13EC4201-DIGITAL IMAGE PROCESSING

(ECE)

Dr. Abraham Mathew **Sri.B. Rajendra Prasad**

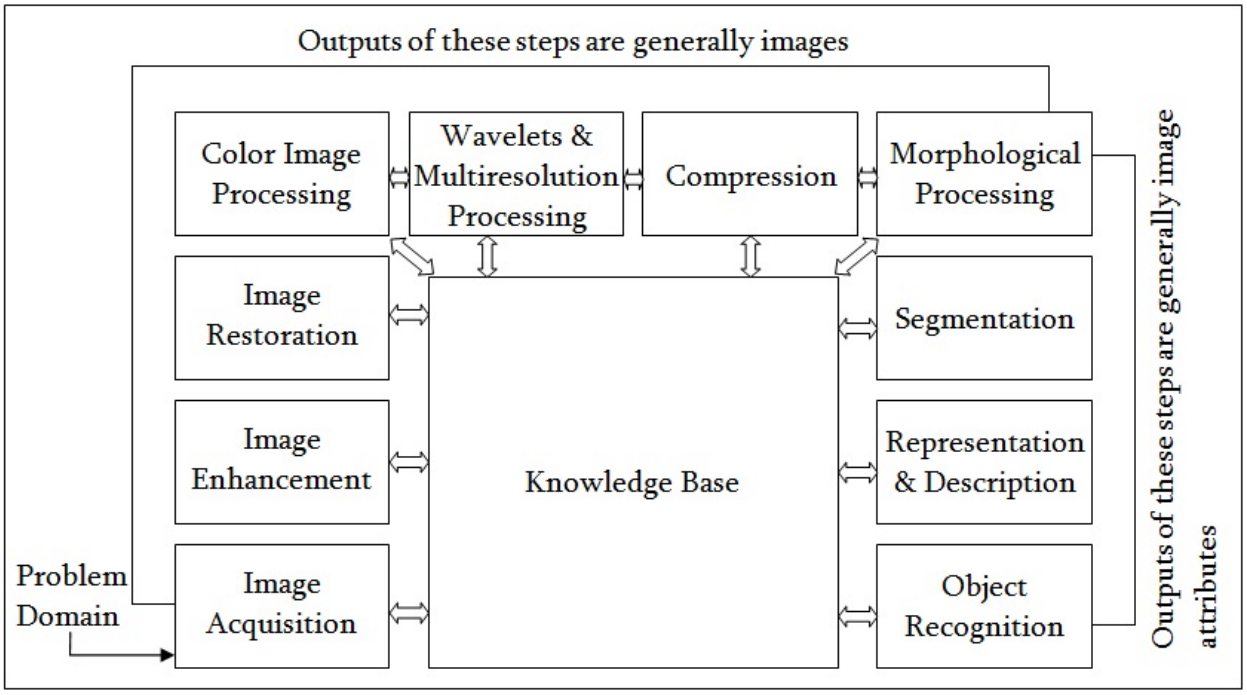
UNIT – I

Digital Image Fundamentals: Digital Image Representation – Digital Image Processing System – Visual Perception – Sampling and quantization – Basic Relationship between pixels – Imaging geometry.

What Is Digital Image Processing

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*. The field of *digital image processing* refers to processing digital images by means of a digital computer. Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as *picture elements*, *image elements*, *pels*, and *pixels*. *Pixel* is the term most widely used to denote the elements of a digital image.

Fundamental Steps in Digital Image Processing:



(i) **Image Acquisition :** This is the first step or process of the fundamental steps of digital image processing. Image acquisition could be as simple as being given an image that is already in digital form. Generally, the image acquisition stage involves preprocessing, such as scaling etc.

(ii) Image Enhancement : Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interest in an image. Such as, changing brightness & contrast etc.

(iii) Image Restoration : Image restoration is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation. Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a "good" enhancement result.

(iv) Color Image Processing : Color image processing is an area that has been gaining its importance because of the significant increase in the use of digital images over the Internet. This may include color modeling and processing in a digital domain etc.

(v) Wavelets and Multiresolution Processing : Wavelets are the foundation for representing images in various degrees of resolution. Images subdivision successively into smaller regions for data compression and for pyramidal representation.

(vi) Compression : Compression deals with techniques for reducing the storage required to save an image or the bandwidth to transmit it. Particularly in the uses of internet it is very much necessary to compress data.

(vii) Morphological Processing : Morphological processing deals with tools for extracting image components that are useful in the representation and description of shape.

(viii) Segmentation : Segmentation procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually.

(ix) Representation and Description : Representation and description almost always follow the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region or all the points in the region itself. Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing. Description deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.

(x) Object recognition : Recognition is the process that assigns a label, such as, "vehicle" to an object based on its descriptors.

(xi) Knowledge Base : Knowledge may be as simple as detailing regions of an image where the information of interest is known to be located, thus limiting the search that has to be conducted in seeking that information. The knowledge base also can be quite complex, such as an interrelated list of all major possible defects in a materials inspection problem or an image database containing high-resolution satellite images of a region in connection with change-detection applications.

Components of an Image Processing System:

Although large-scale image processing systems still are being sold for massive imaging applications, such as processing of satellite images, the trend continues toward miniaturizing and blending of general-purpose small computers with specialized image processing hardware. The function of each component is discussed in the following paragraphs, starting with image sensing. With reference to *sensing*, two elements are required to acquire digital images. The first is a physical device that is sensitive to the energy radiated by the object we wish to image. The second, called a *digitizer*, is a device for converting the output of the physical sensing device into digital form. For instance, in a digital video camera, the sensors produce an electrical output proportional to light intensity. The digitizer converts these outputs to digital data.

Specialized image processing hardware usually consists of the digitizer just mentioned, plus hardware that performs other primitive operations, such as an arithmetic logic unit (ALU), which performs arithmetic and logical operations in parallel on entire images. One example of how an ALU is used is in averaging images as quickly as they are digitized, for the purpose of noise reduction. This type of hardware sometimes is called a *front-end subsystem*, and its most distinguishing characteristic is speed. In other words, this unit performs functions that require fast data throughputs (e.g., digitizing and averaging video images at 30 frames_s) that the typical main computer cannot handle.

The ***computer*** in an image processing system is a general-purpose computer and can range from a PC to a supercomputer. In dedicated applications, sometimes specially designed computers are used to achieve a required level of performance, but our interest here is on general-purpose image processing systems. In these systems, almost any well-equipped PC-type machine is suitable for offline image processing tasks.

Software for image processing consists of specialized modules that perform specific tasks. A well-designed package also includes the capability for the user to write code that, as a minimum, utilizes the specialized modules. More sophisticated software packages allow the integration of those modules and general- purpose software commands from at least one computer language.

Mass storage capability is a must in image processing applications.An image of size 1024*1024 pixels, in which the intensity of each pixel is an 8-bit quantity, requires one megabyte of storage space if the image is not compressed. When dealing with thousands, or even millions, of images, providing adequate storage in an image processing system can be a challenge. Digital storage for image processing applications falls into three principal categories: (1) short term storage for use during processing, (2) on-line storage for relatively fast recall, and (3) archival storage, characterized by infrequent access. Storage is measured in bytes (eight bits), Kbytes (one thousand bytes), Mbytes (one million bytes), Gbytes (meaning giga, or one billion, bytes), and T bytes (meaning tera, or one trillion, bytes). One method of providing short-term storage is computer memory.Another is by specialized boards, called *frame buffers*, that store one or more images and can be accessed rapidly, usually at video rates (e.g., at 30 complete images per second).The latter method allows virtually instantaneous image *zoom*, as well as *scroll* (vertical shifts) and *pan* (horizontal shifts). Frame buffers usually are housed in the specialized image processing hardware unit. Online storage generally takes the form of magnetic disks or optical-media storage. The key factor characterizing on-line storage is frequent access to the stored data. Finally, archival storage is characterized by massive storage requirements but infrequent need for access. Magnetic tapes and optical disks housed in "jukeboxes" are the usual media for archival applications.

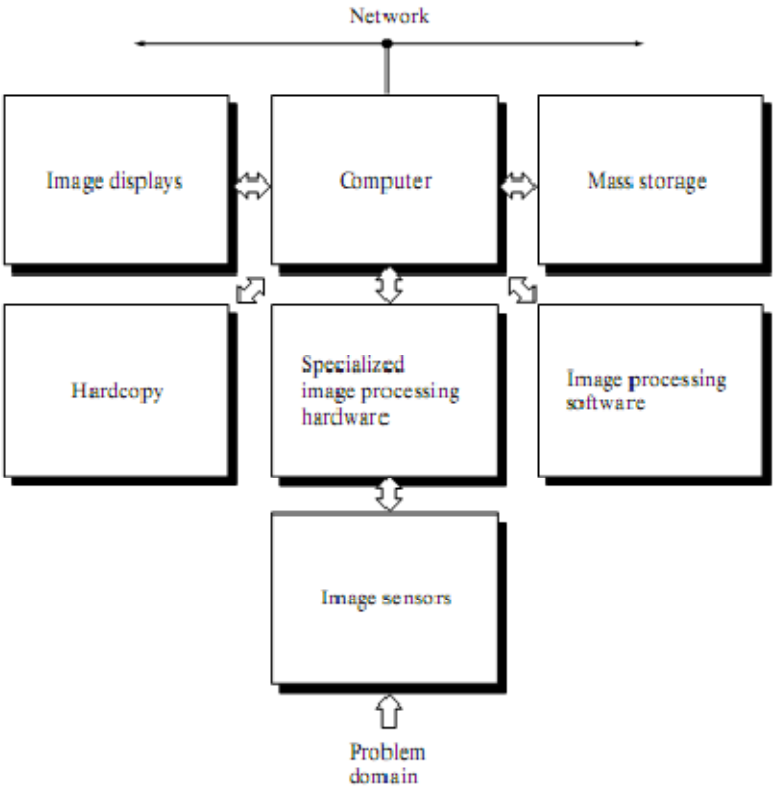


Image displays in use today are mainly color (preferably flat screen) TV monitors. Monitors are driven by the outputs of image and graphics display cards that are an integral part

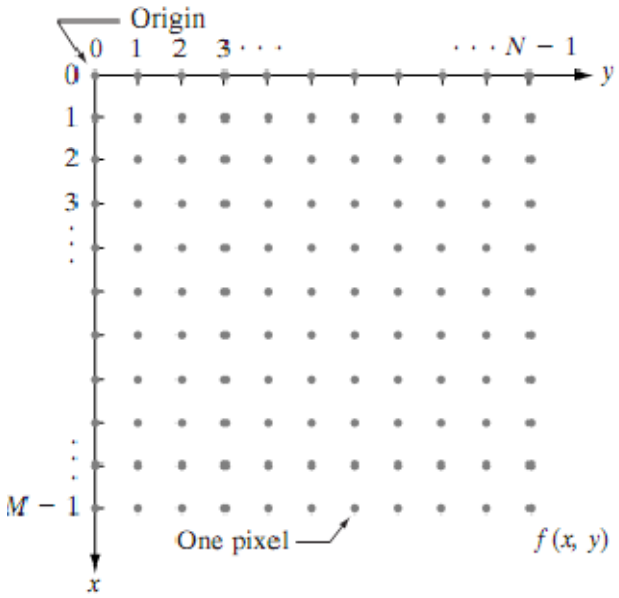
of the computer system. Seldom are there requirements for image display applications that cannot be met by display cards available commercially as part of the computer system. In some cases, it is necessary to have stereo displays, and these are implemented in the form of headgear containing two small displays embedded in goggles worn by the user.

Hardcopy devices for recording images include laser printers, film cameras, heat-sensitive devices, inkjet units, and digital units, such as optical and CD-ROM disks. Film provides the highest possible resolution, but paper is the obvious medium of choice for written material. For presentations, images are displayed on film transparencies or in a digital medium if image projection equipment is used. The latter approach is gaining acceptance as the standard for image presentations.

Networking is almost a default function in any computer system in use today. Because of the large amount of data inherent in image processing applications, the key consideration in image transmission is bandwidth. In dedicated networks, this typically is not a problem, but communications with remote sites via the Internet are not always as efficient. Fortunately, this situation is improving quickly as a result of optical fiber and other broadband technologies.

Image representation and its properties

We will use two principal ways to represent digital images. Assume that an image f(x, y) is sampled so that the resulting digital image has M rows and N columns. The values of the coordinates (x, y) now become discrete quantities. For notational clarity and convenience, we shall use integer values for these discrete coordinates. Thus, the values of the coordinates at the origin are (x, y) = (0, 0). The next coordinate values along the first row of the image are represented as (x, y) = (0, 1). It is important to keep in mind that the notation (0, 1) is used to signify the second sample along the first row. It does not mean that these are the actual values of physical coordinates when the image was sampled. Figure shows the coordinate convention used.



The notation introduced in the preceding paragraph allows us to write the complete M*N digital image in the following compact matrix form: The right side of this equation is by definition a digital image. Each element of this matrix array is called an image element, picture element, pixel, or pel.

The notation introduced in the preceding paragraph allows us to write the complete M*N

Digital image in the following compact matrix form:

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

The right side of this equation is by definition a digital image. Each element of this matrix

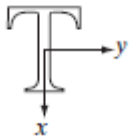
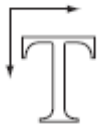
Array is called an image element, picture element, pixel, or pel.

Geometric Transform

Geometric image transformation functions use mathematical transformations to crop, pad, scale, rotate, transpose or otherwise alter an image array to produce a modified view of an image. The transformations described in this chapter are linear transformations. For a description of non-linear geometric transformations, When an image undergoes a geometric transformation, some or all of the pixels within the source image are relocated from their original spatial coordinates to a new position in the output image. When a relocated pixel does not map directly onto the centre of a pixel location, but falls somewhere in between the centres of pixel locations, the pixel's value is computed by sampling the values of the neighbouring pixels. This resampling, also known as interpolation, affects the quality of the output image.

Cropping Images:

Cropping an image extracts a rectangular region of interest from the original image. This focuses the viewer's attention on a specific portion of the image and discards areas of the image that contain less useful information. Using image cropping in conjunction with image magnification allows you to zoom in on a specific portion of the image. This section describes how to exactly define the portion of the image you wish to extract to create a cropped image

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

Padding Images:

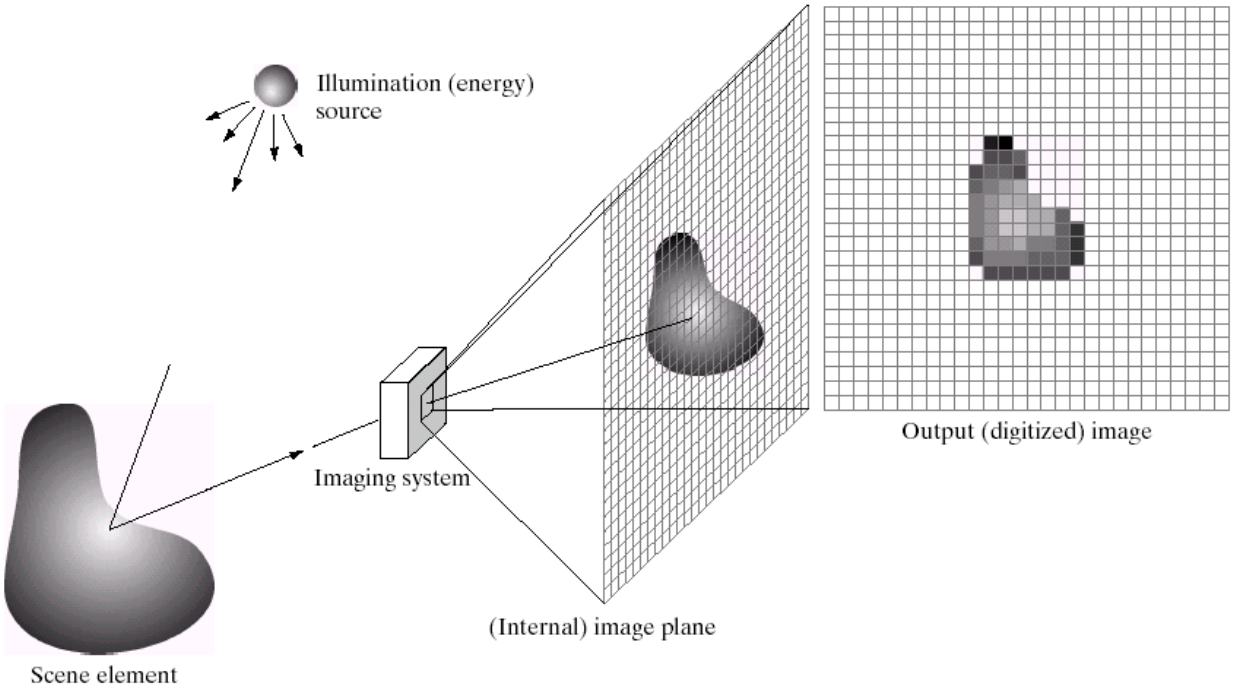
Image padding introduces new pixels around the edges of an image. The border provides space for annotations or acts as a boundary when using advanced filtering techniques. This exercise adds a 10-pixel border to left, right and bottom of the image and a 30-pixel border at the top allowing space for annotation. The diagonal lines in the following image represent the area that will be added to the original image. For an example of padding an image, complete the following steps.

Image Sampling and Quantization,

To create a digital image, we need to convert the continuous sensed data into digital form. This involves two processes: *sampling* and *quantization*. A continuous image, f(x, y), that we want to convert to digital form. An image may be continuous with respect to the x- and y-coordinates, and also in amplitude. To convert it to digital form, we have to sample the function in both coordinates and in amplitude. Digitizing the coordinate values is called *sampling*. Digitizing the amplitude values is called *quantization*.

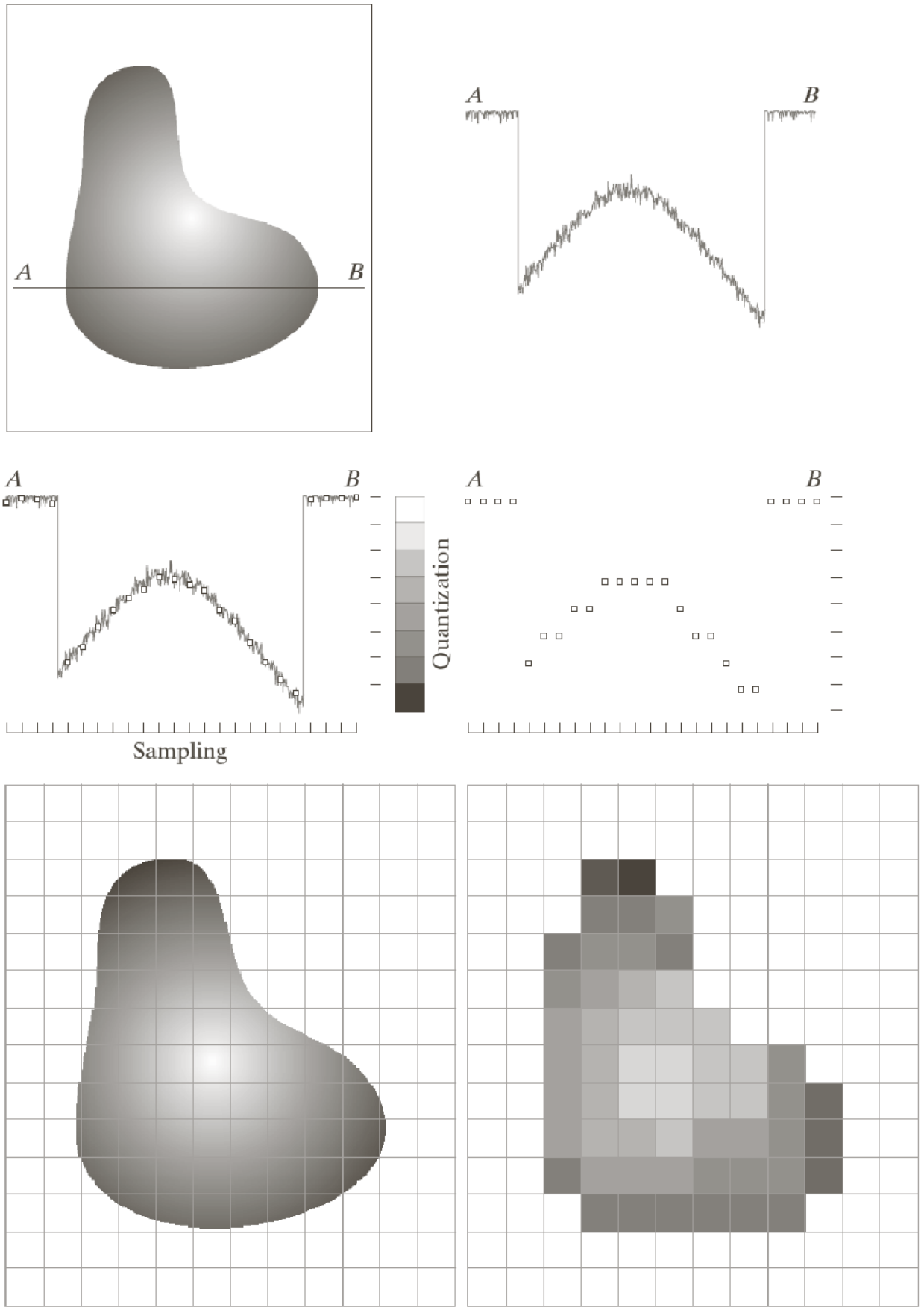
The one-dimensional function shown in Fig. is a plot of amplitude (gray level) values of the continuous image along the line segment AB. The random variations are due to image noise.

To sample this function, we take equally spaced samples along line AB, The location of each sample is given by a vertical tick mark in the bottom part of the figure. The samples are shown as small white squares superimposed on the function. The set of these discrete locations gives the sampled function. However, the values of the samples still span (vertically) a continuous range of gray-level values. In order to form a digital function, the gray-level values also must be converted (*quantized*) into discrete quantities. The right side gray-level scale divided into eight discrete levels, ranging from black to white. The vertical tick marks indicate the specific value assigned to each of the eight gray levels. The continuous gray levels are quantized simply by assigning one of the eight discrete gray levels to each sample. The assignment is made depending on the vertical proximity of a sample to a vertical tick mark. The digital samples resulting from both sampling and quantization.



a b c d e

FIGURE 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.



a b

FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Basic Relationships between Pixels:

In this section, we consider several important relationships between pixels in a digital image. As mentioned before, an image is denoted by f(x, y).When referring in this section to a particular pixel, we use lowercase letters, such as p and q.

Neighbors of a Pixel A pixel p at coordinates (x, y) has four *horizontal* and *vertical* neighbors whose coordinates are given by

(x+1, y), (x-1, y), (x, y+1), (x, y-1)

This set of pixels, called the 4-*neighbors* of p, is denoted by $N4(p)$. Each pixel is a unit distance from (x, y), and some of the neighbors of p lie outside the digital image if (x, y) is on the border of the image. The four *diagonal* neighbors of p have coordinates

(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)

and are denoted by $ND(p)$. These points, together with the 4-neighbors, are called the 8-*neighbors* of p, denoted by $N8(p)$. As before, some of the points in $ND(p)$ and $N8(p)$ fall outside the image if (x, y) is on the border of the image.

Adjacency, Connectivity, Regions, and Boundaries Connectivity between pixels is a fundamental concept that simplifies the definition of numerous digital image concepts, such as regions and boundaries. To establish if two pixels are connected, it must be determined if they are neighbors and if their gray levels satisfy a specified criterion of similarity (say, if their gray levels are equal).For instance, in a binary image with values 0 and 1, two pixels may be 4-neighbors, but they are said to be connected only if they have the same value.

Let V be the set of gray-level values used to define adjacency. In a binary image, $V=\{1\}$ if we are referring to adjacency of pixels with value 1. In a grayscale image, the idea is the same, but set V typically contains more elements.

For example, in the adjacency of pixels with a range of possible gray-level values 0 to 255, set V could be any subset of these 256 values. We consider three types of adjacency:

(a) 4-*adjacency*. Two pixels p and q with values from V are 4-adjacent if q is in the set $N4(p)$.

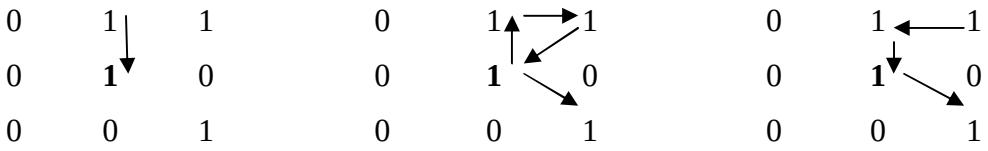
(b) 8-*adjacency*. Two pixels p and q with values from V are 8-adjacent if q is in the set $N8(p)$.

(c) *m-adjacency* (mixed adjacency).Two pixels p and q with values from V are m-adjacent if

(i) q is in *N*4(p), *or*

(ii) q is in *N*D(p) *and* the set has no pixels whose values are from *V*.

- A pixel p is adjacent to a pixel q if they are connected.
- A path from pixel to pixel q will be a sequence of distinct pixels with their own coordinates



4 neighbours **8 neighbours** **m neighbours**

Linear and Nonlinear Operations

Let *H* be an operator whose input and output are images. *H* is said to be a *linear* operator if, for any two images f and g and any two scalars a and b,

H(af + bg) = aH(f) + bH(g).

In other words, the result of applying a linear operator to the sum of two images (that have been multiplied by the constants shown) is identical to applying the operator to the images individually, multiplying the results by the appropriate constants, and then adding those results. For example, an operator whose function is to compute the sum of K images is a linear operator. An operator that computes the absolute value of the difference of two images is not. Linear operations are exceptionally important in image processing because they are based on a significant body of well-understood theoretical and practical results. Although nonlinear operations sometimes offer better performance, they are not always predictable, and for the most part are not well understood theoretically.

UNIT – II

Image Transforms: DiscreteFourier Transform – Properties of 2-D Fourier transform – 2-D Fast Fourier Transform – Walsh Transform – Hadamard Transform – DCT – Haar Transform – Slant Transform – Hotelling Transform.

3.3 DISCRETE FOURIER TRANSFORM

So far we have discussed how to compute Fourier transform for continuous functions or signals. But in digital image processing the use of Fourier spectrum of continuous function is of no use. But to derive the discrete version, the knowledge of continuous function is utilized and hence it is described here. The digital version of the Fourier spectrum is used in digital image processing and it is referred as *discrete Fourier spectrum/transform.* The determination of discrete Fourier transform/spectrum is explained in detail in this section.

Discrete Fourier spectrum/transform: This is the digital version of the Fourier spectrum used in digital image processing.

Let us consider a continuous function as shown in Figure 3.3. In order to derive the equation for discrete Fourier transform (DFT), it is necessary to discretize the given function $f(x)$. The function is discretized at regular intervals, by taking N samples, Δx units apart as shown in Figure (3.3).

Then the function $f(x)$ after discretization can be written as

$$f(x) = f(x_0 + x\Delta x) \qquad (3.13)$$

where x takes values 0, 1, 2, …, N — 1.

The sequences $\{f(0), f(1), \ldots, f(N-1)\}$ represents N uniformly spaced samples of the continuous function. The DFT pair of the sampled function is given by

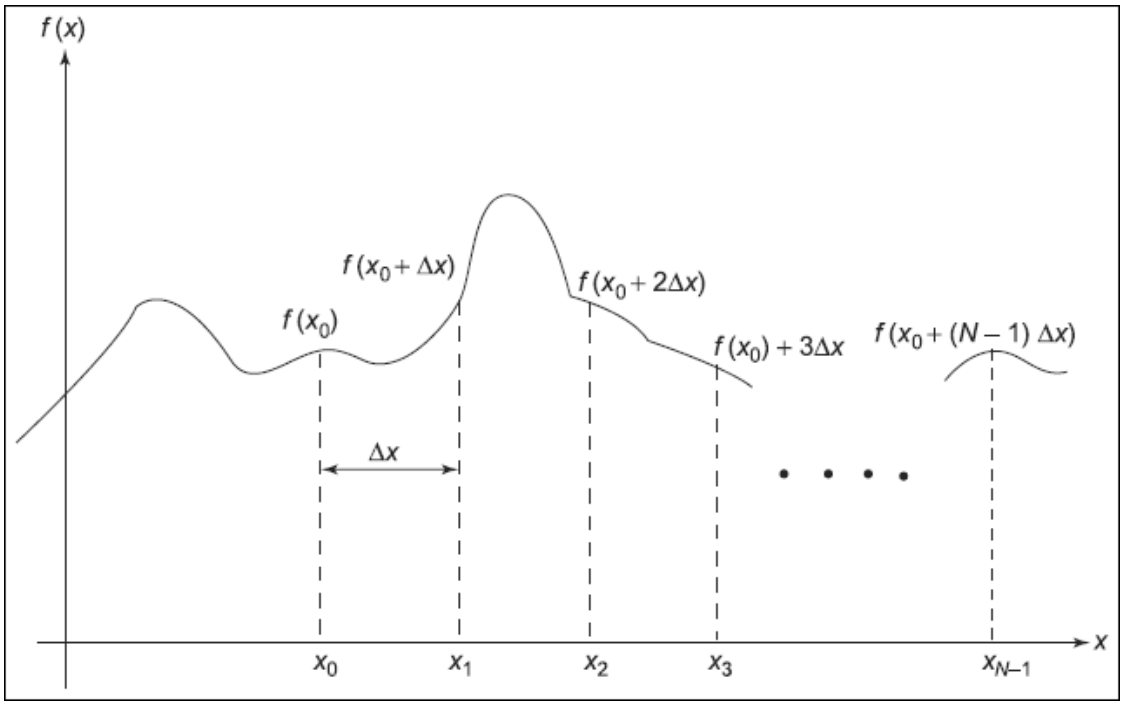


FIGURE 3.3 Sampling a continuous function

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{[-j2\pi ux/N]} \qquad (3.14)$$

for $u = 0, 1, 2, \ldots, N-1$ and

$$f(x) = \frac{1}{N}\sum_{u=0}^{N-1} F(u)e^{[j2\pi ux/N]} \qquad (3.15)$$

for $x = 0, 1, 2, \ldots, N-1$.

The values $u = 0, 1, 2, \ldots, N-1$ in equation (3.14) correspond to the samples of continuous transform at values $0, \Delta u, 2\Delta u, \ldots, (N-1)\Delta u$.

The terms Δu and Δx are related by the expression

$$\Delta u = \frac{1}{N\Delta x}$$

Similarly, the DFT pair for the two variable case is as follows:

$$F(u, v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y)e^{\left[-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right]} \qquad (3.16)$$

for $u = 0, 1, 2, \ldots, M-1$ and $v = 0, 1, 2, \ldots N-1$

$$f(x, y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u, v)e^{\left[j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right]} \qquad (3.17)$$

for $x = 0, 1, 2, \ldots, M-1$ and y = 0, 1,2, …, $N-1$.

Where the function $f(x, y)$ represents the samples of the function

$$f(x_0 + x\Delta x, y_0 + y\Delta y)$$

for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$.

The sampling increments in the spatial and frequency domains are related by

$$\Delta u = \frac{1}{M\Delta x} \qquad (3.18)$$

and

$$\Delta v = \frac{1}{N\Delta y} \qquad (3.19)$$

when images are sampled in a square array, $M = N$, then the DFT pair is given as

$$F(u, v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y)e^{-j2\pi\left(\frac{ux+vy}{N}\right)} \qquad (3.20)$$

For $u, v = 0, 1, \ldots, N - 1$ and

$$f(x, y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u, v)e^{j2\pi\left(\frac{ux+vy}{N}\right)} \qquad (3.21)$$

For $x, y = 0, 1, 2, \ldots, N - 1$.

3.4 PROPERTIES OF FOURIER TRANSFORM

The properties of Fourier transform which are useful in digital image processing are discussed in detail in this section.

For example, using the separability property, the Fourier transform $F(u, v)$ for the function $f(x, y)$ can be obtained in two steps by successively applying one-dimensional Fourier transform and the same is explained in Section 3.4.1.

3.4.1 Separability

The DFT pair that we have already discussed in equations (3.20) and (3.21) is represented here for convenience.

Equations (3.20) and (3.21) are expressed in the separable form as:

$$F(u, v) = \frac{1}{N}\sum_{x=0}^{N-1} e^{\left[\frac{-j2\pi ux}{N}\right]}\sum_{y=0}^{N-1} f(x, y)e^{\left[\frac{-j2\pi vy}{N}\right]} \qquad (3.22)$$

for $u, v = 0, 1, 2, \ldots, N - 1$ and

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} e^{\left[\frac{j2\pi ux}{N}\right]} \sum_{v=0}^{N-1} F(u, v) e^{\left[\frac{j2\pi vy}{N}\right]} \qquad (3.23)$$

for $x, y = 0, 1, 2,\ldots, N-1$.

Equation (3.22) can be written as

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} F(x, v) e^{\left[\frac{-j2\pi ux}{N}\right]} \qquad (3.24)$$

where

$$F(x, v) = N \left[\frac{1}{N} \sum_{y=0}^{N-1} f(x, y) e^{\left[\frac{-j2\pi vy}{N}\right]} \right] \qquad (3.25)$$

Equation (3.25) can be interpreted as one-dimensional transform for each value of x with the frequency values $v = 0, 1, \ldots, N-1$. Therefore, the two-dimensional function $F(x, v)$ is obtained by taking a transform along each row of $f(x, y)$ and multiplying the result by N. The final result $F(u, v)$ is then obtained by taking a transform along each column of $F(x, v)$ as given by equation (3.24). This can be illustrated as shown in Figure 3.4.

The same results can also be obtained by taking transforms along the columns of $f(x, y)$ and then along the rows using columns result.

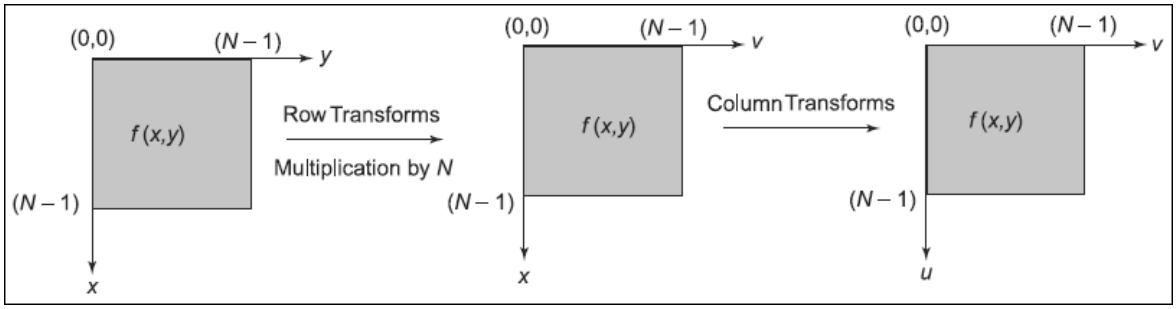


FIGURE 3.4 Computation of the two-dimensional Fourier transform as a series of two one-dimensional transforms

3.4.2 TRANSLATION

In order to shift the origin of the frequency plane to the point (u_0, v_0) the original function $f(x, y)$ should be multiplied by the exponential term $e^{[j2\Pi(u_0x+v_0y)/N]}$. This translation effect is given in the equation (3.26)

$$f(x, y) e^{\left[\frac{j2\pi(u_0x+v_0y)}{N}\right]} \Leftrightarrow F(u - u_0, v - v_0) \qquad (3.26)$$

Similarly, the origin of the spatial plane can be shifted to (x_0, y_0) by multiplying $F(u, v)$ with the exponential term $e^{[-j2\Pi(ux_0+vy_0)/N]}$ and the same is given in the equation (3.27).

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v)e^{\left[\frac{-j2\pi(ux_0+vy_0)}{N}\right]} \qquad (3.27)$$

When we substitute $u_0 = v_0 = \frac{N}{2}$ in equation (3.26), the equation becomes

$$f(x, y)e^{[j\pi(x+y)]} = f(x, y)(-1)^{x+y} \Leftrightarrow F\left(u - \frac{N}{2}, v - \frac{N}{2}\right) \qquad (3.28)$$

Thus the origin of the Fourier transform of $f(x, y)$ can be moved to the center by multiplying $f(x, y)$ by $(-1)^{x+y}$, in the case of one variable this shift reduces to multiplication of $f(x)$ by the term $(-1)^x$. The shift in $f(x, y)$ does not affect the magnitude of its Fourier transform and the same is illustrated as

$$|F(u, v)|e^{\left[\frac{-j2\pi(ux_0+vy_0)}{N}\right]}$$

$$= |F(u, v)| \left| e^{\left[\frac{-j2\pi(ux_0+vy_0)}{N}\right]} \right| \qquad (3.29)$$

$$= |F(u, v)| \left| \cos\left(2\pi\left(\frac{ux_0 + vy_0}{n}\right)\right) - j\sin\left(2\pi\left(\frac{ux_0 + vy_0}{n}\right)\right) \right|$$

$$= |F(u, v)||1|$$

$$= |F(u, v)|$$

3.4.3 PERIODICITY AND CONJUGATE SYMMETRY

The DFT and its inverse are periodic with period N, that is,

$$F(u, v) = F(u + N, v) = F(u, v + N) \qquad (3.30)$$

$$= F(u + N, v + N)$$

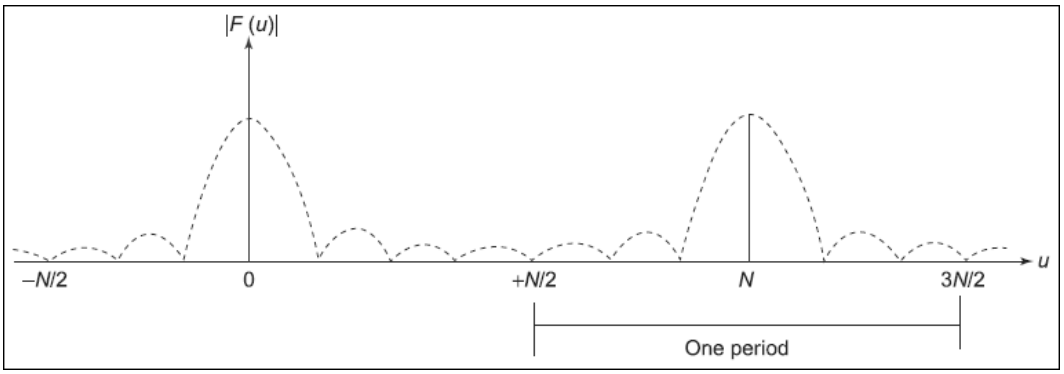


FIGURE 3.5 Fourier transform periodicity

Although, equation (3.30) indicates that $F(u, v)$ repeats itself for infinitely many values of u and v, only the N values of each variable in any one period are required to obtain $f(x, y)$ from $F(u, v)$. In other words, only one period of the transform is necessary to specify $F(u, v)$ completely in the frequency domain. Similar comments can be given to $f(x, y)$ in the spatial domain. The periodicity property is illustrated in Figure 3.5.

The Fourier transform also exhibits conjugate symmetry if $f(x, y)$ is real.

$$|F(u, v)| = F^*(-u, -v) \qquad (3.31)$$

Or

$$|F(u, v)| = |F(-u, -v)| \qquad (3.32)$$

where $F^*(u, v)$ is the complex conjugate of $F(u, v)$.

3.4.4 ROTATION

The function $f(x, y)$ and $F(u, v)$ can be represented in the polar coordinate as $f(r, \theta)$ and $F(\omega,\varphi)$, respectively. The relationship between x, y, r, and θ and the relationship between u, v, w, and φ is as in the following.

$$x = r\cos\theta \qquad y = r\sin\theta$$

$$u = \omega\cos\phi \qquad v = \omega\sin\phi$$

Then the direct substitution of the above relationships in either continuous or discrete Fourier transform pair gives

$$F(r, \theta + \theta_0) \Leftrightarrow F(\omega, \phi + \theta_0) \qquad (3.33)$$

In other words, rotating $f(x, y)$ by an angle θ_0 rotates $F(u, v)$ by the same angle. Similarly, rotating $F(u, v)$ rotates $f(x, y)$ by the same angle.

3.4.5 Distributivity and Scaling

The Fourier transform and its inverse are distributive over addition but not over multiplication. The same is given in equation (3.34)

The Fourier transform of

$$\mathcal{F}\{f_1(x, y) + f_2(x, y)\} = \mathcal{F} f_1(x, y) + \mathcal{F} f_2(x, y) \qquad (3.34)$$

and in general Fourier transform of

$$\mathcal{F}\{f_1(x, y) \cdot f_2(x, y)\} \neq \mathcal{F}\{f_1(x, y)\} \cdot \mathcal{F}\{f_2(x, y)\} \qquad (3.35)$$

For two scalars a and b

$$af(x,y) \Leftrightarrow aF(u, v) \qquad (3.36)$$

and

$$f(ax, by) \Leftrightarrow \frac{1}{|ab|F}\left(\frac{u}{a}, \frac{v}{b}\right) \qquad (3.37)$$

3.4.6 AVERAGE VALUE

Average value of the two-dimensional discrete function in general is given by the expression

$$\overline{f}(x, y) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y) \qquad (3.38)$$

Substituting $u = v = 0$ in equation (3.20) yields

$$F(0, 0) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y) \qquad (3.39)$$

Therefore, $f(x, y)$ is related to the Fourier transform of $f(x, y)$ by

$$\overline{f}(x, y) = \frac{1}{N}F(0, 0) \qquad (3.40)$$

3.4.7 LAPLACIAN

The Laplacian of a two variable function $f(x, y)$ is defined as

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (3.41)$$

From the definition of the two-dimensional Fourier transform

$$F\{\nabla^2 f(x, y)\} \Leftrightarrow -(2\pi)^2(u^2 + v^2)F(u, v) \qquad (3.42)$$

The Laplacian operator is useful for finding edges in an image.

3.4.8 CONVOLUTION AND CORRELATION

In order to establish a link between the spatial and frequency domain, we have to use the concepts called *convolution* and *correlation*. In this section we first discuss the convolution process for one-dimensional case with continuous arguments. We then extend to the discrete case and finally to the two-dimensional continuous and discrete cases. Similarly, the concept of correlation is explained for continuous and discrete cases.

Convolution: A process that provides a way to relate the spatial and frequency domains. For example, the convolution of two functions namely $f(x, y)$ and $g(x, y)$ is equivalent to multiplying the Fourier spectrum of the two functions (i.e.,) $f(u, v)$ and $G(u, v)$.

Convolution The convolution of the two continuous functions $f(x)$ and $g(x)$, denoted by $f(x) * g(x)$ and defined by the integral is given in equation (3.43)

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha) g(x - \alpha) \mathrm{d}\alpha \qquad (3.43)$$

where α is a dummy variable of integration.

The convolution process is not easy to visualize and hence is discussed with graphical illustration. Consider the function $f(x)$ and $g(x)$ as shown in the Figure 3.6(a) and (b), respectively. The function $g(x - \alpha)$ must be formed from $g(\alpha)$ before the integration is carried out. $g(x - \alpha)$ is formed in two steps and is illustrated in Figure 3.6(c) and (d). The first step is simply folding $g(\alpha)$ about the origin to get $g(-\alpha)$ and then in the second step the function is shifted by a distance x.

Then for any value of x, the function $f(\alpha)$ multiplied with $g(x - \alpha)$ and the product is integrated from $-\infty$ to ∞. The product of $f(\alpha)$ and $g(x - \alpha)$ is the shaded portion of Figure 3.6(e). This figure is valid for $0 \leq X \leq 1$. The product is 0 for values of $f(\alpha)$ outside the interval $(0, x)$. So $f(x) * g(x) = \frac{x}{2}$ which is simply the area of the shaded region. For x in the interval (1, 2), Figure 3.6 (f) is used and $f(x) * g(x) = 1 - \frac{x}{2}$.

Finally, we have

$$f(x) * g(x) = \begin{cases} \frac{x}{2} & 0 \leq x \leq 1 \\ 1 - \frac{x}{2} & 1 \leq x \leq 2 \\ 0 & \text{elsewhere} \end{cases}$$

and this result is represented in Figure 3.6(g).

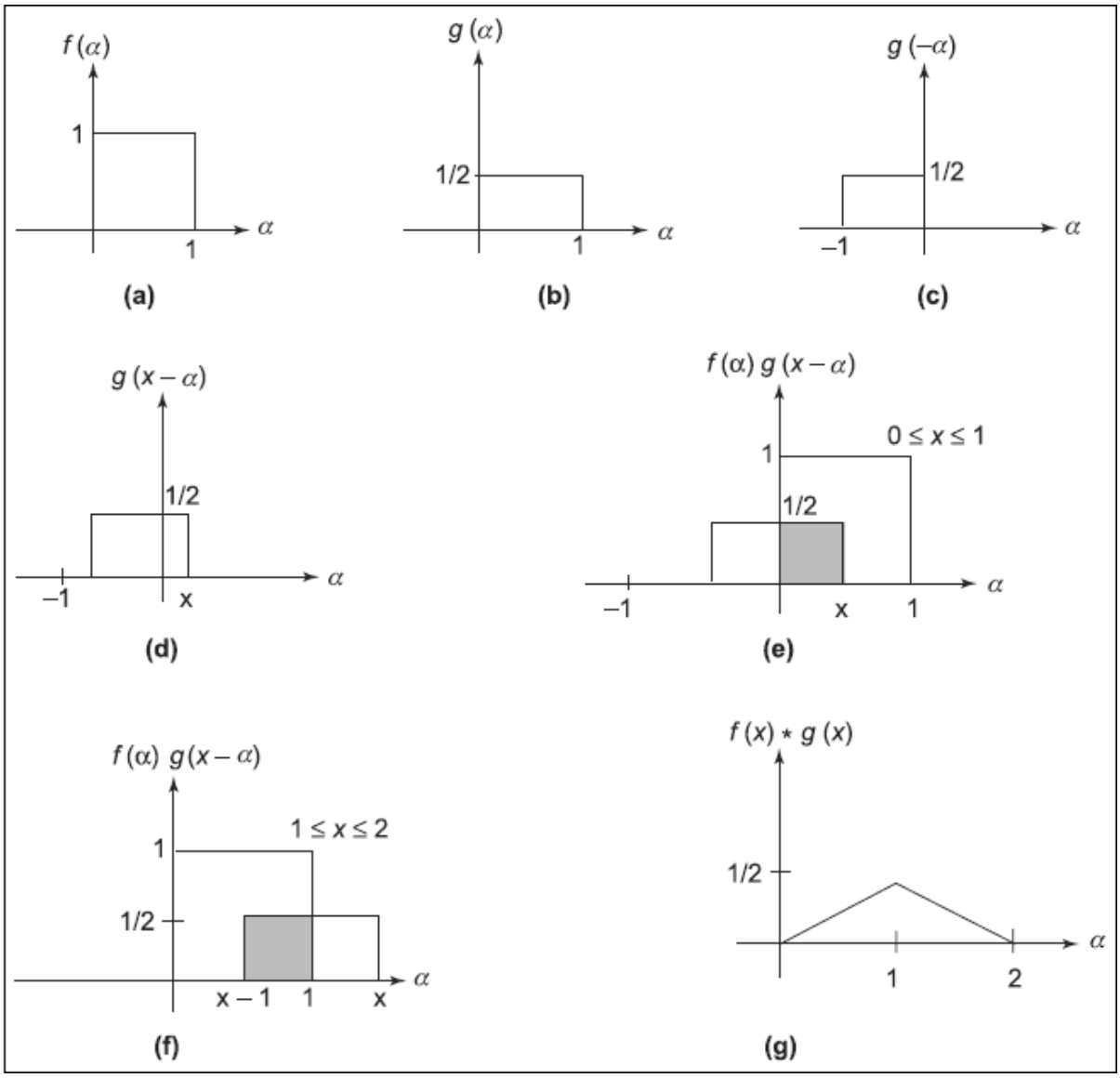


FIGURE 3.6 Graphical illustration of convolution

One of the important uses of the convolution is to convolve the given function with the impulse function $\delta(x - x_0)$ which is given by the relation

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)\mathrm{d}x = f(x_0) \qquad (3.44)$$

Convolution in this case, amounts to merely copying $f(x)$ at the location of impulse. If $F(u)$ and $G(u)$ are the Fourier transform of $f(x)$ and $g(x)$, respectively, then $f(x) * g(x)$ is equal to the product of $F(u)$ and $G(u)$. This result is stated as in equation (3.45)

$$f(x) * g(x) \Leftrightarrow F(u)G(u) \qquad (3.45)$$

From equation (3.45) we infer that, the convolution in the spatial domain can also be obtained by taking the inverse Fourier transform of the product $\{F(u)G(u)\}$. An analogous result

is that the convolution in the frequency domain is simply the multiplication in the spatial domain and it is stated as

$$f(x)g(x) \Leftrightarrow F(u) * G(u) \qquad (3.46)$$

These two results are commonly referred to as convolution theorems. In the discrete convolution process, the functions $f(x)$ and $g(x)$ are discretized and stored in arrays of size A and B, respectively. The two array elements are given as

$$\{f(0), f(1), f(2), \ldots, f(A-1)\}$$

and

$$\{g(0), g(1), g(2), \ldots, g(B-1)\}$$

Assume that the discrete function $f(x)$ and $g(x)$ are periodic with the same period M. The resulting convolution is then periodic with the same period M. The period M must be selected in such a way that

$$M \geq A + B - 1$$

so that the wraparound error can be avoided. Because the assumed period must be greater than A or B, the length of the sampled sequence must be increased so that both are of length M. Appending zeros to the samples forms the extended sequences and are given as

$$f_e(x) = \begin{cases} f(x) & 0 \leq x \leq A-1 \\ 0 & A \leq x \leq M-1 \end{cases}$$

and

$$g_e(x) = \begin{cases} g(x) & 0 \leq x \leq B-1 \\ 0 & B \leq x \leq M-1 \end{cases}$$

Based on these extensions the discrete convolution of functions $f_e(x)$ and $g_e(x)$is defined by

$$f_e(x) * g_e(x) = \frac{1}{M} \sum_{\alpha=0}^{M-1} f_e(\alpha) g_e(x-\alpha) \qquad (3.47)$$

for $x = 0, 1, 2, \ldots, M-1$.

The mechanism of discrete convolution is same as continuous convolution. Figure (3.7) illustrates graphically the convolution of two discrete functions $f_e(\alpha)$ and $g_e(\beta)$.

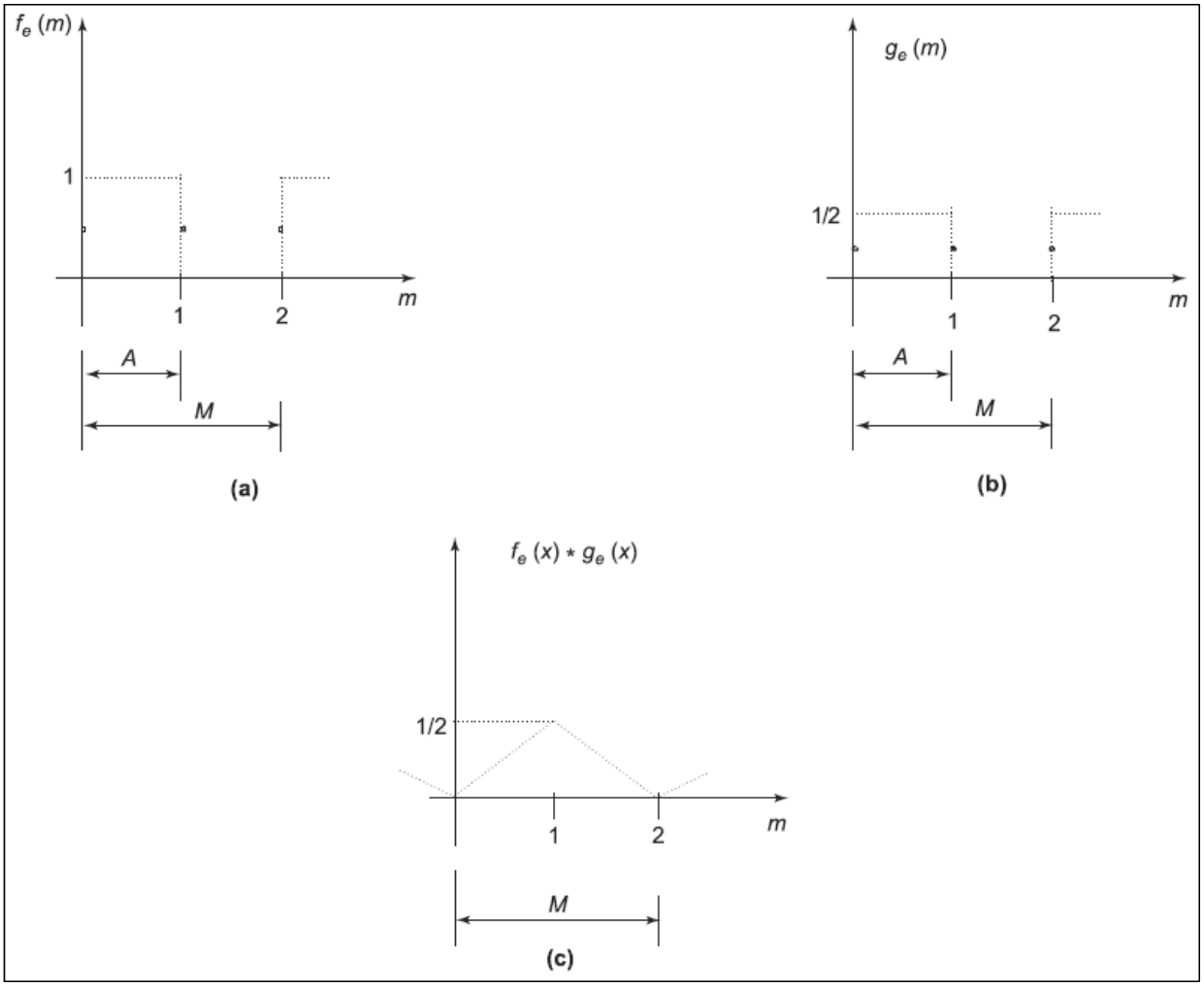


FIGURE 3.7(a–c) Graphical illustration of the convolution of two discrete functions

The two-dimensional convolution for two functions $f\ (x,\ y)$ and $g(x,\ y)$ is given in equation (3.48), which is analogous to equation (3.43). Thus, for two functions $f(x, y)$ and $g(x, y)$

$$f(x, y) * g(x, y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(\alpha, \beta)g(x - \alpha, y - \beta)\mathrm{d}\alpha\mathrm{d}\beta \qquad (3.48)$$

The convolution theorem for two-dimensional functions is expressed by the relations

$$f(x, y) * g(x, y) \Leftrightarrow F(u, v)G(u, v) \qquad (3.49)$$

and

$$f(x, y)g(x, y) \Leftrightarrow F(u, v) * G(u, v) \qquad (3.50)$$

In general, carrying out the convolution process in the spatial domain for the two-dimensional image functions is complex in nature. Hence to overcome this difficulty we employ the convolution theorem. From the convolution theorem we find that the convolution in the spatial domain of two functions is equivalent to multiplying the respective Fourier transforms in the frequency domain. Hence, in order to perform the convolution operations, we have to obtain

the Fourier transform of the given two functions separately. Then the product of these two Fourier transforms is obtained and finally take the inverse Fourier transform of the product thus obtained. The result gives the convolution of the given two functions in the spatial domain.

Consider the 3 × 2 and 2 × 2 arrays $x(m, n)$ and $h(m, n)$ shown here where the boxed element is at the origin. Now let us illustrate the step-by-step procedure for the convolution of the functions $x(m, n)$ and $h(m, n)$ which are shown in Figure 3.8(a).

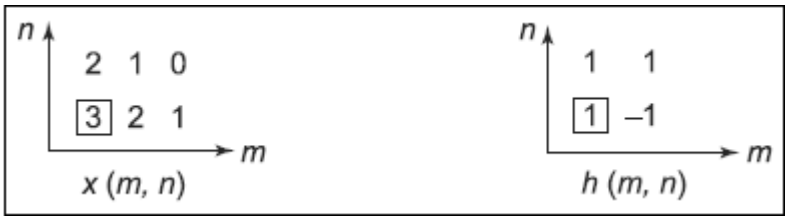
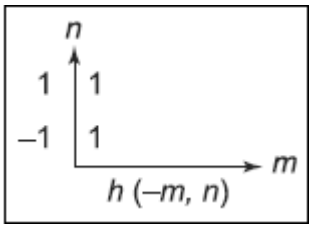
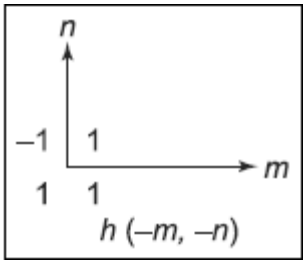


FIGURE 3.8(a) The functions $x(m, n)$ and $h(m, n)$ used for convolution

Step 1 Obtain *h(−m, n)*



Step 2 Obtain $h(-m, -n)$



Step 3 Obtain *h(−m + 1, −n)*

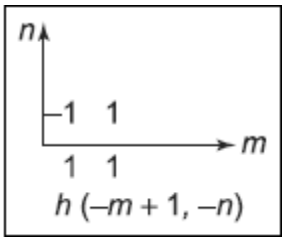


FIGURE 3.8(b) Steps to obtain the function *h(1–m, –n)*

We know that convolution is a process in which the second function is folded (mirror image) and slided over the first function and at every point the area is computed or the sum of products is obtained. The three steps 1 to 3 illustrate how to fold the second function and to slide right by one position. Now the function $h(1 - m, - n)$ can be used to slide over the first function to get the convolved final function or image; in general, the convolution of two arrays of sizes

(M_1, N_1) and (M_2, N_2). In this example, the convolution yields an array of size ($M_1 + M_2 - 1$) × ($N_1 + N_2 - 1$) = (2 + 2 − 1) × (3 + 2 − 1), that is, 3 × 4.

The various elements in the convolved array of size (3 × 4) are obtained in Figures 3.8(c)−(n). The elements in the final convolved array [Figure 3.8(o)] are denoted as *y(m, n),* where *m* takes the values 0, 1, 2 and *n* takes values 0, 1, 2 and 3. Now the first column elements are denoted as *y*(0,0), *y*(0,1), and *y*(0,2) and are given in Figures 3.8(c)−(e).

y(0,0) Substitute *m = 0* and *n = 0* in the function *h*(1 − *m, n*) so that the resulting second function is *h*(1, 0). This means the function shown in Step 3 is moved one position to the left and the function *h*(1,0) is shown in Figure 3.8(c). Now the function *h*(1,0) is placed on the function *x*(*m, n*) and the result is as follows.

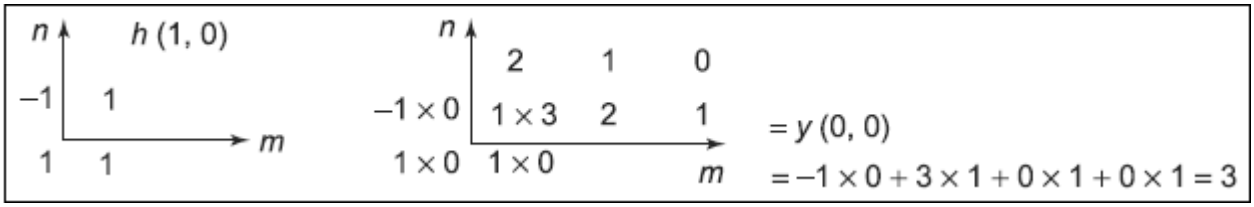
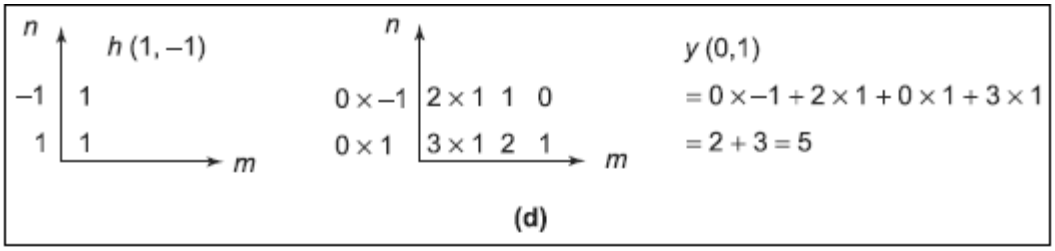


FIGURE 3.8(c)

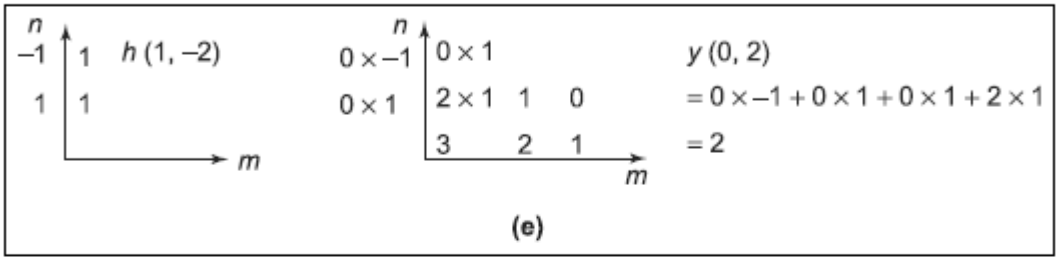
Similarly, the element *y*(0,1) is obtained as illustrated.

***y*(0,1) : *h*(1, − 1)**

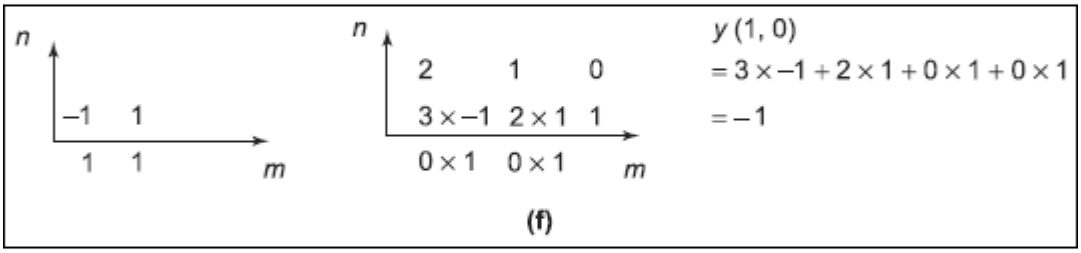
The corresponding second function to be used is *h*(1−0, −1) = *h*(1, −1). This means the function *h*(1−*m*,−*n*) is to be moved one position left and one position up. Then the function *h*(1, −1) is as shown in Figure 3.8(d). The convolution operation of the functions *x(m,n)* * *h*(1, − 1) is also illustrated in Figure 3.8(d). Similar procedure is followed to compute other elements which are shown in Figures 3.8(e)−(n). The final convolved matrix is shown in Figure 3.8(o).



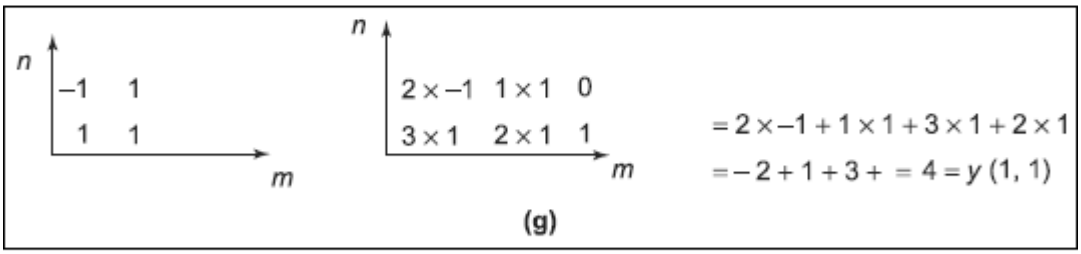
$y(0, 2) : h(1, -2)$



$y(1, 0) : h(0, 0)$



$y(1, 1) : h(0, -1)$



$y(1, 2) : h(0, -2)$

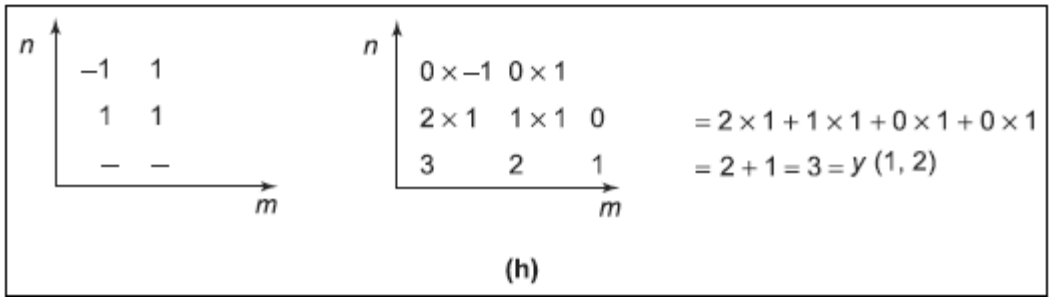
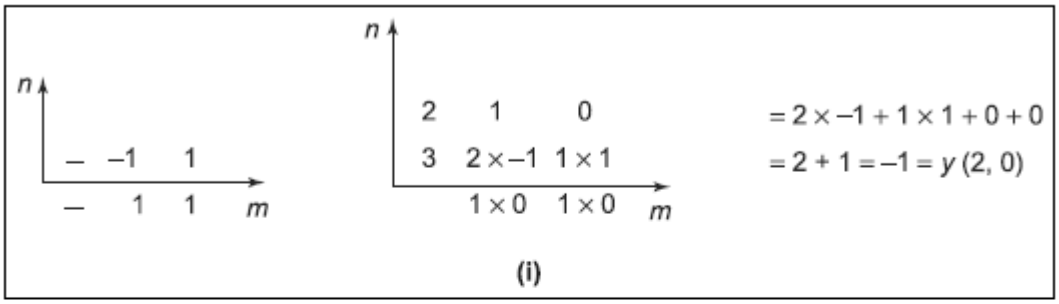
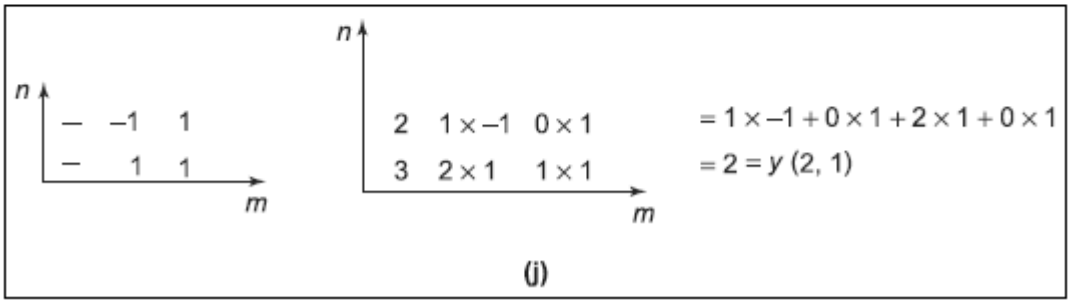


FIGURE 3.8(d–h)

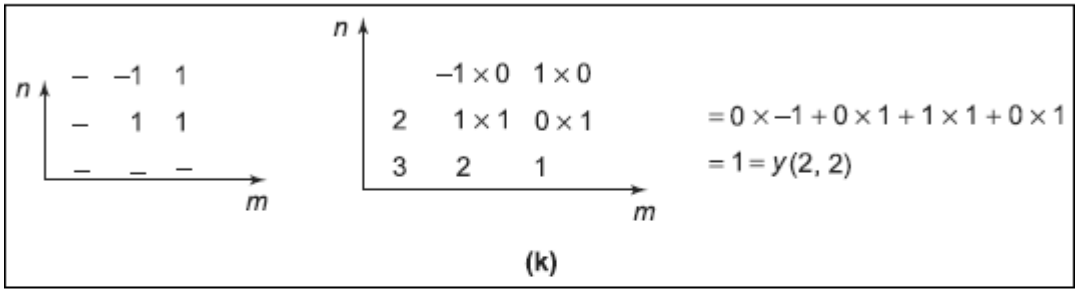
$y(2, 0) : h(-1, 0)$



$y(2, 1) : h(-1, -1)$



$y(2, 2) : h(-1, -2)$



$y(3, 0) : h(-2, 0)$

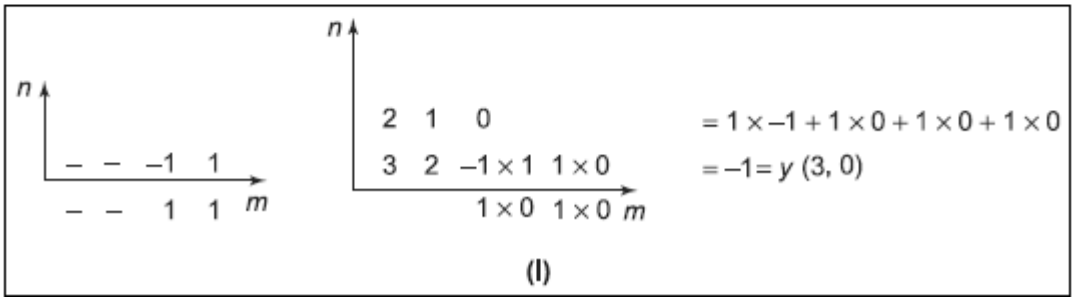
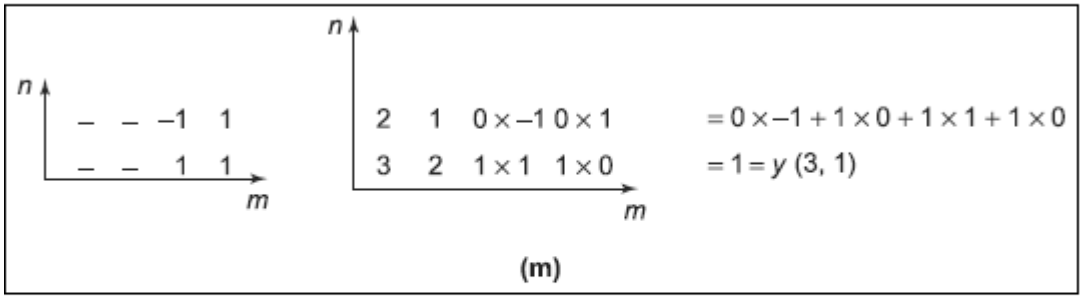
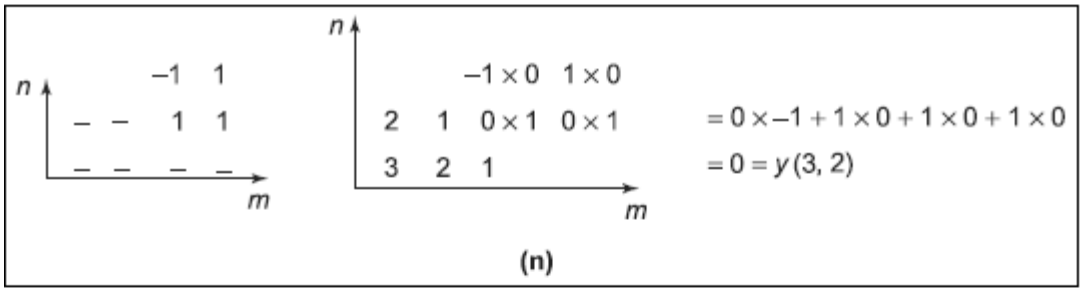


FIGURE 3.8(i–l)

$y(3, 1) : h(-2, -1)$



$y(3, 2) : h(-2, -2)$



Thus the final convoluted matrix of size 3×4 is as shown in the following:

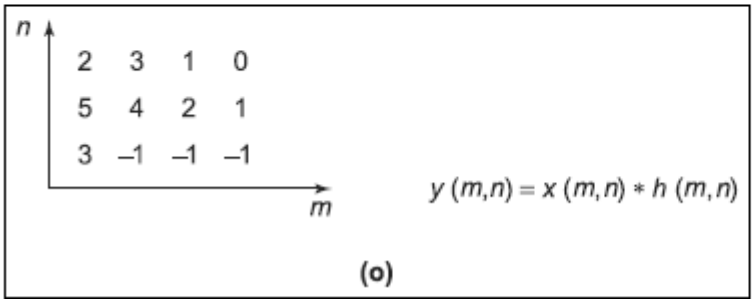


FIGURE 3.8(m–o)

Correlation The correlation of two continuous functions $f(x)$ and $g(x)$ is denoted as $f(x)$ o $g(x)$ and defined by the relation

$$f(x) \circ g(x) = \int_{-\infty}^{\infty} f^*(\alpha) g(x + \alpha) \mathrm{d}\alpha \qquad (3.51)$$

where F^* is the complex conjugate of $f(x)$.

Equation (3.51) is similar to equation (3.43) with the only difference being that the function $g(x)$ is not folded about the origin. Thus to perform correlation, one has to simply slide $g(x)$ by $f(x)$ and integrate the product from $-\infty$ to $+\infty$. The graphical illustration of the correlation process for $f(x)$ and $g(x)$ is shown in Figure 3.9. The equivalent discrete correlation process is discussed later. For each value of displacement x, the discrete correlation can be defined as

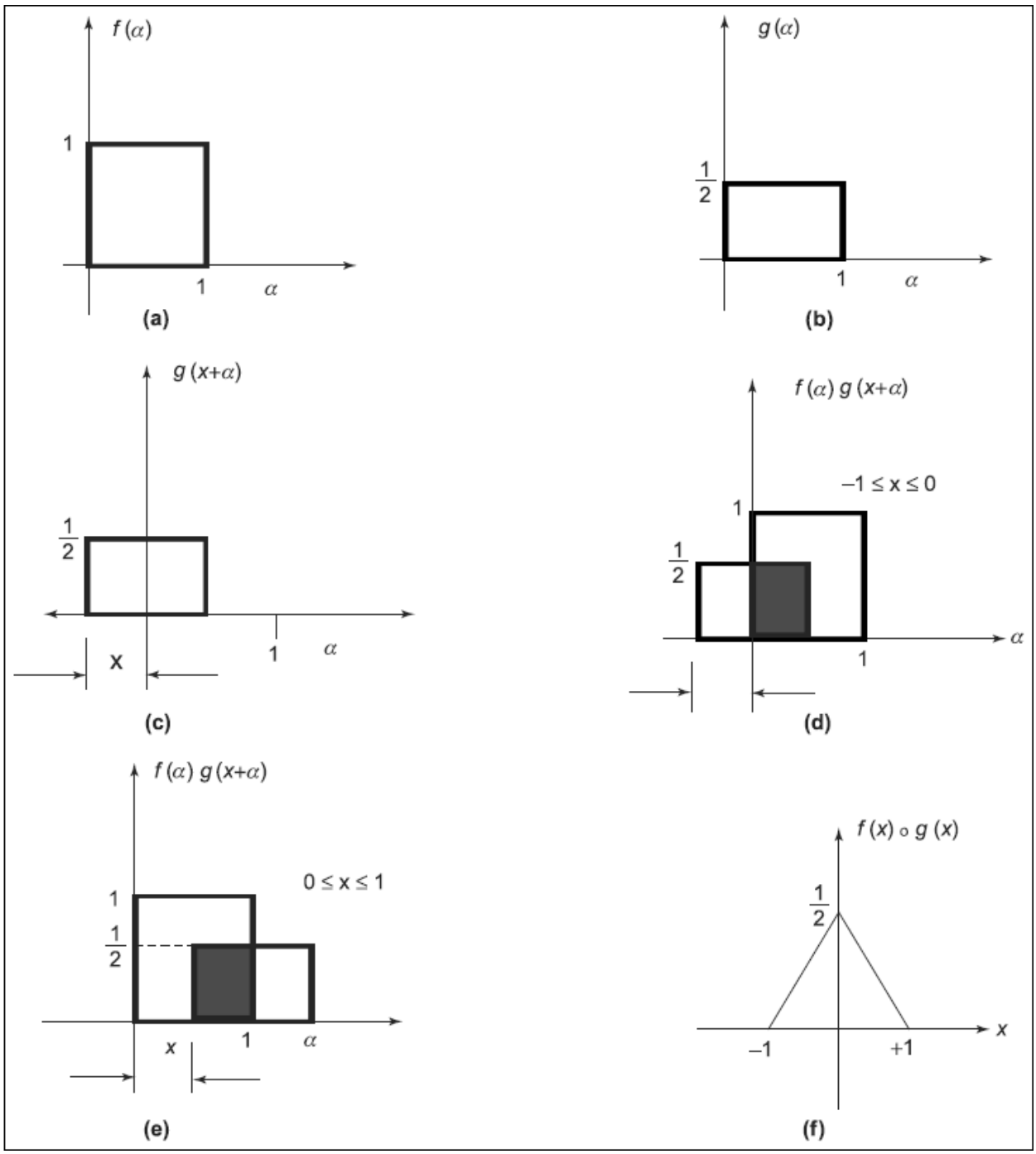


FIGURE 3.9 Graphical illustration of correlation

$$f(x) \circ g(x) = \frac{1}{M} \sum_{\alpha=0}^{M-1} f^*(\alpha) g(x+\alpha) \qquad (3.52)$$

for $x = 0, 1, 2, \ldots, M - 1$.

The correlation procedure can also be extended to discrete functions of $f(x)$ and $g(x)$. To avoid wraparound error a common interval M is used and the functions are discretized. In the case of two-dimensional functions $f(x, y)$ and $g(x, y)$ the correlation is defined as,

$$f(x, y) \circ g(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f^*(\alpha, \beta)\, g(x + \alpha, y + \beta)\, d\alpha d\beta \qquad (3.53)$$

In case of the two-dimensional discrete functions, the correlation of them is defined as

$$f_e(x, y) \circ g_e(x, y) = \frac{1}{MN} \sum_{\alpha=0}^{M-1} \sum_{\beta=0}^{N-1} f_e(\alpha, \beta) g_e(x + \alpha, y + \beta) \qquad (3.54)$$

for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$ where $f_e(x, y)$ and $g_e(x, y)$ are extended functions and M and N are as follows to avoid wraparound error.

$$M \geq A + C - 1 \qquad (3.55)$$

$$N \geq B + D - 1 \qquad (3.56)$$

Then, the correlation theorem can be stated as in equations (3.57) and (3.58).

$$f(x, y) \circ g(x, y) \Leftrightarrow F^*(u, v) G(u, v) \qquad (3.57)$$

$$f^*(x, y) g(x, y) \Leftrightarrow F(u, v) \circ G(u, v) \qquad (3.58)$$

Correlation: Yet another process that provides a way to relate the frequency and spatial domain functions. For example, the correlation of two functions $f(x, y)$ and $g(x, y)$ is equivalent to multiplying the conjugate of the Fourier transform of the first function with the Fourier transform of the second function (i.e.,) $f(x, y)$ and $g(x, y) = F^*(u, v)\, G(u, v)$.

One of the important applications of correlation in image processing is in the area of template or prototype matching, where the problem is to find the closest match between an unknown image and a set of known images. One way of obtaining this is by finding the correlation between the unknown and each of the known images. Then the closest match can be found by selecting the image that gives the correlation function with the largest value.

3.5 FAST FOURIER TRANSFORM

The computation cost of the Fourier transform is very high and hence to reduce it, the fast Fourier transform (FFT) was developed. With the introduction of the FFT the computational complexity is reduced from N^2 to $\log_2 N$. For example, for an image of size 256 × 256 pixels the processing time required is about 2 minutes on a general purpose computer. The same machine would take 30 times longer (60 minutes) to compute the Fourier transform of the same image of size 256 × 256. In this section the concept used in fast Fourier transform is described in detail.

From equation (3.14) it can be found that for each of the N values of u, the expansion of summation require N complex multiplication of $f(x)$ by $e^{\left(\frac{-j2\pi ux}{N}\right)}$ and $N - 1$ additions give the

Fourier coefficient $F(0)$. Therefore, for the computation of N Fourier coefficients, the number of complex multiplications and additions required is proportional to N^2.

The computational complexity in the implementation of equation (3.14) can be reduced from N^2 to $N \log_2 N$ by a decomposition procedure. And this procedure is called Fast Fourier Transform (FFT) algorithm.

The FFT approach offers a considerable reduction in the computational cost when N is relatively large. For example, when $N = 512$, the direct FT computational complexity proportional to $N^2 = 262, 144$ whereas the FFT computational complexity is proportional to $N \log_2 N = 2048$. This means FFT is 32 times faster than direct FT $\left[\frac{262144}{2048} = 32\right]$.

The FFT algorithm for one variable is illustrated in Section 3.5.1 and the two-dimensional FFT can be obtained by successive process of one-dimensional transform.

3.5.1 Fast Fourier Transform Algorithm

Equation (3.14) is rewritten for convenience as

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) W_N^{ux} \qquad (3.59)$$

where

$$W_N = e^{\left[\frac{-j2\pi}{N}\right]} \qquad (3.60)$$

and N is assumed to be of the form $N = 2^n$ where n is a positive integer.

Hence N can be expressed as

$$N = 2M \qquad (3.61)$$

where M is also a positive integer. By substituting equation (3.61) in equation (3.59), we get,

$$F(u) = \frac{1}{2M} \sum_{x=0}^{2M-1} f(x) W_{2M}^{ux}$$

$$= \frac{1}{2}\left[\frac{1}{M} \sum_{x=0}^{M-1} f(2x) W_{2M}^{u(2x)} + \frac{1}{M} \sum_{x=0}^{M-1} f(2x+1) W_{2M}^{u(2x+1)}\right] \qquad (3.62)$$

From equation (3.60), we can prove $W_{2M}{}^{2ux} = W_{uM}{}^{ux}$ and therefore equation (3.62) may be expressed as

$$F(u) = \frac{1}{2}\left[\frac{1}{M} \sum_{x-0}^{M-1} f(2x) W_M^{ux} + \frac{1}{M} \sum f(2x+1) W_M^{ux} W_{2M}^{u}\right] \qquad (3.63)$$

Defining

$$F_{\text{even}}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x)W_M^{ux} \qquad (3.64)$$

for $u = 0, 1, 2, \ldots, M-1$, and

$$F_{\text{odd}}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x+1)W_M^{ux} \qquad (3.65)$$

For $u = 0, 1, 2, \ldots, M-1$. Equation (3.63) reduces to

$$F(u) = \frac{1}{2}\left[F_{\text{even}}(u) + F_{\text{odd}}(u)W_{2M}^{u}\right] \qquad (3.66)$$

Also, since $W_M^{u+M} = W_M^{u}$ and $W_{2M}^{u+M} = -W_{2M}^{u}$, equation (3.66) can also be written as

$$F(u+M) = \frac{1}{2}\left[F_{\text{even}}(u) - F_{\text{odd}}(u)W_{2M}^{u}\right] \qquad (3.67)$$

The analysis of equations (3.64) to (3.67) reveals some interesting properties of these expressions. An N-point transform can be obtained by dividing the original expression into two parts, as given in equations (3.66) and (3.67). Computation of the first-half of $F(u)$ requires an evaluation of two ($\frac{N}{2}$) point transforms as given in equations (3.64) and (3.65). The resulting values of $F_{\text{even}}(u)$ and $F_{\text{odd}}(u)$ are substituted in equation (3.66) to obtain $F(u)$ for $u = 0, 1, 2, \ldots, (\frac{N}{2} - 1)$. Then the other half follows directly from equation (3.67) without any additional computation.

In order to prove that the FFT algorithm computational cost is proportional to $N \log_2 N$ the following analysis is given.

Let us assume $m(x)$ and $a(n)$ represents the number of complex multiplication and addition required to implement FFT. Let the number of samples $b_c 2^n = N = 2M$ where n is a positive integer.

First, assume that $n = 1$, therefore $N = 2$. This means that it is a two-point transformation and this requires the evaluation of $F(0)$ and $F(1)$. To obtain $F(0)$ it is required to compute $F_{even}(0)$ and $F_{odd}(0)$ using equations (3.64) and (3.65). For $M = 1$, $F_{\text{even}}(0)$ and $F_{\text{odd}}(0)$ no multiplications or additions are required. This means that for a single point the evaluation of $F_{even}(0)$, $F_{odd}(0)$ is simply the sample value itself. To obtain $F(0)$ using equation (3.67), one multiplication of $F_{odd}(0)$ by W_2^0 and one more addition are required. Then to obtain $F(1)$ using equation (3.69) one more addition (subtraction is considered as equivalent to addition) is required. As $F_{odd}(0)W_2^0$ had already been computed, the total number of operations required for a two-point transform consists of $m(1) = 1$ multiplication and $a(1) = 2$ two additions. Next consider $n = 2$. This means that the number of point $N = 2^2 (N = 2^n = 2^2 = 4 = 2M)$. According to the

previous explanation the four-point transform can be decomposed into two two-point transforms for $M = 2$.

From the previous analysis for $n = 1$, a two-point transform requires $m(1)$ multiplication and $a(1)$ addition. So the evaluation of the two equations (3.64) and (3.65) requires a total of two $m(1)$ multiplications and two $a(1)$ additions. Two further multiplications and additions are required to obtain $F(0)$ and $F(1)$ using equation (3.66). Since $F_{odd}(0)W^{u}_{2M}$ had already been completed for $u = 0, 1$ two more additions give $F(2)$ and $F(3)$. Thus

$$m(2) = 2m(1) + 2$$

$$a(2) = 2a(1) + 2^2.$$

Similar arguments can be carried out for $n = 3$ and the number of multiplications and additions to compute the FFT are given as

$$m(3) = 2m(2) + 4 = 2m(2) + 2^2$$

$$a(3) = 2a(2) + 8 = 2a(2) + 2^3$$

Then for any integer value n, the number for multiplication and additions required to implement FFT is as in equations (3.68) and (3.69).

$$m(n) = 2m(n-1) + 2^{n-1} \quad n \geq 1 \qquad (3.68)$$

$$a(n) = 2a(n-1) + 2^n \qquad n \geq 1 \text{ and} \qquad (3.69)$$

where $m(0) = 0$ and $a(0) = 0$ because the transform of a single point does not require any additions or multiplications. Hence, the number of multiplications and additions involved in the FFT algorithm is proportional to $a(n) = N \log_2 N$.

3.5.2 THE INVERSE FFT

Any algorithm used for implementing the discrete forward transform may also be used (with minor modifications in the input) to compute the inverse. This statement can be proved by the following steps:

Let us consider the equation,

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{\left[\frac{-j2\pi ux}{N}\right]} \qquad (3.70)$$

and

$$f(x) = \frac{1}{N} \sum_{u=0}^{N-1} F(u) e^{\left[\frac{j2\pi ux}{N}\right]} \qquad (3.71)$$

Taking the complex conjugate of equation (3.71) and dividing both sides by N yields

$$\frac{1}{N} f^*(x) = \frac{1}{N} \sum_{x=0}^{N-1} F^*(u) e^{\left[\frac{-j2\pi ux}{N}\right]} \qquad (3.72)$$

Comparing equation (3.70) with (3.72), right-hand side of equation (3.72) is in the form of the forward Fourier transform. Thus inputting *F*(u)* into an algorithm designed to compute the forward transform gives the quantity $f^* \frac{(x)}{N}$. Taking the complex conjugate and multiplying by *N* yields the desired inverse *f*(*x*).

For the two-dimensional square arrays, equation (3.73) is considered for finding the inverse Fourier transform.

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) e^{\left[\frac{j2\pi(ux+vy)}{N}\right]} \qquad (3.73)$$

Taking the complex conjugate of equation (3.74), that is,

$$f^*(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F^*(u, v) e^{\left[\frac{-j2\pi(ux+vy)}{N}\right]} \qquad (3.74)$$

is in the form of the two-dimensional forward transform which is given in equation (3.75).

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{\left[\frac{-j2\pi(ux+vy)}{N}\right]} \qquad (3.75)$$

Therefore, inputting *F* (u, v)* into an algorithm designed to compute the forward transform gives *f** (*x, y*). Taking the complex conjugate of this result yields *f*(*x, y*). Hence, an algorithm meant for computing forward transform can be used for computing inverse transform.

3.6 DISCRETE COSINE TRANSFORM

The popular transform used for image compression is discrete Cosine transform (DCT) which is explained in detail in this section.

The one-dimensional discrete Cosine transform is defined by equation (3.76)

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \qquad (3.76)$$

for $u = 0, 1, 2, \ldots, N-1$

Similarly, the inverse DCT is defined by equation (3.77), that is,

$$f(x) = \sum_{u=0}^{N-1} \alpha(u) C(u) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \qquad (3.77)$$

for $x = 0, 1, 2, \ldots, N-1$

The *α(u)* used in equations (3.76) and (3.77) is defined in equation (3.78).

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u = 1, 2, 3, \ldots, N-1 \end{cases} \quad (3.78)$$

The corresponding two-dimensional DCT pair is

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (3.79)$$

for $u, v = 0, 1, 2, \ldots, N-1$

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \alpha(u)\alpha(v)C(u, v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (3.80)$$

for $x, y = 0, 1, 2, \ldots, N-1$ where a is defined by equation (3.78).

3.6.1 Properties of Cosine Transform

1. The Cosine transform is real and orthogonal, that is,

$$C = C^* \Rightarrow C^{-1} = C^T \quad (3.81)$$

It is not the real part of the unitary DCT. However, the cosine transform of a sequence is related to the DFT of its symmetric extension.

2. The Cosine transform is a fast transform. The Cosine transform of a vector of N elements can be calculated in $O(N \log_2 N)$ operations via N-point FFT.

3. It has excellent energy compaction for correlated data.

3.7 WALSH TRANSFORM

The Walsh transform was introduced by Walsh in the year 1923 and contains only the entries +1 and −1. The one-dimensional discrete Walsh transform of a function $f(x)$ is denoted by $W(u)$ and is given in equation (3.82)

$$W(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \prod_{i=0}^{n-1} (-1)^{[b_i(x)b_{n-1-i}(u)]} \quad (3.82)$$

where $b_i(x)$ is the ith bit in the binary representation of x.

For example,

$$\text{If } n = 3 \text{ and } x = 6(110 \text{ in binary}),$$

$$b_0(x) = 0, b_1(x) = 1 \text{ and } b_2(x) = 1$$

The term

$$\frac{1}{N}\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)]} \qquad (3.83)$$

is called the kernel and denoted as *g(x, u)*.

In other words

$$g(x,u) = \frac{1}{N}\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)]} \qquad (3.84)$$

The array formed by the Walsh transformation is a symmetric matrix having orthogonal rows and columns. The inverse kernel is identical to the forward kernel except for the multiplication factor $\frac{1}{N}$ which is given in equation (3.85)

$$h(x,u) = \prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)]} \qquad (3.85)$$

Hence, the inverse Walsh transform can be written as

$$f(x) = \sum_{u=0}^{N-1} W(u)\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)]}$$

The Walsh transform consists of a series of expansion of basis functions whose values are +1 and −1. The forward and inverse Walsh kernel for the two-dimensional case is given by the relations

$$g(x,y,u,v) = \frac{1}{N}\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]} \qquad (3.86)$$

and

$$h(x,y,u,v) = \frac{1}{N}\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)]} \qquad (3.87)$$

The forward and inverse transform are given by the relations

$$W(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)+b_i(y)\,b_{n-1-i}(v)]} \qquad (3.88)$$

and

$$f(x,y) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} W(u,v)\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)+b_i(y)\,b_{n-1-i}(v)]} \qquad (3.89)$$

From equations (3.88) and (3.89) it is clear that the two-dimensional forward Walsh transform may also be used without modification to compute the inverse transform.

The Walsh transform kernel is separable and symmetric because

$$\begin{aligned} g(x, y, u, v) &= g_1(x, u)g_1(y, v) \\ &= h_1(x, u)h_1(y, v) \\ &= \left[\frac{1}{\sqrt{N}}\prod_{i=0}^{n-1}(-1)^{[b_i(x)b_{n-1-i}(u)]}\right]\left[\frac{1}{\sqrt{N}}\prod_{i=0}^{n-1}(-1)^{[b_i(y)b_{n-1-i}(v)]}\right] \end{aligned} \tag{3.90}$$

Hence, *W(u, v)* and its inverse may be computed by successive applications of the one-dimensional Walsh transform in equation (3.86). Similar to FFT, the Walsh transform may be computed by a fast algorithm nearby, identical in form to the successive doubling method.

Table 3.1 Walsh transformation kernel for $N = 8$

U \ X	0	1	2	3	4	5	6	7
0	+	+	+	+	+	+	+	+
1	+	+	+	+	–	–	–	–
2	+	+	–	–	+	+	–	–
3	+	+	–	–	–	–	+	+
4	+	–	+	–	+	–	+	–
5	+	–	+	–	–	+	–	+
6	+	–	–	+	+	–	–	+
7	+	–	–	+	–	+	+	–

3.8 HADAMARD TRANSFORM

The Hadamard transform is based on basis functions that are simply +1 or −1, instead of the more complex sine and cosine functions used in the Fourier transform.

The one-dimensional kernel for the Hadamard transform is given in the relation

$$g(x, u) = \frac{1}{N}(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)]} \tag{3.91}$$

where the summation in the exponent is performed in modulo 2 arithmetic and $b_i(x)$ is the *i*th bit in binary representation of *x*. The one-dimensional equation for the Hadamard transform is

$$H(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)]} \qquad (3.92)$$

where $N = 2^n$ and $u = 0, 1, 2, \ldots, N-1$.

As the Hadamard kernel forms the matrix having orthogonal rows and columns the inverse kernel exists and is given by

$$h(x, u) = (-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)]} \qquad (3.93)$$

Hence, the inverse Hadamard transform expression is as in equation (3.94).

$$f(x) = \sum_{u=0}^{N-1} H(u)(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)]} \qquad (3.94)$$

for $x = 0, 1, 2, \ldots, N-1$.

Similarly, the two-dimensional kernels are given by the relations

$$g(x, y, u, v) = \frac{1}{N}(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)+b_i(y)b_i(v)]} \qquad (3.95)$$

and

$$h(x, y, u, v) = \frac{1}{N}(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)+b_i(y)b_i(v)]} \qquad (3.96)$$

From equations (3.94) and (3.95) Hadamard transforms kernels are identical. Hence, the two-dimensional Hadamard transform pair

$$H(u, v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y)(-1)^{\sum_{i=0}^{N-1}[b_i(x)b_i(u)+b_i(y)b_i(v)]} \qquad (3.97)$$

and

$$f(x, y) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} H(u, v)(-1)^{\sum_{i=0}^{N-1}[b_i(x)b_i(u)+b_i(y)b_i(v)]} \qquad (3.98)$$

The Hadamard kernel satisfies the separable and symmetric properties and they are

$$\begin{aligned} g(x, y, u, v) &= g_1(x, u)g_1(y, v) \\ &= h_1(x, u)h_2(y, v) \\ &= \left[\frac{1}{\sqrt{N}}(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)]}\right]\left[\frac{1}{\sqrt{N}}(-1)^{\sum_{i=0}^{n-1}[b_i(y)b_i(v)]}\right] \end{aligned} \qquad (3.99)$$

As the two-dimensional Hadamard kernels are separable, the two-dimensional transform pair may be obtained by successive applications of the one-dimensional Hadamard transform algorithm.

The use of Walsh and Hadamard transforms is intermixed in the image processing literature. The term Walsh-Hadamard is often used to denote either transform. The reason for this is that the Hadamard transform can be obtained from the Walsh transform. For example, the Hadamard transform kernel (Table 3.2) can be obtained from the Walsh transform kernel Table (3.1) by reordering the columns.

Table 3.2 The one-dimensional Hadamard transformation kernel for $N = 8$

U \ X	0	1	2	3	4	5	6	7
0	+	+	+	+	+	+	+	+
1	+	–	+	–	+	–	+	–
2	+	+	–	–	+	+	–	–
3	+	–	–	+	+	–	–	+
4	+	+	+	+	–	–	–	–
5	+	–	+	–	–	+	–	+
6	+	+	–	–	–	–	+	+
7	+	–	–	+	–	+	+	–

By interchanging columns 1 and 4 and interchanging columns 3 and 6 in the Walsh transform, the Hadamard transform can be obtained. The important feature of the Walsh transform is that it can be expressed directly in a successive doubling format in order to obtain fast Walsh transform, whereas the Hadamard transform due to its ordering does not allow the successive doubling format to obtain fast Hadamard transform.

However, a Hadamard transform leads to a simple recursive relationship for generating the transformation matrices from the lower order to higher order. For example, $n = 2$, the Hadamard matrix can be given as

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad (3.100)$$

To obtain, the Hadamard matrix for $N = 4$, the recursive relationship given in equation (3.100) can be used.

$$H_4 = \begin{pmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{pmatrix} = \begin{bmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{bmatrix} \qquad (3.101)$$

In general,

$$H_{2N} = \begin{pmatrix} H_N & H_N \\ H_N & -H_N \end{pmatrix} \qquad (3.102)$$

3.9 THE HAAR TRANSFORM

The Haar transform is considerably less useful in practice compared to the other transforms discussed earlier. This has been included here for completeness. The Haar transform is based on the Haar functions $h_k(z)$ which are defined in the continuous closed interval $z\varepsilon[0, 1]$ and for $k = 0, 1, 2, \ldots, N-1$, where $N = 2^n$.

In order to generate the Haar transform, it is necessary to decompose the integer k uniquely as given in equation (3.103)

$$k = 2^p + q + 1 \qquad (3.103)$$

where P takes values from 0 to $n-1$, $q = 0$ or 1 for $p = 0$ and q takes values from 1 to 2^P for $p \neq 0$.

For example $N = 4$, k, p, q have the following values:

k	P	Q
0	0	0
1	0	1
2	1	1
3	1	2

From the above details, the Haar transform functions are defined as

$$h_0(z) \underline{\underline{\Delta}} h_{00}(z) = \frac{1}{\sqrt{N}} \quad \text{for} \quad z\varepsilon(0, 1) \qquad (3.104)$$

and

$$h_k(z) \underline{\underline{\Delta}} h_{pq}(z) = \frac{1}{\sqrt{N}} \begin{cases} 2^{\frac{p}{2}} & \dfrac{(q-1)}{2^p} \leq z < \dfrac{\left(q-\frac{1}{2}\right)}{2^p} \\ -2^{\frac{p}{2}} & \dfrac{\left(q-\frac{1}{2}\right)}{2^p} \leq z < \dfrac{q}{2^p} \\ 0 & \text{otherwise} \quad \text{for} \quad z \in [0, 1] \end{cases} \qquad (3.105)$$

In general, the Haar transformation matrix order is denoted as $N \times N$. The ith row of the Haar matrix can be obtained from the elements of $h_i(z)$.

For $z = \frac{0}{N}, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{(N-1)}{N}$. For instance, when $N = 2$, the first row of the 2×2 Haar matrix is computed by using $h_0(z)$ with $z = \frac{0}{2}, \frac{1}{2}$. From equation (3.105) the first row of the matrix has two identical elements $\frac{1}{\sqrt{2}}$. The second row is obtained by $h_1(z)$ with $z = \frac{0}{2}, \frac{1}{2}$, when $k = 1$, $p = 0$, and $q = 1$ using the equation (3.103).

Thus from equation (3.104)

$$h_1(0) = \frac{2^0}{\sqrt{2}} = \frac{1}{\sqrt{2}}$$

and

$$h_1\left(\frac{1}{2}\right) = -\frac{2^0}{\sqrt{2}} = -\frac{1}{\sqrt{2}}$$

Therefore, the 2×2 Haar matrix is given by

$$A_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Similarly, for matrix $N = 4$

$$A_4 = \frac{1}{\sqrt{4}} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -\sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & -\sqrt{2} & -\sqrt{2} \end{pmatrix}$$

The Haar matrices are orthogonal and it allows us to have a fast Haar algorithm.

3.10 THE SLANT TRANSFORM

Slant transform plays a vital role in image compression techniques. Slant transform has been proven to be superior from the standpoint of image quality compared to other transforms. Studies of slant transform reveal that the average coding of a monochrome image can be reduced from 8 bits/pixels to 1 bit/pixel without seriously degrading the image quality. For colour images 24 bits/pixels can be reduced to 2 bits per pixels while preserving quality reconstruction.

The Slant transform matrix of order $N \times N$ is given by the recursive expression as

$$S_N = \frac{1}{\sqrt{2}} \left[\begin{array}{cc|c|cc|c} 1 & 0 & & 1 & 0 & \\ a_N & b_N & 0 & -a_N & b_N & 0 \\ \hline & 0 & & & 0 & \\ & & I_{\left(\frac{N}{2}\right)-2} & & & I_{\left(\frac{N}{2}\right)-2} \\ \hline 0 & 1 & & 0 & -1 & \\ -b_N & a_N & 0 & b_N & a_N & 0 \\ \hline & 0 & & & 0 & \\ & & I_{\left(\frac{N}{2}\right)-2} & & & I_{\left(\frac{N}{2}\right)-2} \end{array}\right] \left[\begin{array}{c|c} S_{\frac{N}{2}} & 0 \\ \hline 0 & S_{\frac{N}{2}} \end{array}\right] \tag{3.106}$$

where I_M is the identity matrix of order $M \times M$.

The slant matrix of the order 2×2 is as in the following:

$$S_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.107}$$

The coefficients are defined as

$$a_N = \left[\frac{3N^2}{4(N^2-1)}\right]^{\frac{1}{2}} \tag{3.108}$$

and

$$b_N = \left[\frac{N^2-4}{4(N^2-1)}\right]^{\frac{1}{2}} \quad \text{for} \quad N > 1 \tag{3.109}$$

Using equations the Slant matrix S_4 is

$$S_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ \frac{3}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} \\ 1 & -1 & -1 & 1 \\ \frac{1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} & \frac{3}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \end{pmatrix}$$

The Slant matrixes are orthogonal and have necessary properties to allow implementation of a fast slant transform algorithm based on the above matrix formulation.

3.11 THE HOTELLING TRANSFORM

The Hotelling transform is based on statistical properties and has several useful properties that make it an important tool for image processing. In order to illustrate the use of Hotelling transform in image processing let us proceed with the following mathematical analysis.

Consider the random vectors of the form

$$X = \begin{pmatrix} X_1 \\ X_2 \\ . \\ . \\ . \\ X_n \end{pmatrix} \qquad (3.110)$$

The mean vector of X is defined as

$$m_x = E\{X\}$$

where E {arg} is the expected value of the argument and the subscript denotes that m is associated with the population of X vectors.

The covariance matrix of the vector population is defined as

$$C_X = E\{(X - m_x)(X - m_x)^T\} \qquad (3.111)$$

where T indicates vector transposition.

We know that X is $n \times 1$ dimensional, then C_x and $(X-m_x)(X-m_x)^T$ are matrices of the order of $n \times n$. The element C_{ii} of C_x is the variance of X_i, ith component of the X vectors, and the element C_{ij} of C_x is the covariance between elements X_i and Xj of these vectors. The matrix C_X is real and symmetric.

If $C_{ij} = C_{ji} = 0$ then the elements X_i and X_j are uncorrelated.

For M vector samples from a random population, the mean vector and covariance matrix can be approximated from the samples by the equations

$$m_x = \frac{1}{M}\sum_{k=1}^{M} X_k \qquad (3.112)$$

and

$$C_x = \frac{1}{M}\sum_{k=1}^{M} X_k X_k^T - m_x m_x^T \qquad (3.113)$$

The following illustrative examples show how to compute the mean vector m_x and the covariance matrix C_x from the following column vectors.

$$X_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \qquad X_3 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$$X_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad X_4 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The average vector m_x and the $m_x m_x^T$ are computed as follows

$$m_x = \frac{1}{4}\begin{pmatrix} 3 \\ 2 \\ 3 \end{pmatrix} \qquad m_x m_x^T = \frac{1}{4}\begin{pmatrix} 3 \\ 2 \\ 3 \end{pmatrix} \frac{1}{4}\begin{pmatrix} 3 & 2 & 3 \end{pmatrix}$$

$$= \frac{1}{16}\begin{pmatrix} 9 & 6 & 9 \\ 6 & 4 & 6 \\ 9 & 6 & 9 \end{pmatrix}$$

Then we compute

$$\sum_{k=1}^{4} X_k X_k^T = X_1 X_1^T + X_2 X_2^T + X_3 X_3^T + X_4 X_4^T$$

where

$$X_1 X_1^T = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$X_2 X_2^T = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$X_3 X_3^T = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$X_4 X_4^T = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Then the covariance matrix is given by

$$C_x = \frac{1}{M}\sum_{k=1}^{M} X_k X_k^T - m_x m_x^T$$

$$C_x = \frac{1}{4}\begin{pmatrix} 3 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 3 \end{pmatrix} - \frac{1}{16}\begin{pmatrix} 9 & 6 & 9 \\ 6 & 4 & 6 \\ 9 & 6 & 9 \end{pmatrix}$$

$$= \frac{1}{16}\begin{pmatrix} 12 & 8 & 8 \\ 8 & 8 & 4 \\ 8 & 4 & 12 \end{pmatrix} - \frac{1}{16}\begin{pmatrix} 9 & 6 & 9 \\ 6 & 4 & 6 \\ 9 & 6 & 9 \end{pmatrix}$$

$$= \frac{1}{16}\begin{pmatrix} 3 & 2 & -1 \\ 2 & 2 & -2 \\ -1 & -2 & 3 \end{pmatrix}$$

Since C_x is real and symmetric, finding a set of orthogonal eigen vector is always possible. Let e_i and $\lambda_i$be the eigen vectors and the corresponding eigen values of C_x, where i takes values from 1 to n. The eigen values are arranged in the descending order so that $\lambda_j \geq \lambda_{j+1}$ for j = 1 to $n - 1$.

Let A be a matrix whose rows are formed from the eigen vectors of C_x, ordered so that the first row of A is the eigen vector corresponding to the largest eigen value and the last row is the eigen vector corresponding to the smallest eigen value.

suppose that A is a transformation matrix that maps the x's into vectors denoted by y's and given by equation (3.114).

$$y = A(X - m_x) \qquad (3.114)$$

Equation (3.114) is called the Hotelling transform. The mean of the y vectors resulting from this transformation is zero, that is,

$$m_y = 0 \qquad (3.115)$$

and the covariance matrix of the y's can be obtained in terms of A and C_x by

$$C_y = AC_xA^T \qquad (3.116)$$

C_y is the diagonal matrix whose elements along the main diagonal are the eigen values of C_x, that is,

$$C_y = \begin{pmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & . & \\ & & & . \\ 0 & & & \lambda_n \end{pmatrix} \qquad (3.117)$$

The off diagonal elements of the covariance matrix are 0, so the elements of the y vectors are uncorrelated. Thus C_x and C_y have same eigen values. In fact, the same is true for the eigen vectors. The net effect of equation (3.115) is to establish a new coordinate system whose origin is at the centroid of the population and whose axes are in the direction of the eigen vectors of C_x.

One of the applications of Hotelling transform is to align the two-dimensional object with its principal eigen vectors axis. To illustrate this consider a two-dimensional image as shown in Figure 3.10(a). Equation (3.114) is used to obtain a new coordinate system whose center is at the centroid of the population and whose axes are in the direction of the eigen vectors of the covariance matrix C_x as shown in Figure 3.10(b).

Equation (3.114) is a rotational transformation that aligns the data with the eigen vectors as shown in Figure 3.10(c). The *y* axes are called eigen axes. Thus, if the images under consideration are already rotated, they can be brought to normal conditions by using Hotelling transform.

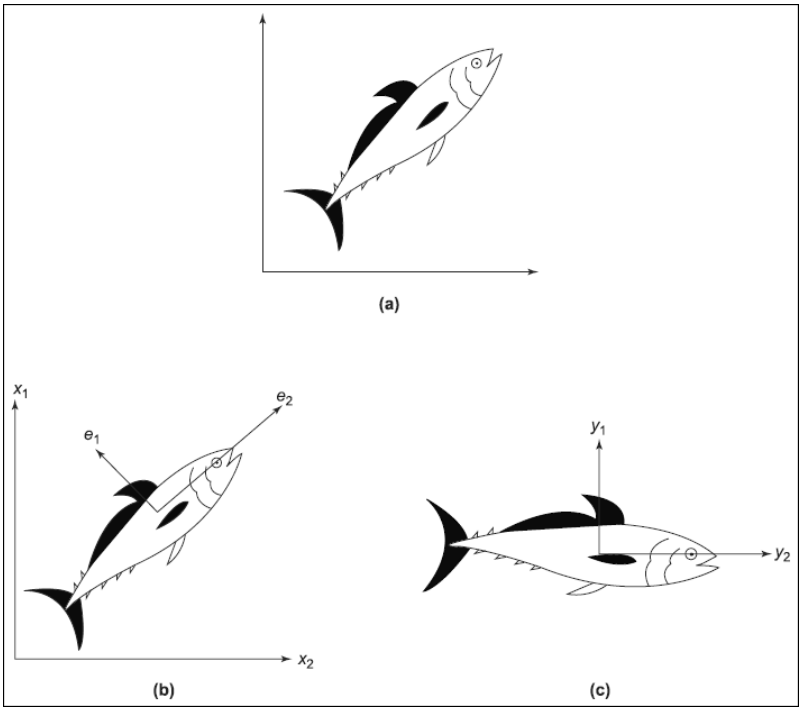


FIGURE 3.10 A two-dimensional image. (a) The original image (b) The eigen vectors superimposed on the image (c) The image rotated by using Hotelling transform

Summary

The principal objective of this chapter is to present the theoretical foundation of digital image transforms. This material is presented in such a way that all the mathematical formulae and theory are readily understandable. Although it illustrates every step in deriving formulas, prior mathematical knowledge is necessary.

Transforms are new image processing tools that are being applied to a wide variety of image processing problems. Fourier transform and similar frequency transform techniques are widely used in image understanding and image enhancement techniques. Fast Fourier transform is the variation of Fourier transform in which the computing complexity is largely reduced. Because of its less computing complexity, many of the transform techniques has its Fast transform counterparts. The various properties of Fourier transforms are explained with relevant examples. This enables the students to gain an in-depth knowledge on Fourier transform and their applications.

This chapter also describes the DCT—one of the important transforms that have a wide range of applications. Discrete Cosine transform is mainly used in image enhancement and image compression. Other equally efficient transforms such as Walsh transform, Hadamard transform and Hotelling transform are also covered in this chapter. An illustrative example to obtain the covariance matrix is well explained. This acts as a hands-on tool for subsequent discussion of image enhancement and image compression techniques.

UNIT – III

Image Enhancement: Back ground enhancement by point processing – Histogram Processing – Spatial Filtering – Enhancement in frequency Domain – Image Smoothing – Image Sharpening

Colour Images: Colour Image Processing – Pseudo colour image processing – Full colour image processing.

INTRODUCTION

In the first chapter, we have described the fundamental steps involved in digital image processing. The various steps are image acquisition, preprocessing, segmentation, representation and description, recognition, and interpretation. Before we proceed with the segmentation process it is necessary to condition the image. The conditioning of the image can be carried out by preprocessing. One of the preprocessing techniques is image enhancement.

Image enhancement technique is defined as a process of an image processing such that the result is much more suitable than the original image for a ‘specific’ application. The word specific is important because the method that is more useful for an application (say for X-ray

images) may not be suitable for another application (say pictures of Mars transmitted by space probe).

The image enhancement approaches can be put into two broad categories and they are

1. Spatial domain approach
2. Frequency domain approach

In the spatial domain approach the pixels of an image are manipulated directly. The frequency domain approach is mainly based on modifying the Fourier transform of an image. The enhancement techniques based on various combinations of methods are given in this chapter. Section 4.2 describes the basic ideas of spatial domain and frequency domain methods. Section 4.3 deals with enhancement techniques under the point processing categories. Section 4.4 deals with the enhancement methods based on mask processing. In Section 4.5 we discuss the enhancement techniques using Fourier transform. The concluding Section 4.6 discusses the enhancement techniques for color images. Tree diagram representation of image enhancement techniques is given in Figure 4.1.

SPATIAL DOMAIN AND FREQUENCY DOMAIN APPROACHES

In the spatial domain method, the pixel composing of image details are considered and the various procedures are directly applied on these pixels. The image processing functions in the spatial domain may be expressed as

$$g(x, y) = T[f(x, y)] \qquad (4.1)$$

Where $f(x, y)$ is the input image, $g(x, y)$ is the processed output image and T represents an operation on 'f' defined over some neighborhood of (x, y). Sometimes T can also be used to operate on a set of input images. Consider an image representation shown in Figure 4.2.

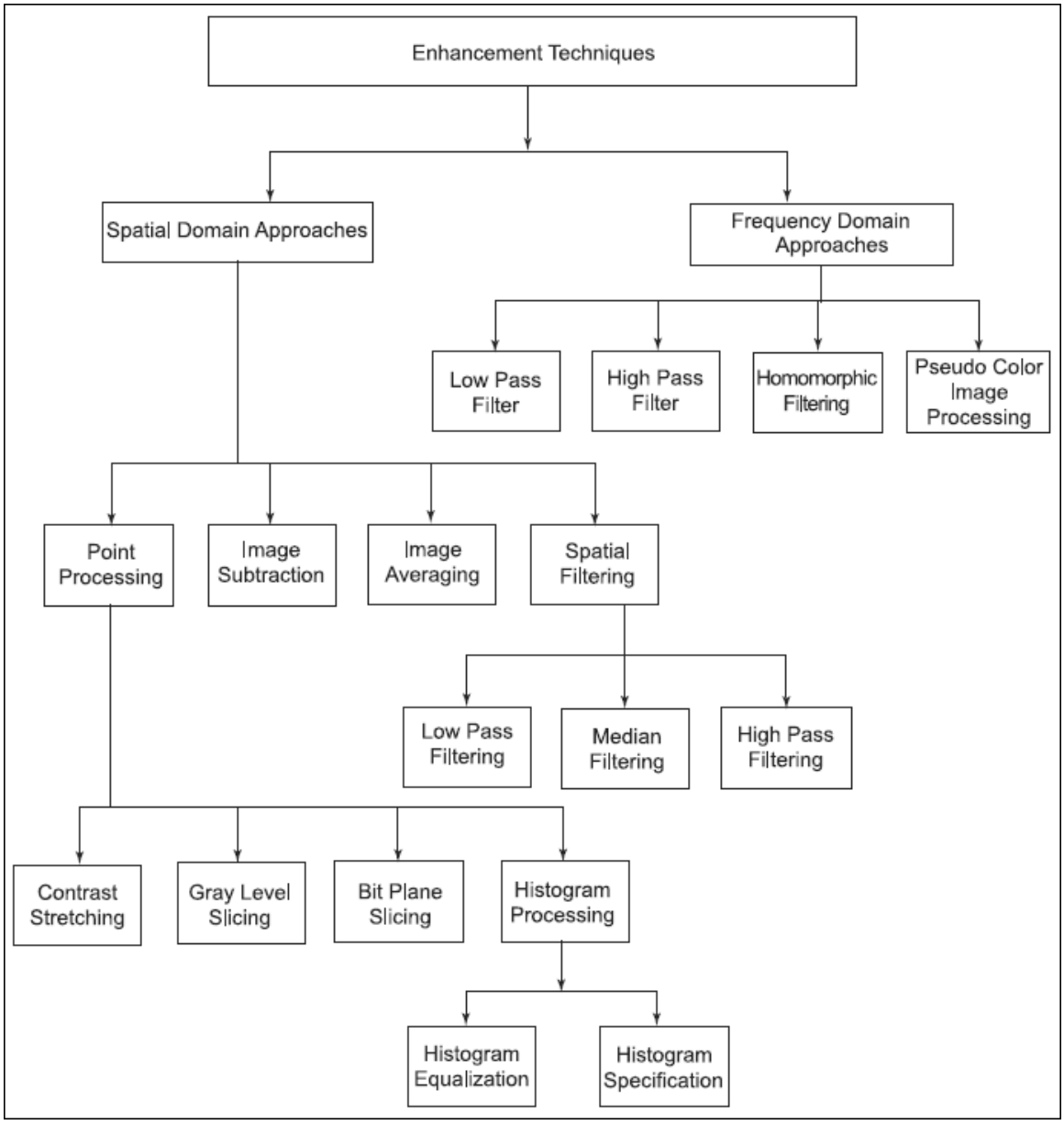


FIGURE 4.1 Tree diagram for image enhancement techniques

In Figure 4.2, a sub image of size (3 × 3) about a point (*x*, *y*) is given. The center of the sub image is moved from pixel to pixel starting at the top left corner and applying the operator at each location (*x*, *y*) to give the output image *'g'* at that location. The subimage considered may be circle, square, or rectangular arrays.

If we consider the simplest case, where the neighborhood is (1 * 1), the output image *g* depends only on the value of *f* at (*x*, *y*) and *T* is called a *gray level transformation function.* The gray level transformation function will then be given in the form

$$S = T(r) \qquad (4.2)$$

Where, *r* and *s* are variables denoting the gray level of *f*(*x*, *y*) and *g*(*x*, *y*) at any point (*x*, *y*).

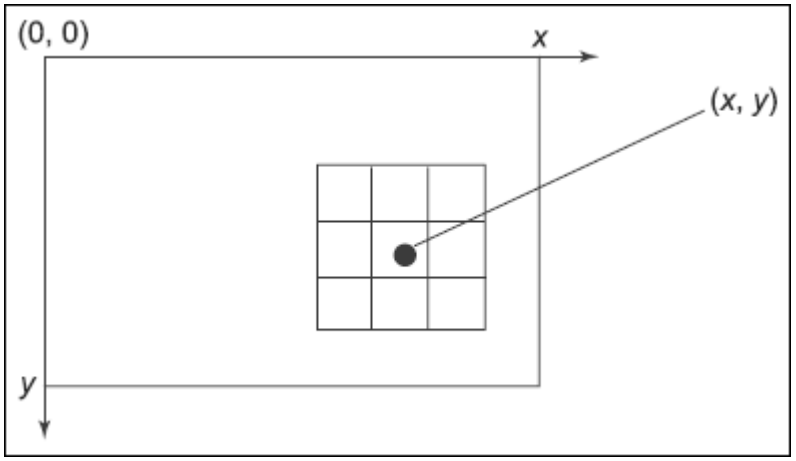


FIGURE 4.2 Digital image processing

If $T(r)$ is of the form shown in Figure 4.3(a) the effect of this transformation is to produce an image of higher contrast compared to the original image by darkening the levels below m and brightening the levels above m in the original image. This approach is called contrast stretching, because the values of r, below m are compressed, and the opposite effect takes place for the values above m.

Contrast Stretching: An enhancement technique used to increase the dynamic range of gray levels in the image being processed.

Figure 4.3(b) shows a typical case of transfer function $T(r)$ that produces a binary image. In general, the methods just described use a transfer function to produce the gray level in output image at the location (x, y) that depends only on the gray level of the input image at that location. These techniques are often referred to as point processing techniques. The larger neighborhoods allow a variety of processing function that go beyond just image enhancement. The neighborhood of pixels considered about the center pixel (x, y) is called as mask, template, or window. A mask is a small two-dimensional array (say, (3×3)) shown in Figure 4.2. The enhancement techniques based on this type are often called as mask processing or filtering.

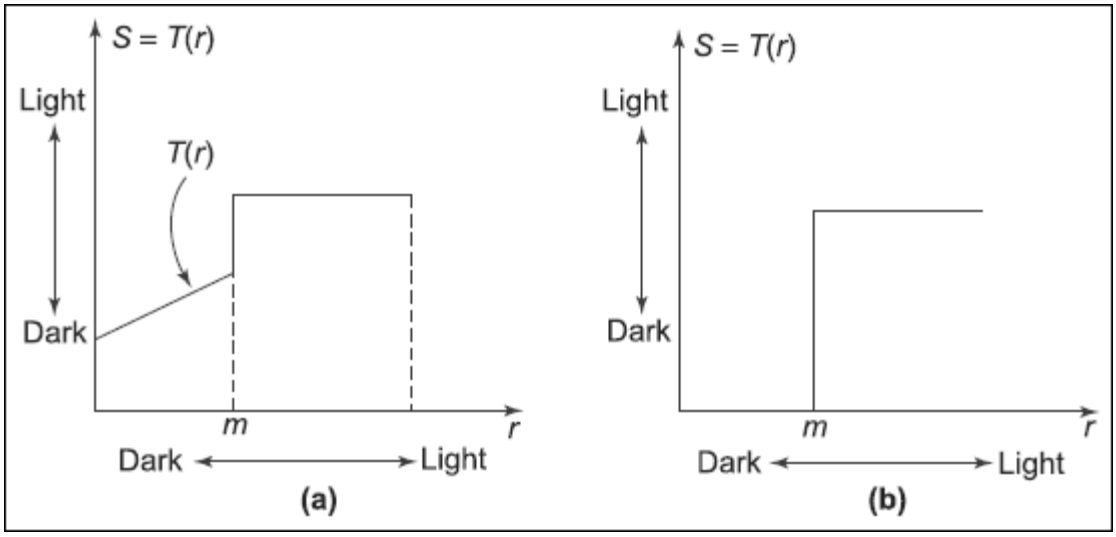


FIGURE 4.3 A typical case of transfer function

4.2.1 FREQUENCY DOMAIN TECHNIQUES

The convolution theorem is the basis for the frequency domain approaches. Let $G(x, y)$ be an image formed by the convolution of the image $f(x, y)$ and a linear position invariant operator $H(x, y)$ and is given by,

$$G(x, y) = H(x, y) * f(x, y) \qquad (4.3)$$

Where '*' represents the convolution operation.

Then, from the convolution theorem, the equation 4.3 can be written in the frequency domain as,

$$G(u, v) = H(u, v)F(u, v) \qquad (4.4)$$

In the linear system theory, the transform $H(u, v)$ is called the *Transfer function.* The various image enhancement problems can be expressed in the form of equation (4.3). In a typical image enhancement application, the image $f(x, y)$ is given and the objective after the computation of $F(u, v)$ is to select $H(u, v)$ so that the desired image can be given by the equation

$$g(x, y) = F^{-1}[H(u, v)F(u, v)] \qquad (4.5)$$

This equation shows some highlighted feature of the original image $f(x, y)$. For example, the edges in $f(x, y)$ can be highlighted using a function $H(u, v)$ that emphasize the high frequency component of $F(u, v)$.

Figure 4.4 illustrates the various steps involved in the enhancement approach based on frequency domain.

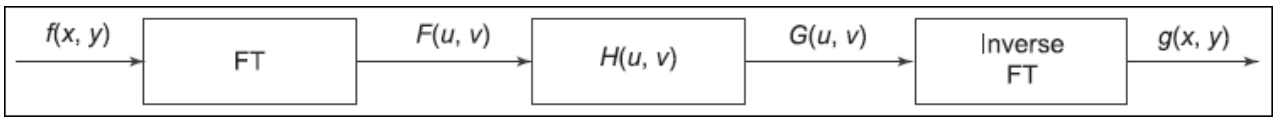


FIGURE 4.4 Enhancement steps in frequency domain approaches

4.3 SPATIAL DOMAIN TECHNIQUES

In this section, the image enhancement techniques based only on the intensity of single pixel is considered. The single point processes are the simplest among the various image enhancement techniques.

Let us consider '*r*' denotes the intensity of pixel in the given original image and '*s*' represents the intensity of the pixels in the enhanced image.

4.3.1 NEGATIVE OF AN IMAGE

There are a number of applications in which negative of the digital images are quiet useful. For example, displaying of medical images and photographing a screen with monochrome positive film with the idea of using the resulting negatives as normal slides. The negative of the digital image is obtained by using the transformation function $s = T(r)$ and it is shown in Figure 4.5(a), where L is the number of gray levels. The idea is to reverse the order from black to white so that the intensity of the output image decreases as the intensity of the input increases. The transformation function given in Figure 4.5(a) is applied to the original Lena image shown in Figure 4.5(b). The resulting negative of the original image is shown in Figure 4.5(c).

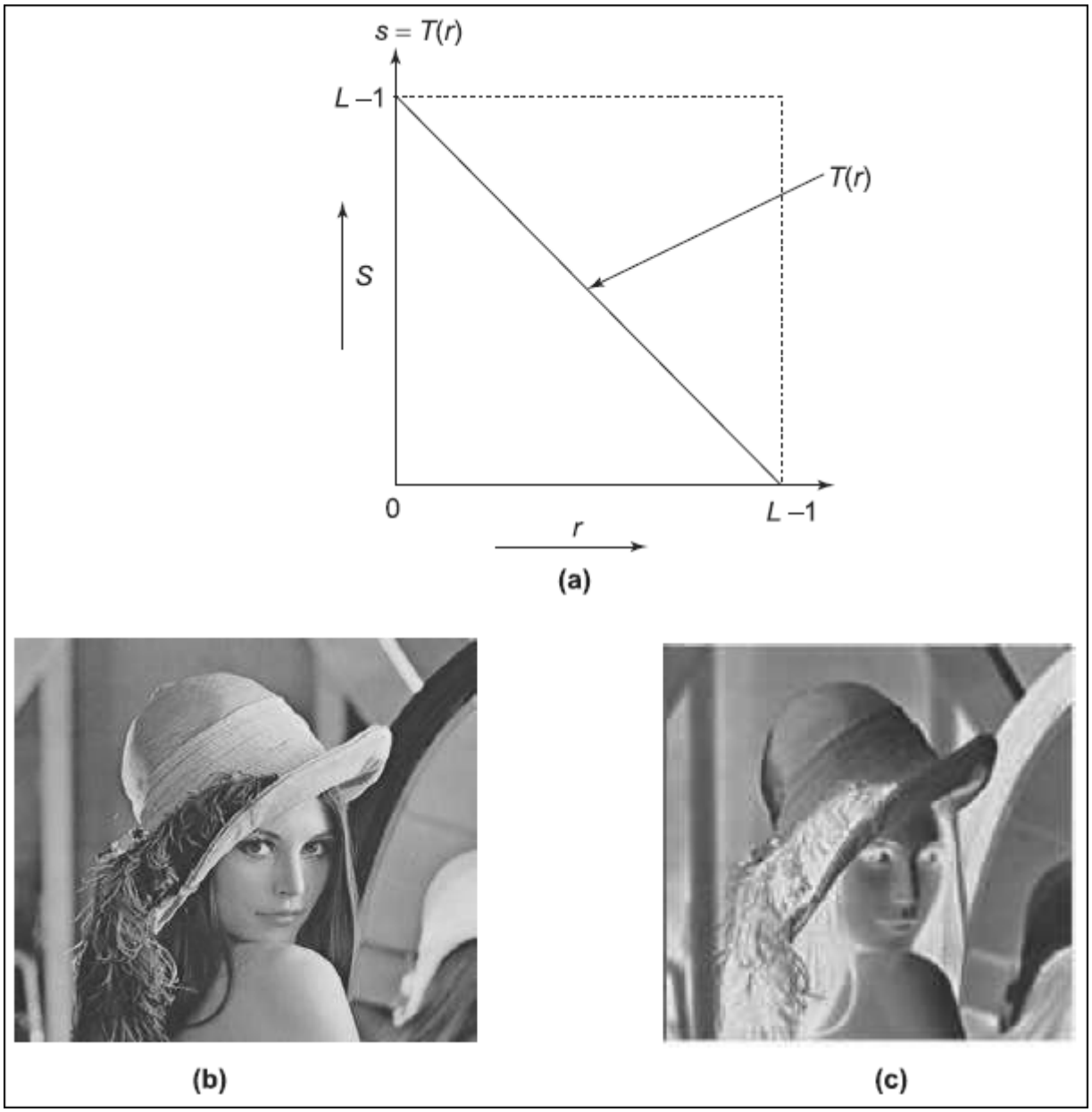


FIGURE 4.5 (a) Gray level transformation function (b) Original image (c) Negative of the original image

4.3.2 CONTRAST STRETCHING

Sometimes during image acquisition low contrast images may result due to one of the following reasons:

- poor illumination
- lack of dynamic range in the image sensor and
- Wrong setting of the lens aperture.

The idea behind the contrast stretching is to increase the dynamic range of gray levels in the image being processed. Figure 4.6(a) shows a typical transformation function used for contrast stretching.

The location of the points (r_1, s_1) and (r_2, s_2) control the shape of the transformation function. If $r_1 = r_2$, $s_1 = 0$, and $s_2 = L - 1$, the transformation becomes the thresholding function and creates a binary image. Intermediate values of r_1, s_1, and r_2, s_2 produces various degree of spread in the gray levels at the output image, thus affecting its contrast.

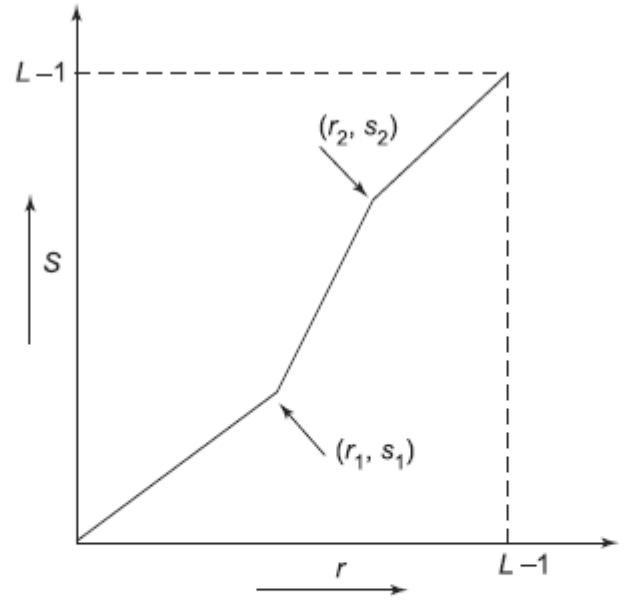
Figure 4.6(b) shows an 8-bit image with low contrast and 4.6(c) shows the result of contrast stretching. Figure 4.6(d) shows the result of thresholding the original image. The thresholding level is $r = 128$, with the output set at 255 for any gray level in the input image of 128 or higher and at 0, for all other values.

4.3.3 Gray Level Slicing

There are numerous applications in which highlighting a specific range of gray levels in an image is often required. For example, enhancing the flaws in X-ray images and enhancing features such as masses of water in satellite imagery. There are two basic approaches for doing gray level slicing.

Gray Level Slicing: An enhancement technique in which all the gray levels in the range of interest are displayed using high values and all other gray levels are displayed using low gray levels.

In the first approach, all the gray levels in the range of interest are displayed using a high value and all other gray values are displayed using low values. The corresponding transformation function used is shown in Figure 4.7(a) and this results in a binary image.



(a)

(b)

(c)

(d)

Min. & Max. value of R = 0, 255
Min. & Max. value of G = 0, 255
Min. & Max. value of B = 0, 255

Threshold Value of R, G, B = 127, 127, 127

FIGURE 4.6 (a) A form of transformation function (b) Low-cost image (c) Image after contrast stretching (d) Thresholded image

The second approach is based on transformation function shown in Figure 4.7(b). This transfer function brightens the desired range of gray levels but preserves the background and the gray level tonalities in the image. Figure 4.7(c) shows the original image and 4.7(d) shows the image resulted after applying the gray level slicing using the transformation function shown in Figure 4.7(a).

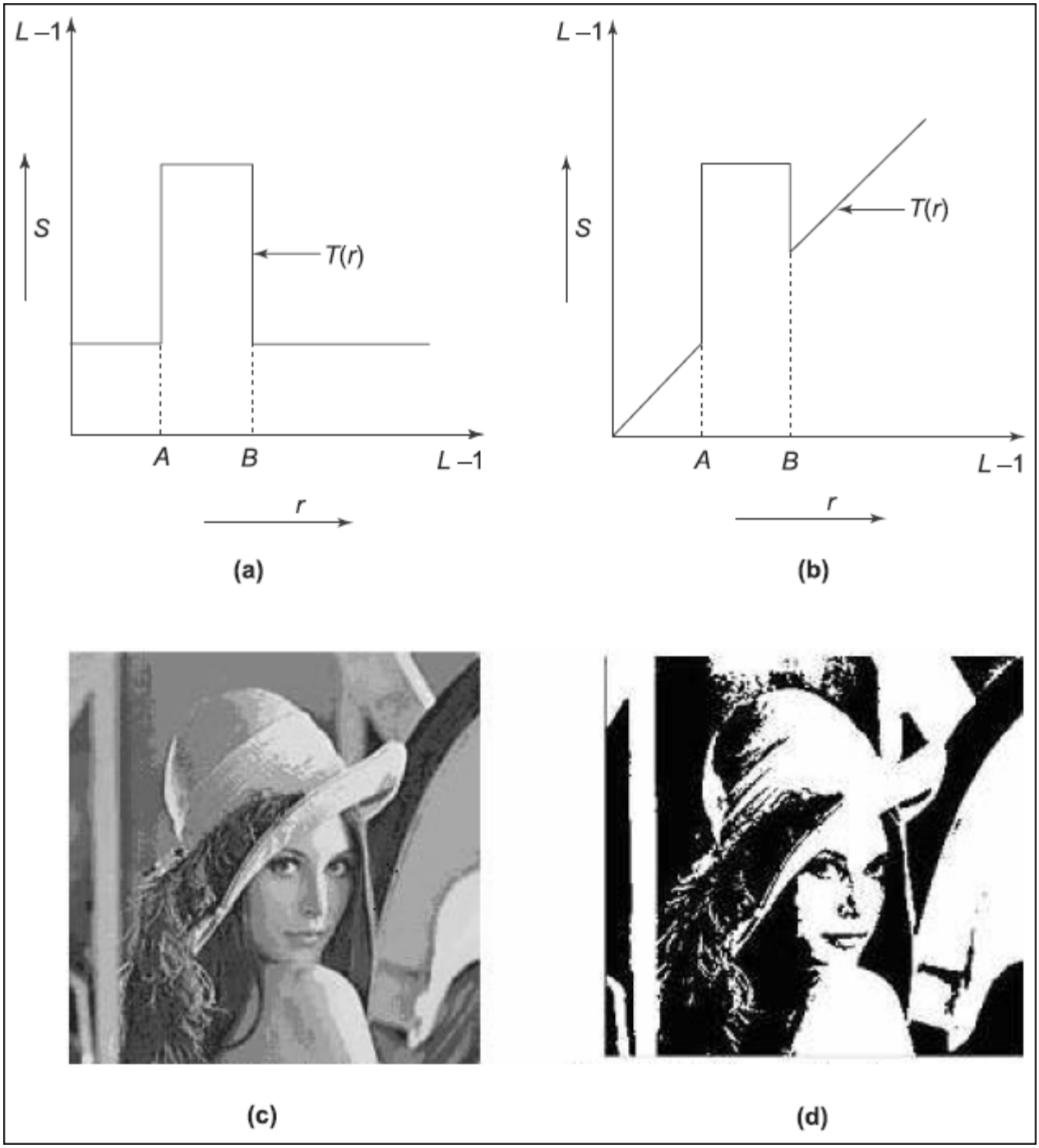


FIGURE 4.7 Transformation function. (a) For a range of pixels of interest (b) For a range of pixels of interest and preserving background gray levels (c) Original image (d) Resulting image after applying transformation function shown in Figure 4.7(a)

4.3.4 BIT PLANE SLICING

Instead of highlighting the intensity ranges sometimes it is desired to highlight the contribution made to total image appearance by considering the specific bits of the image. We know that each pixel in the image is represented by 8 bits. Imagine that the image is composed of eight 1-bit planes corresponding to the 8 bits ranging from plane 0 for the least significant bit and plane 7 for the most significant bit. Plane 0 contains all the lower order bits in the bytes comprising the pixels in the image and plane 7 contains all the higher order bits. This idea is illustrated in the Figure 4.8. Figure 4.9 shows the original image and the Figure 4.10(a)–(h) shows the various bit planes of the image shown in Figure 4.9.

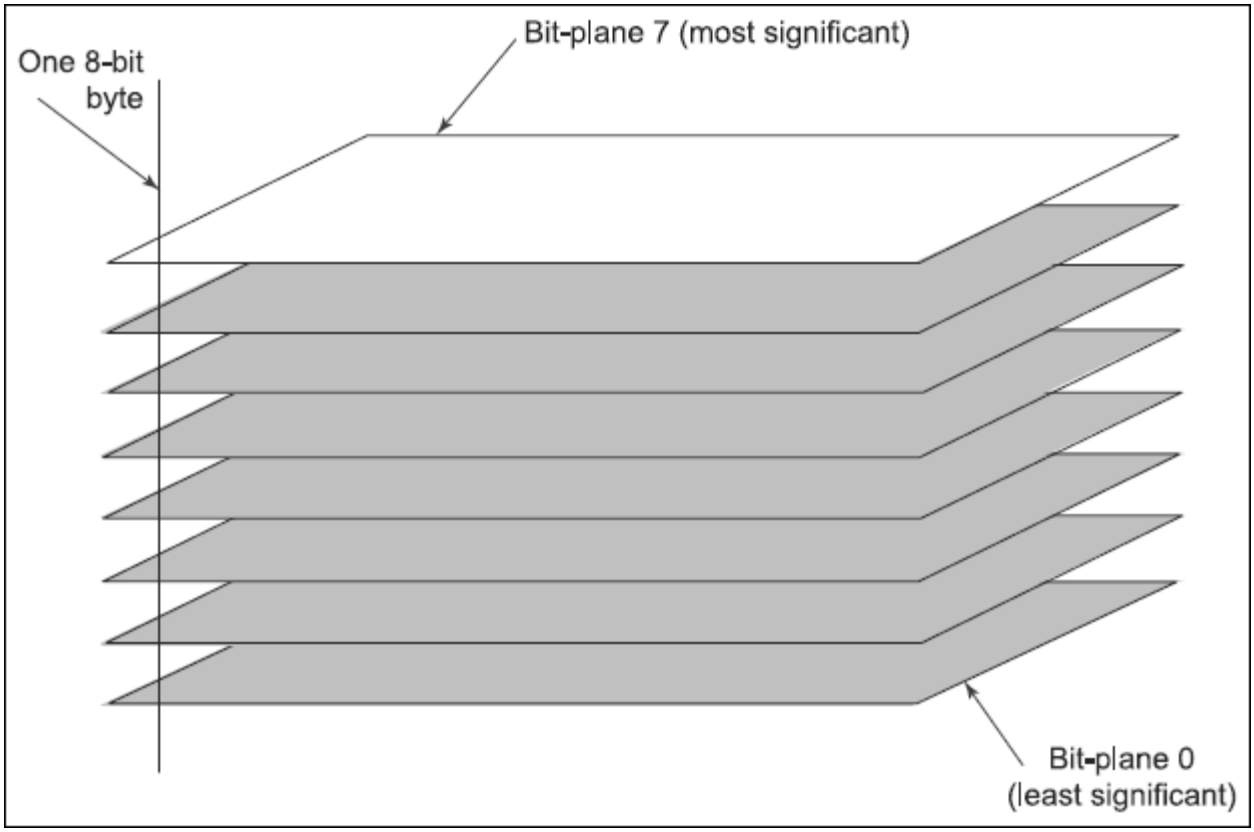


FIGURE 4.8 Bit-plane representation of an 8-bit image

FIGURE 4.9 Original image (*Source:* Signal and Image Processing Institute, University of California)

Note that only the five highest order bits contain visually significant data. The other bit planes contribute to only less details in the image. It is also worth mentioning that the plane 7 shown in Figure 4.10(a) can also be obtained by thresholding the image of the gray level 128. The other plane images shown in Figures (g), (f), (e), (d), (c), (b) and (a) are obtained by thresholding the original image at gray levels 64, 32, 16, 8, 4, 2 and 1, respectively.

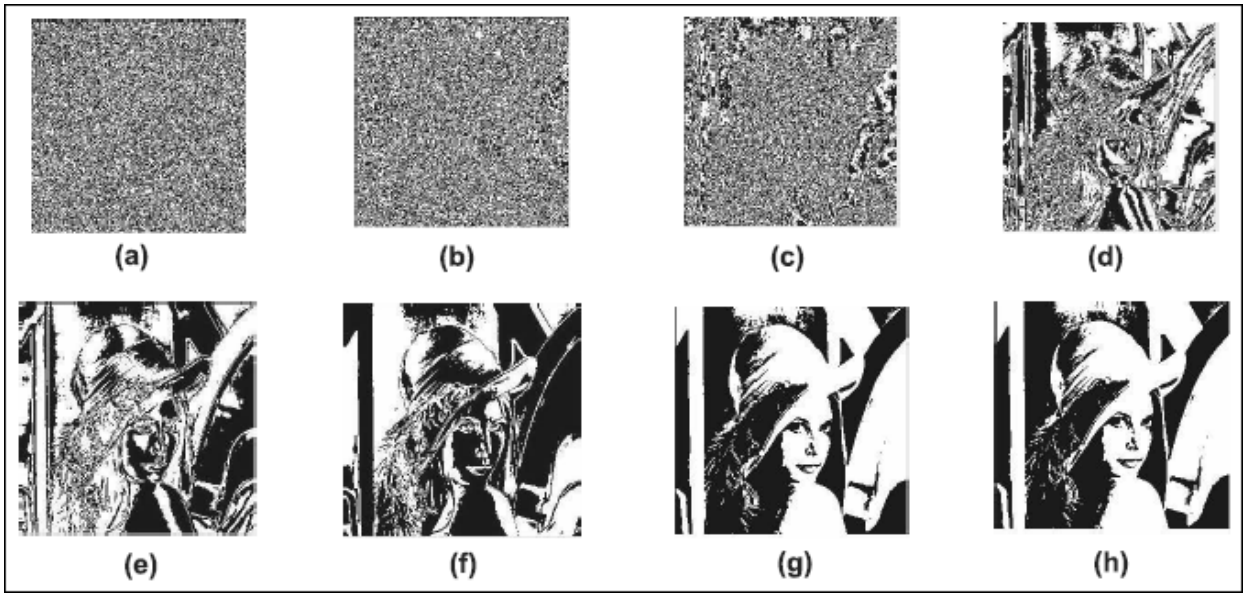


FIGURE 4.10(a–h) Bit planes for image shown in Figure 4.9

4.3.5 HISTOGRAM AND HISTOGRAM EQUALIZATION

Histogram The histogram of a digital image is the probability of occurrence associated with the gray levels in the range 0 to 255. It can be expressed using a discrete function

$$P(r_k) = \frac{n_k}{n} \qquad (4.6)$$

where r_k is the k^{th} gray level, n_k is the number of pixels in the image with that gray level, n is the total number of pixels in the image and k = 0, 1, 2,…, 255. In general, $P(r_k)$ gives an estimate of the probability of occurrence of gray level r_k. The plot of $P(r_k)$ for all values of k is called histogram of the image and it gives a global description of the appearance of an image. The histograms for four different types of images are shown in Figure 4.11.

Histogram: A plot between the probability associated with each gray level versus gray levels in the image. From this one can infer whether the given image is

- A dark image or
- Bright image or
- Low contrast image or
- High contrast image.

The histogram shown in Figure 4.11(a) shows that the gray levels are concentrated towards the dark end of the gray scale range. Thus this histogram corresponds to an image with overall dark characteristics. Figure 4.11(b) shows the histogram for a bright image. Figures 4.11(c) and (d) are the histograms for low-contrast and high-contrast images, respectively. Thus the shape of the histogram gives useful information about the possibility for contrast enhancement. The following discussion is for image enhancement based on histogram manipulation.

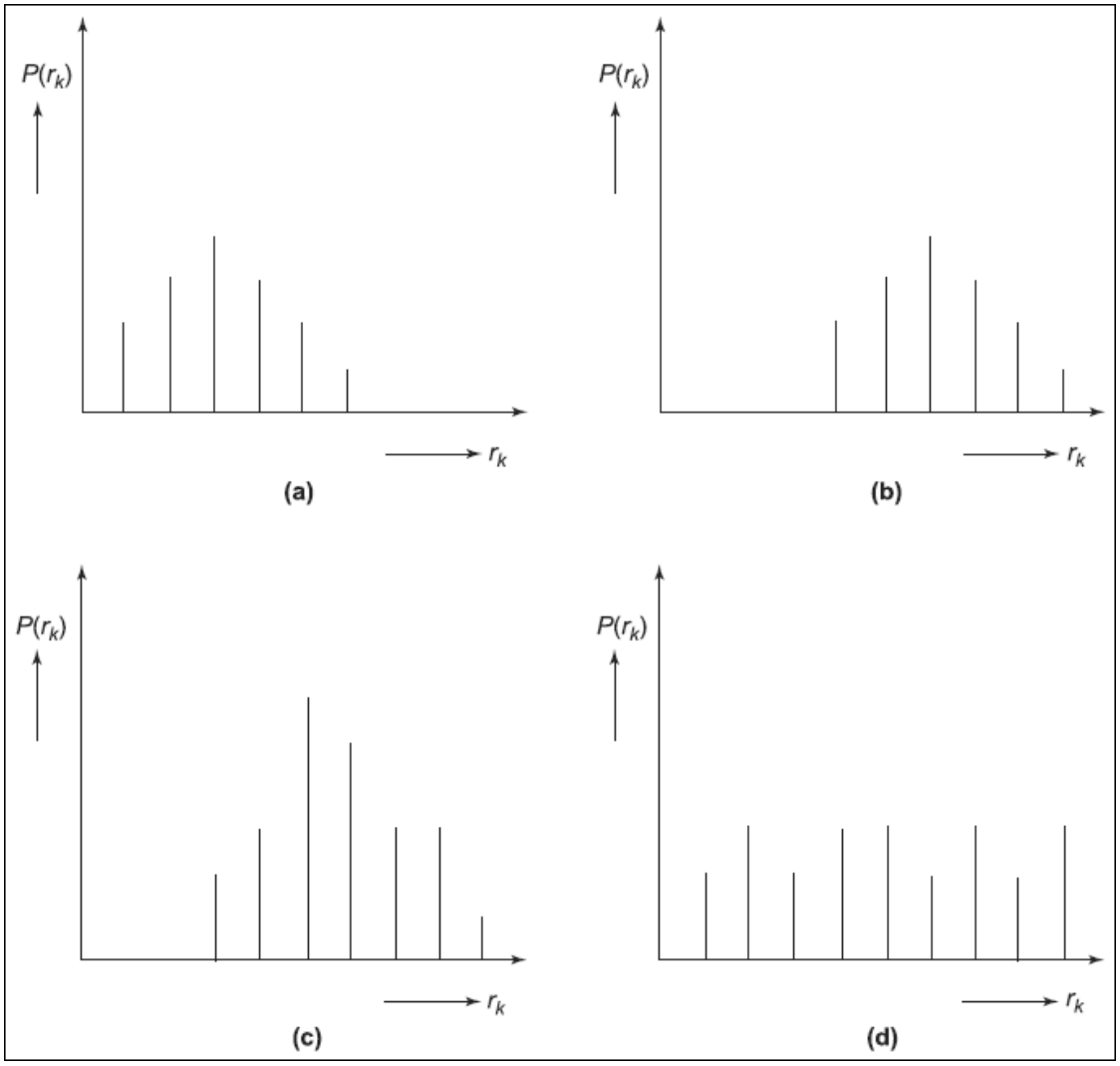


FIGURE 4.11 Histograms for four different types of images. (a) Dark image (b) Bright image (c) Low contrast image (d) High contrast image

Histogram equalization Let *r* be the variable representing the gray levels in the image to be enhanced. Assume that the gray levels in this image after normalization range from 0 to 1. For any value of *r* in the original image in the interval (0, 1) the transformation in the form

$$s = T(r) \qquad (4.7)$$

produces a gray level *s*. It is assumed that equation (4.7) satisfies the following two conditions:

1. *T(r)* is single–valued and monotonically increasing in the interval $0 \leq r \leq 1$
2. $0 \leq T(r) \leq 1$ for $0 \leq r \leq 1$.

The first condition preserves the order from black to white in the gray scale, whereas the second condition guarantees a mapping that is consistent with the allowed range of pixel values.

An example transfer function given in Figure 4.13 satisfies these conditions. The inverse transfer function from *s* back to *r* is given as

$$r = T^{-1}(s) \quad \text{for} \quad 0 \le s \le 1 \qquad (4.8)$$

Where $_T{}^{-1}$(s) also satisfies the conditions (1) and (2) with respect to the variable *s*. The gray levels in an image may be viewed as random quantities in the interval (0,1). The original and transformed gray levels can be characterized by their probability density functions $P_r(r)$ and $P_s(S)$, respectively. If $P_r(r)$ and $T(r)$ are known and $T^{-1}(s)$ satisfies condition (1), the probability density function of the transformed image gray levels is then given by

$$P_s(s) = \left[P_r(r) \frac{\mathrm{d}r}{\mathrm{d}s} \right]_{r=T^{-1}(s)} \qquad (4.9)$$

We now discuss the approach which is based on modifying the appearance of an image by controlling the probability density function of its gray levels using the transformation function $T(r)$ as shown in Figure 4.12.

Consider the transformation function,

$$S = T(r) = \int_0^r P_r(r)\mathrm{d}r \quad 0 \le r \le 1 \qquad (4.10)$$

The right side of equation (4.10) is called as cumulative distribution function (CDF) of *r*. The CDF satisfies the conditions (1) and (2) already stated. Differentiating equation (4.10) with respect to *r* results,

$$\frac{\mathrm{d}s}{\mathrm{d}r} = P_r(r) \qquad (4.11)$$

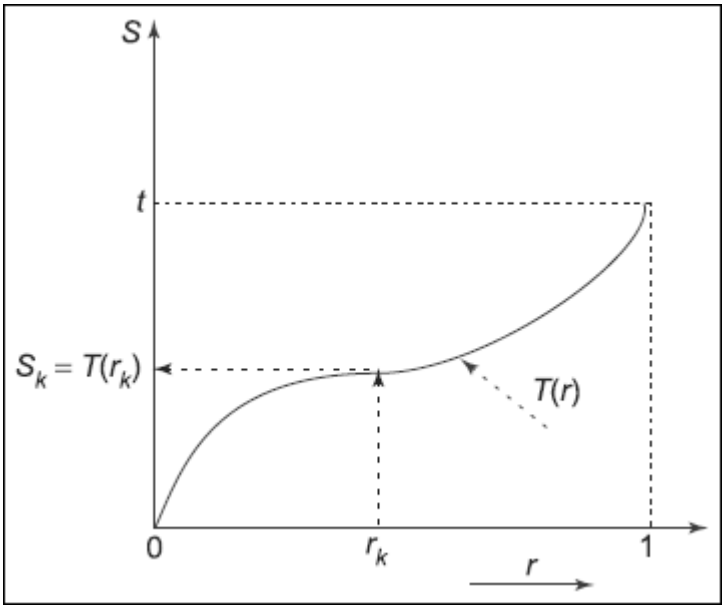


FIGURE 4.12 Gray level transformation function

Substituting d*r*/d*s* into equation (4.9) yields

$$P_s(s) = \left[P_r(r) \frac{1}{P_r(r)} \right]_{r=T^{-1}(s)} \qquad (4.12)$$

$$= [1]_{r=T^{-1}(s)}$$

$$= 1 \quad 0 \le s \le 1$$

Which is a uniform density in the interval [0, 1] for the variable *s*. From this we infer that using the CDF as transformation function results in an image whose gray levels have a uniform density. In terms of enhancement this implies an increase in the dynamic range of pixels which can have a considerable effect on the appearance of the image.

The concepts discussed earlier can be illustrated using a simple example.

Example 1

Assume that the levels *r* have the probability density function shown in Figure 4.13(a).

From Figure 4.13(a) the equation for the function can be given as

$$P_r(r) = \begin{cases} 2r & 0 \le r \le 1 \\ 0 & \text{elsewhere} \end{cases} \qquad (4.13)$$

$$s = T(r) = \int_0^r P(r) = \int_0^r 2r\,dr \qquad (4.14)$$

$$= \frac{2r^2}{2} = r^2$$

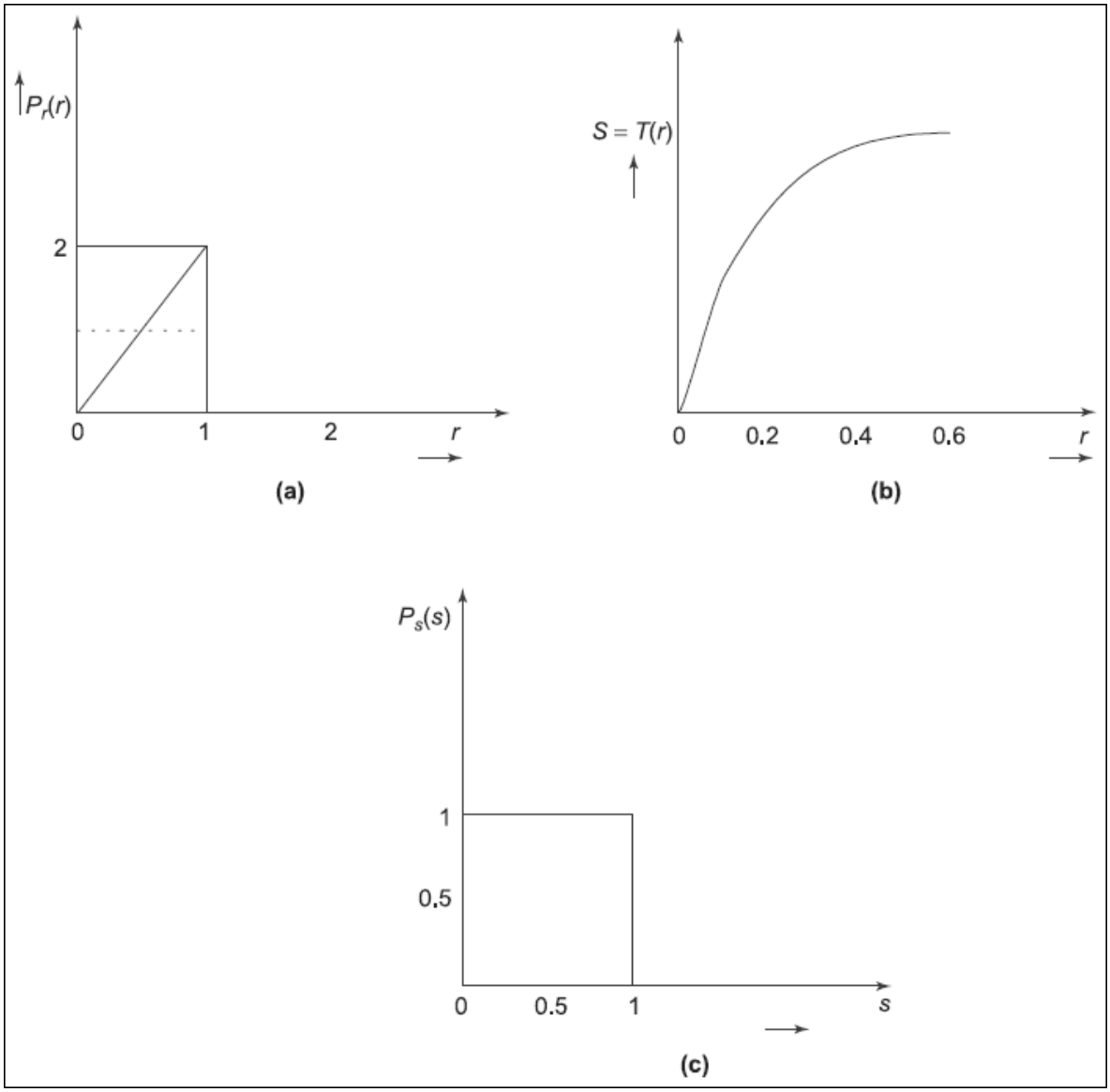


FIGURE 4.13 (a) Probability density $P_r(r)$ function of the image
(b) Transformation function (c) Resulting uniform density

From equation (4.14), $S = r^2$, differentiating the above equation

$$\frac{ds}{dr} = 2r \quad \text{(or)} \quad \frac{dr}{ds} = \frac{1}{2r}$$

then,

$$P_s(s) = P_r(r)\left[\frac{dr}{ds}\right]_{r=T^{-1}(s)} = 2r * \left[\frac{1}{2r}\right] = 1 \quad 0 \le s \le 1$$

which is a uniform density function in the desired range. Figure 4.13(b) shows the transformation function $T(r) = r^2$ and Figure 4.13(c) shows $P_s(s) = 1$.

The concepts discussed so far must be formulated in discrete form so that it will be useful for digital image processing. For the gray levels in the discrete values and the probabilities associated with them can be given as

$$P_r(r_k) = \frac{n_k}{n} \qquad (4.15)$$

where $0 \leq r_k \leq 1$ and $k = 0,1,2,3,\ldots, L-1$.

L is the number of levels, $P_r(r_k)$ is the probability of the kth gray level, n_k is the number of times this level appears in the image and n is the total number of pixels.

The plot of $P_r(r_k)$ versus r_k is called a histogram, and the technique used for obtaining the uniform histogram is known as *histogram equalization* or *histogram linearization.*

Then the discrete form of transformation is given by

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_i}{n} \sum_{j=0}^{k} P_r(r_j) \quad \text{for} \quad 0 \leq n_k \leq 1 \qquad (4.16)$$

The inverse transformation is denoted as

$$r_k = T^{-1}(s_k) \quad \text{for} \quad 0 \leq s_k \leq 1$$

where both $T(r_k)$ and $T^{-1}(s_k)$ are assumed to satisfy the conditions (1) and (2), mentioned previously.

The histogram equivalization technique is applied to the original Lena image shown in Figure 4.14(a)(The colored figure is available here). The Figure 4.14(b) shows the histogram of the Lena image. Figure 4.14(c) is the enhanced Lena image obtained using histogram equivalization technique. The histogram of the enhanced Lena image is given in Figure 4.14(d).

4.3.6 Histogram Specifications

For interactive image enhancement applications, the histogram equalization method is not suitable. The reason is that the histogram equalization method is capable of generating only one result, that is, approximation to uniform histogram.

In practical applications, it is desirable to specify a particular histogram shape capable of highlighting certain gray level ranges in an image.

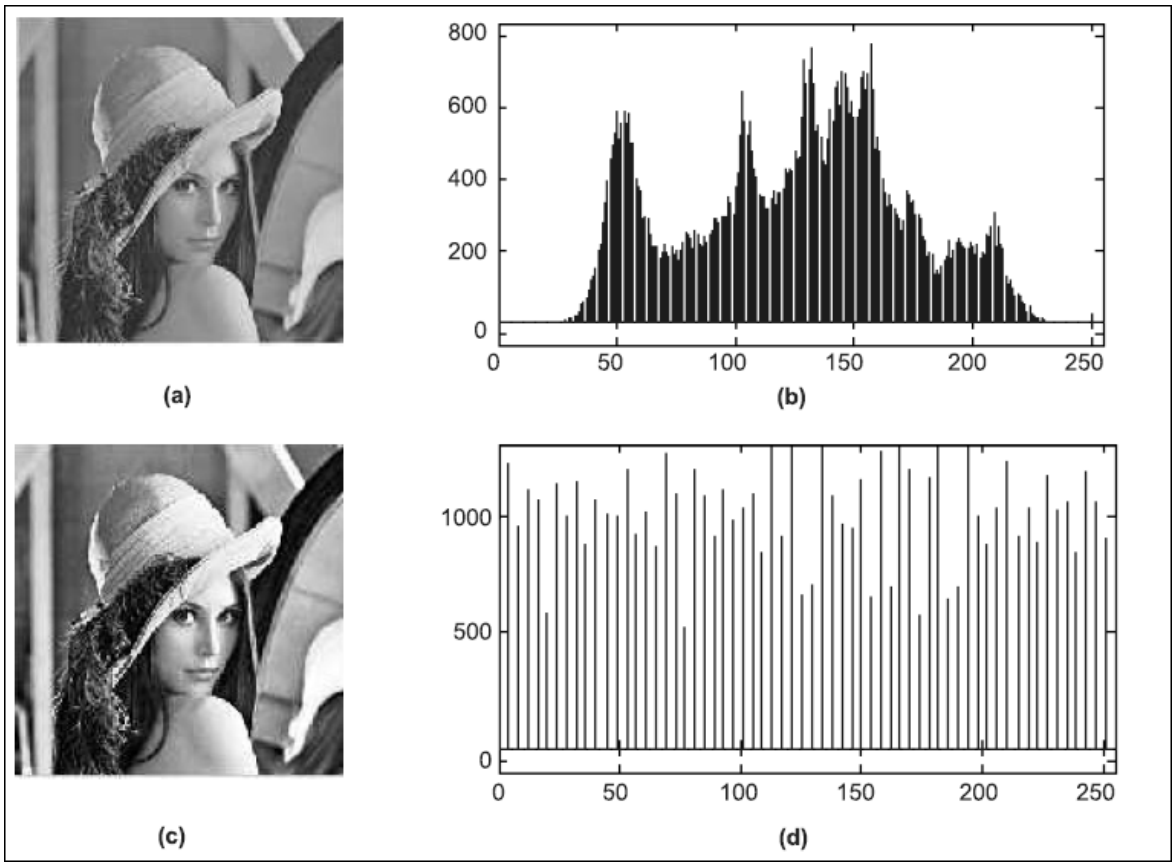


FIGURE 4.14 (a) Original Lena image (b) Histogram of original image (c) Histogram equalized Lena image (d) Histogram of equalized Lena image

To illustrate this concept let us consider $P_r(r)$ and $P_z(z)$ as the original and desired probability density function, respectively.

Suppose, the histogram equalization is applied to the original image, that is,

$$s = T(r) = \int_0^r P_r(r)\mathrm{d}r \qquad (4.17)$$

if the desired image was also available, its gray levels would also be equalized by using equation (4.18).

$$V = G(z) = \int_0^z P_z(z)\mathrm{d}z \qquad (4.18)$$

Then the inverse process $z = G^{-1}(V)$ gives back the gray levels z, of the desired image. This formulation is hypothetical, because the gray levels 'z' are precisely what is being required. However, $P_s(s)$ and $P_v(v)$ would be identical uniform densities because of the final result of the transformation.

Hence we can write

$$s = T(r) = \int_0^r P_r(r)\mathrm{d}r \qquad (4.19a)$$

Which is independent of the density inside the integral. Thus instead of using V in the inverse process one can use the uniform levels s obtained from the original image, resulting in the levels $z = G^{-1}(s)$ and this will have the desired probability density function. This procedure can be summarized as follows.

1. Equalize the original image using the equation $s = T(r) = \int_0^r P_r(r)\,\mathrm{d}r$.
2. Specify the desired density function and obtain the transformation function

$$G(Z) = v = \int_0^z P_z(z)\mathrm{d}z \qquad (4.19b)$$

3. Apply the inverse transformation function $Z = G^{-1}(s)$ to the level obtained in step 1 (since $G(z) = v = s = T(r)$).

This procedure gives a modified version of the original image with the new gray levels characterized by the desired density $P_z(z)$.

The histogram specification technique just described involves two transformation functions $T(r)$ followed by $G^{-1}(s)$. These two steps can be combined into a single step, so that the desired gray levels starting with the original pixels can be obtained. We know that, $Z = G^{-1}(s)$ and substituting equation $s = T(r) = \int_0^r P_r(r)\mathrm{d}r$ in equation 4.19(b), results in the combined transformation function as

$$Z = G^{-1}(s) = G^{-1}[T(r)] \qquad (4.20)$$

Where r relates to z.

The implication of equation (4.20) is simply that, an image need not be histogram equalized explicitly. All that is required is that $T(r)$ be determined and combined with the inverse transformation function G^{-1}.

4.3.7 Local Enhancement Technique

The two approaches discussed earlier are global techniques because all the pixels present in an image are modified using a transformation function. These global approaches are not

suitable to enhance the details over small areas. The reason for this is that the pixels in these small areas have negligible influence on the computation of a global transformation. So it is necessary to develop an approach that will produce the desired local enhancement. The solution is to use transformation functions, which are based on gray level distribution or other properties in the neighborhood of every pixel in the image.

In the local enhancement technique the square or rectangular neighborhood is considered and we move the center of this area from pixel to pixel. At each location the histogram of the points in the neighborhood is computed and either a histogram equalization or histogram specification transformation function is obtained. This function is finally used to map the gray level of the pixel centered in the image. The center of the neighborhood is then moved to the adjacent pixel location and the procedure is repeated. This procedure is called *local enhancement technique.*

Figure 4.15(a) shows an image consisting of five dark squares. Figure 4.15(b) shows the result of global histogram equalization. From this we understand that no new details or structures are resulted. Figure 4.15(c) is the result of local processing using 5×5 neighborhood pixels and it revealed the presence of small squares inside the large darker squares. The small squares are too close in the gray levels and the influence of global histogram equalization is negligible.

The local enhancement can also be achieved using the properties of the pixel, such as intensities in the neighborhood instead of using histograms (Figure 4.15). We know that the mean denotes the average brightness and the variance denotes the contrast. So the intensity mean and variance are two properties that describe the appearance of the image.

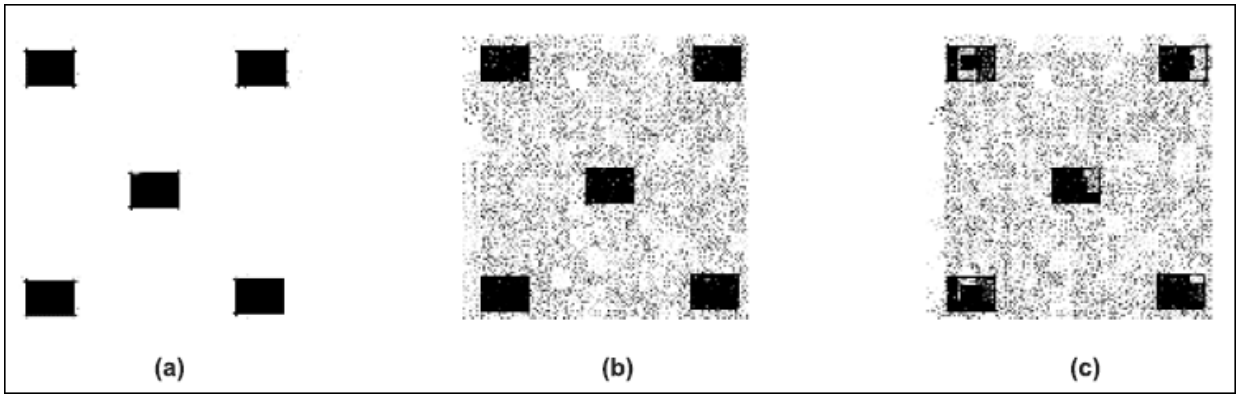


FIGURE 4.15 Local enhancement. (a) Original Image (b) Result of global histogram equalization (c) Image after local enhancement

The typical local transformations that uses these two properties to transform the input image $f(x, y)$ into the new image $g(x, y)$ at each pixel location (x, y) is explained later.

The equation for such a transformation can be given as

$$g(x, y) = A(x, y)\,[f(x, y) - m(x, y)] + m(x, y) \qquad (4.21)$$

where

$$A(x, y) = k\left(\frac{M}{\sigma(x, y)}\right) \quad \text{and} \quad 0 < k < 1 \qquad (4.22)$$

In the equations (4.21) and (4.22) $m(x, y)$ and $\sigma(x, y)$ are the gray level mean and standard deviation calculated in a neighborhood region centered at (x, y), M is the global mean of $f(x, y)$ and k is a constant. The values of the variables A, m, and σ are dependent on a predefined neighborhood of (x, y). From equation (4.22), $A(x, y)$ is inversely proportional to the standard deviation of density and hence it offers high gain to low contrast regions and vice-versa.

4.3.8 IMAGE SUBTRACTION

Image subtraction plays a vital role in medical applications. One of the important applications is in mask mode radiography. The primary use of image subtraction includes background removal and illumination equalization. The image difference between two images $f(x, y)$ and $g(x, y)$ can be expressed as

$$Z(x, y) = f(x, y) - g(x, y) \qquad (4.23)$$

The image subtraction is used to enhance the medical images. It can be used to detect the malfunctioning of the human organ and blocking in the blood-carrying arteries. In order to detect blockage in the arteries, usually iodine dye is injected into the blood stream. Using the camera the image of the blood stream is taken before and after injecting the dye. Figure 4.16(a) shows the image before injection of the dye. Figure 4.16(b) shows the image after injecting the dye and Figure 4.16(c) is the result of subtracting (b) from (a). This image is an enhanced version in which the arterial path is quite bright compared to the other two images. By analyzing the image obtained by subtraction, the doctor is in a position to decide the actual location of blood blockage.

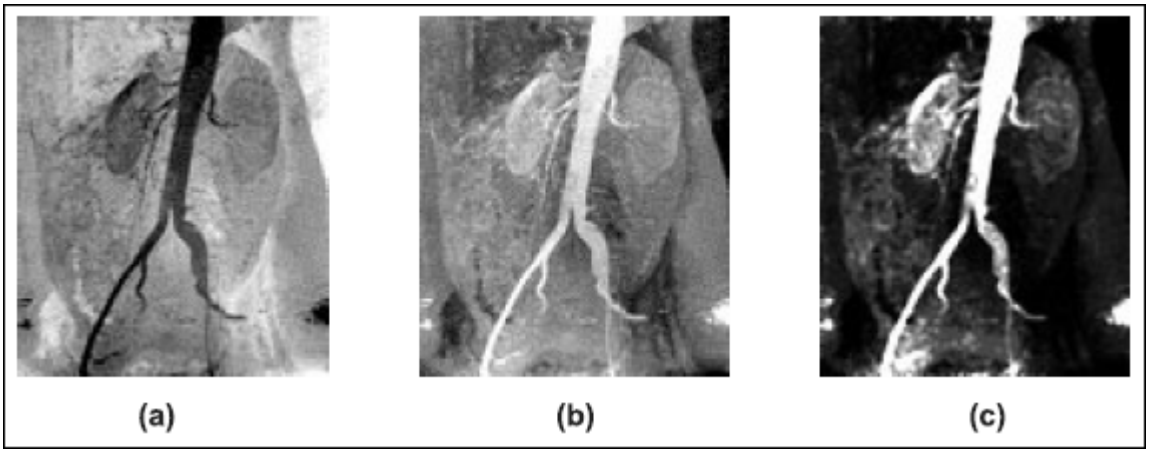


FIGURE 4.16 Enhancement by image subtraction. (a) The mask image of a major blood vessel (b) Image after injecting dye in the blood stream (c) The result of subtraction (a) from (b)

4.3.9 Image Average

Consider a noise image $Z(x, y)$ obtained by adding the noise term $\eta(x, y)$ to the original image $f(x, y)$ and it is given as

$$Z(x, y) = f(x, y) + \eta(x, y) \qquad (4.24)$$

The noise term $\eta(x, y)$ is considered as a random phenomenon and it is uncorrelated; hence the average value of the noise results in a zero value. If our intension is to reduce the noise from the noisy image $Z(x, y)$, one can apply the image averaging technique. If the noise satisfies zero average value then the problem of reducing or eliminating the noise from an image is a simple matter. Let us assume that there are 'm' number of noisy images available and it is denoted as $Z_1(x, y)$, $Z_2(x, y)$,…, $Z_m(x, y)$. Then the average of these images is represented as

$$\overline{Z}(x, y) = \frac{1}{m} \sum_{i=1}^{m} Z_i(x, y) \qquad (4.25)$$

It can then be proved that

$$\overline{Z}(x, y) = f(x, y) \qquad (4.26)$$

As per Papoulis theory, it can be proved that

$$E\left\{\overline{Z}(x, y)\right\} = f(x, y) \qquad (4.27)$$

and

$$\sigma^2_{\overline{Z}(x,\, y)} = \left(\frac{1}{m}\right) \sigma^2_{\eta(x,\, y)} \qquad (4.28)$$

where $E\{\overline{Z}(x,y)\}$ is the expected value of $\overline{Z}$, $\sigma^2_{\overline{Z}(x,\, y)}$, and $\sigma^2_{\eta(x,y)}$ are the variances of Z and η, for all coordinates (x, y). The standard deviation at any point in the average image is

$$\sigma_{\overline{Z}(x,\, y)} = \left(1/\sqrt{m}\right) \sigma_{\eta(x,\, y)} \qquad (4.29)$$

From this equation, we infer that as 'm' increases, the variance of the pixel values at each location decreases. This means that $Z(x, y)$ approaches to $f(x, y)$ as the number of noisy images used in the averaging process increases. Figure 4.17(a) shows the original printed circuit board image and (b) is its noisy version. Figures 4.17(c)−(g) shows the result of averaging 2, 4, 8, 16 and 32 such noisy images. The image obtained by averaging 32 noisy image is free from noise and suitable for all practical purposes.

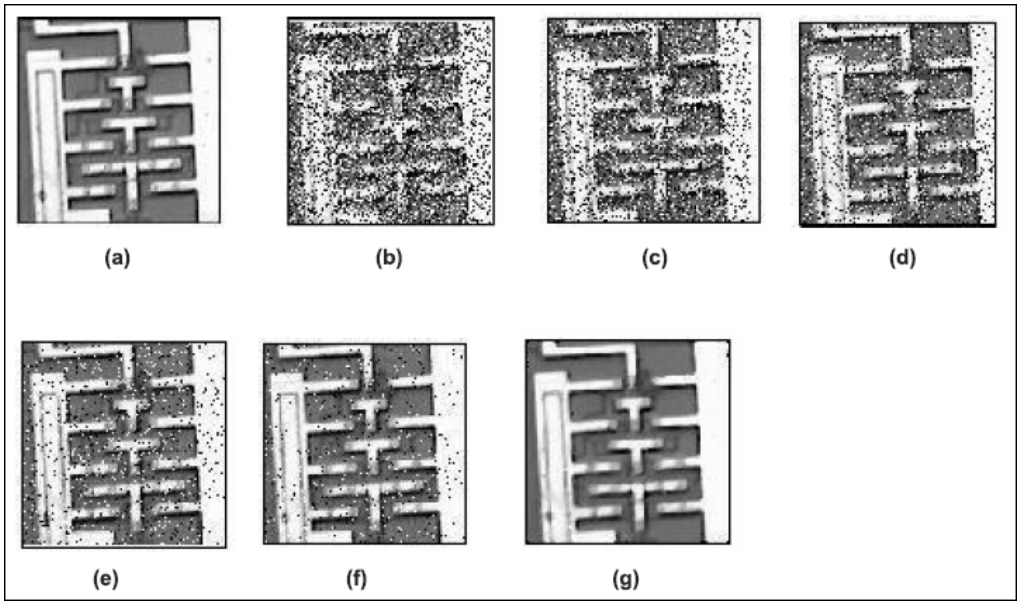


FIGURE 4.17 Noise reduction by averaging technique. (a) The original printed circuit board image (b) The noisy image of (a), (c)–(g) the results of averaging 2, 4, 8, 16 and 32 noisy images

4.4 SPATIAL FILTERING

The spatial filtering concept has been introduced using spatial masks. The spatial filtering approaches are useful in image processing. Sometimes the masks used for implementing the filters are called as spatial filters. The widely used filter categories are

1. Low-pass filters and
2. High-pass filters.

Any image can be viewed as consisting of gray level details corresponding to low frequencies and high frequencies. For example, high-frequency components correspond to edges and other sharp details in an image. So when we employ a high-pass filter to process an image, the details corresponding to edges and other sharp details are highlighted and the low-frequency details are attenuated. Hence the high-pass filter can be used to obtain the boundary of the objects and other sharp details available in an image.

Similarly, the low-frequency details correspond to slowly varying components of an image. So when we employ a low-pass filter to process an image it allows only slowly varying image details and attenuate heavily the details corresponding to edges and sharp transitions and results in a blurred image.

The frequency responses of low-pass filter and high-pass filters are shown in Figure 4.18(a) and (b) and the corresponding spatial domain responses are shown in Figure 4.18(c) and (d). The responses shown in Figure 4.18(c) and (d) provides an idea for specifying the linear spatial filter mask.

In general, in all the spatial domain filters, a mask with its coefficient are used to find the sum of the products between the mask coefficients and the intensity of the image pixels under the mask at specific location in the image. A mask of size 3 × 3 is shown in Figure 4.19, where C_1, C_2,…, C_9 are the coefficient of mask.

Assume that the gray levels of the pixel under the mask are L_1, L_2,…, L_9, then the average response of the linear mask is given by the equation,

$$R_a = (1/9) * [C_1 * L_1 + C_2 * L_2 + C_3 * L_3 + \cdots + C_9 * L_9]. \qquad (4.30)$$

Then the center pixel value at the location (x, y) is replaced by the response R_a obtained above. The mask is then moved to the next position and the response is obtained using the equation (4.30) and the pixel value in the image at the center of the mask is replaced by the response. This procedure is repeated for the remaining pixel in the image.

4.4.1 Low-pass Spatial Filters

The low-pass spatial filters are used to reduce the noise such as bridging of small gaps in the lines or curves in a given image. So the low-pass filter is also called as *smoothing filters.*

Low-pass Spatial Filter: This smoothes the data and makes the image appear less granular, thus suppressing image noise.

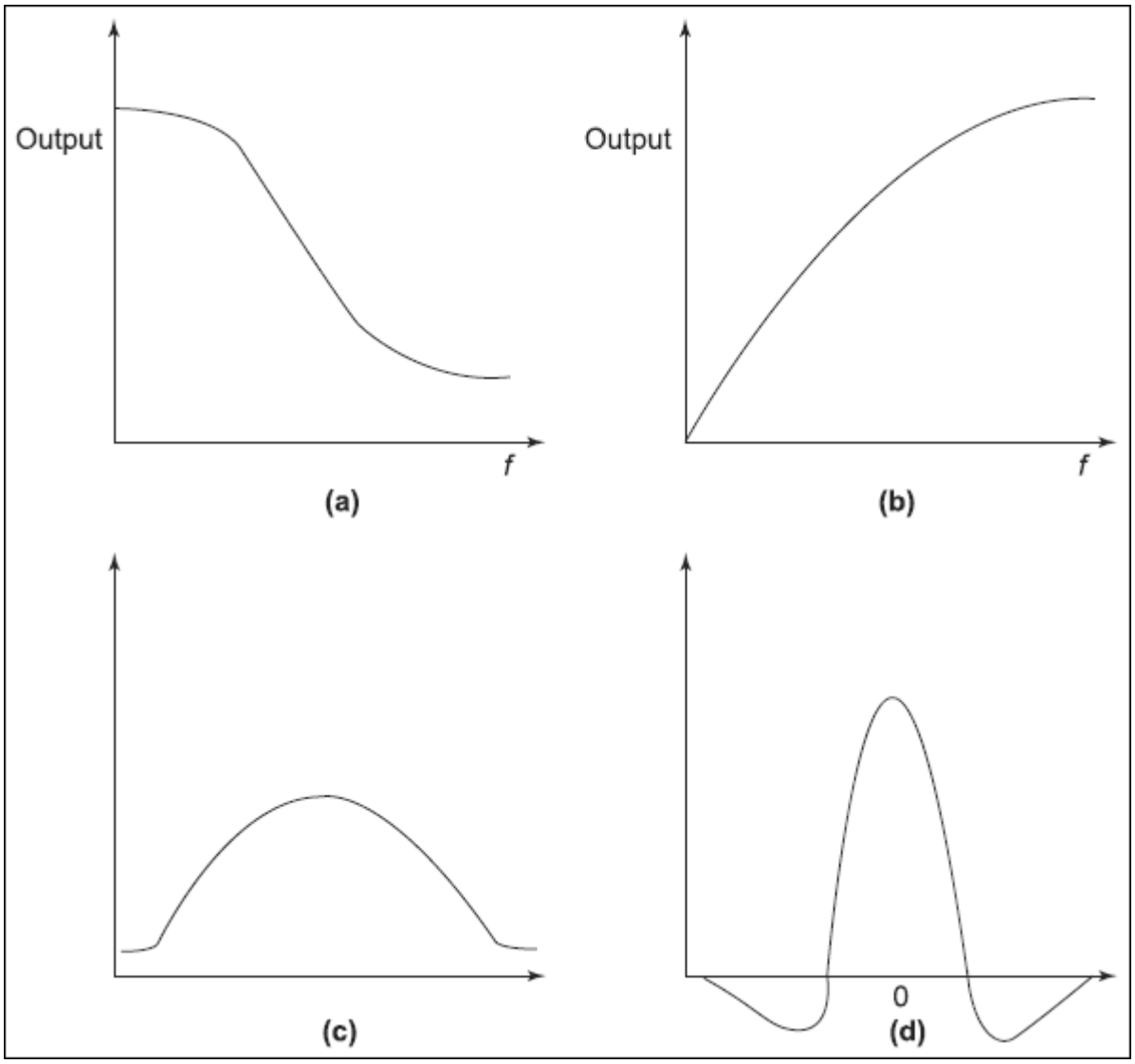


FIGURE 4.18 (a) Low-pass filter and (b) High-pass filter (c) Cross-section of low pass filter (a) (d) Cross-section of high pass filter (b)

C1	C2	C3
C4	C5	C6
C7	C8	C9

FIGURE 4.19 A sample mask of coefficients

The shape of the spatial low-pass filter is shown in Figure 4.18(c). From this curve we infer that all the coefficients of the mask corresponding to the spatial low-pass filter must have all positive values. For a 3 × 3 spatial filter the easiest arrangement is to have a mask in which all the coefficients have a value of 1. The 3 × 3 mask with coefficients 1 is shown in Figure 4.20. The mask can also be larger than 3 × 3.

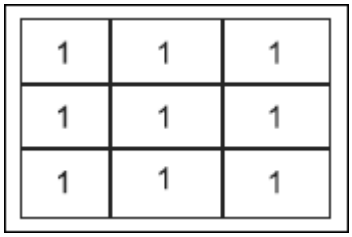


FIGURE 4.20 Low-pass filter

As the size of the mask increases the smoothing effect (blurring effect) also increases.

Figure 4.21(a) shows the original image (The colored figure is available here). Figure 4.21(b) and (c) is the result after applying the spatial low-pass filter of size 3 × 3 and 7 × 7, respectively.

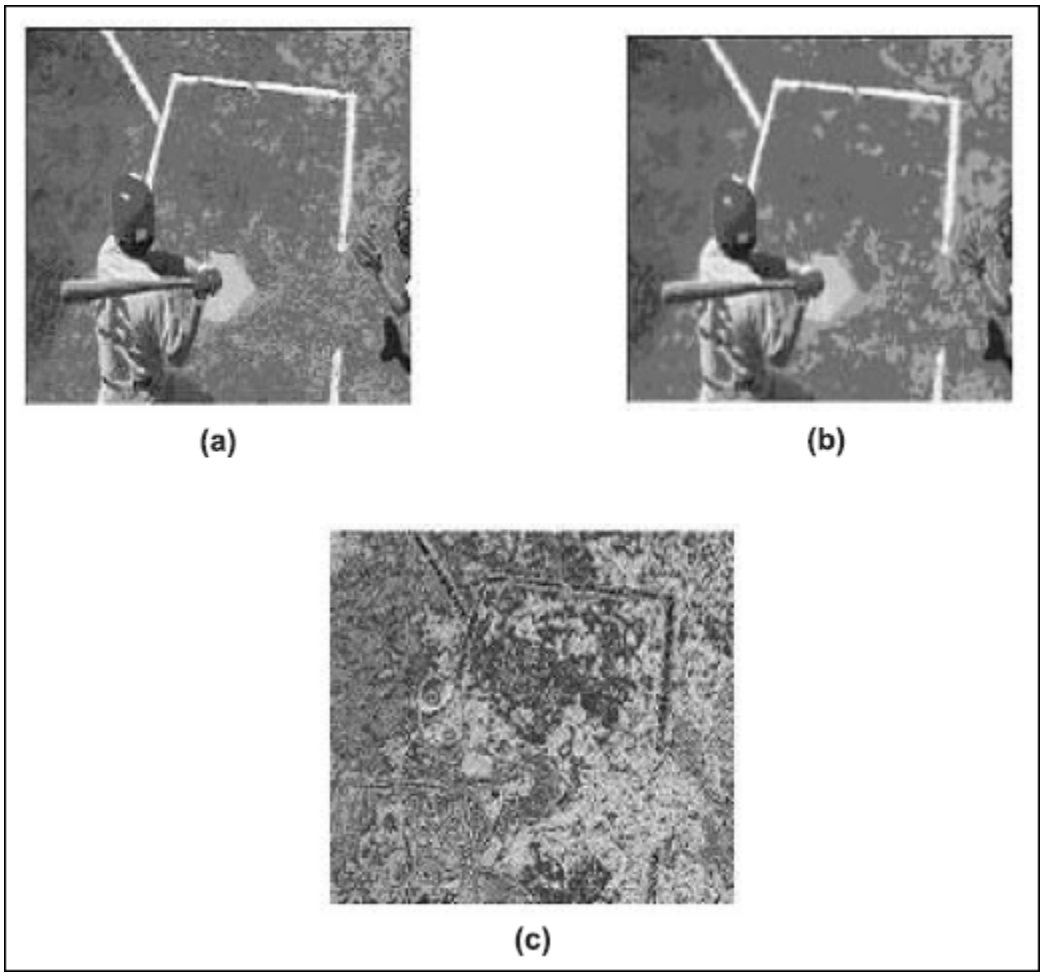


FIGURE 4.21 (a) Original image (b) Result of spatial HFF (c) Result of spatial LPF of size 3 × 3

4.4.2 MEDIAN FILTERING

The low-pass spatial filter smooths out the edges and sharp details. The low-pass filter is not suitable for reducing the noise patterns consisting of strong spike-like components. For such applications the median filter is best suited.

In order to perform the median filtering the mask of size say 3 × 3 can be considered [Figure 4.22(b)]. The coefficients of this mask are all equal to 1. Place the mask in the top left corner and read the pixel values below this mask. Arrange these pixel values in the ascending order. For example, the values of the pixels below the mask 3 × 3 are 30, 15, 3, 7, 40, 28, 18, 65, 4 as shown in Figure 4.22(a). Then the median is found from the sorted values 3, 4, 7, 15, 18, 28, 30, 40, 65. For this example, the median value is '18', the middle value of the sorted pixel values. So the center pixel of the mask is replaced by median thus computed. The procedure is repeated for moving the mask one position after another until the last pixel in the image is processed. The Figure 4.23(a) is an image and to this noise is added and the resulting image is shown in Figure 4.23(b). The noisy image shown in Figure 4.23(b) is given as input to both low-pass and median filters and their output responses are shown in figures (c) and (d), respectively. From the figure it is understood that the median filter performs better than low-pass filter under noisy environment.

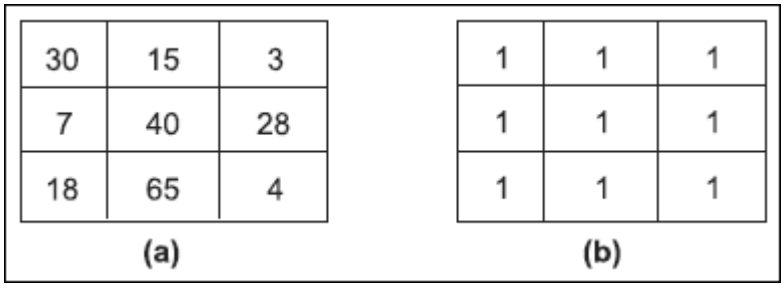


FIGURE 4.22 (a) Pixel values under mask of 3 × 3 (b) Centre value '40' In Figure 4.23(a) is replaced by median value '18'

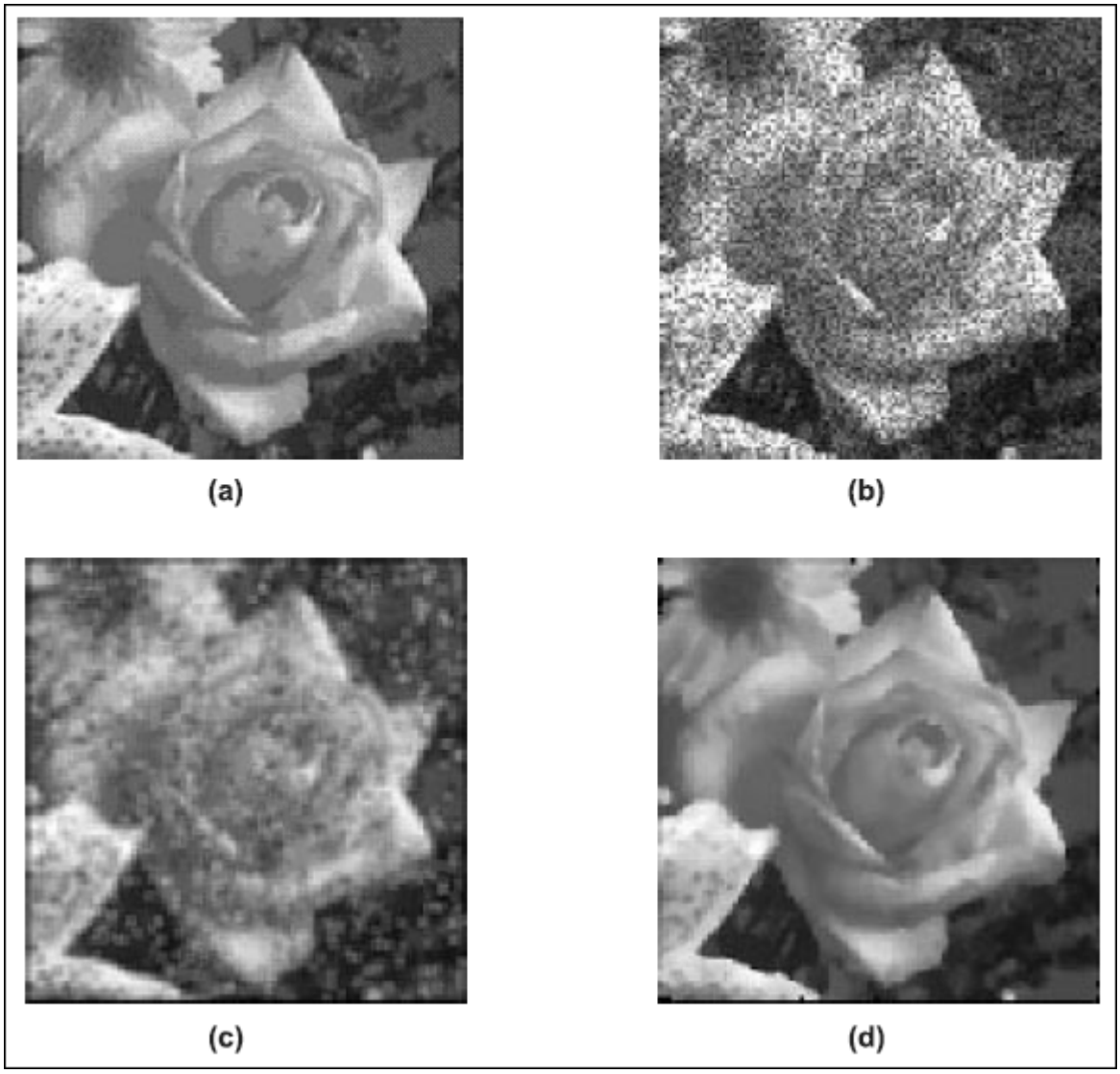


FIGURE 4.23 (a) The original rose image (b) The noisy image (c) The image after applying LPF (d) Image after applying median filtering

4.4.3 High-pass Spatial Filters

The high-pass filters attenuates low-frequency components heavily and pass the high-frequency components. This results in an mage with sharp details such as edges and high contrast. Additionally, high-pass filters can provide more visible details that are obscured, hazy, and of poor focus in the original image.

High-pass Spatial Filter: These attenuates low-frequency components and pass the high-frequency components resulting in an image with details on edges and high contrast.

From the spatial high-pass filter response shown in Figure 4.19, we have to construct a mask such that the center of the mask has a positive value and all its neighbor coefficients are of negative value. Such a mask is shown in Figure 4.24.

–1	–1	–1
–1	8	–1
–1	–1	–1

FIGURE 4.24 A mask for high-pass filter

When we operate this mask to an image starting from top left corner and slide one portion to the right till it reaches the last position in an image, it results in an image which emphasizes sharp details (Figure 4.25). The logic behind this high-pass filter is explained as follows.

Let the mask be at a location where the pixels beneath have equal values corresponding to the background of an image. Let us assume that all the pixels have the gray level values say 10. Then the response of the mask is given by

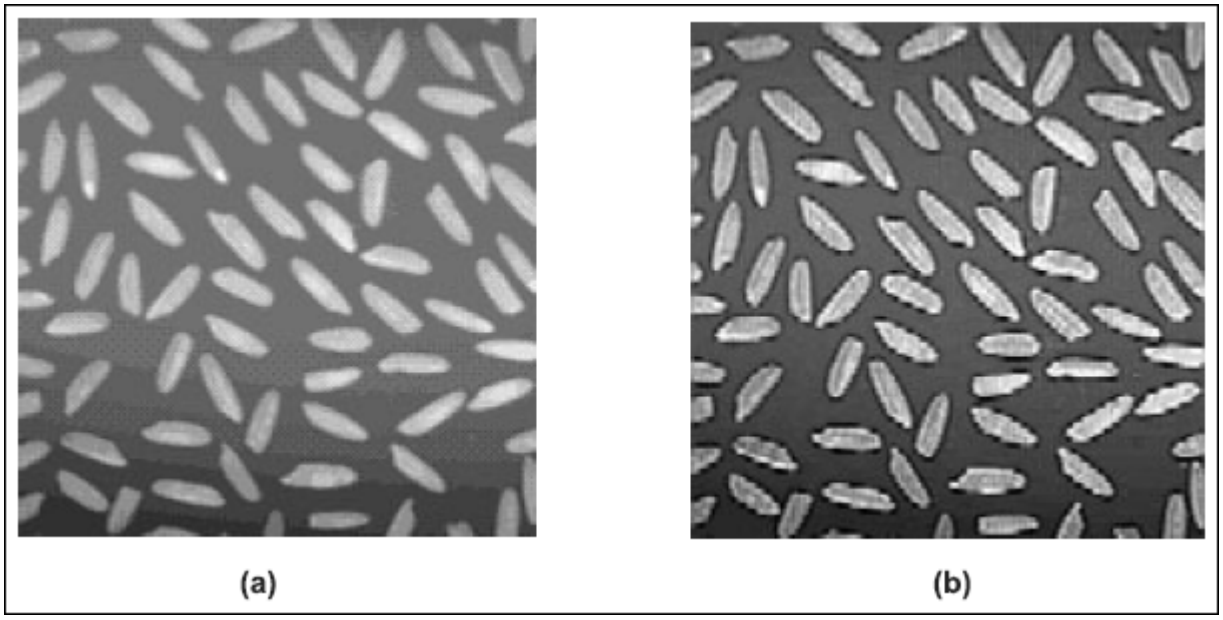


FIGURE 4.25 (a) The original image (b) The high-pass filtered image

$$R_a = \frac{1}{9}[(-1*10) + (-1*10) + (-1*10) + (-1*10) + (8*10)$$

$$+ (-1*10) + (-1*10) + (-1*10) + (-1*10)]$$

$$= 0$$

So the center pixel is replaced by 0. This means that the background pixels correspond to low frequencies and they are attenuated. On the other hand, when we place the center of the mask corresponding to pixel whose gray level value is 150 and all its neighbors have gray level 10, then the corresponding response of the mask is given as

$$R_a = \frac{1}{9}[(-1*10) + (-1*10) + (-1*10) + (-1*10) + (80*150)$$

$$+ (-1*10) + (-1*10) + (-1*10) + (-1*10)]$$

$$= (1200 - 80)/9 = 124$$

So the center pixel is replaced by 124. This means that the pixels which correspond to sharp details are passed without much attenuation. Hence it is clear that the mask we have constructed emphasizes the details corresponding to edges and sharp details.

4.4.4 High-boost Filter

The high-pass filter in general results in a background which is darker than the original image. To overcome this difficulty, a high-boost filter can be employed which restores the original background details and at the same time enhances the sharpness of the image. The high-boost filter effect can be illustrated as follows.

High-boost Filter: Restores the original background details and enhances the sharpness of the image.

We know that high pass filter response = original − low pass. Then the high-boost filter can be given as

$$\text{High-boost} = A[\text{original}] - \text{low-pass filter} \tag{4.31}$$

$$= [A-1]\text{original} + \text{original} - \text{low-pass filter}$$

$$= [A-1]\text{original} + \text{high-pass filter}$$

The mask for high boost filter is shown in Figure 4.26.

As '*A*' increases, the background details of the high-boost filtered image becomes brighter and brighter (Figure 4.27). When we increase the value beyond 1.2, the resulting image becomes unacceptable. Hence we should be careful in choosing the value of '*A*' so that an acceptable background is obtained.

–1	–1	–1
–1	9-A	–1
–1	–1	–1

FIGURE 4.26 Mask for high-boost filtering

FIGURE 4.27 High-boost filter. (a) Original Lena image (b) Image after applying high-boost filter with $A = 1.1$

4.4.5 Derivative Filters

The low-pass filter approach used for image enhancement can be realized by using a mask whose coefficients are equal to 1. This means when we operate this mask over an image the resulting response corresponds to the average value in that location. So the averaging increases the smoothness of the image. We can also view averaging as corresponding to integration. Similarly, the high-pass filter is viewed as opposite to low-pass filter, that is, equivalent to doing differentiation. So we can say integration results in smoothing or blurring and differentiation results in sharpening of an image. Hence, the high-pass filter can be realized by using the differentiation concept. The effect of differentiation can be implemented using the gradient operator. The gradient operation can be illustrated as in equation (4.32). Consider an image $f(x, y)$. The gradient of the image at the coordinates (x, y) is given by the vector $\overrightarrow{\Delta f}$

$$\overrightarrow{\Delta f} = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix} \qquad (4.32)$$

Then, the magnitude of the vector is given as

$$\Delta f = \left|\overrightarrow{\nabla f}\right| = \text{mag}(\overrightarrow{\Delta f}) \left[\left[\frac{\delta f}{\delta x}\right]^2 + \left[\frac{\delta f}{\delta y}\right]^2 \right]^{\frac{1}{2}} \qquad (4.33)$$

Equation (4.33) is the basis of image differentiation. The differentiation can be effected in many ways. Roberts proposed a technique and it is illustrated as follows.

Consider a part of an image of size 3 × 3 the gray levels of which are denoted as $r_1, r_2, \ldots, r_9$ as is shown in Figure 4.28.

r_1	r_2	r_3
r_4	r_5	r_6
r_7	r_8	r_9

FIGURE 4.28 Subimage of size 3 × 3

Equation (4.33) can be approximated at the point r_5 and it is given as

$$\nabla f = \left[(r_5 - r_8)^2 + (r_5 - r_6)^2\right]^{\frac{1}{2}} \qquad (4.34)$$

where $r_5 - r_8$ is the difference in the x-direction and $(r_5 - r_6)$ is the difference in the y-direction.

Equation (4.34) can also be simplified further and given as

$$\nabla f = |r_5 - r_8| + |r_5 - r_6| \qquad (4.34a)$$

According to Roberts technique the other definition is given in equation (4.35).

$$\Delta f = |r_5 - r_9| + |r_6 - r_8| \qquad (4.35)$$

Equation (4.34a) can be represented in the form of masks for *x*-direction and *y*-direction and they are given as in Figure 4.29. These masks are also known as Roberts' cross-gradient operators.

Another way of representing mask proposed by the scientist Prewitt using 3 × 3 mask is given in equation (4.36).

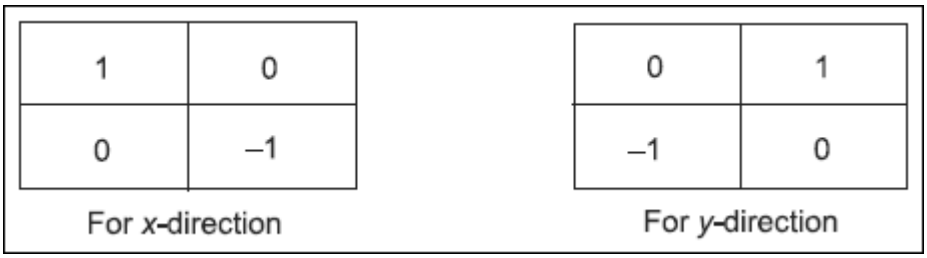


FIGURE 4.29 Roberts' mask

$$\nabla f = |(r_7 + r_8 + r_9) - (r_1 + r_2 + r_3)| + |(r_3 + r_6 + r_9) - (r_1 + r_4 + r_7)| \qquad (4.36)$$

The mask for equation (4.36) is shown in Figure 4.30. The equation (4.36) has two components and they correspond to *x* and *y* components. The first component is the difference between the third and first row of 3 × 3 region which approximates the derivative in the *x*-direction and the second component is the difference between the third and first column which approximates the derivative in the *y*-direction. The masks are also known as *Prewitt operators.* There is yet another pair of mask proposed by Sobel which is given in Figure 4.31.

The three different masks described here are applied to the original Lena image shown in Figure 4.32(a) (The colored figure is available here). Figures 4.32(b), (c), and (d) are the result of Roberts', Prewitt, and Sobel operators, respectively. A VC++ program to implement the Roberts', Prewitt, and Sobel operators is also given for the readers reference.

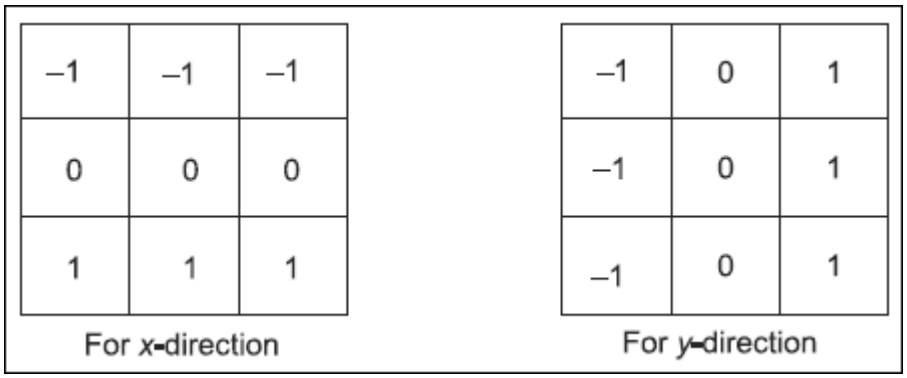


FIGURE 4.30 Prewitt mask

–1	–2	–1
0	0	0
1	2	1

For *x*-direction

–1	0	1
–2	0	2
–1	0	1

For *y*-direction

FIGURE 4.31 Sobel mask

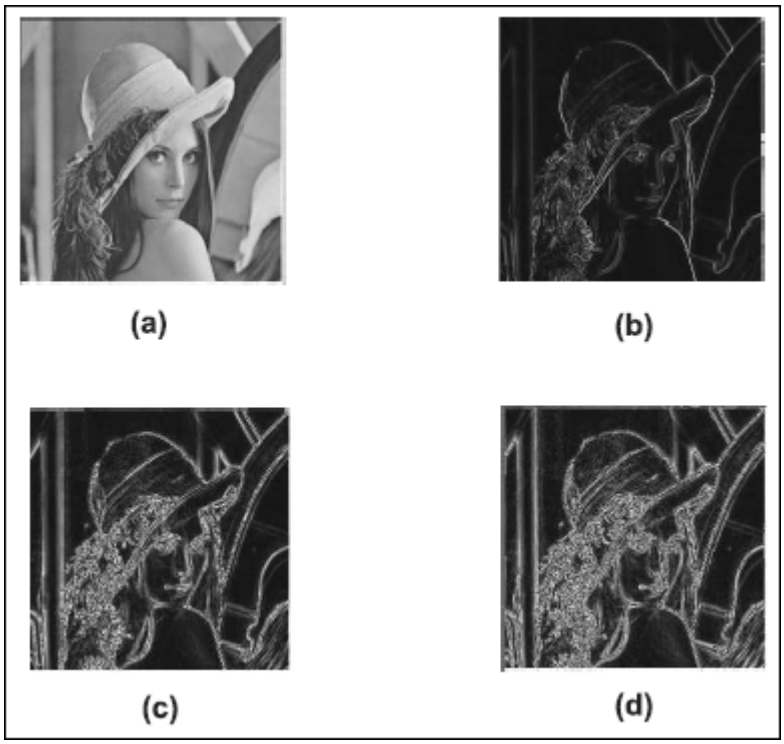


FIGURE 4.32 Output of the program given below. (a) Original image (b) Result of gradient approach using Roberts‘ approach (c) Result of gradient approach using Perwitt approach (d) Result of gradient approach using Sobel approach

4.5 FREQUENCY DOMAIN

The spatial domain filters used for image enhancement have been discussed earlier. The spatial technique methods are simple and easy to implement and also the speed of operation is high. In spite of these advantages, there are certain situations in which the spatial domain filters are not easily addressable. Under such circumstances it is more appealing and intuitive to use the frequency domain filtering approach. All the frequency domain filters are based on computing the Fourier transform of an image to be enhanced. Then the result is multiplied with a filter transfer function and the inverse Fourier transform is applied to the product so that it results in the enhanced image.

The low-pass filter is used to smooth the image and remove the high-frequency components related to noise. Smoothing effect is achieved in the frequency domain by attenuating a specified range of high-frequency components in the transformed image.

The low-pass filter equation can be represented as

$$Z(u, v) = H(u, v)F(u, v) \qquad (4.37)$$

where $F(u, v)$ is the Fourier transform of the image to be enhanced, $H(u, v)$ is the filter transfer function, and $Z(u, v)$ is the enhanced image in the frequency domain. In order to get the enhanced image in the spatial domain the inverse transform is applied and the corresponding equation is given by

$$z(x, y) = \text{inverse Fourier transform of } \{H(u, v)F(u, v)\} \qquad (4.38)$$

In equation (4.37) we have considered the transfer function $H(u, v)$ that gives $Z(u, v)$ by attenuating the high-frequency components of $F(u, v)$. Now our difficulty lies in selecting an appropriate filter transfer function. In general, most of the filter transfer functions affect the real and imaginary parts of $F(u, v)$ in the same manner. These filters are called as zero phase shift filters because they do not change the phase of the transform. In the following section, selection of the filter transfer function is discussed.

4.5.1 Ideal Low-pass Filter

The two-dimensional ideal low-pass filter's transfer function can be given by the relation

$$H(u, v) = \begin{cases} 1 & L(u, v) \leq L_0 \\ 0 & L(u, v) > L_0 \end{cases} \qquad (4.39)$$

where L_0 is a specified positive quantity and $L(u, v)$ is the distance from the point (u, v) to the origin of the frequency plane, that is,

$$L(u, v) = \left(u^2 + v^2\right)^{\frac{1}{2}} \qquad (4.40)$$

The frequency response of $H(u, v)$ as a function of u and v is shown in Figure 4.33(a). The cross-sectional view of Figure 4.33(a) is also shown in Figure 4.33(b).

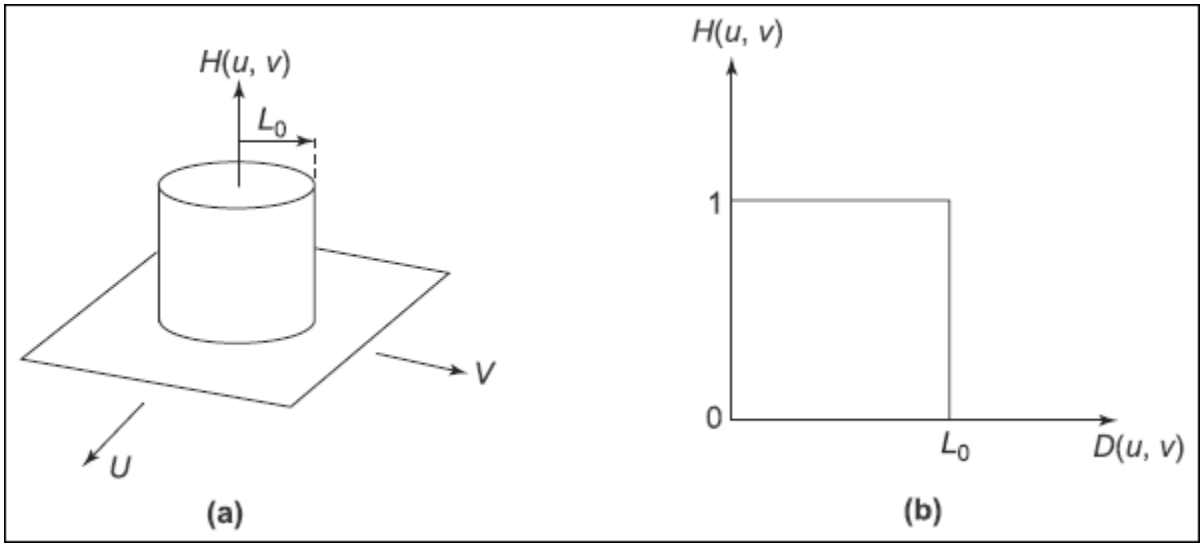


FIGURE 4.33 (a) Perspective plot of a low-pass filter transfer function (b) 2D filter transfer function

From Figure 4.33(a) it can be seen that the filter passes all the frequencies inside the circle of radius $L_0$where it attenuates all the frequencies outside this circle. Hence this filter is called *ideal low-pass filter.*

4.5.2 Butterworth Low-pass Filter

The response of the Butterworth low-pass filter of order *n* is defined by the equation

$$H(u, v) = \frac{1}{1 + [L(u, v)/L_0]^{2n}} \qquad (4.41)$$

where $L(u, v)$ is the distance from the point (u, v) to the origin and is given by $(u^2 + v^2)^{\frac{1}{2}}$. The three-dimensional and cross-sectional views of the Butterworth low-pass filter responses are shown in Figure 4.34.

When $L(u, v) = L_0$, $H(u, v) = 0.5$, this indicates that at cut-off frequency the response is half of its maximum value (50%).

In most of the cases, at cut-off frequency, the response will be equal to $\frac{1}{\sqrt{2}}$ times the maximum value of $H(u, v)$. To have this effect, the equation should be modified as given in equations (4.42) and (4.43).

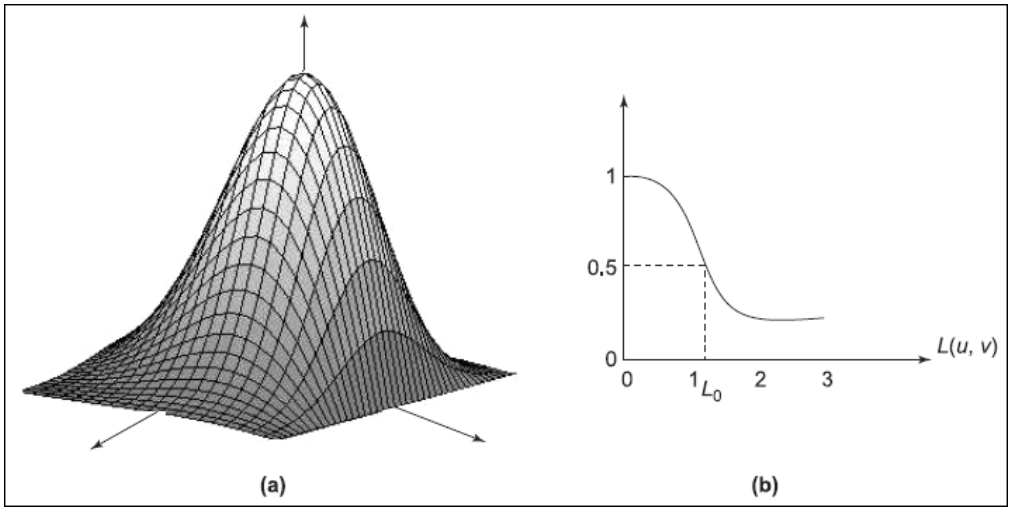


FIGURE 4.34 The three-dimensional and cross-sectional views of Butterworth low-pass filter

$$H(u, v) = \frac{1}{1 + [\sqrt{2} - 1][L(u, v)/L_0]^{2n}} \qquad (4.42)$$

$$H(u, v) = \frac{1}{1 + [0.414][L(u, v)/L_0]^{2n}} \qquad (4.43)$$

Figure 4.35(a) shows the original image (The colored figure is available here). Figure 4.35(b) shows the result of applying low-pass Butterworth filter of order n = 1 for different radii. From this example it can be understood that the low-pass filtering process reduces the spurious effect.

FIGURE 4.35 (a) The original image (b) The result of applying low-pass Butterworth filter of order 1

4.5.3 HIGH-PASS FILTER

In the previous sections we have discussed the ideal low-pass and Butterworth low-pass filters in detail. The low-pass filter results in the smoothing effect by attenuating the high-frequency components. The opposite effect, that is, the sharpening of the details in an image can be obtained using high-pass filter. The high-pass filter passes the high frequency components and it attenuates low-frequency components corresponding to slow-varying details of the image.

The ideal high-pass filter which has sharp or abrupt transition is given by the equation

$$H(u, v) = \begin{cases} 0, & \text{if } L(u, v) \leq L_0 \\ 1, & \text{if } L(u, v) > L_0 \end{cases} \qquad (4.44)$$

where L_0 is the cut-off frequency which is measured from the origin and $L(u, v)$ is the distance from the origin and is given by

$$\left[u^2 + v^2\right]^{\frac{1}{2}}$$

The filter response is opposite to the ideal low-pass filter discussed earlier. The ideal low-pass filter which has the abrupt transition at the cut-off frequency cannot be realized using the electronic circuit components. However, it can be realized with smooth transition frequency and such filters are called *Butterworth filters.*

The transfer function of the high-pass Butterworth filter of order n with a cut-off frequency L_0 from the origin is given by the equation

$$H(u, v) = \frac{1}{1 + [L_0/L(u, v)]^{2n}} \qquad (4.45)$$

4.5.4 HOMOMORPHIC FILTERING

Homomorphic Filtering: This filter controls both high-frequency and low-frequency components.

We have already discussed that an image function can be represented using illumination and reflectance components. The illumination component of an image generally represents the slow-varying details of the image, while the reflected component represents sharp details and are abruptly varying junctions of dissimilar object. Hence, we can interpret that the illumination corresponds to low-frequency components and the reflectance corresponds to high-frequency components in the frequency domain, respectively. It is possible to develop a filter which will control both high-frequency and low-frequency components. The filter which controls both high-frequency and low-frequency components are sometimes called as *Homomorphic filter.* The equation for Homomorphic filter can be derived from the illumination reflectance model given by the equation

$$f(x, y) = i(x, y)r(x, y) \qquad (4.46)$$

The Fourier transform of the product of two functions is not separable, that is,

$$F[f(x, y)] \neq F[i(x, y)]F[r(x, y)]$$

To overcome this difficulty, we rewrite equation (4.46) in the logarithmic form as

$$g(x, y) = \ln f(x, y) + \ln r(x, y) \qquad (4.47)$$

Then taking the Fourier transform of equation (4.47) results in

$$F[g(x, y)] = F[\ln f(x, y)] \qquad (4.48)$$

$$= F[\ln i(x, y)] + F[\ln r(x, y)]$$

$$G(u, v) = I(u, v) + R(u, v) \qquad (4.49)$$

where $I(u, v)$ and $R(u, v)$ are the Fourier transform of ln $r(x, y)$ and ln $i(x, y)$.

Let $H(u, v)$ be the Homomorphic filter function. The response of the $H(u, v)$ on the function $G(u, v)$ can be given by the relation

$$Z(u, v) = H(u, v).G(u, v) \qquad (4.50)$$

$$= H(u, v).I(u, v) + H(u, v).R(u, v) \qquad (4.51)$$

The inverse Fourier transform of equation (4.51)

$$Z(x, y) = F^{-1}\{H(u, v).I(u, v)\} + F^{-1}\{H(u, v).R(u, v)\} \qquad (4.52)$$

$$Z(x, y) = i'(x, y) + r'(x, y) \qquad (4.53)$$

where

$$i'(x, y) = F^{-1}\{H(u, v).I(u, v)\}$$

$$r'(x, y) = F^{-1}\{H(u, v).R(u, v)\}$$

As $g(x, y)$ is formed by taking a logarithm of original image $f(x, y)$, the inverse operation gives the desired enhanced image $Z(x, y)$, that is,

$$g(x, y) = \exp[z(x, y)]$$

$$= \exp[i'(x, y) + r'(x, y)]$$

$$g(x, y) = \exp[i'(x, y)].\exp[r'(x, y)]$$

where $g(x, y)$ is the enhanced image in the spatial domain. The procedure adopted can be given by a schematic diagram shown in Figure 4.35(a). Figure 4.36(b) is used as an input to the homomorphic filter and its response is shown in Figure 4.36(c).

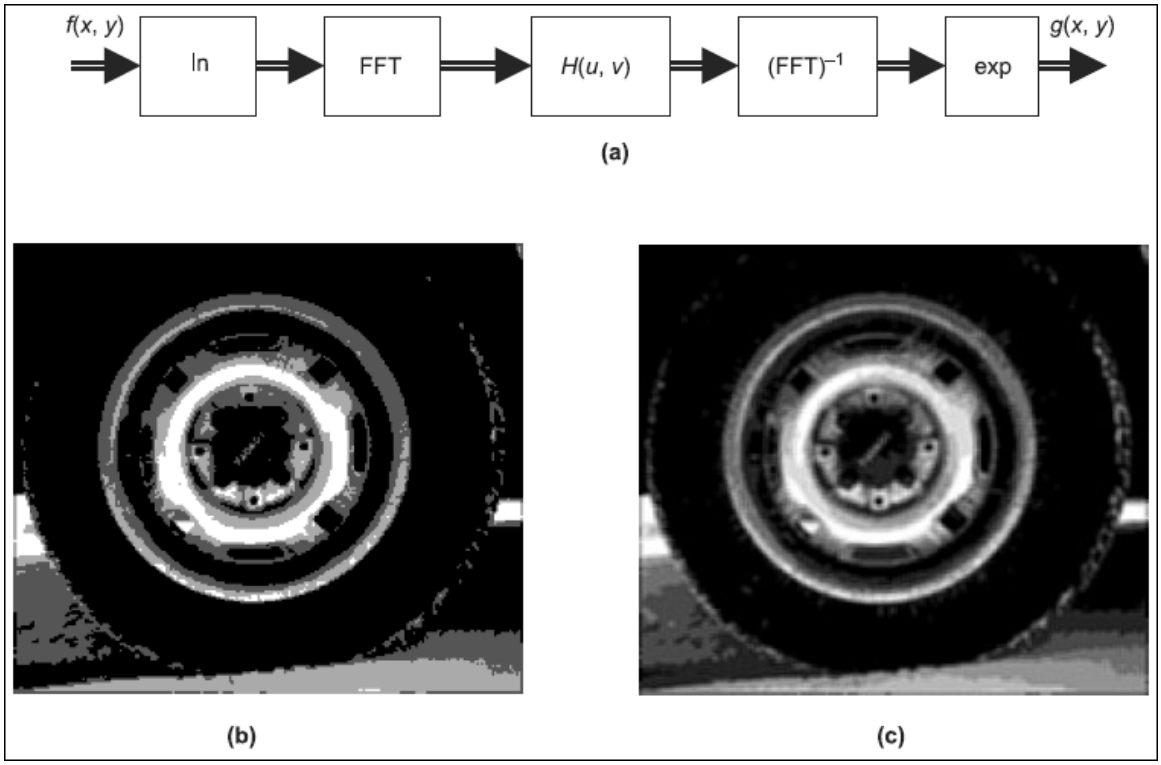


FIGURE 4.36 (a) Homomorphic filtering approach for image enhancement (b) The tire image (*Source*: MathWorks Inc., USA (MATLab)) (c) The response of the homomorphic filter

4.5.5 Pseudo Color Image

The assigning of color to monochrome images based on certain properties of the gray level content is called *pseudo color image processing.* This gives an illusion to a common person as if the image under consideration is a color image.

Pseudo Color: A tricolor image formed using red, green, and blue primary colors and it gives an illusion to the common man as if the image under consideration is a color image.

The intensity slicing and color coding is one of the simplest examples of pseudo color image processing. If an image is viewed as a two-dimensional intensity function then a parallel plane can be placed at each coordinate corresponding to the plane which will slice the area of intersection. Figure 4.37 shows an example of slicing the image at a height $h_1 = f(x, y)$ to slice into two parts. In general, many planes are located at different heights $h_1, h_2, \ldots, h_m$ and the gray levels ranging from l_0 to L, where $l_0$corresponds to dark and L to white. So the gray levels ranging from $0 < m < L$, have m planes to partition the image into $m + 1$ regions. The regions which have the same gray levels are denoted as $r_1, r_2, \ldots, r_5$. Then the color assigned to each of the region can be given by the relation

$$f(x, y) = c_k \quad \text{if} \quad f(x, y) \varepsilon r_k \qquad (4.54)$$

where C_k is color associated with kth region.

An example for intensity slicing is shown in the Figure 4.37 (The colored figure is available here) where the different gray level regions are highlighted with different colors.

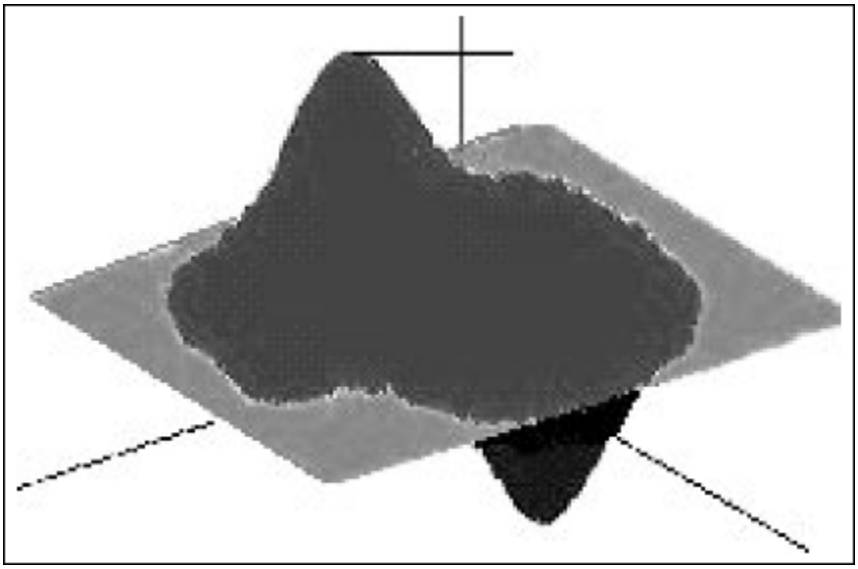


FIGURE 4.37 Slicing the image at height h_1

4.6 GRAY LEVEL TO COLOR TRANSFORMATION

The pseudo color enhancement for the monochrome image can be achieved using three independent transformations on the gray levels of a given input image. The results of the three transformations are then separately sent to the red, green, and blue guns of a color television monitor. This approach produces a composite image color content of which is modulated by the nature of the transfer function used.

4.6.1 Filter Approach for Color Coding

In this approach the Fourier transform of the monochrome image to be enhanced is first obtained. The Fourier transform of the image is then subjected to three different filters. The first filter may allow the low-frequency components of the image. In order to design the filter, a cut-off frequency corresponding to the low-frequency must be selected. The second filter will pass the frequency components corresponding to the high frequency in the images. The third filter may pass the band of frequencies in the image.

The output of these filters is subjected to the inverse Fourier transform followed by proper preprocessing techniques. Then the final output from these three different streams is fed to a tricolor picture tube so that the final output will be highlighted with red, blue, and green colors corresponding to low-frequency components, a desired band of frequencies, and high-frequency components of the image.

Summary

Image enhancement techniques are used to improve the image quality or appearance for human interpretation. This chapter introduces spatial domain and frequency domain

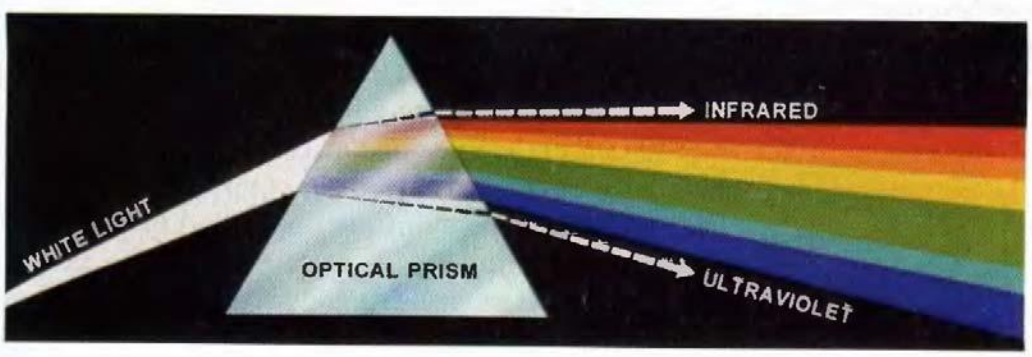
enhancement techniques. It lays stress on the importance and necessity of image enhancement processes in various applications. It gives a detailed description of the steps involved in spatial and frequency domain techniques. Most of the techniques covered are explained with sample programs implemented in VC++. The images obtained as output for these programs are also given. Students can make use of these programs to have a better practical insight of these techniques.

The spatial domain technique such as histogram equalization is a fundamental tool, not only in image enhancement, but also in satellite and medical imaging applications. This chapter also provides a fair description of the spatial masks and filters. The neighborhood pixel processing techniques discussed in Chapter 1 are used in spatial masking techniques. Various filtering techniques are also discussed with examples. The examples for various filtering applications are self-explanatory.

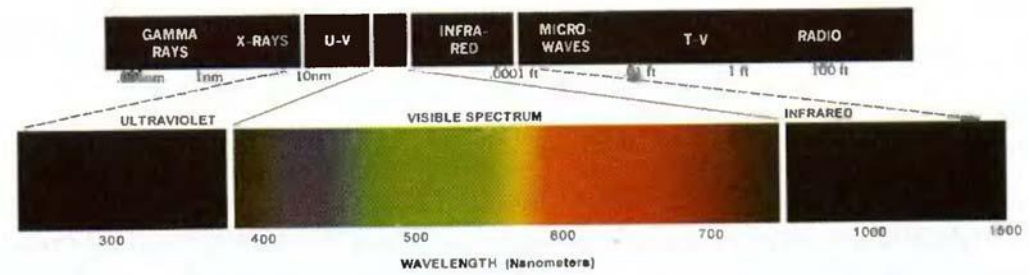
The chapter concludes with the application of pseudo color image processing for image enhancement. It also gives a brief description about the filter approach for color coding. The fact that these tools were introduced in the context of image enhancement is likely to aid in the understanding of how they operate on digital images. The students can improve their understanding capability and logical reasoning by answering the questions and solving the problems given at the end of the chapter.

Colour Image Processing

Color of an object is determined by the nature of the light reflected from it. When a beam of sunlight passes through a glass prism, the emerging beam of light is not white but consists instead of a continuous spectrum of colors ranging from violet at one end to red at the other. As Fig. 5.1.1 shows, the color spectrum may be divided into six broad regions: violet, blue, green, yellow, orange, and red. When viewed in full color (Fig. 5.1.2), no color in the spectrum ends abruptly, but rather each color blends smoothly into the next.



5.1.1 Color spectrum seen by passing white light through a prism.



5.1.2 Wavelengths comprising the visible range of the electromagnetic spectrum.

As illustrated in Fig. 5.1.2, visible light is composed of a relatively narrow band of frequencies in the electromagnetic spectrum. A body that reflects light that is balanced in all visible wavelengths appears white to the observer. However, a body that favors reflectance in a limited range of the visible spectrum exhibits some shades of color. For example, green objects reflect light with wavelengths primarily in the 500 to 570 nm range while absorbing most of the energy at other wavelengths.

Characterization of light is central to the science of color. If the light is achromatic (void of color), its only attribute is its intensity, or amount. Achromatic light is what viewers see on a black and white television set.

Three basic quantities are used to describe the quality of a chromatic light source: radiance, luminance, and brightness.

Radiance:

Radiance is the total amount of energy that flows from the light source, and it is usually measured in watts (W).

Luminance:

Luminance, measured in lumens (lm), gives a measure of the amount of energy an observer perceives from a light source. For example, light emitted from a source operating in the far infrared region of the spectrum could have significant energy (radiance), but an observer would hardly perceive it; its luminance would be almost zero.

Brightness:

Brightness is a subjective descriptor that is practically impossible to measure. It embodies the achromatic notion of intensity and is one of the key factors in describing color sensation.

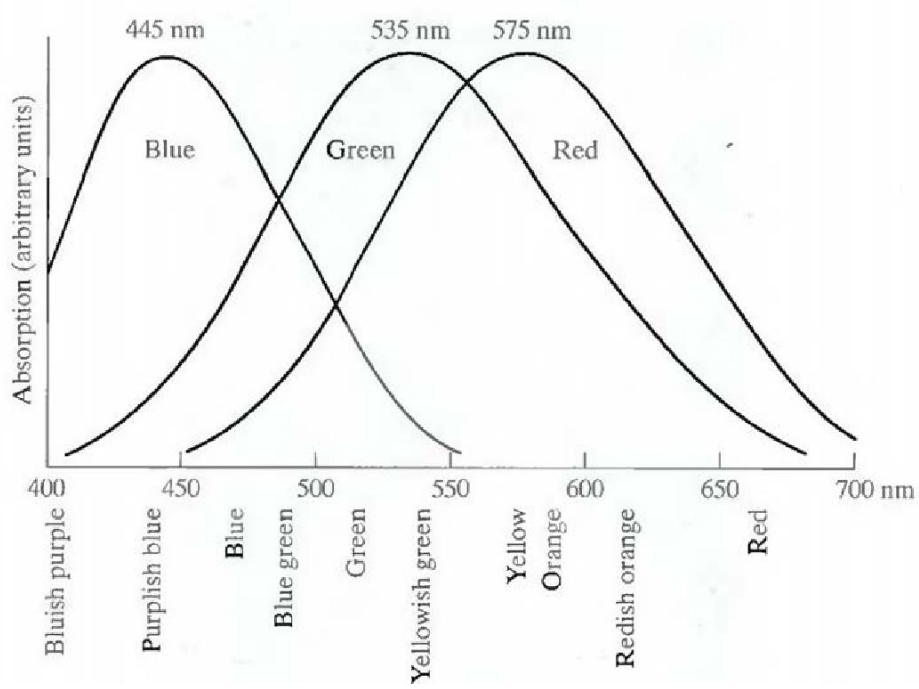


Fig. 5.1.3 Absorption of light by the red, green, and blue cones in the human eye as a function of wavelength.

Cones are the sensors in the eye responsible for color vision. Detailed experimental evidence has established that the 6 to 7 million cones in the human eye can be divided into three principal sensing categories, corresponding roughly to red, green, and blue. Approximately 65% of all cones are sensitive to red light, 33% are sensitive to green light, and only about 2% are sensitive to blue (but the blue cones are the most sensitive). Figure 5.1.3 shows average experimental curves detailing the absorption of light by the red, green, and blue cones in the eye. Due to these absorption characteristics of the human eye, colors arc seen as variable combinations of the so- called primary colors red (R), green (G), and blue (B).

The primary colors can be added to produce the secondary colors of light --magenta (red plus blue), cyan (green plus blue), and yellow (red plus green). Mixing the three primaries, or a secondary with its opposite primary color, in the right intensities produces white light.

The characteristics generally used to distinguish one color from another are brightness, hue, and saturation. Brightness embodies the chromatic notion of intensity. Hue is an attribute associated with the dominant wavelength in a mixture of light waves. Hue represents dominant color as perceived by an observer. Saturation refers to the relative purity or the amount of white light mixed with a hue. The pure spectrum colors are fully saturated. Colors such as pink (red and white) and lavender (violet and white) are less saturated, with the degree of saturation being inversely proportional to the amount of white light-added.

Hue and saturation taken together are called chromaticity, and. therefore, a color may be characterized by its brightness and chromaticity.

RGB COLOR MODEL

The purpose of a color model (also called color space or color system) is to facilitate the specification of colors in some standard, generally accepted way. In essence, a color model is a specification of a coordinate system and a subspace within that system where each color is represented by a single point.

The RGB Color Model:

In the RGB model, each color appears in its primary spectral components of red, green, and blue. This model is based on a Cartesian coordinate system. The color subspace of interest is the cube shown in Fig. 5.2, in which RGB values are at three corners; cyan, magenta, and yellow are at three other corners; black is at the origin; and white is at the corner farthest from the origin. In this model, the gray scale (points of equal RGB values) extends from black to white along the line joining these two points. The different colors in this model arc points on or inside the cube, and are defined by vectors extending from the origin. For convenience, the assumption is that all color values have been normalized so that the cube shown in Fig. 5.2 is the unit cube. That is, all values of R, G. and B are assumed to be in the range [0, 1].

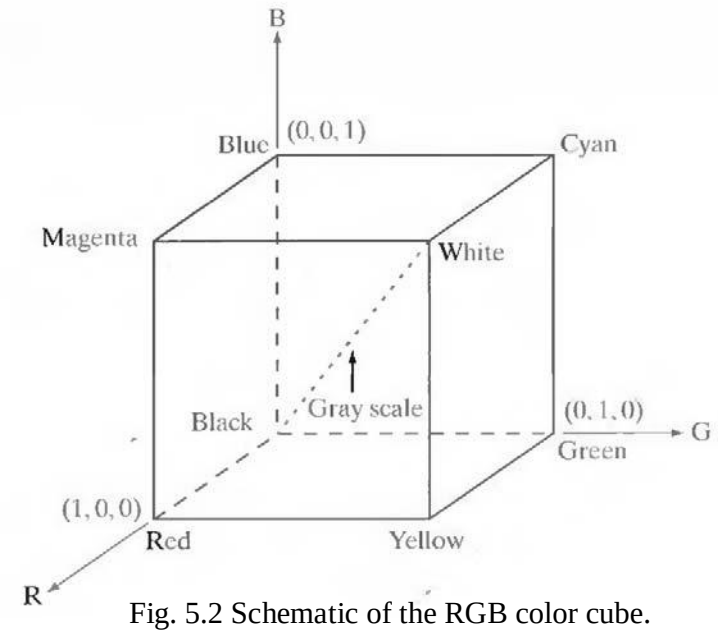


Fig. 5.2 Schematic of the RGB color cube.

Images represented in the RGB color model consist of three component images, one for each primary color. When fed into an RGB monitor, these three images combine on the phosphor screen to produce a composite color image. The number of bits used to represent each pixel in RGB space is called the pixel depth.

Consider an RGB image in which each of the red, green, and blue images is an 8-bit image. Under these conditions each RGB color pixel [that is, a triplet of values (R, G, B)] is said to have a depth of 24 bits C image planes times the number of bits per plane). The term full-color image is used often to denote a 24-bit RGB color image. The total number of colors in a 24-bit RGB image is $(2^8)^3 = 16{,}777{,}216$.

RGB is ideal for image color generation (as in image capture by a color camera or image display in a monitor screen), but its use for color description is much more limited.

CMY color model.

have been normalized to the range [0, 1]. Equation (1) demonstrates that light reflected from a surface coated with pure cyan does not contain red (that is, C = 1 — R in the equation).

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Similarly, pure magenta does not reflect green, and pure yellow does not reflect blue. Equation (1) also reveals that RGB values can be obtained easily from a set of CMY values by subtracting the individual CMY values from 1. As indicated earlier, in image processing this color model is used in connection with generating hardcopy output, so the inverse operation from CMY to RGB generally is of little practical interest.

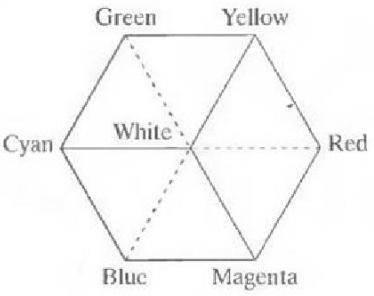
Equal amounts of the pigment primaries, cyan, magenta, and yellow should produce black. In practice, combining these colors for printing produces a muddy-looking black.

HSI color model

When humans view a color object, we describe it by its hue, saturation, and brightness. Hue is a color attribute that describes a pure color (pure yellow, orange, or red), whereas saturation gives a measure of the degree to which a pure color is diluted by white light. Brightness is a subjective descriptor that is practically impossible to measure. It embodies the achromatic notion of intensity and is one of the key factors in describing color sensation.

Intensity (gray level) is a most useful descriptor of monochromatic images. This quantity definitely is measurable and easily interpretable. The HSI (hue, saturation, intensity) color model, decouples the intensity component from the color-carrying information (hue and saturation) in a color image. As a result, the HSI model is an ideal tool for developing image processing algorithms based on color descriptions that are natural and intuitive to humans.

In Fig 5.4 the primary colors are separated by 120°. The secondary colors are 60° from the primaries, which means that the angle between secondaries is also 120°. Figure 5.4(b) shows the same hexagonal shape and an arbitrary color point (shown as a dot).The hue of the point is determined by an angle from some reference point. Usually (but not always) an angle of 0° from the red axis designates 0 hue, and the hue increases counterclockwise from there. The saturation (distance from the vertical axis) is the length of the vector from the origin to the point. Note that the origin is defined by the intersection of the color plane with the vertical intensity axis. The important components of the HSI color space are the vertical intensity axis, the length of the vector to a color point, and the angle this vector makes with the red axis.



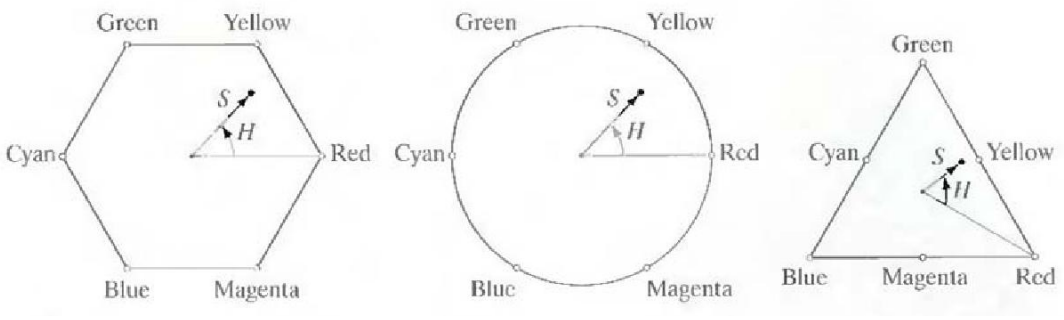


Fig 5.4 Hue and saturation in the HSI color model.

Conversion from RGB color model to HSI color model

Given an image in RGB color format, the H component of each RGB pixel is obtained using the equation

$$H = \begin{cases} \theta & \text{if } B \le G \\ 360 - \theta & \text{if } B > G \end{cases}$$

With

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\}.$$

The saturation component is given by

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)].$$

Finally, the intensity component is given by

$$I = \frac{1}{3}(R+G+B).$$

It is assumed that the RGB values have been normalized to the range [0, 1] and that angle θ is measured with respect to the red axis of the HST space. Hue can be normalized to the range [0, 1] by dividing by 360° all values resulting from Eq. (1). The other two HSI components already are in this range if the given RGB values are in the interval [0, 1].

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)].$$

Conversion from HSI color model to RGB color model

Given values of HSI in the interval [0,1], one can find the corresponding RGB values in the same range. The applicable equations depend on the values of H. There are three sectors of interest, corresponding to the 120° intervals in the separation of primaries.\

RG sector ($0^o \leq H < 120°$):

When H is in this sector, the RGB components are given by the equations

$$B = I (1 – S)$$

$$G = 3 I – (R + B)$$

$$R = I [1 + (S * \cos H/ \cos(60^o – H)]$$

GB sector ($120^o \leq H < 240^o$):

If the given value of H is in this sector, first subtract 120° from it.

$$H = H - 120^0$$

Then the RGB components are

$$R = I (1 – S)$$

$$B = 3 I – (R + G)$$

$$G = I [1 + (S * \cos H/ \cos(60^o – H)]$$

BR sector ($240^o \leq H \leq 360^o$):

If H is in this range, subtract 240^o from it

$$H = H - 240^0$$

Then the RGB components are

G = I (1 – S)

$$R = 3 I – (B + G)$$

$$B = I [1 + (S * \cos H/ \cos(60^o – H)]$$

PSEUDO COLOR IMAGE PROCESSING

Pseudocolor (also called false color) image processing consists of assigning colors to gray values based on a specified criterion. The term pseudo or false color is used to differentiate the process of assigning colors to monochrome images from the processes associated with true color images. The process of gray level to color transformations is known as pseudocolor image processing.

The two techniques used for pseudocolor image processing are,

(i) Intensity Slicing

(ii) Gray Level to Color Transformation

(i) Intensity Slicing:

The technique of intensity (sometimes called density) slicing and color coding is one of the simplest examples of pseudocolor image processing. If an image is interpreted as a 3-D function (intensity versus spatial coordinates), the method can be viewed as one of placing planes parallel to the coordinate plane of the image; each plane then "slices" the function in the area of intersection. Figure 5.8 shows an example of using a plane at $f(x, y) = l_i$ to slice the image function into two levels.

If a different color is assigned to each side of the plane shown in Fig. 5.8, any pixel whose gray level is above the plane will be coded with one color, and any pixel below the plane will be coded with the other. Levels that lie on the plane itself may be arbitrarily assigned one of the two colors. The result is a two-color image whose relative appearance can be controlled by moving the slicing plane up and down the gray-level axis.

In general, the technique may be summarized as follows. Let [0, L - 1] represent the gray scale, let level l_o represent black [f(x, y) = 0], and level l_{L-1} represent white [f(x, y) = L - 1]. Suppose that P planes perpendicular to the intensity axis are defined at levels $l_1, l_2,\ldots,l_p$. Then, assuming that $0 < P < L - 1$, the P planes partition the gray scale into P + 1 intervals, $V_1, V_2,..., V_p + 1$. Gray-level to color assignments are made according to the relation

$$f(x, y) = c_k \quad \text{if } f(x, y) \in V_k$$

where c_k is the color associated with the kth intensity interval V_k defined by the partitioning planes at $l = k - 1$ and $l = k$.

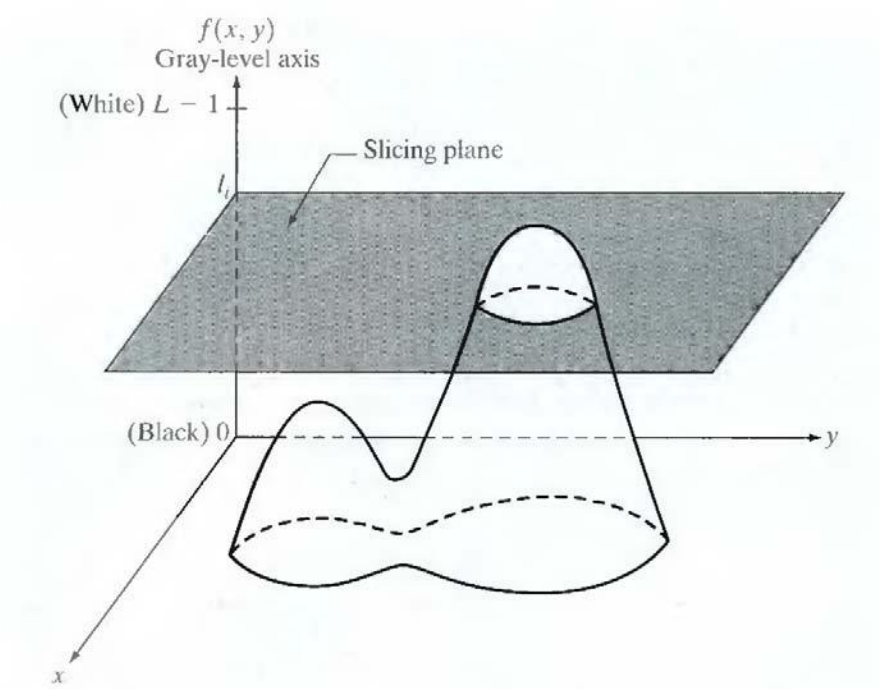


Fig 5.8.1 Geometric interpretation of the intensity-slicing technique.

The idea of planes is useful primarily for a geometric interpretation of the intensity-slicing technique. Figure 5.8.2 shows an alternative representation that defines the same mapping as in Fig. 5.8.1. According to the mapping function shown in Fig. 5.8.2, any input gray level is assigned one of two colors, depending on whether it is above or below the value of l_i When more levels are used, the

mapping function takes on a staircase form.

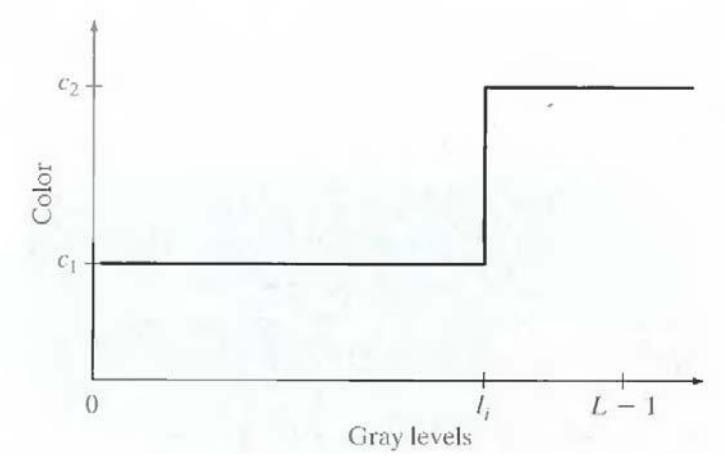


Fig 5.8.2 An alternative representation of the intensity-slicing technique.

(ii) Gray Level to Color Transformation:

The idea underlying this approach is to perform three independent transformations on the gray level of any input pixel. The three results are then fed separately into the red, green, and blue channels of a color television monitor. This method produces a composite image whose color content is modulated by the nature of the transformation functions. Note that these are transformations on the gray-level values of an image and are not functions of position.

In intensity slicing, piecewise linear functions of the gray levels are used to generate colors. On the other hand, this method can be based on smooth, nonlinear functions, which, as might be expected, gives the technique considerable flexibility.

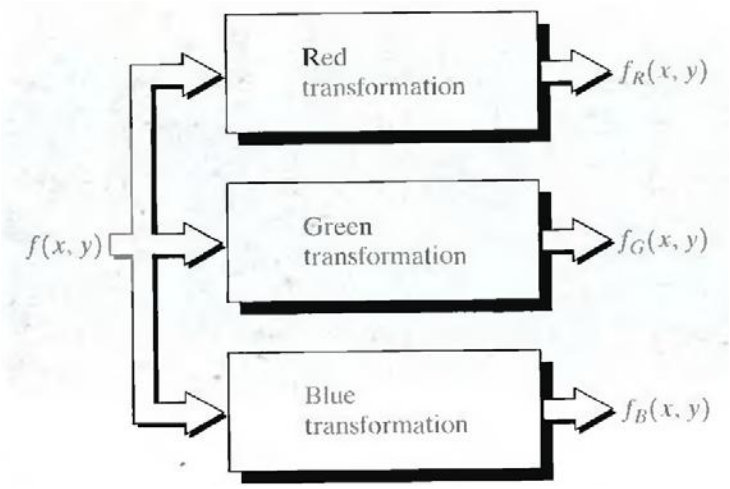


Fig. 5.8.3 Functional block diagram for pseudocolor image processing.

The output of each transformation is a composite image.

FULL COLOR IMAGE PROCESSING

Full-color image processing approaches fall into two major categories. In the first category, each component image is processed individually and then form a composite processed

color image from the individually processed components. In the second category, one works with color pixels directly. Because full-color images have at least three components, color pixels really are vectors. For example, in the RGB system, each color point can be interpreted as a vector extending from the origin to that point in the RGB coordinate system.

Let c represent an arbitrary vector in RGB color space:

$$\mathbf{c} = \begin{bmatrix} c_R \\ c_G \\ c_B \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

This equation indicates that the components of c are simply the RGB components of a color image at a point. If the color components are a function of coordinates (x, y) by using the notation

$$\mathbf{c}(x, y) = \begin{bmatrix} c_R(x, y) \\ c_G(x, y) \\ c_B(x, y) \end{bmatrix} = \begin{bmatrix} R(x, y) \\ G(x, y) \\ B(x, y) \end{bmatrix}.$$

For an image of size M X N, there are MN such vectors, c(x, y), for
x = 0,1, 2,...,M- l; y = 0,1,2,..., N- 1.

It is important to keep clearly in mind that Eq. (2) depicts a vector whose components are spatial variables in x and y.

In order for per-color-component and vector-based processing to be equivalent, two conditions have to be satisfied: First, the process has to be applicable to both vectors and scalars. Second, the operation on each component of a vector must be independent of the other components.

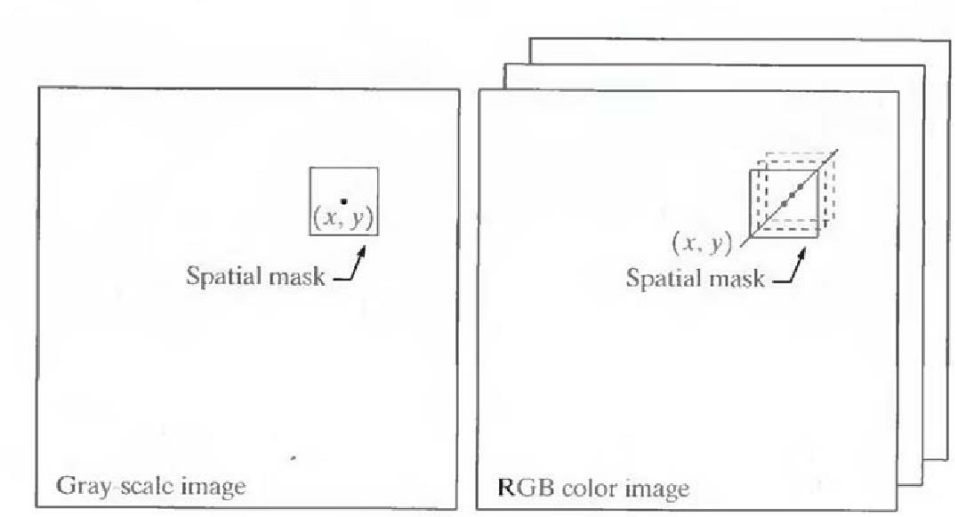


Fig 9 Spatial masks for gray-scale and RGB color images.

Fig 9 shows neighborhood spatial processing of gray-scale and full-color images. Suppose that

the process is neighborhood averaging. In Fig. 9(a), averaging would be accomplished by summing the gray levels of all the pixels in the neighborhood and dividing by the total number of pixels in the neighborhood. In Fig. 9(b), averaging would be done by summing all the vectors in the neighborhood and dividing each component by the total number of vectors in the neighborhood. But each component of the average vector is the sum of the pixels in the image corresponding to that component, which is the same as the result that would be obtained if the averaging were done on a per-color-component basis and then the vector was formed.

UNIT – IV

Image Restoration: Degradation model – Algebraic approach to restoration – Inverse filtering – Least Mean Square filters – Constrained Least Mean Square restoration – Inverse Restoration.

MODEL OF THE IMAGE DEGRADATION/RESTORATION PROCESS

The Fig. 6.3 shows, the degradation process is modeled as a degradation function that, together with an additive noise term, operates on an input image f(x, y) to produce a degraded image g(x, y). Given g(x, y), some knowledge about the degradation function H, and some knowledge about the additive noise term ç(x, y), the objective of restoration is to obtain an estimate *f*(x, y) of the original image. the estimate should be as close as possible to the original input image and, in general, the more we know about H and ç, the closer *f*(x, y) will be to f(x, y).

The degraded image is given in the spatial domain by

$$g(x, y) = h(x, y) * f(x, y) + ç(x, y)$$

where h (x, y) is the spatial representation of the degradation function and, the symbol * indicates convolution. Convolution in the spatial domain is equal to multiplication in the frequency domain, hence

$$G(u, v) = H(u, v)\, F(u, v) + N(u, v)$$

where the terms in capital letters are the Fourier transforms of the corresponding terms in above equation.

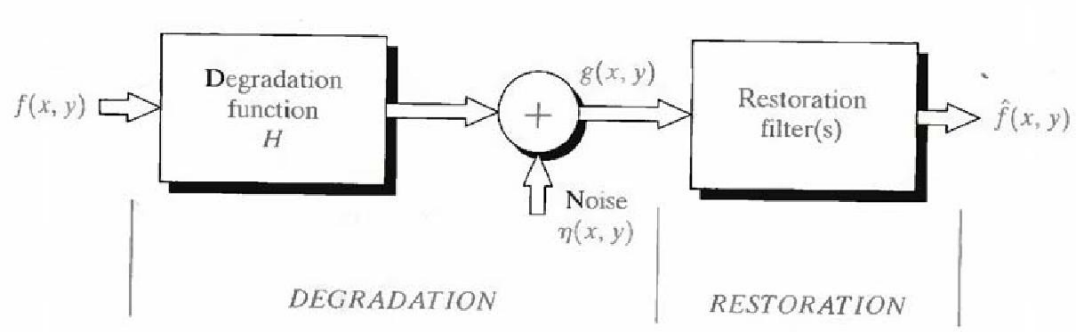


Fig. 6.3 model of the image degradation/restoration process.

WIENER FILTER USED FOR IMAGE RESTORATION

The inverse filtering approach makes no explicit provision for handling noise. This approach incorporates both the degradation function and statistical characteristics of noise into the restoration process. The method is founded on considering images and noise as random processes, and the objective is to find an estimate *f* of the uncorrupted image f such that the mean square error between them is minimized. This error measure is given by

$$e^2 = E\{(f - f)^2\}$$

where E{•} is the expected value of the argument. It is assumed that the noise and the image are

uncorrelated; that one or the other has zero mean; and that the gray levels in the estimate are a linear function of the levels in the degraded image. Based on these conditions, the minimum of the error function is given in the frequency domain by the expression

$$\hat{F}(u,v) = \left[\frac{H^*(u,v)S_f(u,v)}{S_f(u,v)|H(u,v)|^2 + S_\eta(u,v)}\right]G(u,v)$$
$$= \left[\frac{H^*(u,v)}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right]G(u,v)$$
$$= \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right]G(u,v)$$

where we used the fact that the product of a complex quantity with its conjugate is equal to the magnitude of the complex quantity squared. This result is known as the Wiener filter, after N. Wiener [1942], who first proposed the concept in the year shown. The filter, which consists of the terms inside the brackets, also is commonly referred to as the minimum mean square error filter or the least square error filter. The Wiener filter does not have the same problem as the inverse filter with zeros in the degradation function, unless both H(u, v) and $S_ç$(u, v) are zero for the same value(s) of u and v.

The terms in above equation are as follows:

H (u, v) = degradation function

H*(u, v) = complex conjugate of H (u, v)

$|H(u, v)|^2 = H^*(u, v)* H(u, v)$

$S_ç(u, v) = |N(u, v)|^2$ = power spectrum of the noise

$S_f(u, v) = |F(u, v)|^2$ = power spectrum of the undegraded image.

As before, H (u, v) is the transform of the degradation function and G (u, v) is the transform of the degraded image. The restored image in the spatial domain is given by the inverse Fourier transform of the frequency-domain estimate F (u, v). Note that if the noise is zero, then the noise power spectrum vanishes and the Wiener filter reduces to the inverse filter.

When we are dealing with spectrally white noise, the spectrum $|N(u, v)|^2$ is a constant, which simplifies things considerably. However, the power spectrum of the undegraded image seldom is known. An approach used frequently when these quantities are not known or cannot be estimated is to approximate the equation as

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right]G(u,v)$$

RESTORATION FILTERS USED WHEN THE IMAGE DEGRADATION IS DUE TO NOISE ONLY

If the degradation present in an image is only due to noise, then,

$$g(x, y) = f(x, y) + ç(x, y)$$

$$G(u, v) = F(u, v) + N(u, v)$$

The restoration filters used in this case are,

1. Mean filters
2. Order static filters and
3. Adaptive filters

(i) Arithmetic mean filter

This is the simplest of the mean filters. Let S_{xy} represent the set of coordinates in a rectangular subimage window of size m X n, centered at point (x, y).The arithmetic mean filtering process computes the average value of the corrupted image g(x, y) in the area defined by S_{xy}. The value of the restored image *f* at any point (x, y) is simply the arithmetic mean computed using the pixels in the region defined by S_{xy}. In other words

$$\hat{f}(x, y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s, t).$$

This operation can be implemented using a convolution mask in which all coefficients have value 1/mn

(ii) Geometric mean filter

An image restored using a geometric mean filter is given by the expression

$$\hat{f}(x, y) = \left[\prod_{(s,t) \in S_{xy}} g(s, t) \right]^{\frac{1}{mn}}.$$

Here, each restored pixel is given by the product of the pixels in the subimage window, raised to the power 1/mn. A geometric mean filter achieves smoothing comparable to the arithmetic mean filter, but it tends to lose less image detail in the process.

(iii) Harmonic mean filter

The harmonic mean filtering operation is given by the expression

$$\hat{f}(x, y) = \frac{mn}{\sum_{(s,t) \in S_{xy}} \frac{1}{g(s, t)}}.$$

The harmonic mean filter works well for salt noise, but fails for pepper noise. It does well also with other types of noise like Gaussian noise.

(iv) Contra harmonic mean filter

The contra harmonic mean filtering operation yields a restored image based on the expression

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

where Q is called the order of the filter. This filter is well suited for reducing or virtually eliminating the effects of salt-and-pepper noise. For positive values of Q, the filter eliminates pepper noise. For negative values of Q it eliminates salt noise. It cannot do both simultaneously. Note that the contra harmonic filter reduces to the arithmetic mean filter if Q = 0, and to the harmonic mean filter if Q = -1.

(i) Median filter

The best-known order-statistics filter is the median filter, which, as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel:

$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{\text{median}}\{g(s,t)\}.$$

The original value of the pixel is included in the computation of the median. Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of both bipolar and unipolar impulse noise.

INVERSE FILTERING

The simplest approach to restoration is direct inverse filtering, where F (u, v), the transform of the original image is computed simply by dividing the transform of the degraded image, G (u, v), by the degradation function

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}.$$

The divisions are between individual elements of the functions.

But G (u, v) is given by

$$G(u, v) = F(u, v) + N(u, v)$$

Hence

$$\hat{F}(u, v) = F(u, v) + \frac{N(u, v)}{H(u, v)}.$$

It tells that even if the degradation function is known the undegraded image cannot be recovered [the inverse Fourier transform of F(u, v)] exactly because N(u, v) is a random function whose Fourier transform is not known.

If the degradation has zero or very small values, then the ratio N(u, v)/H(u, v) could easily dominate the estimate F(u, v).

One approach to get around the zero or small-value problem is to limit the filter frequencies to values near the origin. H (0, 0) is equal to the average value of h(x, y) and that this is usually the highest value of H (u, v) in the frequency domain. Thus, by limiting the analysis to frequencies near the origin, the probability of encountering zero values is reduced.

Image Segmentation: Detection of Discontinuities – Edge Linking – Boundary detection and Boundary Description – Thresholding – Region Oriented Segmentation.

6.1 INTRODUCTION

Image analysis is an area used for extracting the information from an image. Before we extract the information, the image has to be subdivided into constituent parts or objects. The process of subdividing the given image into its constituent parts or objects is called *image segmentation.* Image segmentation is the first step in image analysis. The level at which the subdivision is carried out depends on the problem being solved. For example, let us consider the image of a basket containing fruits like apples, oranges, and grapes. To know the size of the orange, the image is subdivided into its constituent parts until we get a subimage of the orange. Further subdivision is then not required.

One of the most difficult task in image processing is segmentation process. It plays a vital role in any application and its success is based on the effective implementation of the segmentation technique.

The segmentation algorithms can be divided into two broad categories based on the two important properties, namely,

1. Discontinuity and

2. Similarity.

The various segmentation techniques based on (1) gray level discontinuity and (2) gray level similarity are well depicted in a graph as shown in Figure 6.1.

The forthcoming sections deal with the detection of isolated points, lines, and edges.

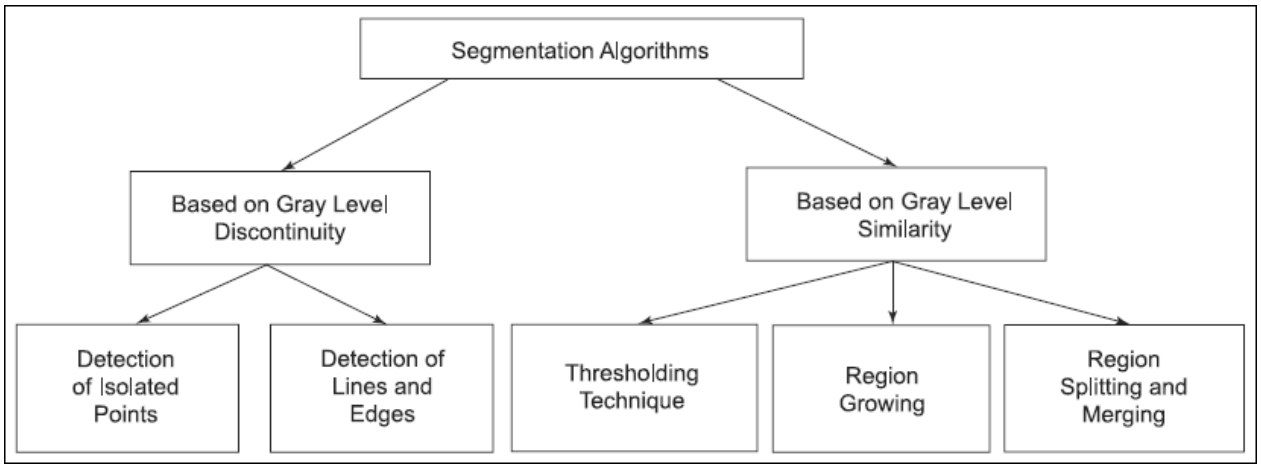


FIGURE 6.1 Segmentation techniques

6.2 DETECTION OF ISOLATED POINTS

In order to detect the isolated points due to noise or any other interference, the general mask employed is shown in Figure 6.2(a) and the typical values of the weights '*W*' are shown in Figure 6.2(b). This mask consists of coefficients '−1' everywhere except at the center which is 8. The sum of these coefficients is 0. When we place the mask over an image it covers 9 pixels in the image. The average response to the mask is computed as

$$R = \frac{1}{9}\{W_1Z_1 + W_2Z_2 + \cdots + W_9Z_9\} = \frac{1}{9}\sum_{i=1}^{9} W_iZ_i \qquad (6.1)$$

where W_i is the coefficient in the mask and Z_i denotes the gray level values of the pixel in the image under the mask. Now the mask is placed at the top left corner of the image and the response to the mask is computed using equation (6.1).

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

(a)

−1	−1	−1
−1	8	−1
−1	−1	−1

(b)

FIGURE 6.2 (a) The general representation of the mask (b) The mask with coefficient values

If the mask is over a uniform intensity area, the response due to this mask is equal to 0. This means there are no isolated pixels with different gray level values. On the other hand, if the mask is placed over the area having an isolated point with different gray levels, the response to the mask will be a nonzero value. The average response will be maximum when the isolated points are just below the center of the mask. Therefore, from the mask response it is possible to locate the isolated points resulting due to noise.

6.3 LINE DETECTION

The various masks used for detecting horizontal, vertical, +45°, and −45° slanting line are shown in Figure 6.3.

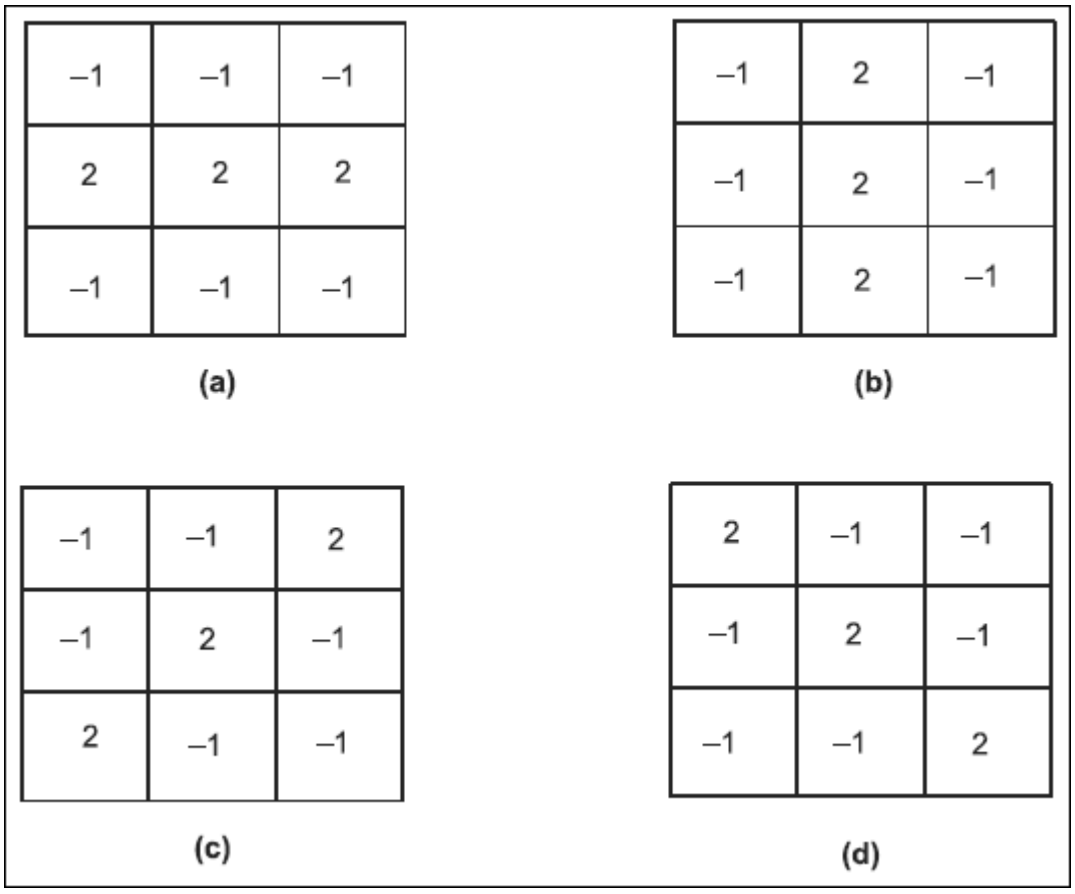


FIGURE 6.3 Masks for detecting lines. (a) Mask for horizontal line detection (b) Mask for + 45° slanting line detection (c) Mask for vertical line detection (d) Mask for − 45° slanting line detection

If the first mask shown in Figure 6.3(a) is moved around an image, its response will be a large value to lines oriented horizontally. The response will be maximum when the line passes through the middle row of the mask with constant background. For example, when we move the

mask over an image consisting of all ‘1’s as background and with a line of different gray level ‘10’s, then the response due to the first mask is computed as

$$(1 * -1) + (1 * -1) + (1 * -1) + (2 * 10) + (2 * 10) + (2 * 10)$$
$$+ (+1 * -1) + (+1 * -1) + (+1 * -1) = 54$$

This high response indicates that the mask is moving along a horizontal line with different gray levels compared to the background pixel gray level values. Similar experiments with second mask results in high response to vertical lines and the third mask to the lines +45° and the fourth mask to the lines in the −45° direction.

Suppose all the masks are applied to an image and the responses computed are denoted as R_1, R_2, R_3, and R_4. If at a certain point in the image $|R_i| > |R_j|$ for all $j \neq i$, then the point is more likely to be associated with the line in the direction of mask i. For example, if a point in the image where $|R_1| > |R_j|$ for $j = 2$, 3, and 4 then that particular point is more likely to be associated with a horizontal line.

6.4 EDGE DETECTION

In image processing, the points and line detection are seldom used. In most of the practical image applications edge detection plays a vital role and the concept involved in the edge detection is illustrated in this section with the help of the image shown in Figure 6.4.

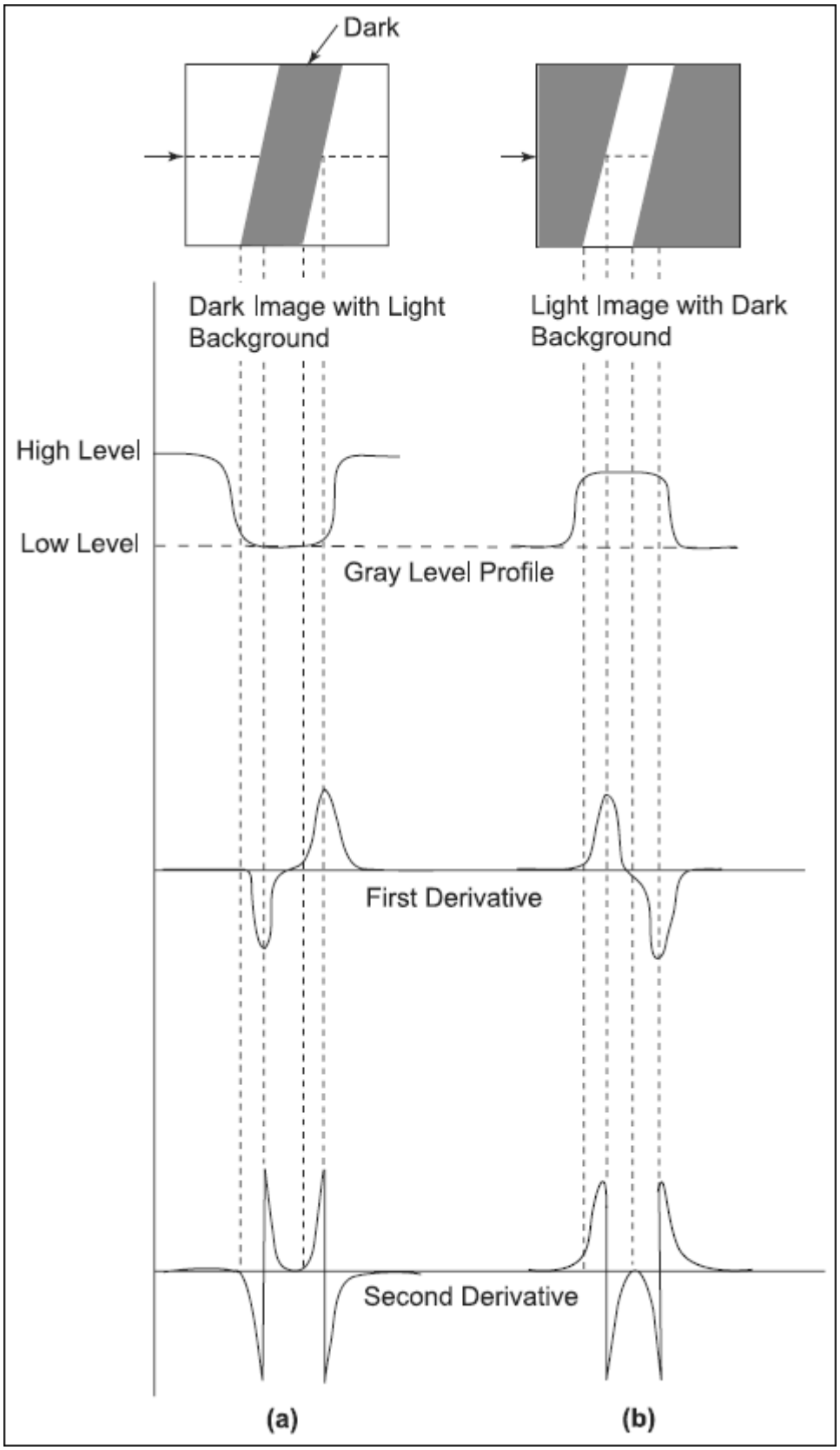


FIGURE 6.4 Edge detection. (a) The dark object on a light background with its derivatives (b) The bright object on the dark background with its derivatives

An edge is a boundary between two regions with relatively distinct gray level properties. Consider the image shown in the Figure 6.4(a) consisting of a dark object in a light background.

The gray level profile along the horizontal line of the image corresponding to the location shown by the arrow line is also given in the Figure 6.4(a).

Edge: An edge is a boundary between two regions with relatively distinct gray level properties.

The first derivative of the gray level profile is negative at the leading edge of the transition, positive at the trailing edge, and zero in the areas of constant gray levels. The second derivative is negative for that part of transition associated with the light side of the edge, positive for that part of the transition associated with the dark side of the edge, and zero for pixels lying exactly on edges. By analyzing the first derivative and second derivative of the image profile corresponding to a horizontal line, the following inference is obtained.

The magnitude of the first derivative is used to detect the presence of an edge in the image and the sign of the second derivative is used to determine whether the edge pixel lies on the dark side or light side of an edge. For example, if the second derivative is positive it shows that the corresponding pixel lies on the dark side of the edge and vice versa.

The second derivative has a zero crossing at the midpoint of the transition in gray level. The first derivative and the second derivative at any point in an image are obtained by using the magnitude of the gradient at that point and Laplacian operator, respectively. The detailed discussion of the gradient and Laplacian operator is given in the following sections.

6.4.1 Gradient Operators

The gradient of an image $f(x, y)$ at the location (x, y) is given by the vector

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \qquad (6.2)$$

The gradient vector points in the direction of the maximum rate of change of f at (x, y). In the edge detection we employ the magnitude of the gradient vector and it is denoted as

$$\nabla f = \text{mag}(\nabla f) = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}} \qquad (6.3)$$

To reduce the computational complexity the magnitude of the gradient vector can be approximated as given in equation (6.4).

$$\Delta f = |G_x| + |G_y| \qquad (6.4)$$

The direction of the gradient vector is another important quantity and is given in equation (6.5).

$$\alpha(x, y) = \tan^{-1}\left[\frac{G_y}{G_x}\right] \qquad (6.5)$$

where the angle is measured with respect to *x*-axis.

The computation of the gradient of an image is obtained from the partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ at every pixel in the image. It is always possible to implement the derivatives in digital form in different ways. One of the equivalent digital forms for the gradient is given by Sobel operators and they are given by the following equations.

$$G_x = (P_7 + 2P_8 + P_9) - (P_1 + 2P_2 + P_3) \qquad (6.6)$$

and

$$G_y = (P_3 + 2P_6 + P_9) - (P_1 + 2P_4 + P_7) \qquad (6.7)$$

where P_1 to P_9 are pixel values in a subimage as shown in Figure 6.5(a).

The equations (6.6) and (6.7) can be represented by two 3 × 3 masks as given in Figure 6.5(b) and (c).

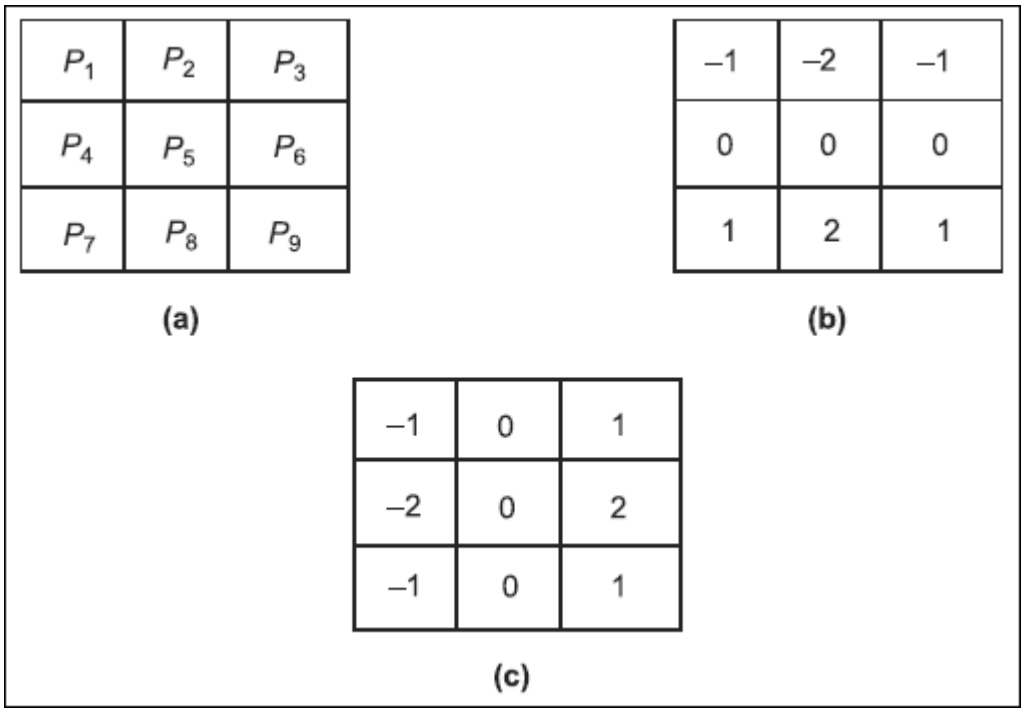


FIGURE 6.5 Sobel masks. (a) Sub image (b) Sobel mask for horizontal direction (c) Sobel mask for vertical direction

The mask in Figure 6.5(b) is used to compute G_x at the center point of the 3 × 3 region and mask in Figure 6.5(c) is used to compute G_y. The other mask called Prewitts can also be used to compute the gradient G_x and G_y as shown in Figure 6.6.

The following two equations give the computations of G_x and G_y components.

$$G_x = (P_7 + P_8 + P_9) - (P_1 + P_2 + P_3) \qquad (6.8)$$

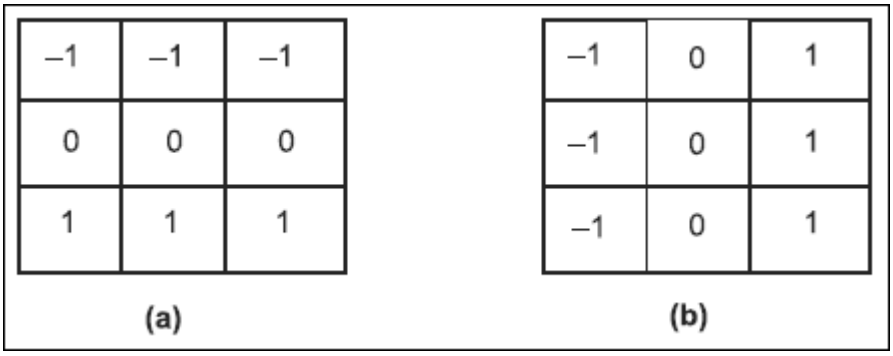


FIGURE 6.6 Prewitt's masks for horizontal and vertical components. (a) Mask to compute G_x (b) Mask to compute G_y

and

$$G_y = (P_3 + P_6 + P_9) - (P_1 + P_4 + P_7) \qquad (6.9)$$

The simplest possible way to implement the partial derivative at the center of the 3 × 3 mask is to use the Roberts, cross gradient operators.

$$G_x = P_9 - P_5 \qquad (6.10)$$

and

$$G_y = P_8 - P_6 \qquad (6.11)$$

The gradient image computed using Sobel operators is given in Figure 6.7.

Figure 6.7(a) shows the original image and Figure 6.7(b) shows the result of computing the modulus of G_x. This result gives the horizontal edges, which is perpendicular to the x-axis. Figure 6.7(c) gives the computation of gradient $|G_y|$, for vertical edges, which is perpendicular to the y-direction. Combining the above two components results in Figure 6.7(d), which is nothing but the gradient image.

6.4.2 Laplacian Operator

The Laplacian of the two-dimensional image $f(x, y)$ is the second-order derivative and is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (6.12)$$

For a 3 × 3 subimage the digital form equivalent to the Laplacian operator is given as

$$\nabla^2 f = 4P_5 - (P_2 + P_4 + P_6 + P_8) \qquad (6.13)$$

From equation (6.13) it is possible to define the digital Laplacian mask so that the coefficient associated with the center pixels should be positive and that associated with the outer pixels should be negative. Moreover, the sum of the coefficients should be zero. Such a spatial mask [corresponding to equation (6.13)] is shown in Figure 6.8.

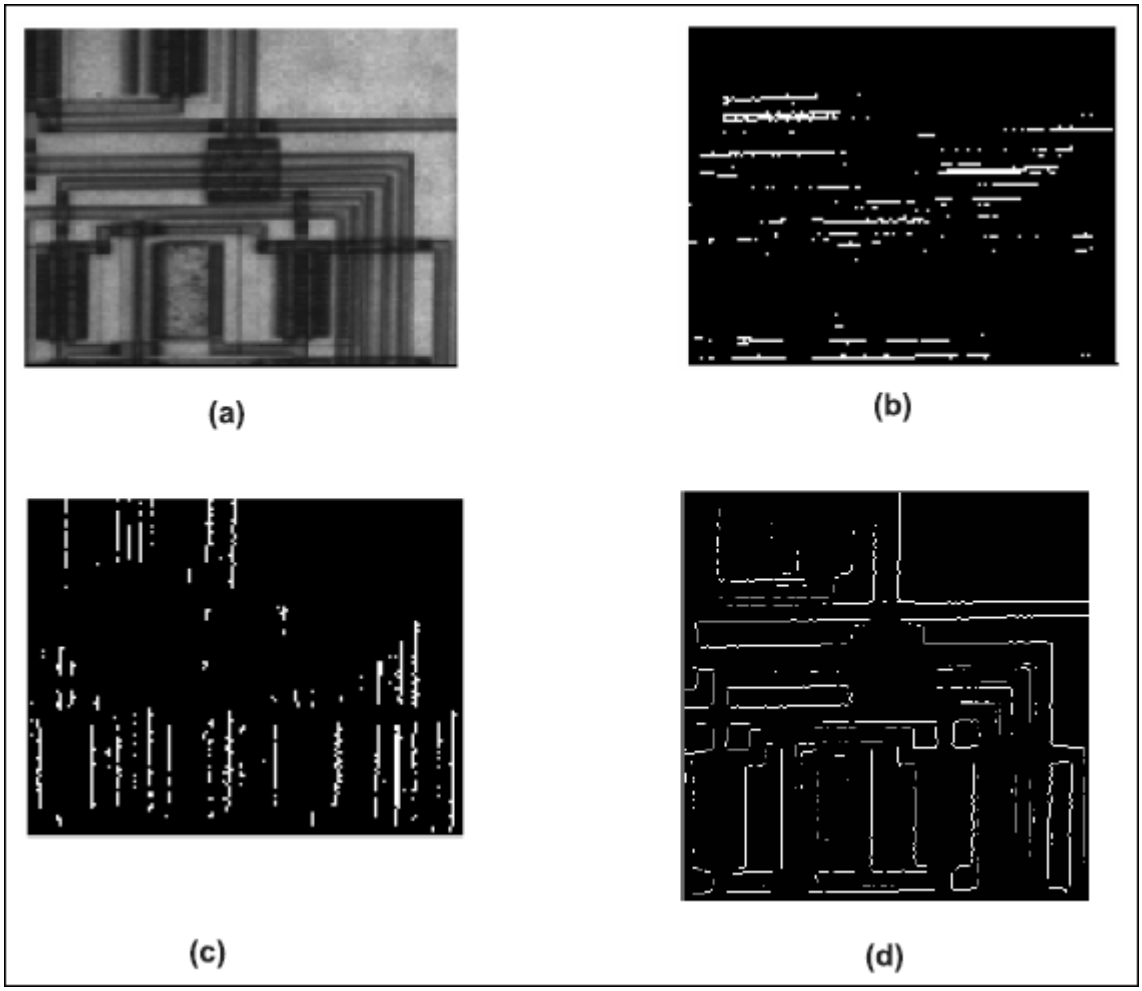


FIGURE 6.7 The gradient images using Sobel operators. (a) Original image (b) Image obtained using gradient G_x (c) Image obtained using G_y (d) Complete gradient image ($G_x + G_y$)

The Laplacian response is sensitive to noise and is rarely used for edge detection.

0	–1	0
–1	4	–1
0	–1	0

FIGURE 6.8 The mask used to compute Laplacian

6.5 EDGE LINKING AND BOUNDARY DETECTION

In practice, the set of pixels detected by the gradient operators do not form a complete boundary due to noise, non-uniform illumination, and other effects. Thus a linking and other boundary detection procedures to assemble the edge pixels into meaningful boundary follow the

edge detection algorithm. There are a number of techniques available for this purpose and they are

1. Local processing
2. Global processing using Hough transform
3. Graph theoretic approach
4. Thresholding
5. Region growing and
6. Region splitting and merging.

The fourthcoming section explains the local processing technique used to get the complete boundary of the objects in the given image.

6.5.1 Local Processing

The edge pixels are determined using the gradient operators. If we closely analyze the boundary, constructed by the edge pixels one may come across small openings or gaps. Thus the boundary is not completely defined by the edge pixels. In order to fill the gaps or openings at each pixel, we also consider the characteristics of pixels in a small neighborhood (3 × 3 or 5 × 5). All the neighborhood pixels which are similar to this boundary pixel are linked.

Two important properties used for checking the similarity of the neighborhood pixels with respect to the edge pixels are

1. The strength of the gradient operator response to produce the edge pixel
2. The direction of the gradient.

Now let us consider an edge pixel with coordinates (x, y) and a neighborhood pixel at the coordinate (x', y'). Then the neighborhood pixel (x', y') is similar to the edge pixel (x, y) if

$$|\nabla f(x, y) - \nabla f(x', y')| \leq T \qquad (6.14)$$

where T is a non-negative threshold value. Then neighborhood pixel (x', y') with respect to the edge pixel at (x, y) has an angle similar to the edge pixel if

$$|\propto (x, y) - \propto (x', y')| < A \qquad (6.15)$$

where

$$\propto (x, y) = \tan^{-1} \frac{G_y}{G_x} \qquad (6.16)$$

and A is an angle of threshold.

A neighborhood pixel (x', y') is linked to the edge pixel (x, y) if both magnitude and direction given in equations (6.14) and (6.15) are satisfied.

This process is repeated for every edge pixel location and a record must be kept for the linked neighborhood pixels. This procedure is applied to the image shown in Figure 6.9(a).

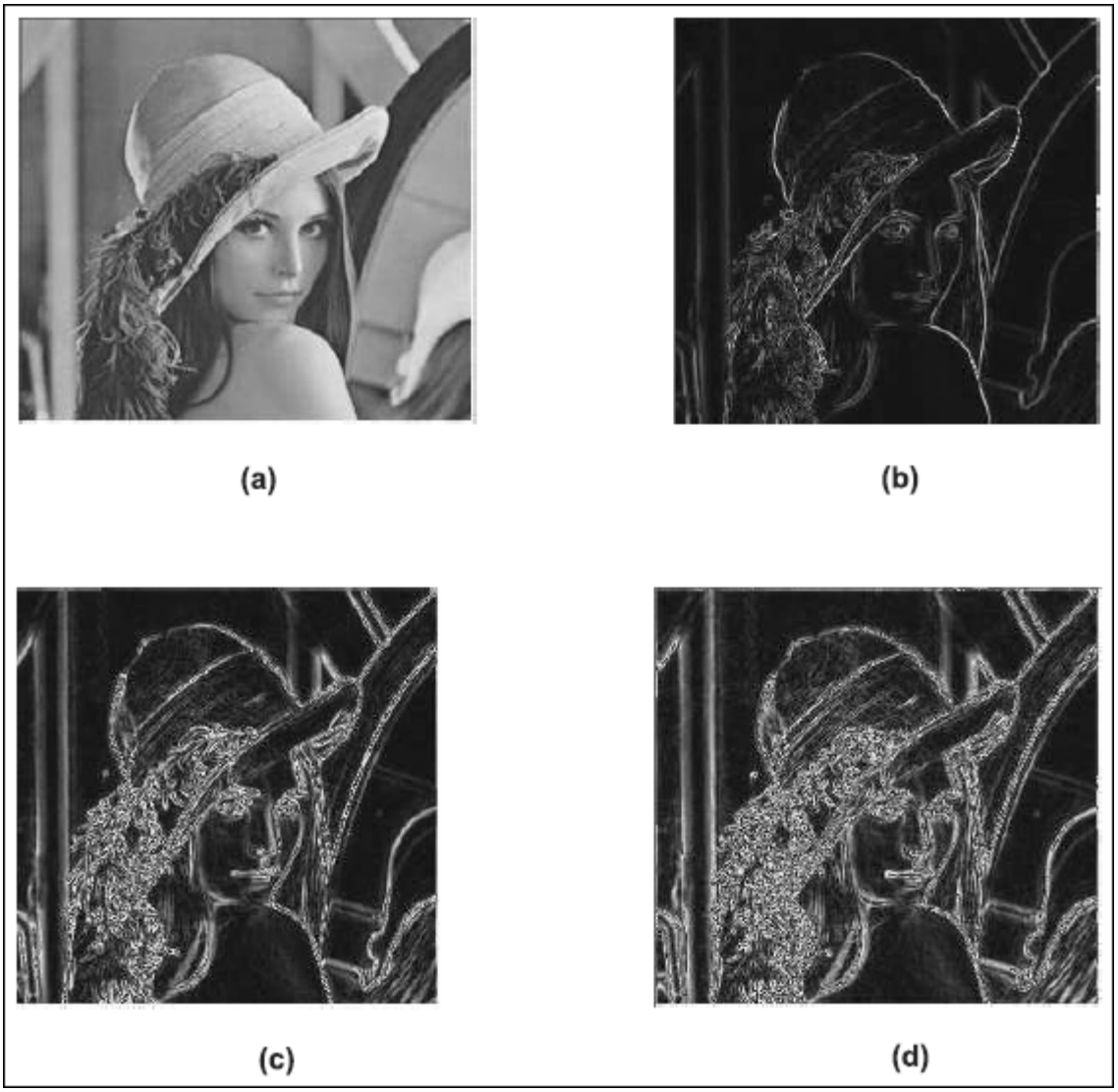


FIGURE 6.9 (a) Input image (b) *Gx* component (c) *Gy* component (d) Result of edge linking

Figure 6.9(b) and (c) shows the components of Sobel operators, Figure 6.9(d) shows the result of linking all the points that had a gradient difference less than 25° and the difference of the gradient direction not more than 15°.

6.5.2 Global Processing using Graph Theoretic Approach

In this section, a global approach for edge detection and linking based on representing the edge elements in the form of a graph and searching the graph for the low-cost path that correspond to the significant edges are discussed.

This graph approach performs well in the presence of noise. Before we explain the graph approach in detail, we introduce some basic definitions. A graph can be represented as

$$G = (N, V) \qquad (6.17)$$

where N is a set of non-empty nodes and V is a set of unordered pairs of nodes in N. Each pair $(N_i, N_j)$of V is called an *edge.* A graph in which the edges are directed is called

a *directed graph*. If an edge is directed from node N_i to N_j then the node N_i is called a parent of the child node N_j. The process of identifying the successor of a node is called expansion of the node. In a graph, the levels are also defined and root node or the start node is node at level 0. The nodes in the least level are called goal nodes. The cost associated with the edge between two nodes n_i and n_j is denoted as $C(n_i, n_j)$. The sequence of nodes $n_1, n_2, \ldots, n_k$ with each node n_i being the child of the node n_{i-1} is called a path from $n_1$to n_k. Then the cost of the path is given in equation (6.18).

$$\text{Path cost} = \sum_{i=1}^{k-1} C(n_i, n_{i+1}) \qquad (6.18)$$

An edge element is defined as the boundary between two pixels p and q such that p and q are four neighbors as given in Figure 6.10.

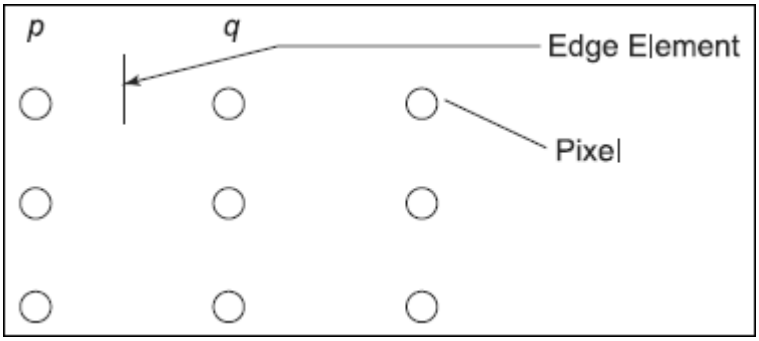


FIGURE 6.10 An edge element

The edge elements are identified by the (x, y) coordinates of p and q. In other words the edge element between p and q is given by the pairs (x_p, y_p) and (x_q, y_q), which define the edge sequence of connected edge elements. Now let us consider a 3 × 3 image and apply the concepts discussed so far to detect an edge. The 3 × 3 image is shown in Figure 6.11.

The outer numbers are pixel coordinates, the numbers in the bracket represent gray level values. Each edge element between pixels p and q is associated with the cost, as given in equation (6.19).

$$C(p, q) = H - [f(p) - f(q)] \qquad (6.19)$$

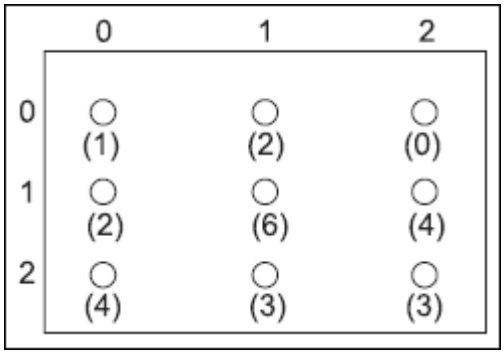


FIGURE 6.11 A 3 × 3 image

where H is the highest gray level in the image and $f(p)$ and $f(q)$ are the gray level values of pixels p and q. For the image 3 × 3, shown in Figure 6.11 the graph is drawn and depicted in Figure 6.12. Each node in this graph corresponds to an edge element. There exists an arc between two nodes if the corresponding edge elements are considered in succession. The cost of each edge element is computed and shown against the edge leading into it and the goal nodes are shown as shaded rectangles. Let us assume that the edge starts in the top row and terminates in the last row. Therefore, the possible start edge elements are [(0,0), (0,1)] and [(0,1), (0, 2)].

The terminating edge elements are [(2,0), (2,1)] or [(2,1), (2,2)]. The minimum cost path computed is indicated in the graph by means of a thick line. The edge with minimum cost is shown in Figure 6.12(b). In Figure 6.12(b), the node denoted by the pair of pixels (0, 0) and (0, 1) has a cost 7, denoted on the arc terminating on it. The cost is computed using equation (6.19). The image has a maximum gray level of 6 and pixels under consideration have gray level values of 1 and 2. Therefore, the cost is computed as

$$C(p,q) = 6 - (1 - 2)$$
$$= 7$$

Similar computations are used for all the pairs of pixels and the final graph is drawn as in Figure 6.12.

In general, finding a minimum cost path is not trivial and in order to reduce the search time an effective heuristic procedure is used. The steps involved in the heuristic procedure are explained as follows.

Let S be the start node and pass through an intermediate node n to reach the goal node. Let $R(n)$ be the estimate of the cost of the minimum cost path from start node to goal node. Then $R(n)$ can be given by the expression

$$R(n) = G(n) + H(n) \qquad (6.20)$$

where

$G(n)$ is the cost of the lowest cost path from S to n found so far.

$H(n)$ is obtained by using available heuristic information.

$H(n)$ is revised as and when we move from one node to another node on the basis of minimum cost path.

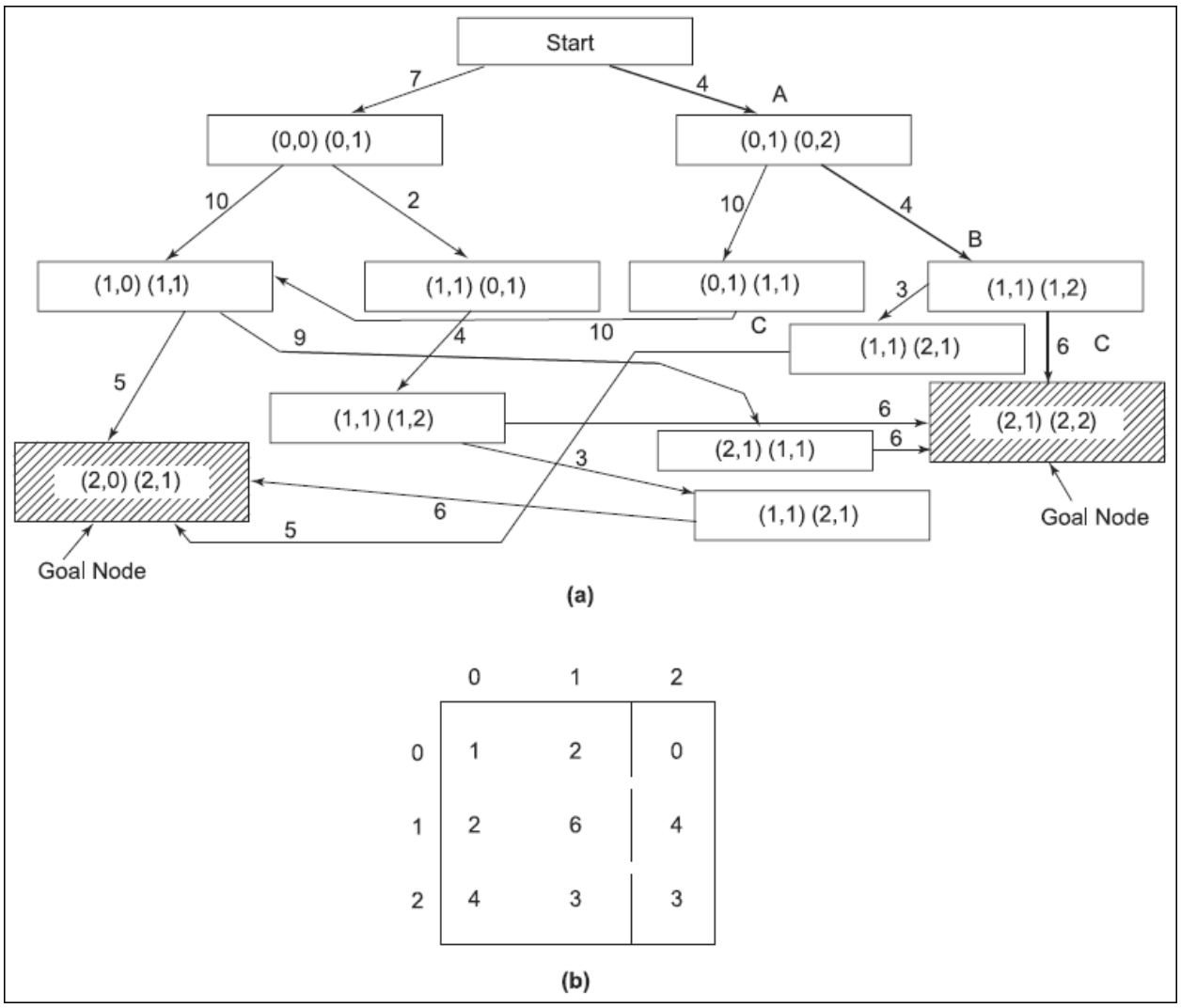


FIGURE 6.12 (a) Graph for finding an edge and the minimum cost path is as ABC (4 + 4 + 6 = 14) (b) Minimum cost edge

6.6 REGION-ORIENTED SEGMENTATION

In this section, we discuss segmentation technique that finds the region of interest directly. This technique uses the neighborhood pixel properties. The basic rules used in this approach are explained in Section 6.6.1.

6.6.1 Basic Rules for Segmentation

Let R be the entire image region. Then by using the segmentation algorithm the image region R is subdivided into 'n' subregions $R_1, R_2, \ldots, R_n$ such that

1. $\bigcup_{i=1}^{n} R_i = R$. This expression indicates that the segmentation process is complete in all respects.

2. R_i is a connected region for $i = 1, 2, \ldots, n$. This condition indicates that the points in the R_i must be connected.

3. $R_i \cap R_j = \Phi$ for all i and j, where $i \neq j$. This condition indicates that region must be disjoint.

4. $P(R_i)$ = True for i = 1,2,…, n. This means all the pixels in the region Ri have the same intensity.

5. $P(R_i \cup R_i)$ = False for $i \neq j$. This condition indicates that the regions R_i and R_j are different in the sense of the predicate P.

6.6.2 Region Growing by Pixel Aggregation

Region growing is a technique that groups the pixels or subregions into larger regions. The simplest of these approaches is pixel aggregation. In this approach a set of seed points are used as starting points and from this, regions grow by appending to each seed point those neighboring pixels that have similar properties.

Pixel Aggregation: A procedure used to group pixels with similar gray level properites in an image or a region in the image.

To illustrate this procedure, consider a small region of image 5 × 5 shown in Figure (6.13), where the gray levels are indicated by the numbers inside the cells. Let the pixel 2 and 7 with the co-ordinate (4, 2) and (3, 4), respectively be the two seeds. Using the two seeds as starting point, the regions are grown by applying the predicate property P. The predicate P to be used to include a pixel in either region is that the absolute difference between the gray level of that pixel and the gray level of the seed be less than a threshold value T.

Any pixel that satisfies this property simultaneously for both seeds is arbitrarily assigned to region R_1.

Figure 6.13(b) shows the result obtained using the threshold value T = 3. The segmentation results show two regions namely, R_1 and R_2 and they are denoted by a′s and b′s. It is also noted that the two regions formed are independent of the starting point. However, if we choose T = 8, the resulting region is shown in Figure 6.13(c).

The pixel aggregation technique used to grow the region is simple and suffers from two difficulties. The first one is the selection of initial seeds that properly represent the region of interest and the second is the selection of suitable properties for including points in the various regions during the region growing processes. The number of seed points selected can be based on the nature of the problem. Figure 6.14(a) shows an image with a single seed point. The threshold used for the region growing is the absolute difference in the gray level between the seed and a candidate point, not exceeding 10% of the difference between maximum and minimum gray level in the entire image.

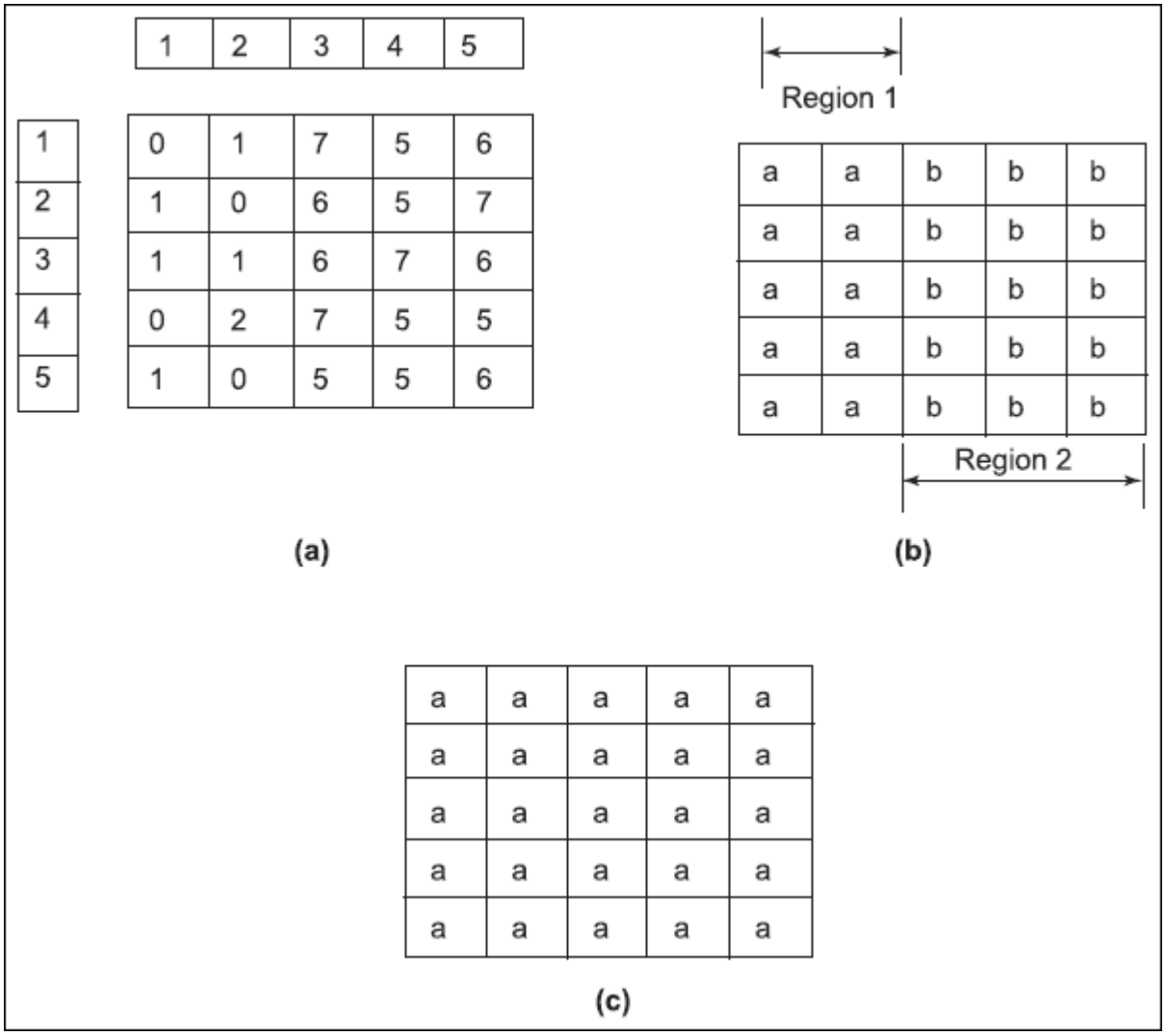


FIGURE 6.13 (a) A subimage with co-ordinates and gray values (b) Region growing with $T = 3$ (c) Region growing with $T = 8$

The property used to add a pixel into the region is an 8-connected one. Figure 6.14(b) shows the region in the early stages of region growing. Figure 6.14(c) shows the region in an intermediate stage of the region growing and Figure 6.14(d) shows the complete region grown by using this technique.

6.6.3 Region Splitting and Merging

In this technique an image is initially divided into a set of arbitrary subimages of disjoint regions, and then merge and/or split operations are carried out based on certain criteria. The split and merge algorithm is explained as follows.

Let R represent the entire image region and a predicate $P(Ri)$ is used to check the conditions. For any region R_i for $i = 1$ to 4, $P(R_i)$ = TRUE is applied. For any image of region R, subdivide the image into smaller and smaller regions so that for any region R_i, $P(R_i)$ = FALSE. This means if $P(R_i)$ = FALSE, divide the image into quadrants. If $P(R_i)$ is false for any quadrant,

subdivide that quadrant into subquadrant, and so on. This process can be conveniently represented by means of a quad tree shown in Figure 6.15(b). For a square region [shown in Figure 6.15(a)] the splitting and merging procedure is applied and the result is given by means of a quad tree as shown in Figure 6.15(b).

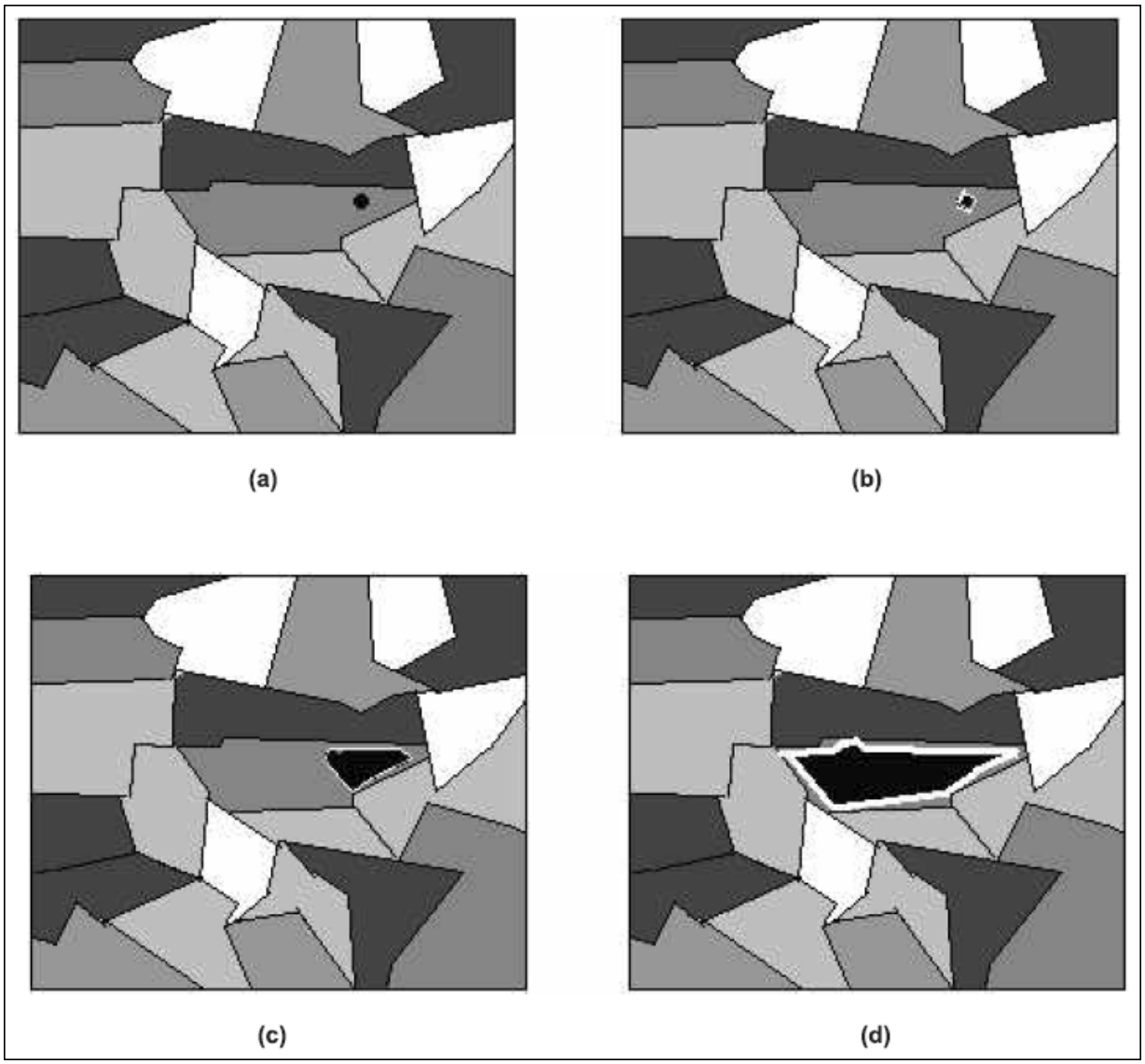


FIGURE 6.14 (a) An image with seed point (given as dark point) (b) Region growing after few iterations, (c) Intermediate stage of region growing (d) Final growth

In this approach splitting must be followed by merging operation.

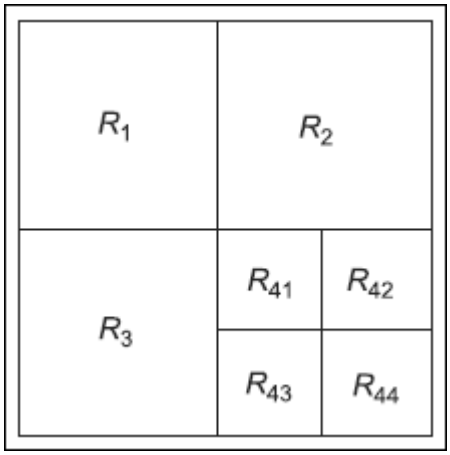


FIGURE 6.15(a) An image subdivided into quadrants

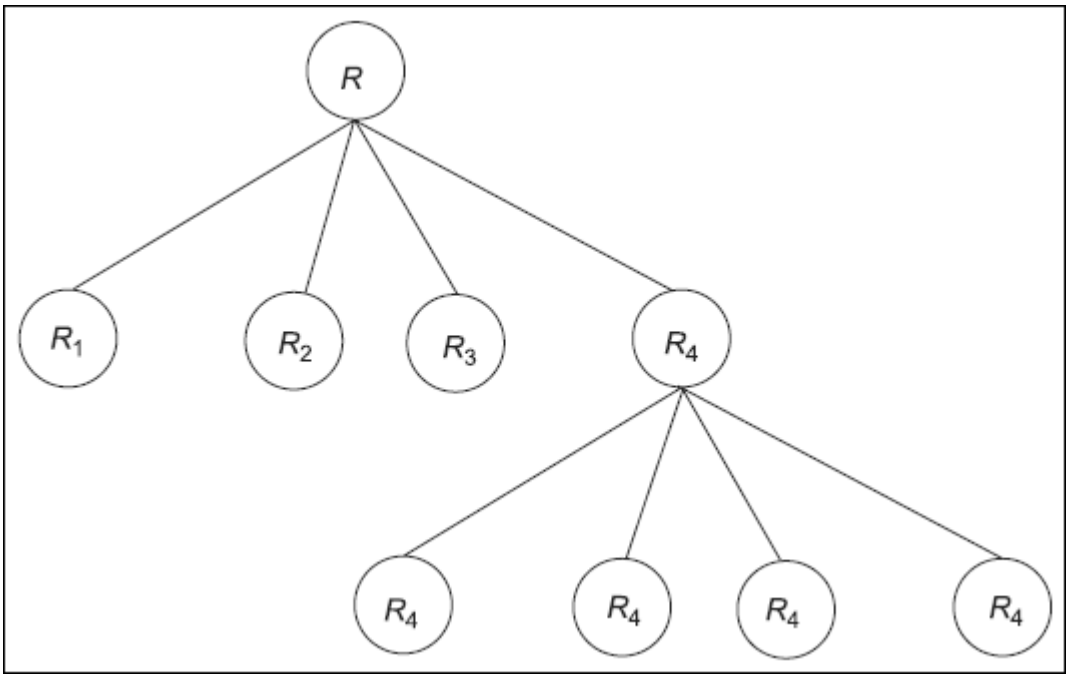


FIGURE 6.15(b) Quad tree

The procedure used can be given by the following three steps:

1. Split the given region R_i into four disjointed quadrants when $P(R_i)$ = False
2. Merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k)$ = True
3. Stop when no further merging or splitting is possible.

The split and merge algorithm is illustrated with the sample image shown in Figure 6.16(a).

For simplicity, a dark object in a white background is considered as shown in Figure 6.16(a). Then for the entire image region R, $P(R)$ = false and hence the image is split into four equal regions as shown in Figure 6.16(b).

In the second step all the four regions do not satisfy the predicate $P(R_i)$ where R_i is one of the quadrant. Hence each quadrant is further subdivided into four smaller regions as depicted in Figure 6.16(c).

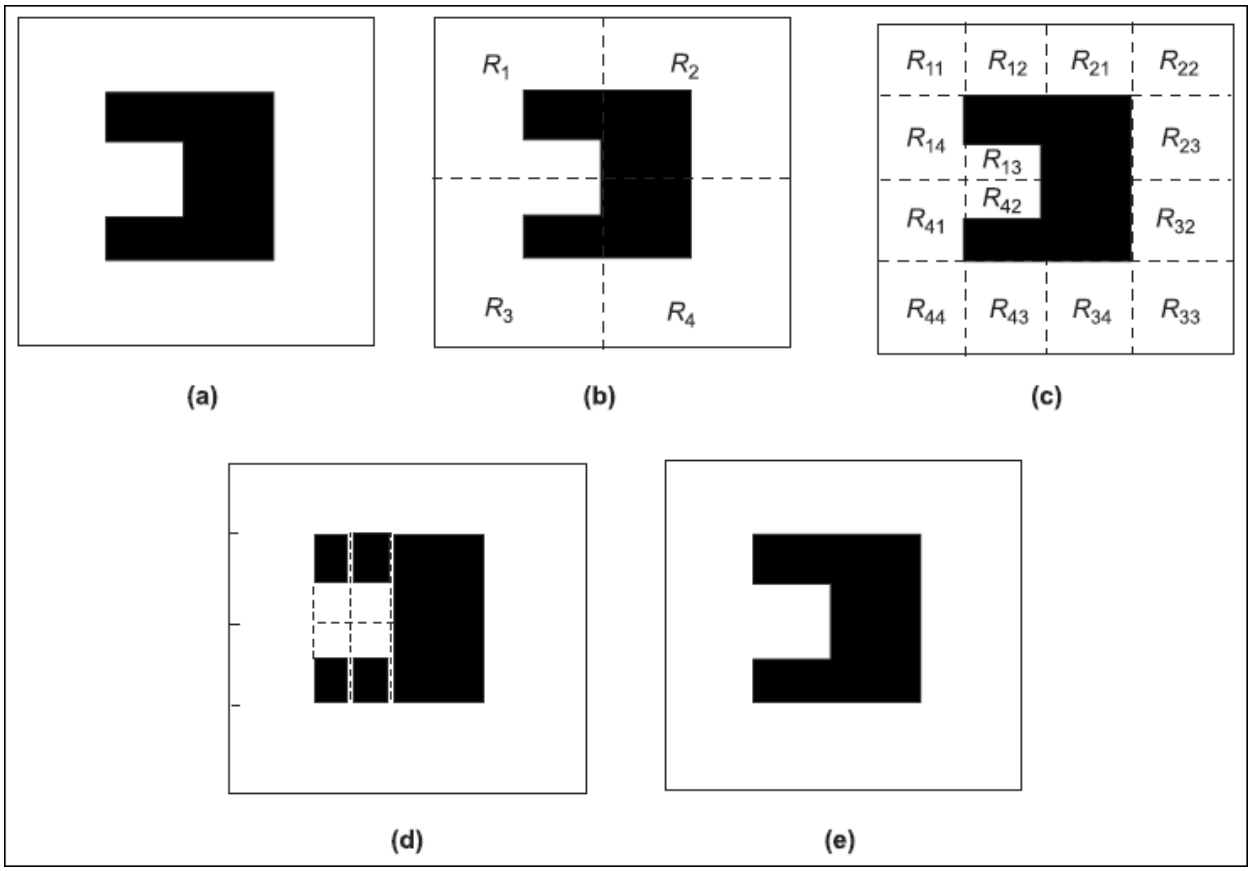


FIGURE 6.16 The steps of split and merge algorithm

At this point several regions can be merged with the exception of two subquadrants that include some points of the object. In Figure 6.16(c) the regions R_{11}, R_{12}, R_{21}, R_{22}, R_{23}, R_{33},…, R_{14} satisfy the predicate $P(R_i)$ = TRUE and all of them are merged to form a larger region whereas the regions R_{13} and R_{42} do not satisfy the predicate $P(R_i)$. These two subquadrants do not satisfy the predicate and hence they split further as in Figure 6.16(d). At this point all the regions satisfy the predicate P and merging the appropriate region from the last split operation gives the final segmented region as in Figure 6.16(e).

Thresholding: Thresholding is an operation in which a reference gray level is selected using histogram such that the object and background can be separated.

6.7 SEGMENTATION USING THRESHOLD

Thresholding is one of the most important techniques used for image segmentation. In this section we discuss the various thresholding techniques for image segmentation. The merits and demerits of various techniques are also discussed.

6.7.1 Fundamental Concepts

The histogram of an image $f(x, y)$ consisting of a light object on the dark background is shown in Figure 6.17(a). This histogram consists of two dominant regions, one for the object and the other for the background. For such an image it is easy to select a threshold T that separates

the object and background region. Then for any point (x, y), $f(x, y) > T$ is called an object point, otherwise the point is called a background point. A general case of this approach is shown in Figure 6.17(b).

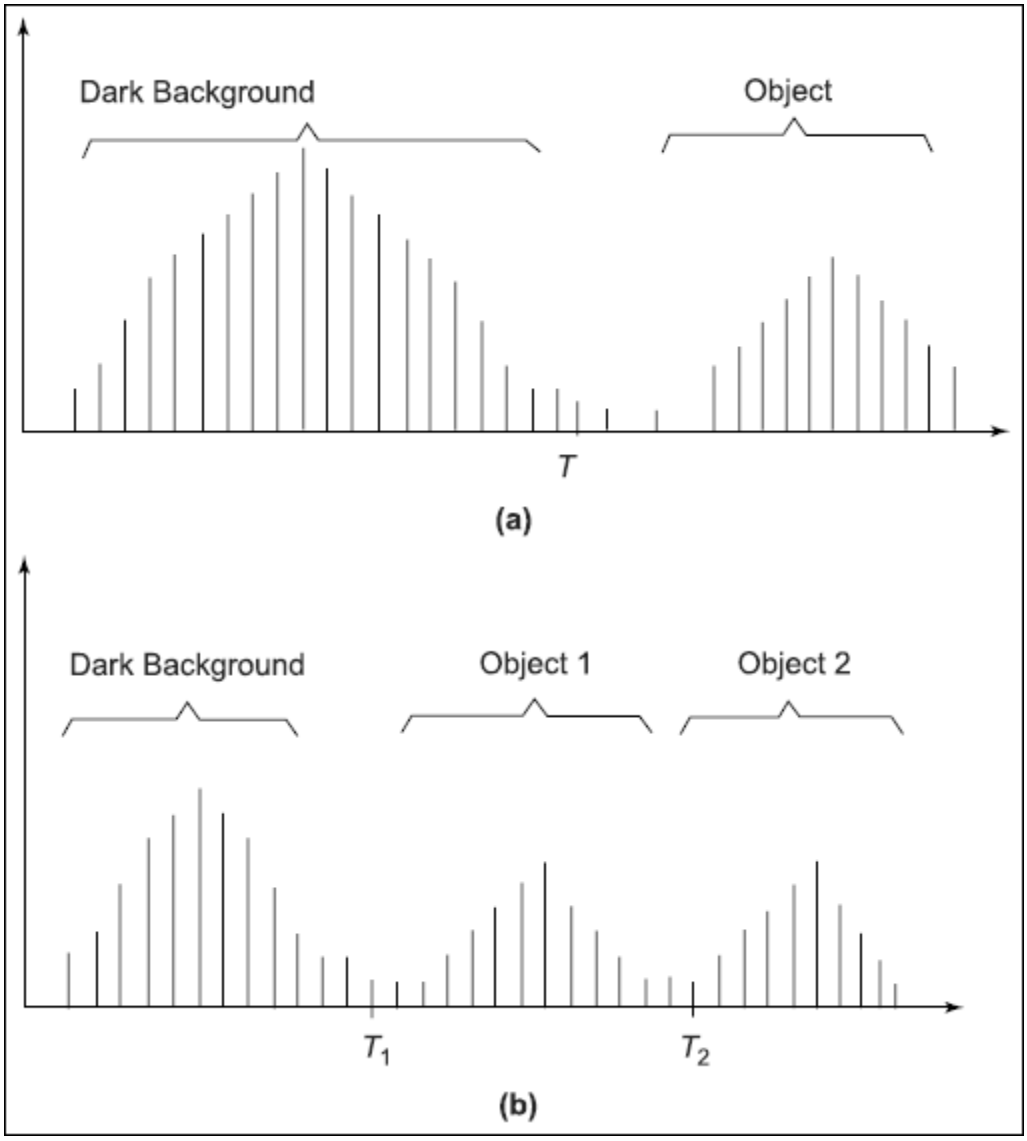


FIGURE 6.17 (a) Histogram of an image consisting of dark background and a light object (b) Histogram for two objects in a dark background

Figure 6.17(b) has three dominant regions that characterize the histogram of the given image. This histogram corresponds to two different light objects on a dark background. From the histogram it is possible to select two different threshold values T_1 and T_2, respectively. Then a point (x, y) belongs to the first object if $T_1 < f(x, y) \leq T_2$, or it belongs to the second object if $f(x, y) > T_2$ and to the background if $f(x, y) \leq T_1$. Usually this kind of thresholding is called multilevel thresholding and is less reliable than its single threshold counterpart.

The reason for this is, that it is difficult to locate multiple thresholds in a given histogram for a real image. The thresholding technique can be put into three different types based on the function *T* and its associated parameters as given in equation (6.21).

$$T = T[(x, y), p(x, y), f(x, y)] \qquad (6.21)$$

where *f*(*x*, *y*) is the gray level at the point (*x*, *y*) and *p(x, y)* denotes some local property at that point (e.g. the average gray level of neighborhood center on (*x*, *y*)). The threshold image *g(x, y)* is given in equation (6.22).

$$G(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases} \qquad (6.22)$$

In the thresholded image the pixel labeled 1 corresponds to the object, whereas pixels labeled 0 corresponds to the background. When the threshold value *T* depends only on *f*(*x*, *y*) the threshold technique is called *global*. If *T* depends on both *f*(*x*, *y*) and *p*(*x*, *y*), the threshold is called *local*. If *T* depends on all the three parameters, that is, the coordinates (*x*, *y*), local property *p(x, y)*, and *f*(*x*, *y*) then the threshold is called *dynamic*.

6.7.2 Optimal Thresholding

Let us assume that an image has only two principal brightness regions. Let *p*(*z*) be the probability density function (histogram) of the gray level values in the image. The overall density function *p*(*z*) is the sum of two densities, one for the light and the other for the dark regions in the image. Further, the mixture parameters are proportional to the area of the picture of each brightness. It is possible to determine the optimal threshold for segmenting the image into two brightness regions if the form of the density function is known. Suppose an image contains two brightness values, the overall density function can be given by the equation (6.23).

$$P(z) = P_1 \times P_1(z_1) + P_2 \times P_2(z) \qquad (6.23)$$

In the case of Gaussian distribution the *P*(*z*) can be given as

$$P(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{(z-\mu_1)^2}{2\sigma_1^2}\right] + \frac{P_2}{\sqrt{2\pi}\sigma_2} \exp\left[-\frac{(z-\mu_2)^2}{2\sigma_2^2}\right] \qquad (6.24)$$

where μ_1 and μ_2 are the mean values of the two brightness levels, $\sigma_1$and σ_2 are the standard deviations of the means. P_1 and P_2 are a priori probabilities of the two gray levels. The constraint $P_1 + P_2 = 1$ must be satisfied so that these are the only five unknown parameters.

Suppose the dark region corresponds to the background and the bright region corresponds to an object, then a threshold *T* may be defined so that all the pixels with gray level below *T* are considered as background points and all pixels with gray levels above *T* are considered as object points.

The probability of error in classifying an object point as background point is given by

$$E_1(T) = \int_{-\infty}^{T} P_2(z)\,dz \qquad (6.25)$$

Similarly, the probability of error in classifying the background point as an object point is given by

$$E_2(T) = \int_{T}^{\infty} P_1(z)\,dz \qquad (6.26)$$

Therefore, the overall probability of error is given by

$$E(T) = P_2(E_1(T)) + P_1(E_2(T))$$

To find a threshold value for which the error is minimum, requires differentiating $E(T)$ with respect to T and equating the result to zero.

Thus

$$P_1 P_1(T) = P_2(P_2(T)) \qquad (6.27)$$

Applying the above result to Gaussian density, taking logarithm and simplifying gives the quadratic equation

$$A \times T^2 + B \times T + C \qquad (6.28)$$

where

$$A = \sigma_1^2 - \sigma_2^2$$

$$B = 2 \times (\mu_1\sigma_2^2 - \mu_2\sigma_1^2)$$

$$C = \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2\mu_2^2 \quad \ln\left[\frac{\sigma_2 P_1}{\sigma_1 P_2}\right]$$

If the variances are equal $\sigma^2 = {\sigma_1}^2 = {\sigma_2}^2$ a single threshold is sufficient.

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 + \mu_2} \ln\left(\frac{P_2}{P_1}\right) \qquad (6.29)$$

If the power probabilities are equal $P_1 = P_2$,the nth optimal threshold is the average of the means, that is,

$$T = \frac{\mu_1 + \mu_2}{2} \qquad (6.30)$$

Thus the optimum threshold is obtained using equation (6.30).

6.7.3 Threshold Selection Based on Boundary Characteristics

The threshold selection depends on the mode peaks in a given histogram. The chances of selecting a good threshold should be considerably enhanced if the histogram peaks are tall,

narrow, symmetrical, and separated by deep valleys. One way of improving the histogram is by considering only those pixels that lie on or near the boundary between the objects and background. By doing so, the appearance of the histogram will be less dependent on the relative sizes of the object and background. This will inturn reduce ambiguity and lead to improvement in threshold selection.

For example, consider an image of large background area of constant intensity with a small object then the histogram of such an image will be a large peak for background region and small peak for object. On the other hand, if we consider only the pixels on or near the boundary between the object and background, the resulting histogram will contain peaks of approximately the same height. In addition, the probability that a pixel lies on an object will be approximately equal to that on the background, hence improving the symmetry of the histogram peaks. Finally, selecting the pixels that satisfy the gradient and Laplacian operators will give rise to histogram peaks with deep valley between the peaks.

A three level image is formed using the gradient ∇f at any point (x, y) in the image and the Laplacian $\nabla^2 f$ at the image at the same point. The three level image is denoted by $U(x, y)$ and is given in equation (6.31).

$$U(x,y) = \begin{cases} 0 & \text{if} \quad \nabla f < T \\ + & \text{if} \quad \nabla f \geq T \quad \text{and} \quad \nabla^2 f \geq 0 \\ - & \text{if} \quad \nabla f \geq T \quad \text{and} \quad \nabla^2 f < 0 \end{cases} \qquad (6.31)$$

where the symbols 0,+, and − represent any three distinct gray levels, T is the threshold and the gradient and Laplacian are computed at every point in the image $f(x, y)$.

For an image with a dark object on a light background, the meaning for the labels 0, +, − can be given as follows:

In the three level image $U(x, y)$ the label '0'represents all the pixels that are not on the edge, the label '+'represents all the pixels on the dark side of an edge and the label '−'represents all the pixels on the light side of an edge. From the three level image it is possible to generate a segmented binary image in which 1′s correspond to the object of interest and 0′s correspond to the background. The transition from light background to dark object is represented by the occurrence of the label '−' followed by the label '+' in the image $U(x, y)$. The interior of the object consists of the labels either '0' or '+'. The transition from the object back to the background is represented by the occurrence of the label '+' followed by the label '−'. Thus when we scan the image either in the horizontal direction or vertical direction, the string of labels will be as follows:

$$(\cdots)(-,+)(0\text{ or }+)(+,-)(\cdots)$$

where (…) represents any combination of '+', '−', and '0'. The innermost parenthesis '0' or '+' corresponds to the object points and they are labeled as 1. The remaining pixels along the scan line are labeled 0. A sample image for a blank cheque is shown in Figure 6.18(a).

Figure 6.18(b) shows the histogram as a function of gradient values for pixels with gradients greater than 8. This histogram has two dominant modes that are symmetrical nearly of the same height and are separated by a distinct valley.

Figure 6.18(c) gives the segmented image obtained by using equation (6.31) with T at or near the midpoint of the valley (T = 19).

UNIT – V

Image Coding & Compression: FidelityCriteria – Encoding Process – Transform Encoding – Redundancies and their removal methods – Image compression models and methods – Source coder and decoder – Error free compression – Lossy compression.

FIDELITY CRITERION

The removal of psychovisually redundant data results in a loss of real or quantitative visual information. Because information of interest may be lost, a repeatable or reproducible means of quantifying the nature and extent of information loss is highly desirable. Two general classes of criteria are used as the basis for such an assessment:

(iii) Objective fidelity criteria and

(iv) Subjective fidelity criteria.

When the level of information loss can be expressed as a function of the original or input image and the compressed and subsequently decompressed output image, it is said to be based on an objective fidelity criterion. A good example is the root-mean-square (rms) error between an input and output image. Let f(x, y) represent an input image and let f(x, y) denote an estimate or approximation of f(x, y) that results from compressing and subsequently decompressing the input. For any value of x and y, the error e(x, y) between f (x, y) and f^ (x, y) can be defined as

$$e(x, y) = \hat{f}(x, y) - f(x, y)$$

so that the total error between the two images is

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]$$

where the images are of size M X N. The root-mean-square error, e_{rms}, between f(x, y) and f^(x, y) then is the square root of the squared error averaged over the M X N array, or

$$e_{rms} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2 \right]^{1/2}$$

A closely related objective fidelity criterion is the mean-square signal-to-noise ratio of the compressed-decompressed image. If f^ (x, y) is considered to be the sum of the original image f(x, y) and a noise signal e(x, y), the mean-square signal-to-noise ratio of the output image, denoted SNR_{rms}, is

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2}.$$

The rms value of the signal-to-noise ratio, denoted SNR_{rms}, is obtained by taking the square root of Eq. above.

Although objective fidelity criteria offer a simple and convenient mechanism for evaluating information loss, most decompressed images ultimately are viewed by humans. Consequently, measuring image quality by the subjective evaluations of a human observer often is more appropriate. This can be accomplished by showing a "typical" decompressed image to an appropriate cross section of viewers and averaging their evaluations. The evaluations may be made using an absolute rating scale or by means of side-by-side comparisons of f(x, y) and f^(x, y).

IMAGE COMPRESSION MODELS

Fig. 3.1 shows, a compression system consists of two distinct structural blocks: an encoder and a decoder. An input image f(x, y) is fed into the encoder, which creates a set of symbols from the input data. After transmission over the channel, the encoded representation is fed to the decoder, where a reconstructed output image f^(x, y) is generated. In general, f^(x, y) may or may not be an exact replica of f(x, y). If it is, the system is error free or information preserving; if not, some level of distortion is present in the reconstructed image. Both the encoder and decoder shown in Fig. 3.1 consist of two relatively independent functions or subblocks. The encoder is made up of a source encoder, which removes input redundancies, and a channel encoder, which increases the noise immunity of the source encoder's output. As would be expected, the decoder includes a channel decoder followed by a source decoder. If the channel between the encoder and decoder is noise free (not prone to error), the channel encoder and decoder are omitted, and the general encoder and decoder become the source encoder and decoder, respectively.

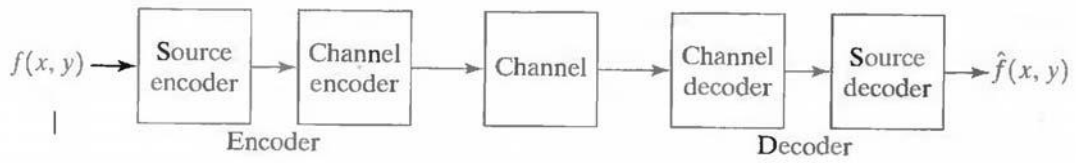
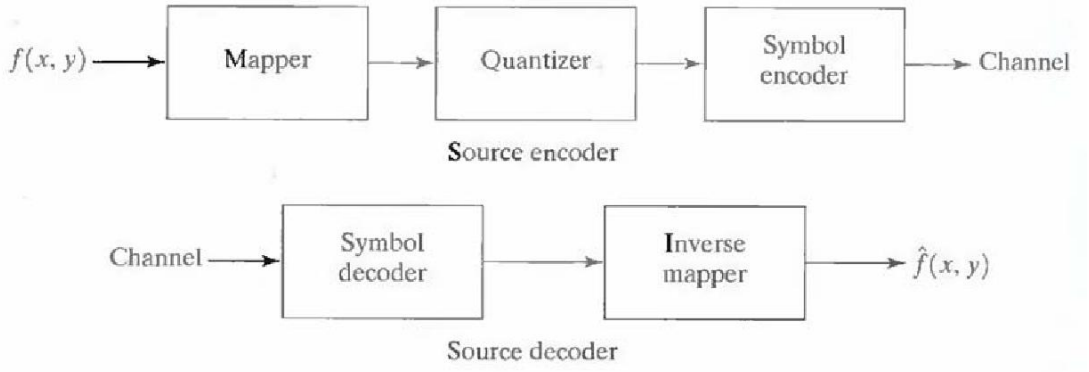


Fig.3.1 A general compression system model

The Source Encoder and Decoder:

The source encoder is responsible for reducing or eliminating any coding, interpixel, or psychovisual redundancies in the input image. The specific application and associated fidelity requirements dictate the best encoding approach to use in any given situation. Normally, the

approach can be modeled by a series of three independent operations. As Fig. 3.2 (a) shows, each operation is designed to reduce one of the three redundancies. Figure 3.2 (b) depicts the corresponding source decoder. In the first stage of the source encoding process, the mapper transforms the input data into a (usually nonvisual) format designed to reduce interpixel redundancies in the input image. This operation generally is reversible and may or may not reduce directly the amount of data required to represent the image.



a
b

Fig.3.2 (a) Source encoder and (b) source decoder model

Run-length coding is an example of a mapping that directly results in data compression in this initial stage of the overall source encoding process. The representation of an image by a set of transform coefficients is an example of the opposite case. Here, the mapper transforms the image into an array of coefficients, making its interpixel redundancies more accessible for compression in later stages of the encoding process.

The second stage, or quantizer block in Fig. 3.2 (a), reduces the accuracy of the mapper's output in accordance with some preestablished fidelity criterion. This stage reduces the psychovisual redundancies of the input image. This operation is irreversible. Thus it must be omitted when error-free compression is desired.

In the third and final stage of the source encoding process, the symbol coder creates a fixed- or variable-length code to represent the quantizer output and maps the output in accordance with the code. The term symbol coder distinguishes this coding operation from the overall source encoding process. In most cases, a variable-length code is used to represent the mapped and quantized data set. It assigns the shortest code words to the most frequently occurring output values and thus reduces coding redundancy. The operation, of course, is reversible. Upon completion of the symbol coding step, the input image has been processed to remove each of the three redundancies.

Figure 3.2(a) shows the source encoding process as three successive operations, but all three operations are not necessarily included in every compression system. Recall, for example, that the quantizer must be omitted when error-free compression is desired. In addition, some compression techniques normally are modeled by merging blocks that are physically separate in Fig. 3.2(a). In the predictive compression systems, for instance, the mapper and quantizer are often represented by a single block, which simultaneously performs both operations.

The source decoder shown in Fig. 3.2(b) contains only two components: a symbol decoder and an inverse mapper. These blocks perform, in reverse order, the inverse operations of the source encoder's symbol encoder and mapper blocks. Because quantization results in irreversible information loss, an inverse quantizer block is not included in the general source decoder model shown in Fig. 3.2(b).

The Channel Encoder and Decoder:

The channel encoder and decoder play an important role in the overall encoding-decoding process when the channel of Fig. 3.1 is noisy or prone to error. They are designed to reduce the impact of channel noise by inserting a controlled form of redundancy into the source encoded data. As the output of the source encoder contains little redundancy, it would be highly sensitive to transmission noise without the addition of this "controlled redundancy." One of the most useful channel encoding techniques was devised by R. W. Hamming (Hamming [1950]). It is based on appending enough bits to the data being encoded to ensure that some minimum number of bits must change between valid code words. Hamming showed, for example, that if 3 bits of redundancy are added to a 4-bit word, so that the distance between any two valid code words is 3, all single-bit errors can be detected and corrected. (By appending additional bits of redundancy, multiple-bit errors can be detected and corrected.) The 7-bit Hamming (7, 4) code word $h_1, h_2, h_3 \ldots, h_6, h_7$ associated with a 4-bit binary number $b_3b_2b_1b_0$ is

$$
\begin{aligned}
h_1 &= b_3 \oplus b_2 \oplus b_0 \qquad & h_3 &= b_3 \\
h_2 &= b_3 \oplus b_1 \oplus b_0 \qquad & h_5 &= b_2 \\
h_4 &= b_2 \oplus b_1 \oplus b_0 \qquad & h_6 &= b_1 \\
& & h_7 &= b_0
\end{aligned}
$$

where denotes the exclusive OR operation. Note that bits h_1, h_2, and h_4 are even- parity bits for the bit fields $b_3\ b_2\ b_0$, $b_3b_1b_0$, and $b_2b_1b_0$, respectively. (Recall that a string of binary bits has even parity if the number of bits with a value of 1 is even.) To decode a Hamming encoded result, the channel decoder must check the encoded value for odd parity over the bit fields in which even parity was previously established. A single-bit error is indicated by a nonzero parity word $c_4c_2c_1$, where

$$
\begin{aligned}
c_1 &= h_1 \oplus h_3 \oplus h_5 \oplus h_7 \\
c_2 &= h_2 \oplus h_3 \oplus h_6 \oplus h_7 \\
c_4 &= h_4 \oplus h_5 \oplus h_6 \oplus h_7.
\end{aligned}
$$

If a nonzero value is found, the decoder simply complements the code word bit position indicated by the parity word. The decoded binary value is then extracted from the corrected code word as $h_3h_5h_6h_7$.

TRANSFORM CODING SYSTEM

4. Transform Coding:

All the predictive coding techniques operate directly on the pixels of an image and thus are spatial domain methods. In this coding, we consider compression techniques that are based on modifying the transform of an image. In transform coding, a reversible, linear transform (such as the Fourier transform) is used to map the image into a set of transform coefficients, which are then quantized and coded. For most natural images, a significant number of the coefficients have small magnitudes and

can be coarsely quantized (or discarded entirely) with little image distortion. A variety of transformations, including the discrete Fourier transform (DFT), can be used to transform the image data.

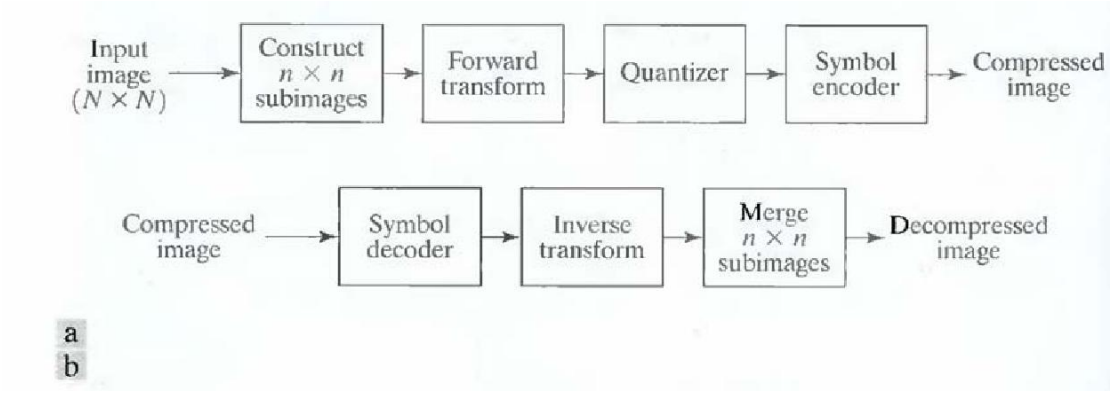


Fig. 10 A transform coding system: (a) encoder; (b) decoder.

Figure 10 shows a typical transform coding system. The decoder implements the inverse sequence of steps (with the exception of the quantization function) of the encoder, which performs four relatively straightforward operations: subimage decomposition, transformation, quantization, and coding. An N X N input image first is subdivided into subimages of size n X n, which are then transformed to generate $(N/n)^2$ subimage transform arrays, each of size n X n. The goal of the transformation process is to decorrelate the pixels of each subimage, or to pack as much information as possible into the smallest number of transform coefficients. The quantization stage then selectively eliminates or more coarsely quantizes the coefficients that carry the least information. These coefficients have the smallest impact on reconstructed subimage quality. The encoding process terminates by coding (normally using a variable-length code) the quantized coefficients. Any or all of the transform encoding steps can be adapted to local image content, called adaptive transform coding, or fixed for all subimages, called nonadaptive transform coding.

WAVELET CODING

Wavelet Coding:

The wavelet coding is based on the idea that the coefficients of a transform that decorrelates the pixels of an image can be coded more efficiently than the original pixels themselves. If the transform's basis functions—in this case wavelets—pack most of the important visual information into a small number of coefficients, the remaining coefficients can be quantized coarsely or truncated to zero with little image distortion.

Figure 11 shows a typical wavelet coding system. To encode a 2^J X 2^J image, an analyzing wavelet, Ø, and minimum decomposition level, J - P, are selected and used to compute the image's discrete wavelet transform. If the wavelet has a complimentary scaling function ö, the fast wavelet transform can be used. In either case, the computed transform converts a large portion of the original image to horizontal, vertical, and diagonal decomposition coefficients with zero mean and Laplacian-like distributions.

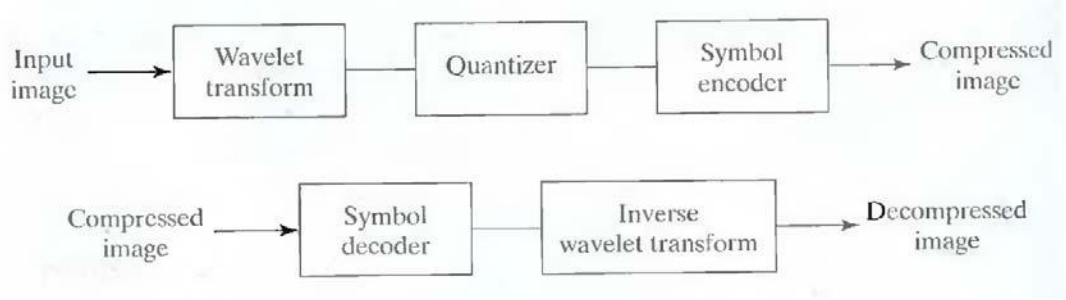


Fig.11 A wavelet coding system: (a) encoder; (b) decoder.

Since many of the computed coefficients carry little visual information, they can be quantized and coded to minimize intercoefficient and coding redundancy. Moreover, the quantization can be adapted to exploit any positional correlation across the P decomposition levels. One or more of the lossless coding methods, including run-length, Huffman, arithmetic, and bit-plane coding, can be incorporated into the final symbol coding step. Decoding is accomplished by inverting the encoding operations—with the exception of quantization, which cannot be reversed exactly.

The principal difference between the wavelet-based system and the transform coding system is the omission of the transform coder's subimage processing stages. Because wavelet transforms are both computationally efficient and inherently local (i.e., their basis functions are limited in duration), subdivision of the original image is unnecessary.

LOSSLESS PREDICTIVE CODING

Lossless Predictive Coding:

The error-free compression approach does not require decomposition of an image into a collection of bit planes. The approach, commonly referred to as lossless predictive coding, is based on eliminating the interpixel redundancies of closely spaced pixels by extracting and coding only the new information in each pixel. The new information of a pixel is defined as the difference between the actual and predicted value of that pixel.

Figure 8.1 shows the basic components of a lossless predictive coding system. The system consists of an encoder and a decoder, each containing an identical predictor. As each successive pixel of the input image, denoted f_n, is introduced to the encoder, the predictor generates the anticipated value of that pixel based on some number of past inputs. The output of the predictor is then rounded to the nearest integer, denoted $f\hat{}_n$ and used to form the difference or prediction error which is coded using a variable-length code (by the symbol encoder) to generate the next element of the compressed data stream.

$$e_n = f_n - \hat{f}_n,$$

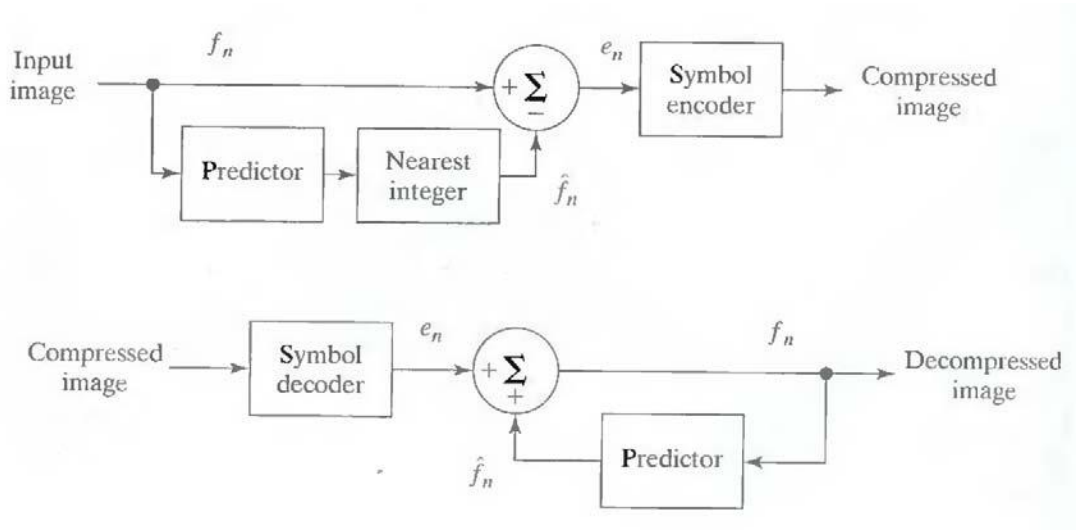


Fig.8.1 A lossless predictive coding model: (a) encoder; (b) decoder

The decoder of Fig. 8.1 (b) reconstructs en from the received variable-length code words and performs the inverse operation

$$f_n = e_n + \hat{f}_n.$$

Various local, global, and adaptive methods can be used to generate f^$_n$. In most cases, however, the prediction is formed by a linear combination of m previous pixels. That is,

$$\hat{f}_n = \text{round}\left[\sum_{i=1}^{m} \alpha_i f_{n-i}\right]$$

where m is the order of the linear predictor, round is a function used to denote the rounding or nearest integer operation, and the á$_i$, for i = 1,2,..., m are prediction coefficients. In raster scan applications, the subscript n indexes the predictor outputs in accordance with their time of occurrence. That is, f_n, f^$_n$ and e_n in Eqns. above could be replaced with the more explicit notation f (t), f^(t), and e (t), where t represents time. In other cases, n is used as an index on the spatial coordinates and/or frame number (in a time sequence of images) of an image. In 1-D linear predictive coding, for example, Eq. above can be written as

$$\hat{f}_n(x, y) = \text{round}\left[\sum_{i=1}^{m} \alpha_i f(x, y - i)\right]$$

where each subscripted variable is now expressed explicitly as a function of spatial coordinates x and y. The Eq. indicates that the 1-D linear prediction f(x, y) is a function of the previous pixels on the current line alone. In 2-D predictive coding, the prediction is a function of the previous pixels in a left-to-right, top-to-bottom scan of an image. In the 3-D case, it is based on these pixels and the previous pixels of preceding frames. Equation above cannot be evaluated for the first m pixels of

each line, so these pixels must be coded by using other means (such as a Huffman code) and considered as an overhead of the predictive coding process. A similar comment applies to the higher-dimensional cases.

LOSSY PREDICTIVE CODING

Lossy Predictive Coding:

In this type of coding, we add a quantizer to the lossless predictive model and examine the resulting trade-off between reconstruction accuracy and compression performance. As Fig.9 shows, the quantizer, which absorbs the nearest integer function of the error-free encoder, is inserted between the symbol encoder and the point at which the prediction error is formed. It maps the prediction error into a limited range of outputs, denoted $e\hat{}_n$ which establish the amount of compression and distortion associated with lossy predictive coding.

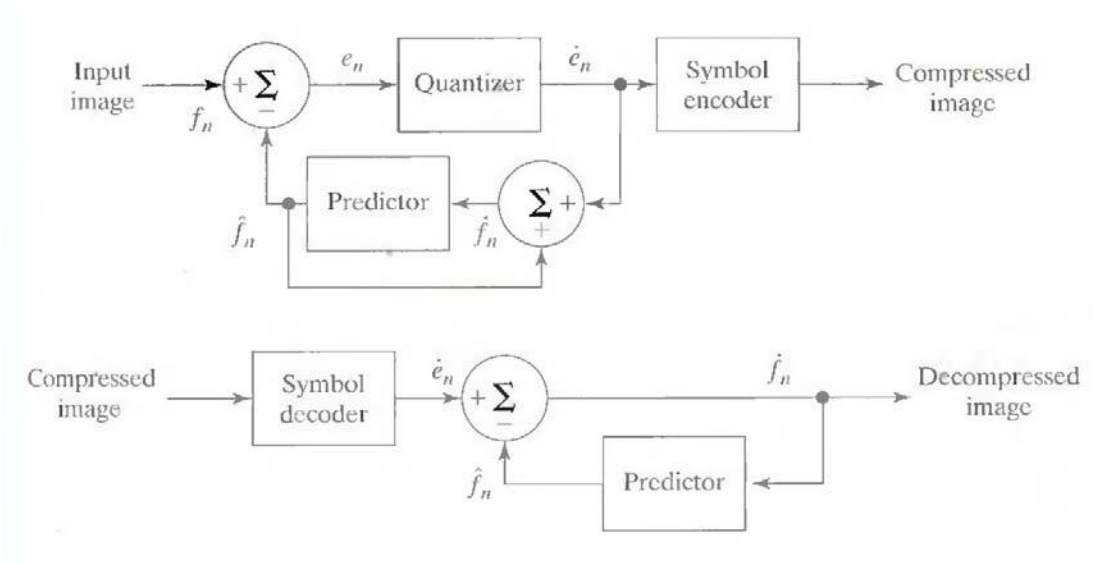


Fig. 9 A lossy predictive coding model: (a) encoder and (b) decoder.

In order to accommodate the insertion of the quantization step, the error-free encoder of figure must be altered so that the predictions generated by the encoder and decoder are equivalent. As Fig.9 (a) shows, this is accomplished by placing the lossy encoder's predictor within a feedback loop, where its input, denoted $f\ddot{y}_n$, is generated as a function of past predictions and the corresponding quantized errors. That is,

$$\dot{f}_n = \dot{e}_n + \hat{f}_n$$

This closed loop configuration prevents error buildup at the decoder's output. Note from Fig. 9 (b) that the output of the decoder also is given by the above Eqn.

Optimal predictors:

The optimal predictor used in most predictive coding applications minimizes the encoder's mean-square prediction error

$$E\{e_n^2\} = E\{[f_n - \hat{f}_n]^2\}$$

subject to the constraint that

$$\dot{f}_n = \dot{e}_n + \hat{f}_n \approx e_n + \hat{f}_n = f_n$$

and

$$\hat{f}_n = \sum_{i=1}^{m} \alpha_i f_{n-i}.$$

That is, the optimization criterion is chosen to minimize the mean-square prediction error, the quantization error is assumed to be negligible ($e\ddot{y}_n \approx e_n$), and the prediction is constrained to a linear combination of m previous pixels.1 These restrictions are not essential, but they simplify the analysis considerably and, at the same time, decrease the computational complexity of the predictor. The resulting predictive coding approach is referred to as differential pulse code modulation (DPCM).

REDUNDANCIES IN A DIGITAL IMAGE

The term data compression refers to the process of reducing the amount of data required to represent a given quantity of information. A clear distinction must be made between data and information. They are not synonymous. In fact, data are the means by which information is conveyed. Various amounts of data may be used to represent the same amount of information. Such might be the case, for example, if a long-winded individual and someone who is short and to the point were to relate the same story. Here, the information of interest is the story; words are the data used to relate the information. If the two individuals use a different number of words to tell the same basic story, two different versions of the story are created, and at least one includes nonessential data. That is, it contains data (or words) that either provide no relevant information or simply restate that which is already known. It is thus said to contain data redundancy.

Data redundancy is a central issue in digital image compression. It is not an abstract concept but a mathematically quantifiable entity. If n_1 and n_2 denote the number of information-carrying units in two data sets that represent the same information, the relative data redundancy R_D of the first data set (the one characterized by n_1) can be defined as

$$R_D = 1 - \frac{1}{C_R}$$

where C_R , commonly called the compression ratio, is

$$C_R = \frac{n_1}{n_2}.$$

For the case $n_2 = n_1$, $C_R = 1$ and $R_D = 0$, indicating that (relative to the second data set) the first representation of the information contains no redundant data. When $n_2 << n_1$, $C_R \to \infty$ and $R_D \to 1$, implying significant compression and highly redundant data. Finally, when $n_2 >> n_1$, $C_R \to 0$ and $R_D \to \infty$, indicating that the second data set contains much more data than the original representation. This, of course, is the normally undesirable case of data expansion. In general, C_R and R_D lie in the open intervals $(0,\infty)$ and $(-\infty, 1)$, respectively. A practical compression ratio, such as 10 (or 10:1), means that the first data set has 10 information carrying units (say, bits) for every 1 unit in the second

or compressed data set. The corresponding redundancy of 0.9 implies that 90% of the data in the first data set is redundant.

In digital image compression, three basic data redundancies can be identified and exploited: **coding redundancy**, **interpixel redundancy**, and **psychovisual redundancy**. Data compression is achieved when one or more of these redundancies are reduced or eliminated.

Coding Redundancy:

In this, we utilize formulation to show how the gray-level histogram of an image also can provide a great deal of insight into the construction of codes to reduce the amount of data used to represent it.

Let us assume, once again, that a discrete random variable r_k in the interval [0, 1] represents the gray levels of an image and that each r_k occurs with probability p_r (r_k).

$$p_r(r_k) = \frac{n_k}{n} \qquad k = 0, 1, 2, \ldots, L - 1$$

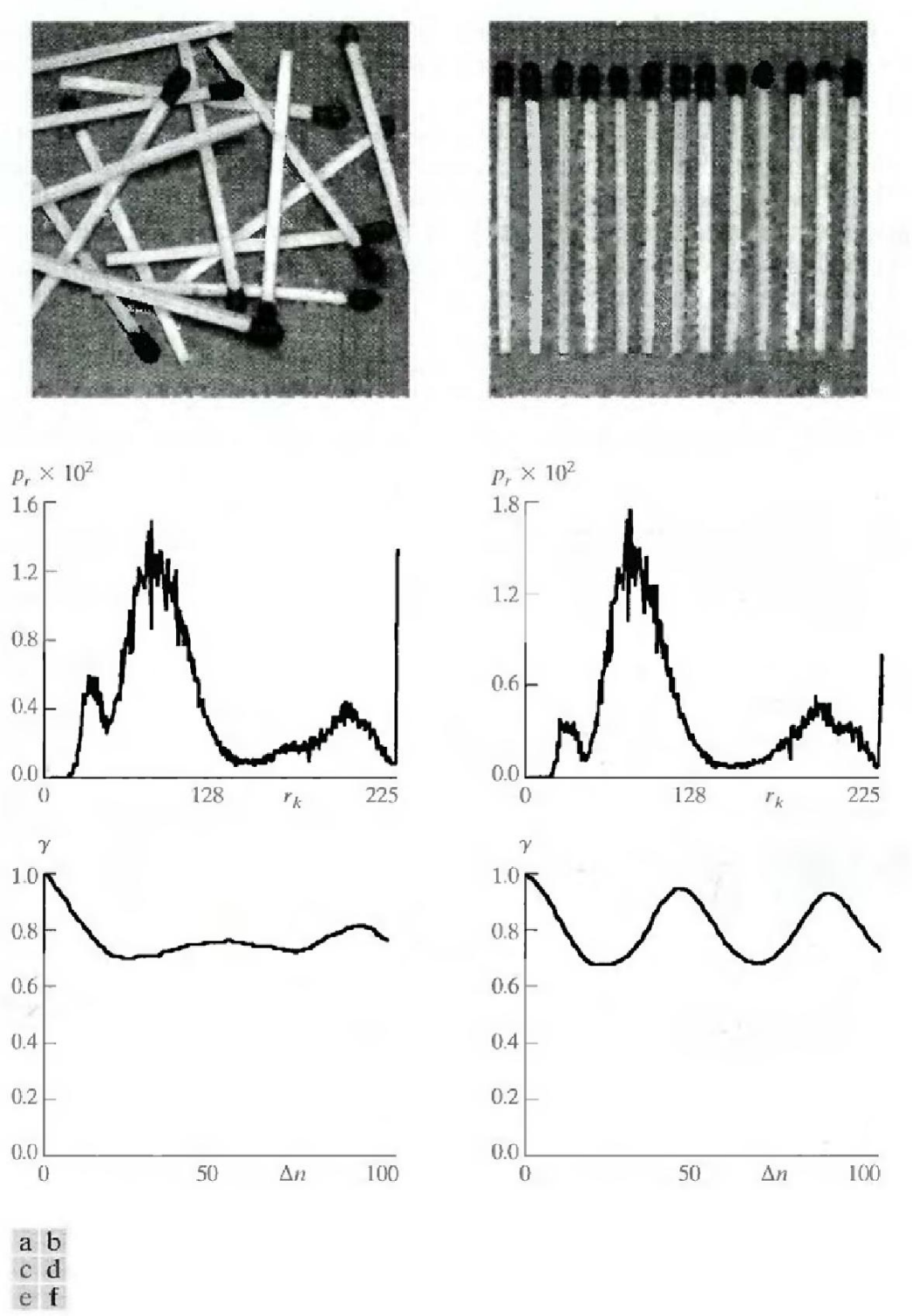
where L is the number of gray levels, n_k is the number of times that the kth gray level appears in the image, and n is the total number of pixels in the image. If the number of bits used to represent each value of r_k is l (r_k), then the average number of bits required to represent each pixel is

$$L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k).$$

That is, the average length of the code words assigned to the various gray-level values is found by summing the product of the number of bits used to represent each gray level and the probability that the gray level occurs. Thus the total number of bits required to code an M X N image is MNL_{avg}.

Interpixel Redundancy:

Consider the images shown in Figs. 1.1(a) and (b). As Figs. 1.1(c) and (d) show, these images have virtually identical histograms. Note also that both histograms are trimodal, indicating the presence of three dominant ranges of gray-level values. Because the gray levels in these images are not equally probable, variable-length coding can be used to reduce the coding redundancy that would result from a straight or natural binary encoding of their pixels. The coding process, however, would not alter the level of correlation between the pixels within the images. In other words, the codes used to represent the gray levels of each image have nothing to do with the correlation between pixels. These correlations result from the structural or geometric relationships between the objects in the image.



a b
c d
e f

Figures 1.1(e) and (f) show the respective autocorrelation coefficients computed along one line of each image.

$$\gamma(\Delta n) = \frac{A(\Delta n)}{A(0)}$$

where

$$A(\Delta n) = \frac{1}{N - \Delta n} \sum_{y=0}^{N-1-\Delta n} f(x, y) f(x, y + \Delta n).$$

The scaling factor in Eq. above accounts for the varying number of sum terms that arise for each integer value of n. Of course, n must be strictly less than N, the number of pixels on a line. The variable x is the coordinate of the line used in the computation. Note the dramatic difference between the shape of the functions shown in Figs. 1.1(e) and (f). Their shapes can be qualitatively related to the structure in the images in Figs. 1.1(a) and (b).This relationship is particularly noticeable in Fig. 1.1 (f), where the high correlation between pixels separated by 45 and 90 samples can be directly related to the spacing between the vertically oriented matches of Fig. 1.1(b). In addition, the adjacent pixels of both images are highly correlated. When n is 1, ã is 0.9922 and 0.9928 for the images of Figs. 1.1 (a) and (b), respectively. These values are typical of most properly sampled television images.

These illustrations reflect another important form of data redundancy—one directly related to the interpixel correlations within an image. Because the value of any given pixel can be reasonably predicted from the value of its neighbors, the information carried by individual pixels is relatively small. Much of the visual contribution of a single pixel to an image is redundant; it could have been guessed on the basis of the values of its neighbors. A variety of names, including spatial redundancy, geometric redundancy, and interframe redundancy, have been coined to refer to these interpixel dependencies. We use the term interpixel redundancy to encompass them all.

In order to reduce the interpixel redundancies in an image, the 2-D pixel array normally used for human viewing and interpretation must be transformed into a more efficient (but usually "nonvisual") format. For example, the differences between adjacent pixels can be used to represent an image. Transformations of this type (that is, those that remove interpixel redundancy) are referred to as mappings. They are called reversible mappings if the original image elements can be reconstructed from the transformed data set.

PSYCHOVISUAL REDUNDANCY

The brightness of a region, as perceived by the eye, depends on factors other than simply the light reflected by the region. For example, intensity variations (Mach bands) can be perceived in an area of constant intensity. Such phenomena result from the fact that the eye does not respond with equal sensitivity to all visual information. Certain information simply has less relative importance than other information in normal visual processing. This information is said to be psychovisually redundant. It can be eliminated without significantly impairing the quality of image perception.

That psychovisual redundancies exist should not come as a surprise, because human perception of the information in an image normally does not involve quantitative analysis of every pixel value in the image. In general, an observer searches for distinguishing features such as edges or textural regions and mentally combines them into recognizable groupings. The brain then correlates these groupings with prior knowledge in order to complete the image interpretation process. Psychovisual redundancy is fundamentally different from the redundancies discussed earlier. Unlike coding and interpixel redundancy, psychovisual redundancy is associated with real or quantifiable visual information. Its elimination is possible only because the information itself is not essential for normal visual processing. Since the elimination of psychovisually redundant data results in a loss of quantitative information, it is commonly referred to as quantization.

This terminology is consistent with normal usage of the word, which generally means the mapping of a broad range of input values to a limited number of output values. As it is an irreversible operation (visual information is lost), quantization results in lossy data compression.

TEXT BOOKS:

1. Digital Image Processing – Rafael C. Gonzalez, Richard E. Woods, 3rd Ed, Pearson.
2. Fundamentals of Image Processing – A. K. Jain, Prentice Hall India.

REFERENCE BOOKS:

1. Digital Image Processing – William K. Pratt, John Wiley Publications
2. Digital Image Processing – K. R. Castleman, Pearson Publications
3. Fundamentals of Electronic Image Processing – Weeks Jr, SRIC/IEEE series, PHI.