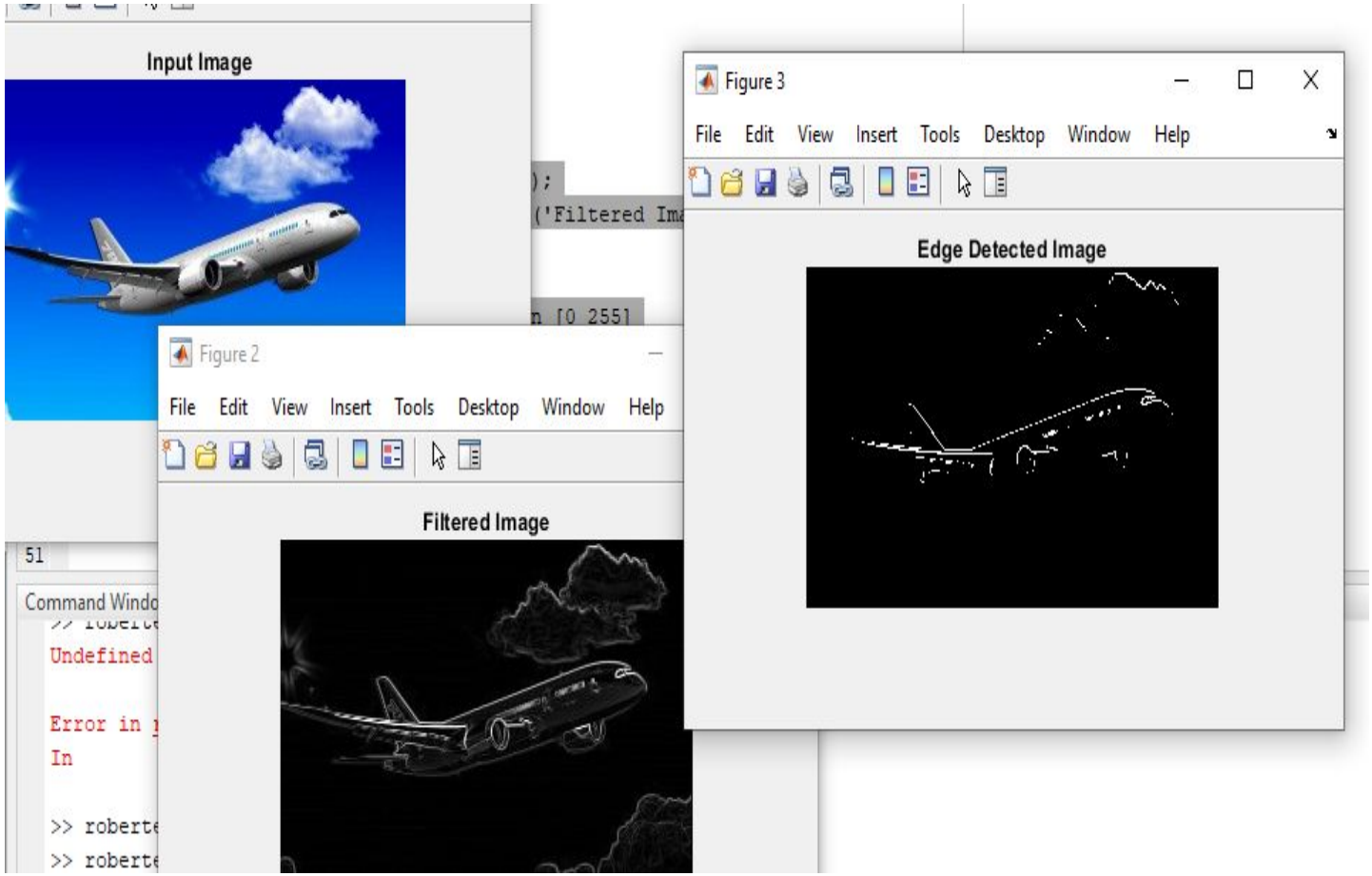
- % Define a threshold value

```
thresholdValue = 100; % varies between [0 255]
output_image = max(filtered_image, thresholdValue);
output_image(output_image == round(thresholdValue)) = 0;
```

- % Displaying Output Image

```
output_image = im2bw(output_image);
figure, imshow(output_image); title('Edge Detected Image');
```

output


Input Image
Figure 2
File Edit View Insert Tools Desktop Window Help
Filtered Image
Figure 3
File Edit View Insert Tools Desktop Window Help
Edge Detected Image
Command Windo
Undefined
Error in
In
>> roberte
>> roberte

ROBERT CROSS OPERATOR TEMPLATE

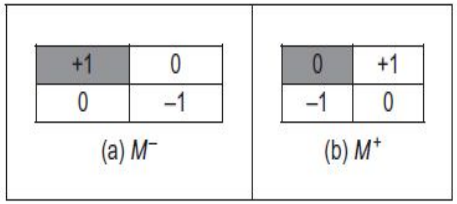


FIGURE 4.5

Templates for Roberts cross operator.

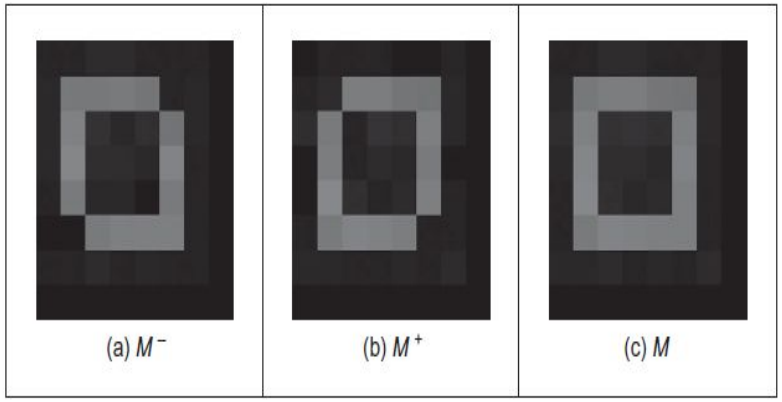


FIGURE 4.6

Applying the Roberts cross operator.

prewitt

- We can then extend the vertical template, Mx, along three rows, and the horizontal template, My, along three columns. These give the Prewitt edge-detection operator (Prewitt and Mendelsohn, 1966) that consists of two templates (Figure 4.8).
- This gives two results: the rate of change of brightness along each axis. As such, this is the vector illustrated in Figure 4.7:
- the edge magnitude, M, is the length of the vector and the edge direction, θ, is the angle of the vector.

$$M(x,y) = \sqrt{Mx(x,y)^2 + My(x,y)^2} \qquad (4.12)$$

$$\theta(x,y) = \tan^{-1}\left(\frac{My(x,y)}{Mx(x,y)}\right) \qquad (4.13)$$

Templates of prewitt

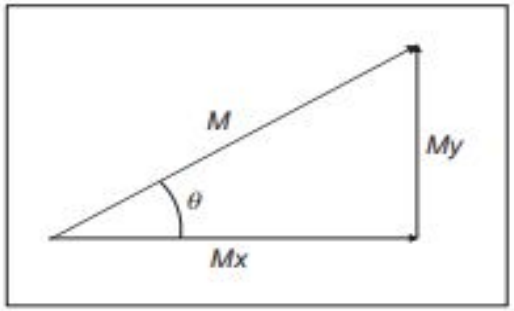


FIGURE 4.7

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*

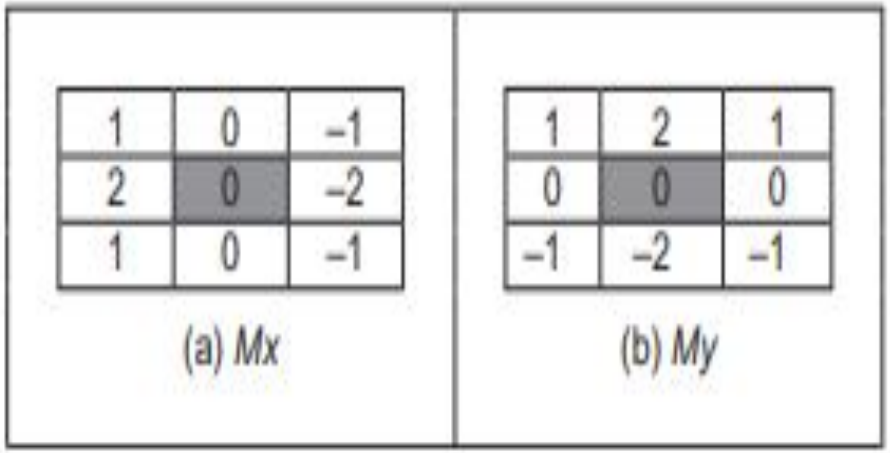
FIGURE 4.8

Templates for Prewitt operator.

sobel

- When the weight at the central pixels, for both Prewitt templates, is doubled, this gives the famous Sobel edge-detection operator which, again, consists of two masks to determine the edge in vector form.
- The Sobel operator was the most popular edge-detection operator until the development of edge-detection techniques with a theoretical basis.
- It proved popular because it gave, overall, a better performance than other contemporaneous edge-detection operators, such as the Prewitt operator. The templates for the Sobel operator can be found in Figure 4.10

Templates of sobel operator



1	0	−1
2	0	−2
1	0	−1

(a) *Mx*

1	2	1
0	0	0
−1	−2	−1

(b) *My*

FIGURE 4.10

Templates for Sobel 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 DERIVATIVE

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

SECOND ORDER

- **Second Derivative for Image Sharpening:**

- This approach consists of defining a discrete formulation of the second-order derivative and then constructing a filter mask based on that formulation.
- We are interested in *isotropic filters, whose response is independent of the direction of* the discontinuities in the image to which the filter is applied.
- Isotropic filters are *rotation invariant, in the sense that rotating the image and* then applying the filter gives the same result as applying the filter to the image first and then rotating the result.
- The simplest *isotropic derivative* operator is the Laplacian, which, for a function (image) $f(x,y)$ of two variables, is defined as

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

SECOND DERIVATIVE….

- **Second Derivative for Image Sharpening:**

- 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

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

LAPLACIAN

- **Second Derivative for Image Sharpening:**

0	1	0
1	−4	1
0	1	0

Filter mask used to implement discrete Laplacian of two variables

Isotropic result for rotations in increments of 90^0

1	1	1
1	−8	1
1	1	1

Mask used to implement an extension of this equation that includes the diagonal terms

Isotropic results in increments of 45°

0	−1	0
−1	4	−1
0	−1	0

Implementations of the Laplacian

−1	−1	−1
−1	8	−1
−1	−1	−1

Implementations of the Laplacian

- **Second Derivative for Image Sharpening:**

- As Laplacian is a derivative operator, its use highlights intensity discontinuities in an image and deemphasizes regions with slowly varying intensity levels.

- This will produce images that have grayish edge lines and other discontinuities.

- If the definition used has a negative center coefficient, then we *subtract, rather than add, the Laplacian image to obtain a* sharpened result.

- Thus, the basic way in which we use the Laplacian for image sharpening is

$$g(x, y) = f(x, y) + c\left[\nabla^2 f(x, y)\right]$$

The constant is c = -1, if the Laplacian filters are mask 1 or 2 used, and c = 1 if either of the other two filters is used.

gaussian

More formally, given a pixel (x, y), the Laplacian L(x,y) of an image with intensity values I_i can be written mathematically as follows:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

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

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.

LOG

laplacian of guassian: LOG

- Which was invented by Marr and Hildreth (1980) who combined Gaussian filtering with the Laplacian.
- This algorithm is not used frequently in machine vision.
- The second derivative of smoothed step edge is a function that crosses zero at the location of edge.

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:

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

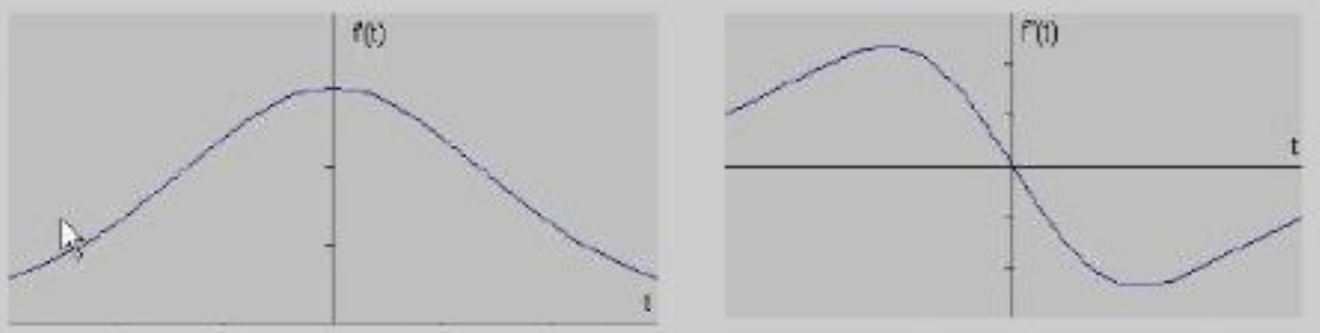
The above equation is continuous, so we need to discretize it so that we can use it on discrete pixels in an image.

Here is an example of a LoG approximation kernel where $\sigma = 1.4$. This is just

Zero crossing

Zero Crossing based Edge Detection

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



Cont. Zero Crossing based Edge Detection - LOG

- Defined as:

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

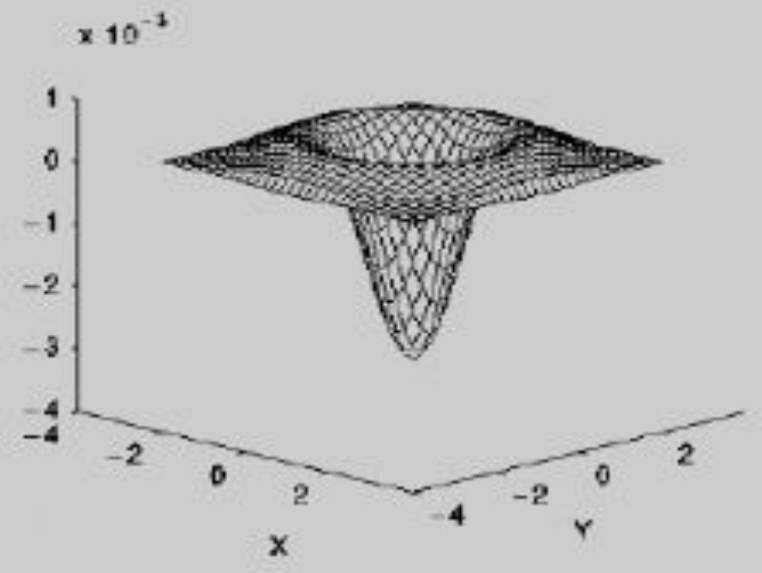
- Greater the value of σ, broader is the Gaussian filter, more is the smoothing

Cont. Zero Crossing based Edge Detection

- Laplacian of Gaussian

1	1	1
1	8	1
1	1	1

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



Cont. Zero Crossing based Edge Detection - LOG

- 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

log

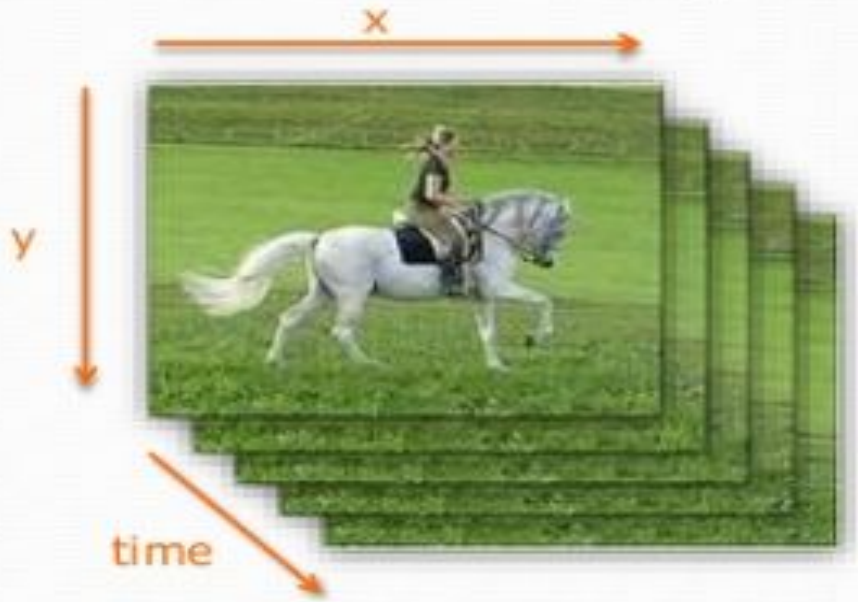
- Computationally cheaper to implement since we can combine the two filters into one filter but it.
- Doesn't provide information about the direction of the edge.
- Probability of false and missing edges remain.
- Localization is better than Gradient Operators

Laplacian Of Gaussain

Low level feature Extraction-DESCRIBING IMAGE MOTION

Describing image motion

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



DESCRIBING IMAGE MOTION

Describing image motion

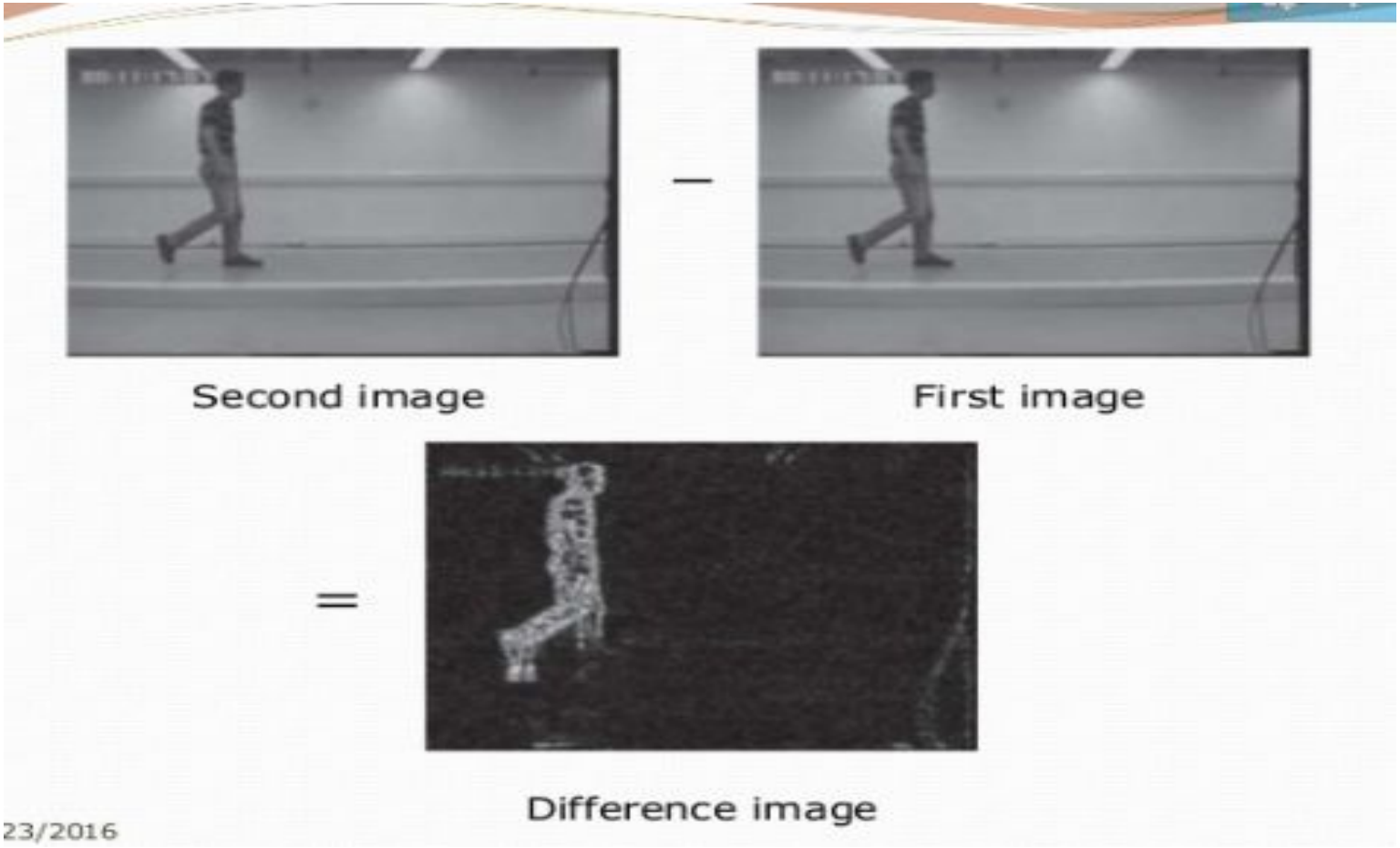
- The simplest way we can detect *motion* is by image differencing.

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

$D(t) = 0$
no motion

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

DIFFERENCE IMAGE



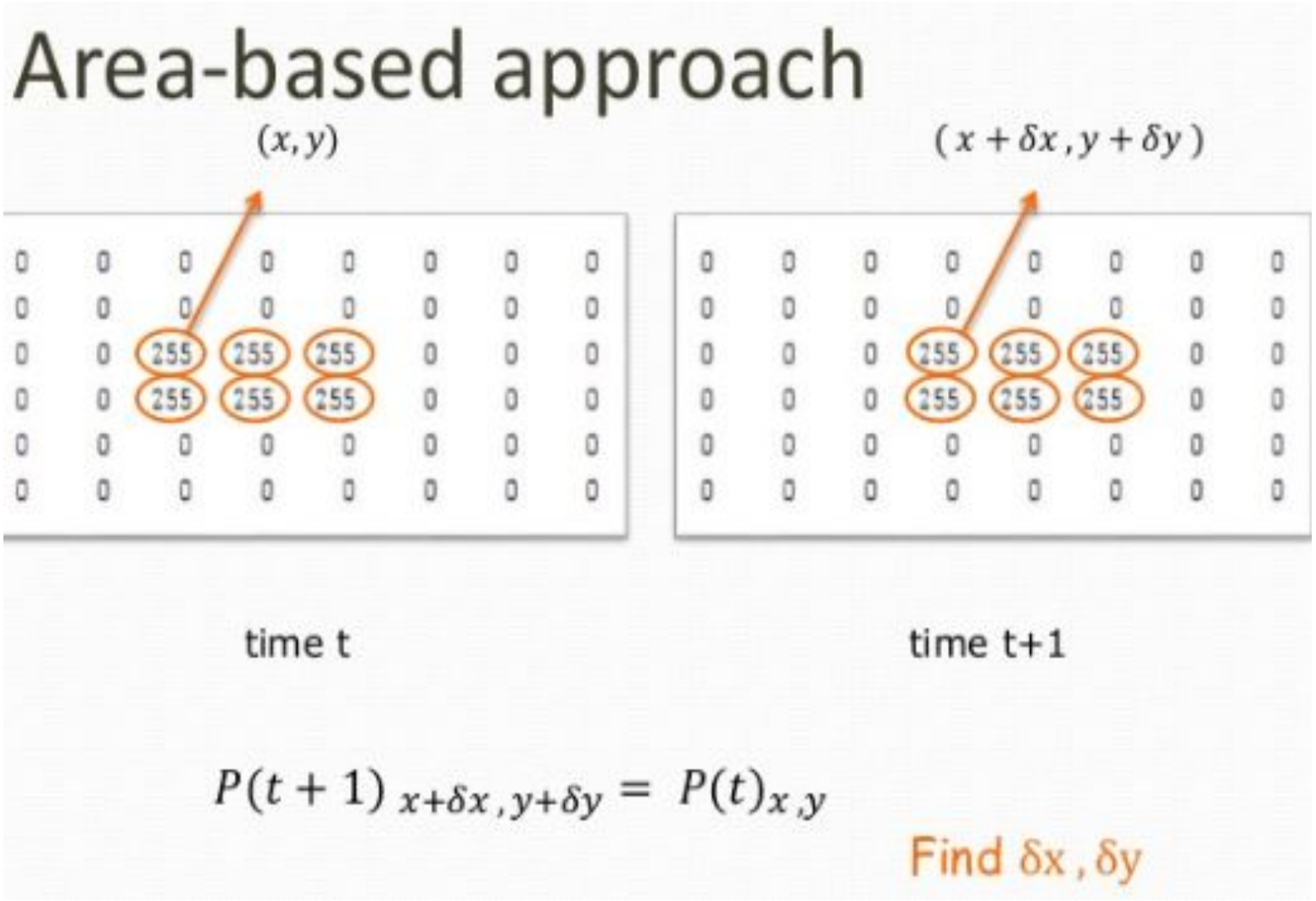
APPROACHES TO DETECT MOTION

Describing image motion

There are a lot of approaches to detect the motion in the image such as:

- Area-based approach.
- Differential approach.

AREA BASED APPROACH

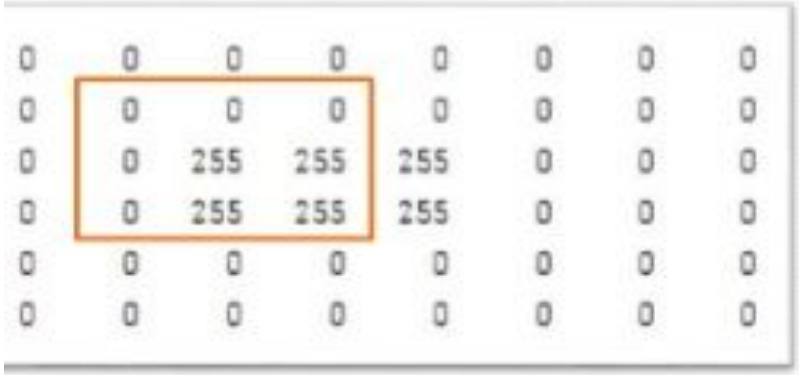


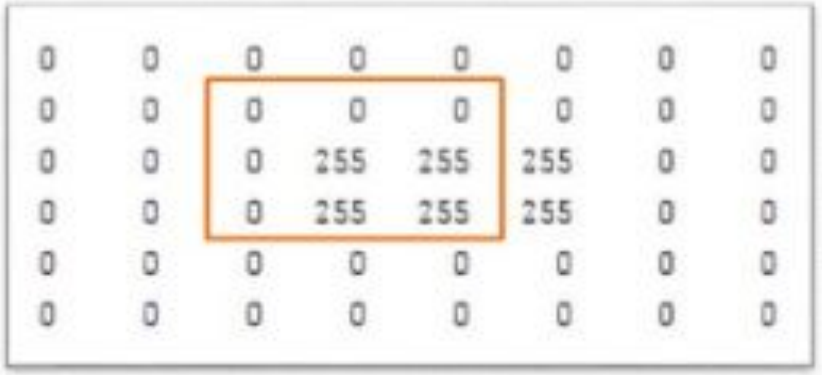
AREA BASED APROACH

Area-based approach

- Consider neighborhood pixels.

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





AREA BASED APPROACH

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.

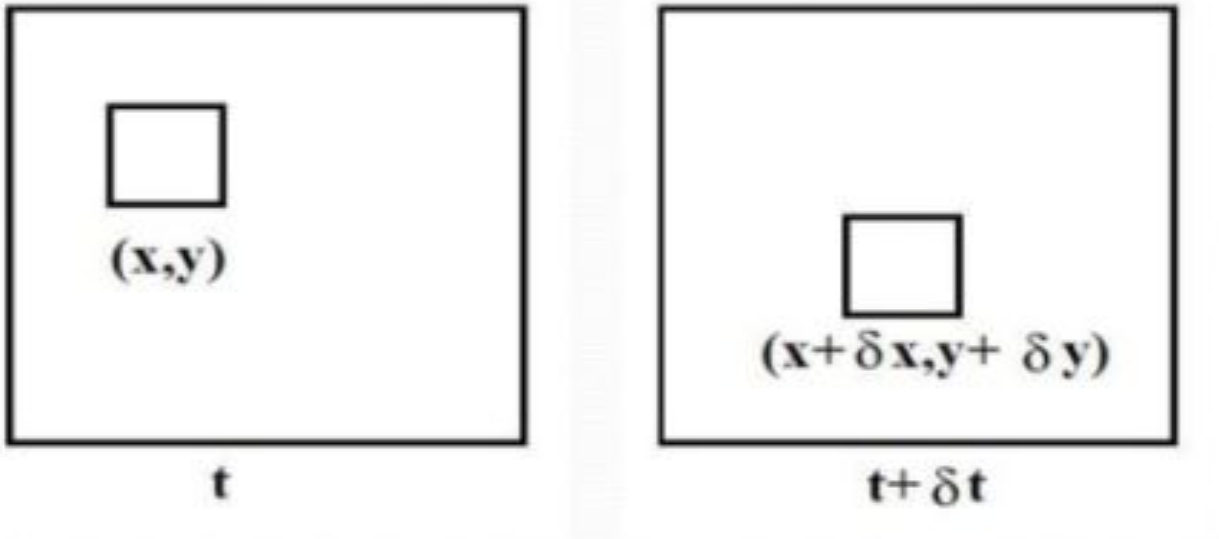
- *The neighboring* points move with *similar* velocity.

DIFFERENTIAL APPROACH

Differential approach

- Differential techniques compute image velocity from derivatives of image intensities:

$$P(x, y, t) \text{ and } P(x + \delta x, y + \delta y, t + \delta t)$$



DIFFERENTIAL APPROACH

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

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.

DIFFEENTIAL APPROACH

Differential approach

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

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

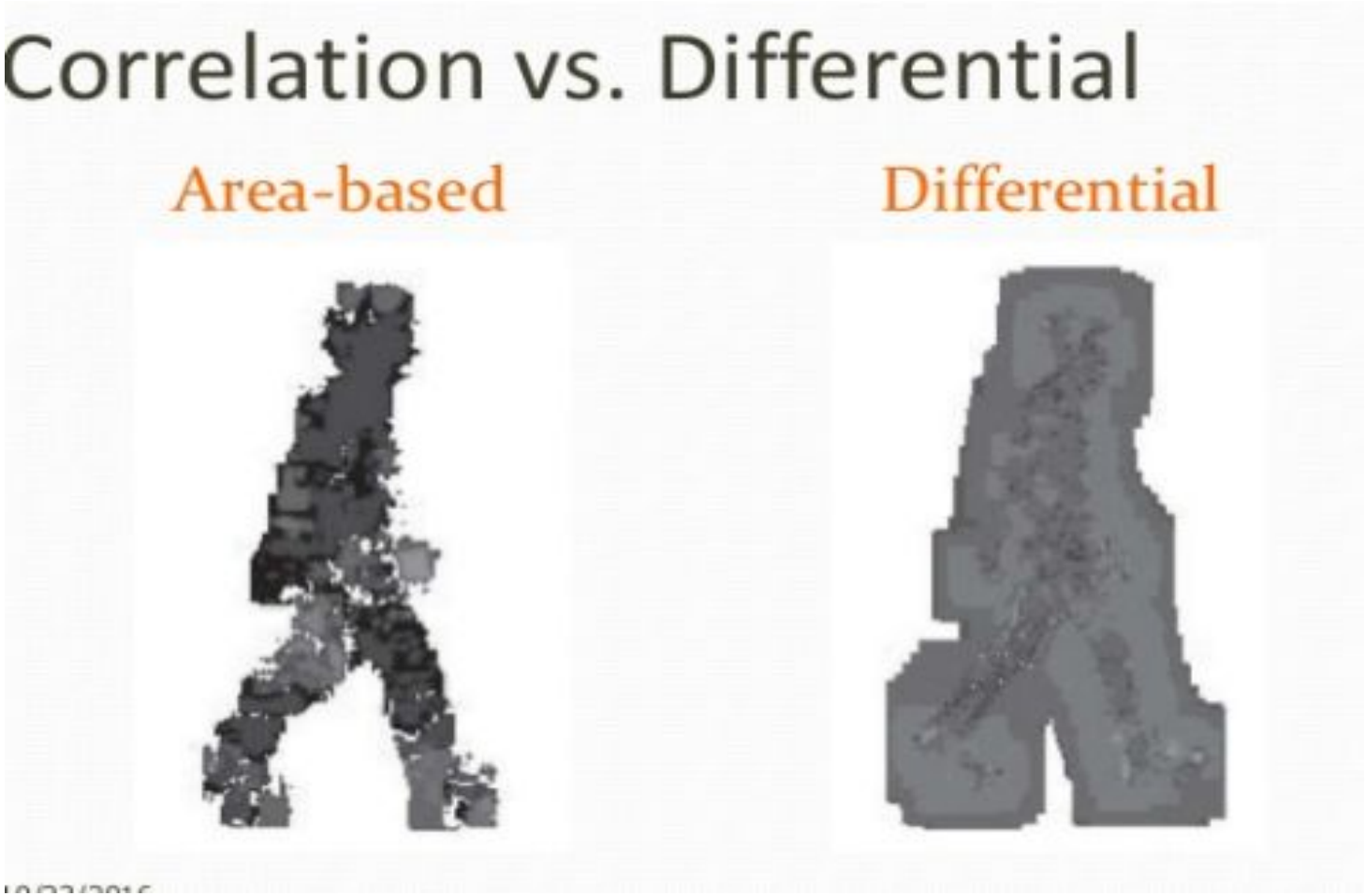
$$v_{x,y}^{n+1} = \bar{v}_{x,y}^{n} - \lambda \left(\frac{\nabla x_{x,y}\, \bar{u}_{x,y} + \nabla y_{x,y}\, \bar{v}_{x,y} + \nabla t_{x,y}}{1 + \lambda \left(\nabla x_{x,y}^{2} + \nabla x_{x,y}^{2} \right)} \right) (\nabla y_{x,y})$$

DIFFERENCE

Correlation vs. Differential

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

DIFFERENCE



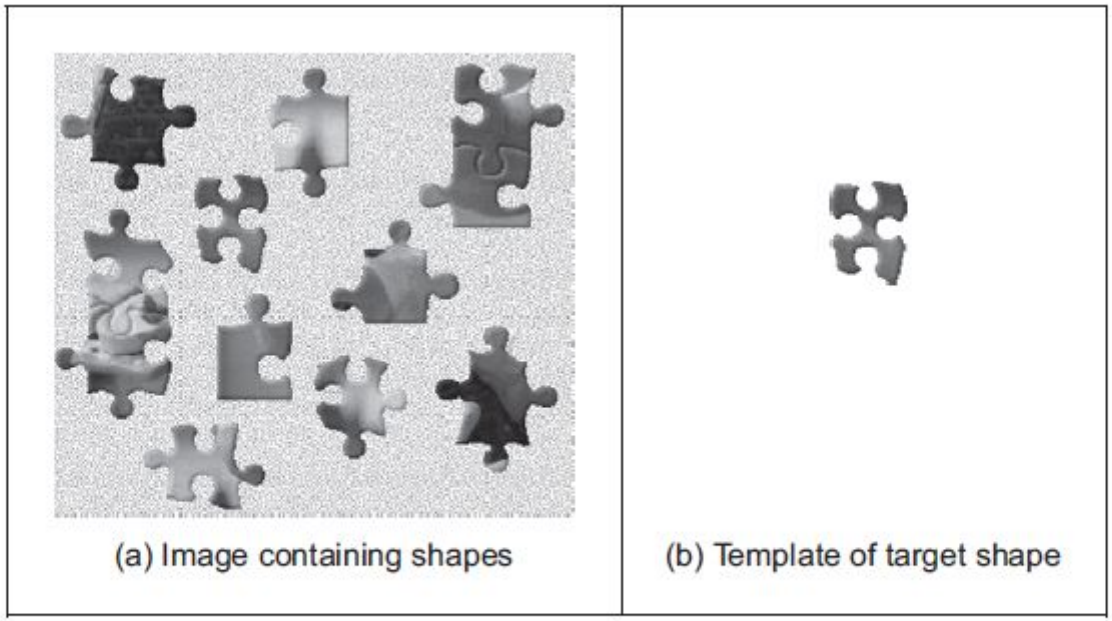
HIGH LEVEL FEATURES

- 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

FIGURE 5.3

Illustrating template matching.[1]

Template matching

```
%Template Matching Implementation

function accum=TMatching(inputimage,template)

%Image size & template size
  [rows,columns]=size(inputimage);
  [rowsT,columnsT]=size(template);

  %Centre of the template
cx=floor(columnsT/2)+1;    cy=floor(rowsT/2)+1;

%Accumulator
accum=zeros(rows,columns);
%Template Position
for i=cx:columns-cx
  for j=cy:rows-cy
     %Template elements
       for x=1-cx:cx-1
         for y=1-cy:cy-1
             err=(double(inputimage(j+y,i+x))
             -double(template(y+cy,x+cx)))^2;
         accum(j,i)=accum(j,i)+err;
       end
     end
  end
end
```

- The Matlab code to implement template matching is the function TMatching

given in Code 5.1.

This function first clears an accumulator array, accum, then searches the whole picture, using pointers i and j, and then searches the whole template for matches, using pointers x and y. Note that the position of the template is given by its center. The accumulator elements are incremented according Eq. (5.7).

The accumulator array is delivered as the result. The match for each position is stored in the array. After computing all the matches, the minimum element in the array defines the position where most pixels in the template matched those in the image.

As such, the minimum is deemed to be the coordinates of the point where the template's shape is most likely to lie within the original image.

It is possible to implement a version of template matching without the accumulator array, by storing the location of the minimum alone. This will give the same

result though it requires little storage.

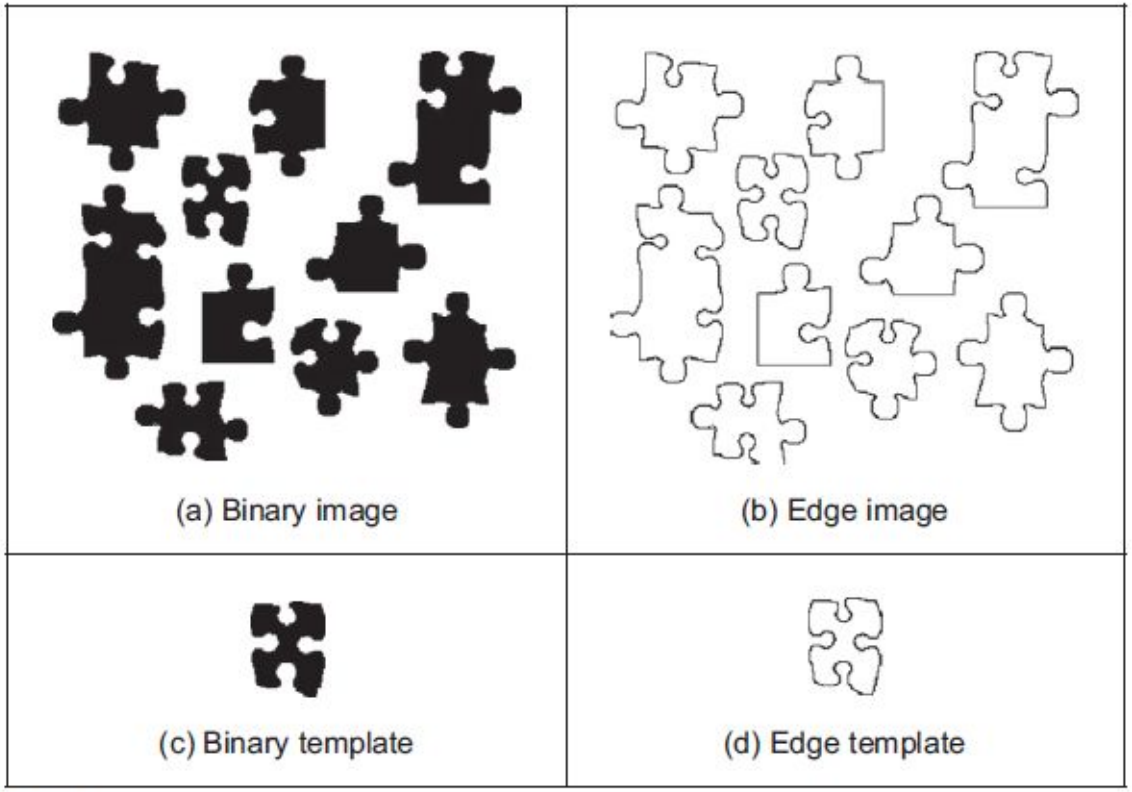


FIGURE 5.4

Example of binary and edge template matching.

Hough transform

- The Hough transform (HT) (Hough, 1962) is a technique that locates shapes in images. In particular, it has been used to extract lines, circles, and ellipses

5.5.2 Lines

We will first consider finding lines in an image. In a Cartesian parameterization, collinear points in an image with coordinates (x,y) are related by their slope m and an intercept c according to

$$y = mx + c \tag{5.24}$$

This equation can be written in homogeneous form as

$$Ay + Bx + 1 = 0 \tag{5.25}$$

where $A = -1/c$ and $B = m/c$. Thus, a line is defined by giving a pair of values (A,B). However, we can observe a symmetry in the definition in Eq. (5.25). This

Ht for lines

```
%Hough Transform for Lines
function HTLine(inputimage)
%image size
[rows,columns]=size(inputimage);
%accumulator
acc1=zeros(rows,91);
acc2=zeros(columns,91);
%image
for x=1:columns
  for y=1:rows
     if(inputimage(y,x)==0)
      for m=-45:45
        b=round(y-tan((m*pi)/180)*x);
        if(b<rows & b>0)
           acc1(b,m+45+1)=acc1(b,m+45+1)+1;
         end
       end
       for m=45:135
         b=round(x-y/tan((m*pi)/180));
         if(b<columns & b>0)
           acc2(b,m-45+1)=acc2(b,m-45+1)+1;
         end
       end
    end
  end
end
```

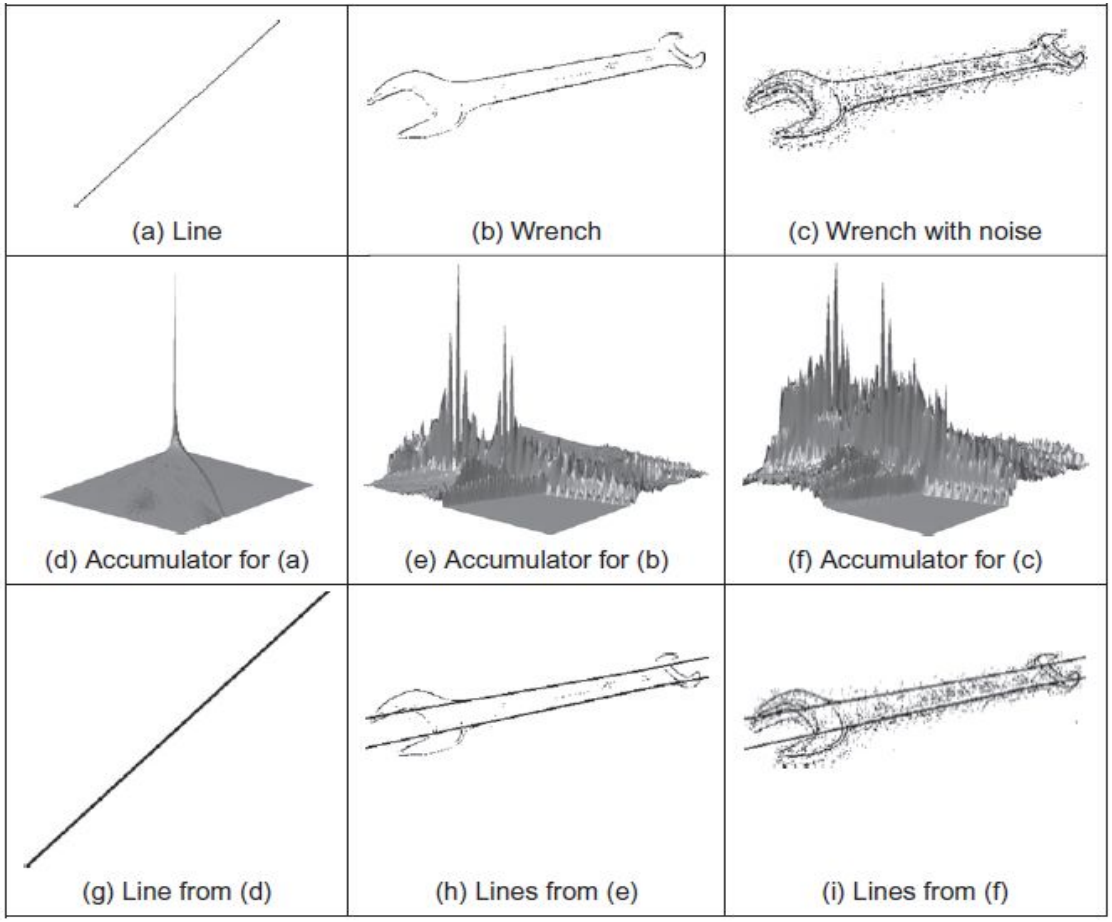

(a) Line
(b) Wrench
(c) Wrench with noise
(d) Accumulator for (a)
(e) Accumulator for (b)
(f) Accumulator for (c)
(g) Line from (d)
(h) Lines from (e)
(i) Lines from (f)

FIGURE 5.15

Circles

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.

HT for circle

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**.

HT for Circles

```
%Hough Transform for Circles

function HTCircle(inputimage,r)

%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-r*cos(t));
            y0=round(y-r*sin(t));
            if(x0<columns & x0>0 & y0<rows & y0>0)
               acc(y0,x0)=acc(y0,x0)+1;
             end
         end
      end
   end
end
```

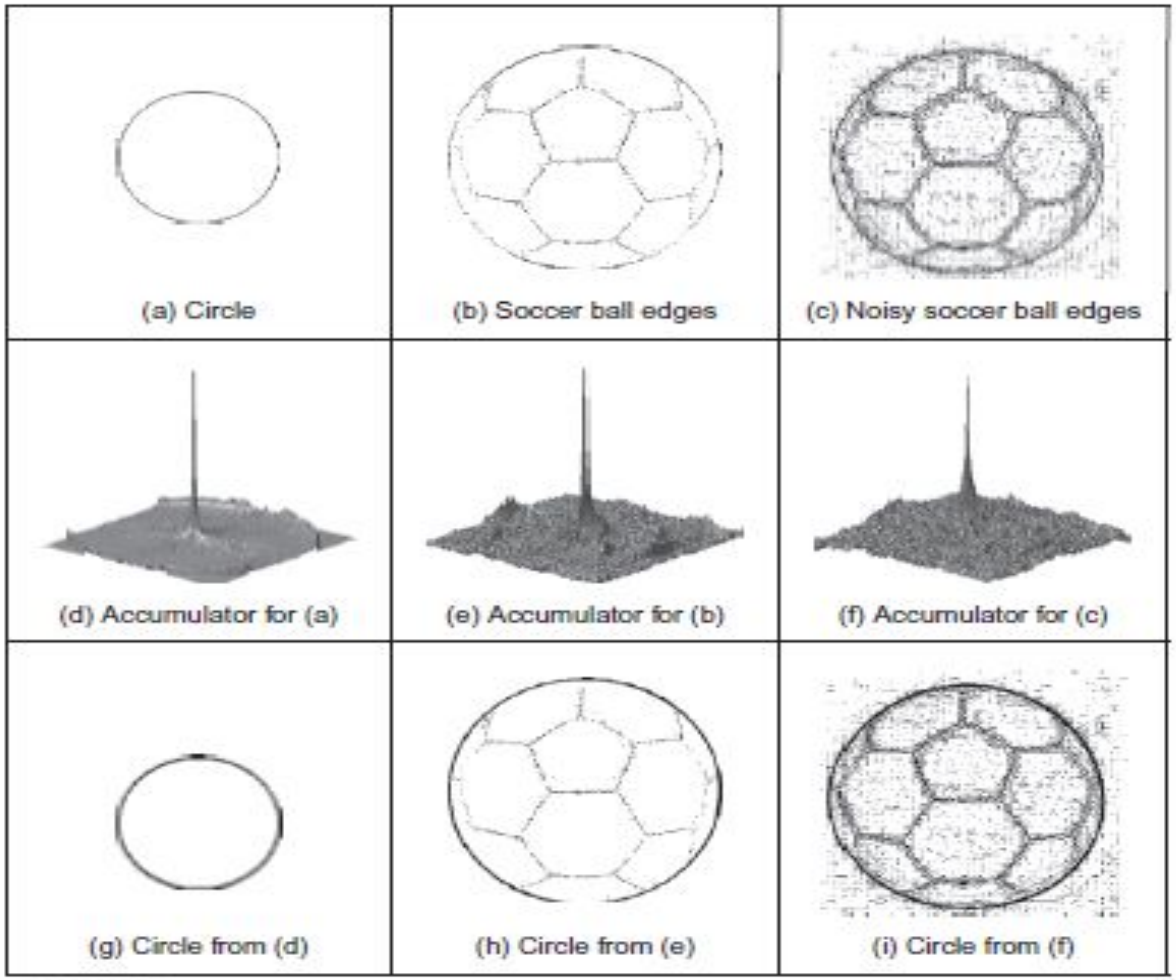


FIGURE 5.20

Applying the HT for circles.

Ellipses

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

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

then the circle is deformed into

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

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,

Ellipses

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

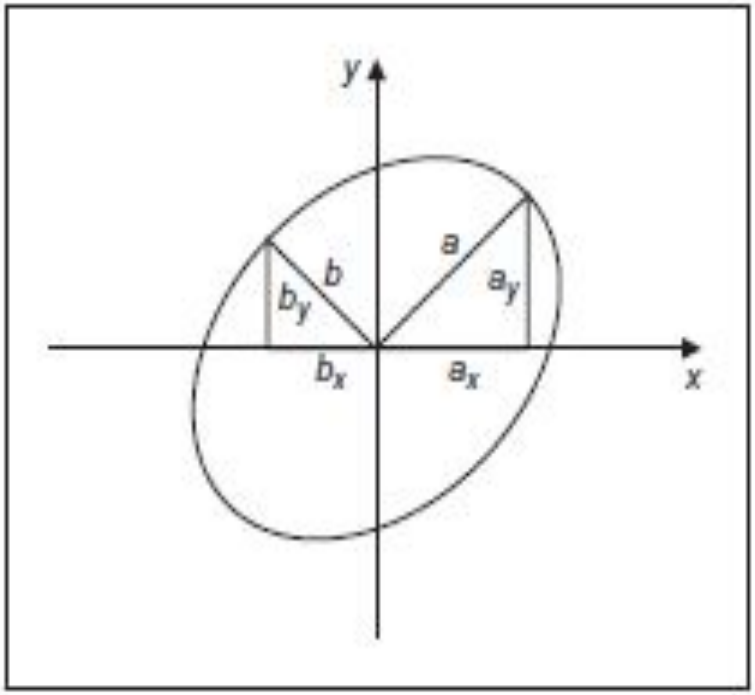


FIGURE 5.22

Definition of ellipse axes.

HT for ellipses

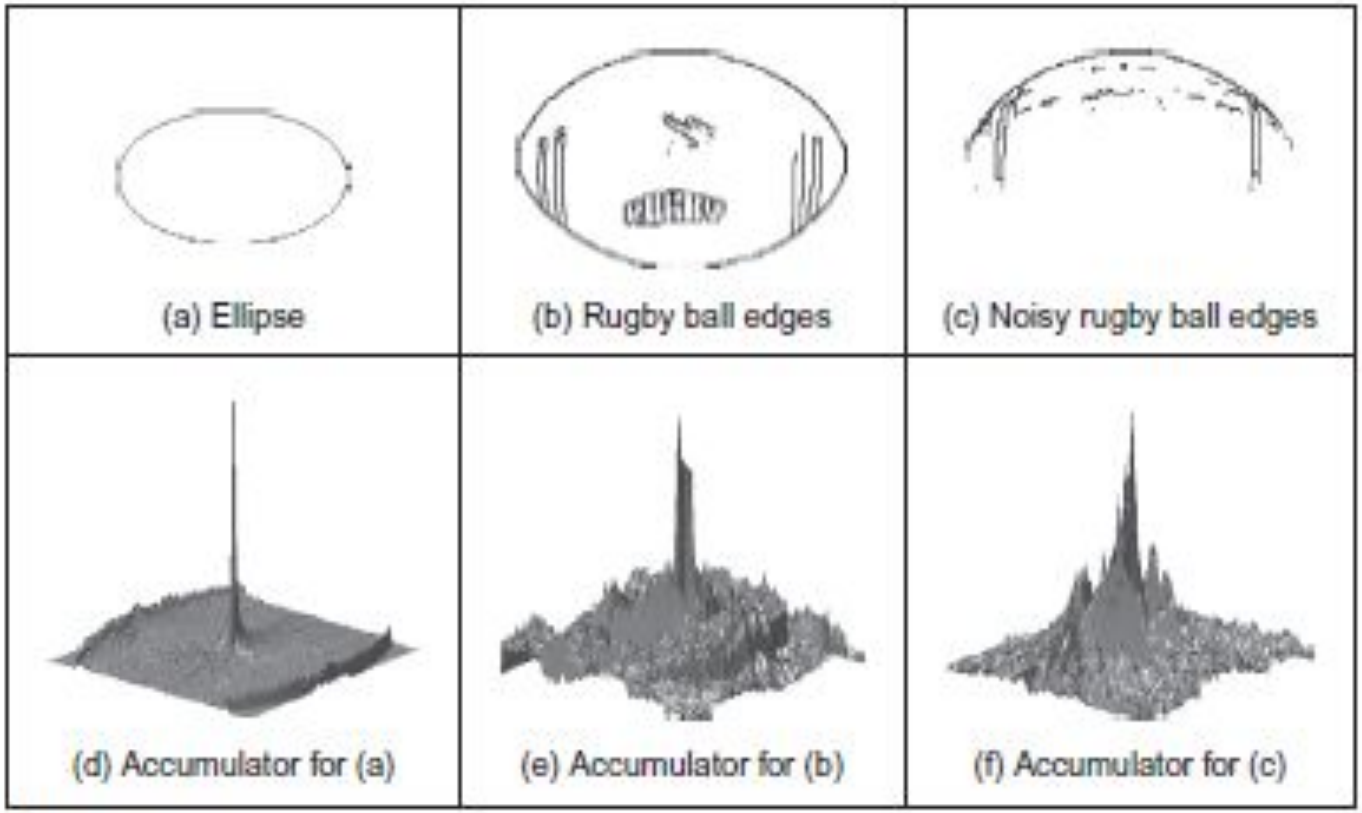


FIGURE 5.23

Applying the HT for ellipses.

Hough transform 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
```

CODE 5.6

Implementation of the HT for ellipses.