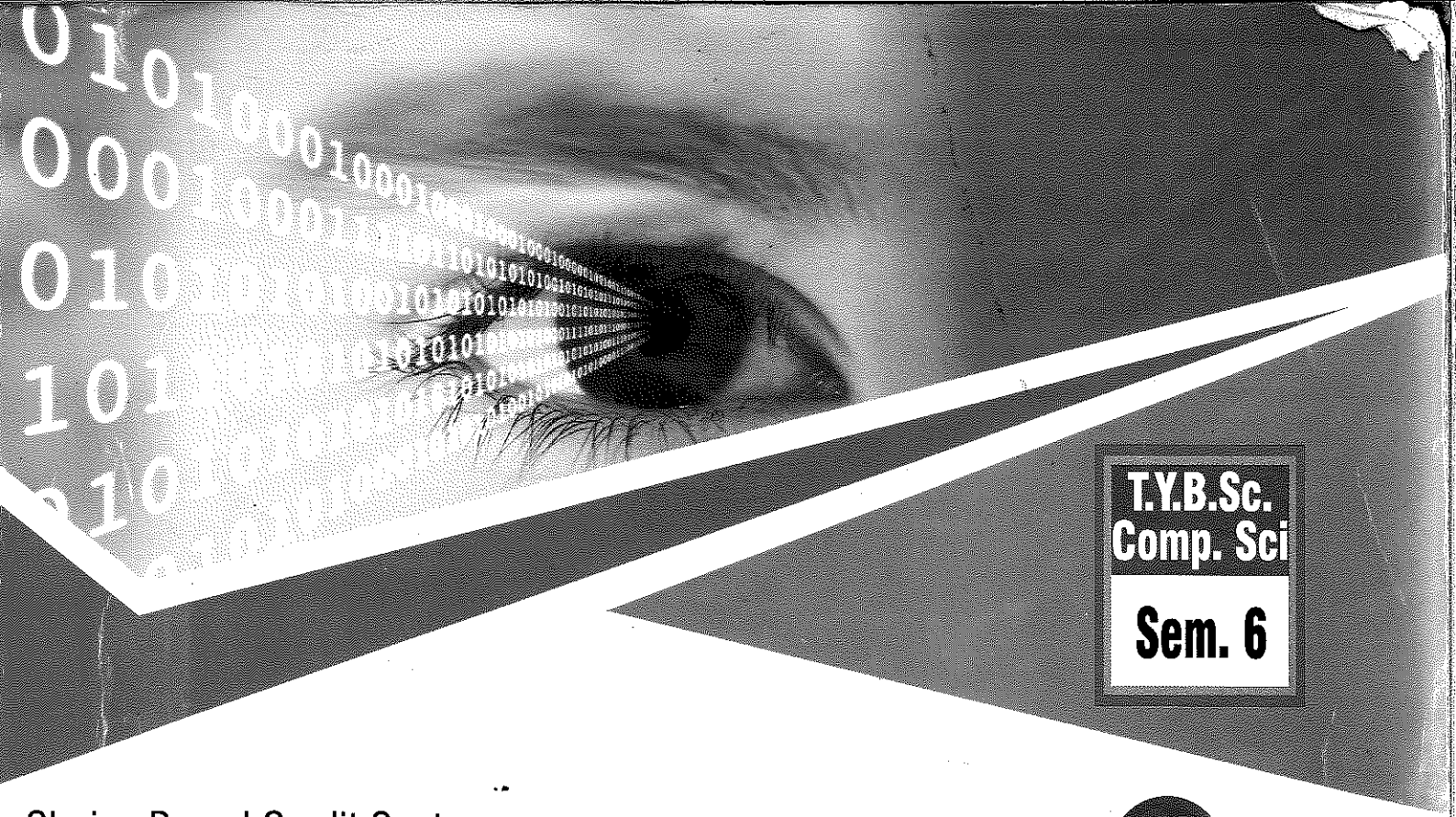


T.Y.B.Sc.
Comp. Sci
Sem. 6

Choice Based Credit System
with effect from the Academic Year 2018-2019

MU

DIGITAL IMAGE PROCESSING

(Using MATLAB Codes)

(Elective II) **(USCS605)**

Dhananjay K. Theckedath

Important Features of the Book

- Model Question Papers as per Examination Pattern
- With Multicolour Diagrams

Tech-Max Publications, Pune
Innovation Throughout
Computer Science Division

CBSCE19A Price ₹ 230/-

SYLLABUS

Course : USCS605	TOPICS (Credits : 03 Lectures / Week : 03) **Digital Image Processing**	

Objectives

To study two-dimensional Signals and Systems. To understand image fundamentals and transforms necessary for image processing. To study the image enhancement techniques in spatial and frequency domain. To study image segmentation and image compression techniques.

Expected Learning Outcomes

Learner should review the fundamental concepts of a digital image processing system. Analyze the images in the frequency domain using various transforms. Evaluate the techniques for image enhancement and image segmentation. Apply various compression techniques. They will be familiar with basic image processing techniques for solving real problems.

Unit I	**Introduction to Image-processing System :** Introduction, Image Sampling, Quantization, Resolution, Human Visual Systems, Elements of an Image-processing System, Applications of Digital Image Processing. **2D Signals and Systems :** 2D signals, separable sequence, periodic sequence, 2D systems, classification of 2D systems, 2D Digital filter. **Convolution and Correlation :** 2D Convolution through graphical method, Convolution through 2D Z-transform, 2D Convolution through matrix analysis, Circular Convolution, Applications of Circular Convolution, 2D Correlation. **Image Transforms :** Need for transform, image transforms, Fourier transform, 2D Discrete Fourier Transform, Properties of 2D DFT, Importance of Phase, Walsh transform, Hadamard transform, Haar transform, Slant transform, Discrete Cosine transform, KL transform. **(Refer Chapters 1, 2, 3, 6, 8 and Appendix)**	**15L**
Unit II	**Image Enhancement :** Image Enhancement in spatial domain, Enhancement through Point operations, Histogram manipulation, Linear and nonlinear Gray Level Transformation, local or neighborhood operation, Median Filter, Spatial domain High pass filtering, Bit-plane slicing, Image Enhancement in frequency domain, Homomorphic filter, Zooming operation, Image Arithmetic. **Binary Image Processing :** Mathematical morphology, Structuring elements, Morphological image processing, Logical operations, Morphological operations, Dilation and Erosion, Distance Transform. **Colour Image Processing :** Colour images, Colour Model, Colour image quantization, Histogram of a colour image. **(Refer Chapters 4, 5, 6, 10 and 11)**	**15L**
Unit III	**Image Segmentation :** Image segmentation techniques, Region approach, Clustering techniques, Thresholding, Edge-based segmentation, Edge detection, Edge Linking, Hough Transform. **Image Compression :** Need for image compression, Redundancy in images, Image-compression scheme, Fundamentals of Information Theory, Run-length coding, Shannon-Fano coding, Huffman Coding, Arithmetic Coding, Transform-based compression, Image-compression standard. **(Refer Chapters 7 and 9)**	**15L**

□□□

❑❑❑

CHAPTER 1

Introduction to Image Processing

1.1 Introduction

Human beings are primarily visual creatures who depend on their eyes to gather information around them. Of the five senses that human beings have, sight is what we depend upon the most. Not many animals depend on their visual systems; the way human beings do.

Bats use high frequency sound waves. They emit sound waves which reflect back when they encounter some obstruction. Cats have poor vision but an excellent sense of smell. Snakes locate prey by heat emission and fish have organs that sense electrical fields.

1.2 What Do We Mean by Image Processing ?

What happens when we look at an object ?

The eye records the scene and sends signals to the brain. These signals get processed in the brain and some meaningful information is obtained. Let us take a simple example : when we see fire, we immediately identify it as something hot. Two things have happened here.

(1) The scene has been recorded by the eye.

(2) The brain processed this scene and gave out a warning signal.

This is image processing !!!

- We start processing images from the day we are born. Hence image processing is an integral part of us and we continue to process images till the day we die. So even if this subject seems to be new, we have been subconsciously doing this, all these years. The human eye-brain mechanism represents the ultimate imaging system.
- Apart from our vision, we have another important trait that is common to all human beings. We like to store information, analyse it, discuss it with others and try to better it. This trait of ours is responsible for the rapid development of the human race.
- Early human beings strove to record their world by carving crude diagrams on stone. All the drawings that we see in old caves is just that; storing images seen, trying to analyse them and discussing it with others in the tribe. Refer Fig. 1.2.1.
- This art developed through the ages by way of materials and skill. By the mid-nineteenth century, photography was well established. Image processing that we study starts from this era.

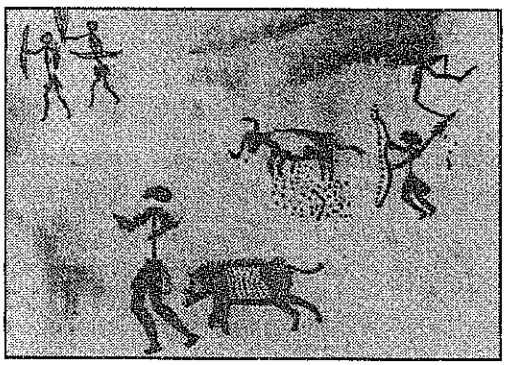


Fig. 1.2.1

- Though it was stated earlier that the human eye-brain mechanism represents the ultimate imaging system, image processing as a subject involves processing images obtained by a camera. With the advent of computers, image processing as a subject grew rapidly.

- Images from a camera are fed into a computer where algorithms are written to process these images. Here, the camera replaces the human eye and the computer does the processing.
- Hence image processing as an engineering subject is basically manipulation of images by a computer.

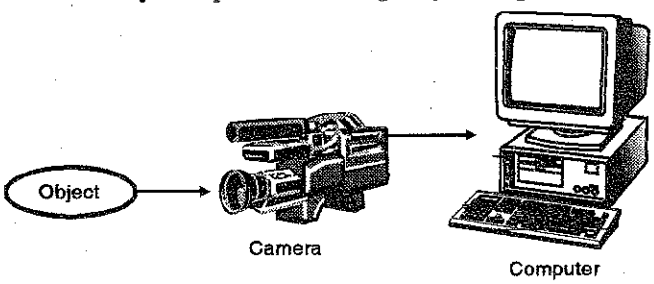


Fig. 1.2.2

1.3 Images

Let us now define what we mean by an image. The world around us is 3-dimensional, while images obtained through a camera are 2-dimensional. Hence an image can be defined as a 2-dimensional representation of a 3-dimensional world. Consider the image shown in Fig. 1.3.1.

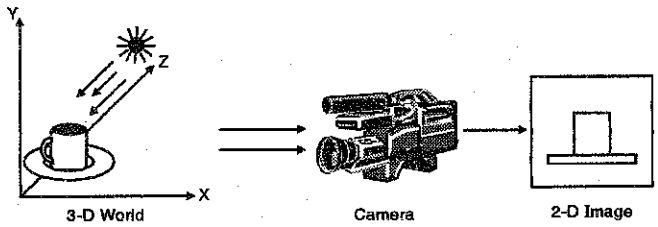


Fig. 1.3.1

In moving from the 3-dimensional world to the 2-dimensional image, we loose one dimension. Depth information is lost. As can be seen from the Fig. 1.3.1, the handle of the tea cup is missing.

All family pictures, photographs on identity cards etc. are 2-dimensional. If this statement is not clear, let us take a simple example.

Example of a 1-dimensional and a 2-dimensional signal

Consider a voltage signal shown in Fig. 1.3.2. We are all familiar with a signal of this kind. Here the voltage is varying with respect to time. This is a typical 1-dimensional signal. If we want to locate a dot on the wave, all we need to know is its corresponding time.

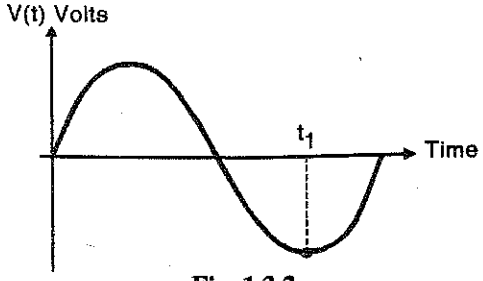


Fig. 1.3.2

Let us see why images are 2-dimensional functions. Consider the image shown in Fig. 1.3.3.

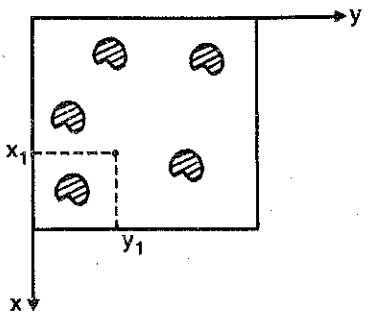


Fig. 1.3.3

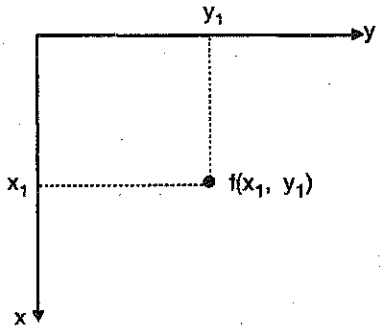


Fig. 1.3.4

In this case, to locate the dot shown, we need to know its position in two directions (x and y) Fig. 1.3.4. Hence all images that we see are 2-dimensional functions. A typical image is represented as shown in Fig. 1.3.5. Here (x_1, y_1) are the spatial coordinates and f is the grey level (colour in the case of colour image) at that point. Hence grey level f varies with respect to the x and y coordinates.

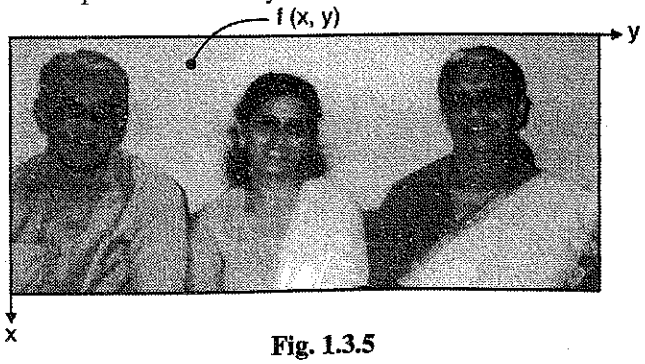


Fig. 1.3.5

1.4 The Electromagnetic Spectrum

The apparatus shown in Fig. 1.2.2 will work only if light is incident on the object. What we call light is actually a very small section of the electromagnetic energy spectrum.The entire spectrum is shown in Fig. 1.4.1.

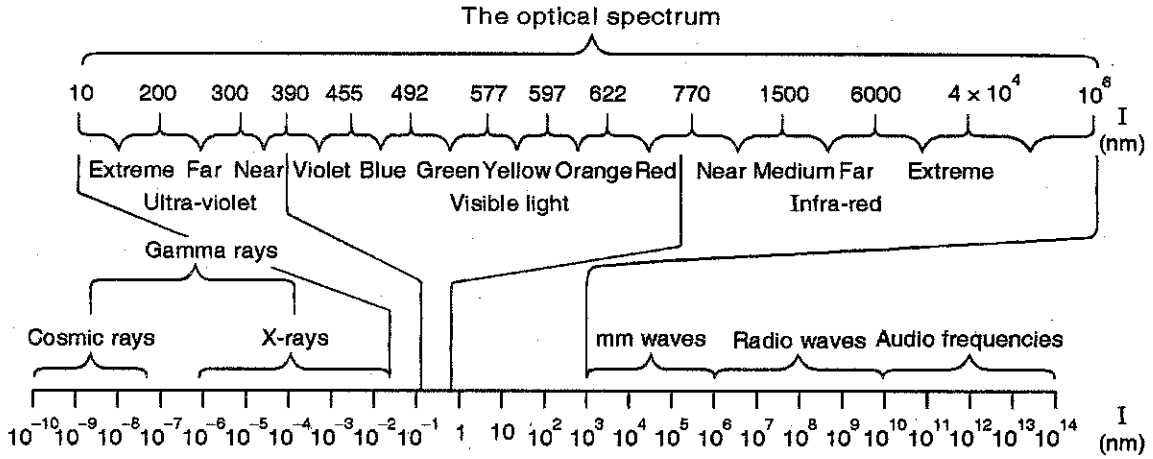


Fig. 1.4.1

Electromagnetic energy, as the name suggests, exists in the simultaneous form of electricity and magnetism. These two forms of the same energy are transmitted together as electromagnetic radiation. One cannot exist without the other. A flow of electric current always produces magnetism, and magnetism is used to produce electricity. Electromagnetic radiation is propagated outwards from its source at a velocity of 300,00,0000 meters per second (3×10^8 m/sec).

Although our natural source of electromagnetic radiation is the sun, there are also a number of man-made sources which, among many others, include tungsten filament lamps, gas discharge lamps and lasers. Light is a band of electromagnetic radiation mediated by the human eye and is limited to a spectrum extending from 380 nm to 760 nm.

Most of the images that we encounter in our day to day life are taken from cameras which are sensitive to this range of the electromagnetic spectrum (380 - 760 nm). We must not forget though, that there are cameras which are capable of detecting infrared, ultraviolet light, X-rays and radio waves too.

The electromagnetic spectrum can be expressed in terms of wavelength and frequency. The wavelength (λ) and the frequency (ν) are related by the expression

$$\lambda = \frac{c}{\nu} \qquad \ldots(1.4.1)$$

Here c is the speed of light = 3×10^8 m/sec

Ex. 1.4.1

Calculate the frequency of oscillation of green light.

Soln. :

It has been known that green light has a wavelength of approximately 500 nm

(500×10^{-9}m)

Its frequency of oscillations can be calculated using Equation (1.4.1).

$$\lambda = \frac{c}{\nu}$$

$$\nu = \frac{c}{\lambda} = \frac{3 \times 10^8 \text{ m/sec}}{500 \times 10^{-9} \text{ m}}$$

$$\nu = 6 \times 10^{14} \text{ Hz}$$

i.e., the frequency of green light is 600,000,000,000,000 cycles/sec !

Hence it is more convenient to discuss electromagnetic radiation in terms of wavelengths (nm) rather than frequencies (Hz).

1.5 Units of Intensity

We know that for an object to be seen, some amount of light has to fall on it.

The unit of luminous intensity (I) is candela (cd) (SI unit) which by definition, is $(1/60)^{th}$ of a square centimetre of the surface of a black body radiator at the absolute temperature of 2045 K.

Note : A black body radiator is one that absorbs all the radiation incident upon it and has no reflecting power. The radiation from a heated black body source is called black body radiation and is always quoted in Kelvin.

The unit is called candela because initially it was defined as a standard candle burning a specific amount of wax per hour. There is another important unit apart from candela that we encounter. This unit is called lux.

1.6 The Human Visual System

It is important for designers and users of image processing to understand the characteristics of the human vision system. For efficient design of algorithms whose output is a photograph or a display viewed by a human observer, it is beneficial to have an understanding of the mechanism of human vision.

Many interesting studies have been carried out and the subject of visual perception has grown over the years. We begin with the structure of the human eye.

The Fig. 1.6.1 shows the horizontal as well as the vertical cross section of a human eye ball. The front of the eye is covered by a transparent surface called cornea. The remaining outer cover, sclera, is composed of a fibrous coat that surrounds the choroid, a layer containing blood capillaries. Sclera is the white portion of the eye.

Inside the choroid, is the retina which is composed of two types of photoreceptors. These photoreceptors are specialised to convert light rays into receptor potentials. These photoreceptors are called rods and cones (named due to their shape). Rods are long and slender while cones are shorter and thicker. Nerves connecting the retina leave the eyeball through the optic nerve bundle. Light entering the cornea is focused on the retina surface by a lens that changes shape under muscular control to perform proper focusing of near and distant objects. The iris acts as a diaphragm to control the amount of light entering the eye.

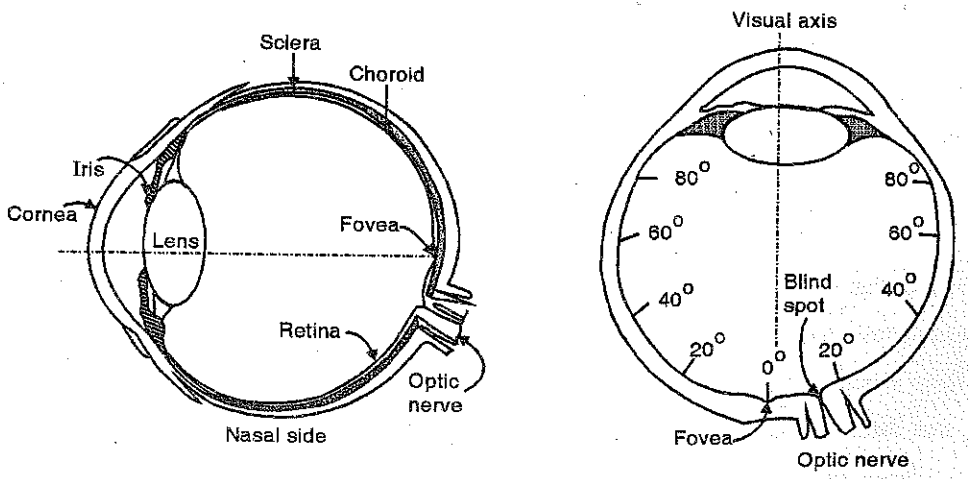


Fig. 1.6.1

Each retina has about 6 million cones and 120 million rods. Rods are most important for black and white vision in dim light. They also allow us to discriminate between different shades of dark and light and permit us to see shapes and movements. At low levels of illumination, the rods provide a visual response called Scotopic Vision. The fact that we can see in the dark is due to this scotopic vision.

Cones provide colour vision and high visual acuity (sharpness of vision) in bright light. Cones respond to higher levels of illumination, their response is called Photopic Vision. In the intermediate range of illumination, both rods and cones are active and provide Mesopic Vision.

In moon light, we cannot see colour because only the rods are functioning. The distribution of rods and cones in the retina are shown Fig. 1.6.2.

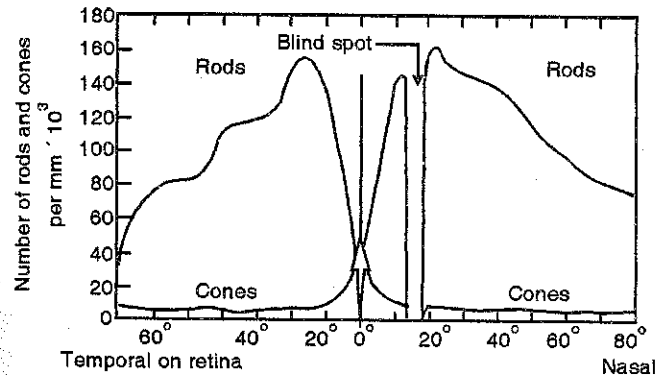


Fig. 1.6.2 : Permetric angle, degrees

At a point near the optic nerve, called fovea, the density of the cones is the greatest. Due to this, it is the region of sharpest photopic vision. The photoreceptors are connected to the nerves and these nerves, combined together, leave the eyeball as the optic nerve. (It is not as simple as stated and interested students can refer to medical journals). The optical nerve is a bundle of nerves (approx 800,000 nerve fibres) leaving the eyeball. The region from where this optic nerve leaves the eyeball has no rods and cones. This is called the blind spot. We cannot see an image that strikes the blind spot. Normally we are not aware of having a blind spot but we can easily convince ourselves of its presence.

☞ Try this experiment

Cover your left eye and gaze directly at the word 'Deep' shown below. You should also see the square drawn next to it. Now increase or decrease the distance between the book and your eye. At some point, the square will disappear. This is because the square falls on the blind spot !!

Deep ■

When a light stimulus activates a rod or a cone, a photochemical transition occurs, producing a nerve impulse. The manner in which these nerve impulses propagate through the visual system has yet to be fully understood by the medical fraternity and hence let us end our discussion on the anatomy of the human eye here. As stated earlier, this discussion is in no way exhaustive and interested students can refer medical journals. But as students of image processing this information would be sufficient.

1.6.1 Image Formation in Eye

The main difference between the lens of the eye and an ordinary optical lens is that the former is flexible. The shape of the lens is controlled by the tension of the ciliary muscles. These muscles help in focussing of objects near or far. The distance between the centre of the lens and the retina (focal length) varies from 14 mm to about 17 mm. When the eye focuses on an object far away (approx. > 3 m) the lens exhibits its lowest refractive power (Focal length = 17 mm). When the eye focuses on nearby objects, the lens of the eye is strongly refractive (Focal length = 14 mm)

Let us take an example :

Ex. 1.6.1

Consider an observer looking at a lamp-post which is at a distance of 50 m. If the height of the lamp post is 10 m, find the size of the image formed in the retina (Retinal image).

Soln. :

Since the image is far away, the focal length is approximately 17 mm. We use the similarity of triangles.

Let the retinal image be r

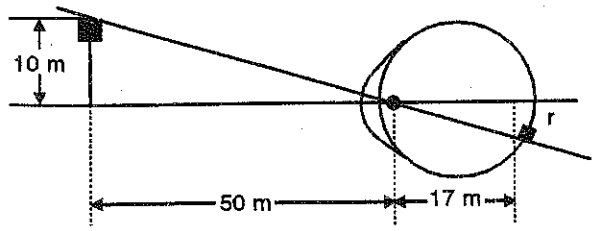


Fig. P.1.6.1

$$\frac{10}{50} = \frac{r}{17}$$

$$r = 3.4 \text{ mm}$$

∴The height of the retinal image is 3.4 mm

If the same image is observed at a distance of 100 m, we get

$$\frac{10}{100} = \frac{r}{17}$$

$$r = 1.7 \text{ mm}$$

1.6.2 Visual Phenomena

The human eye is a complex system and the images that we perceive are also equally complex. We shall now explain the visual phenomena.

What the human eye senses are in general intensity images- Intensity, Brightness and Contrast are three different phenomena. Intensity of a light source depends on the total amount of light emitted by it. Hence intensity is a physical property and can be measured. Brightness on the other hand is a psycho-visual concept and hence is actually a sensation to light intensity. Contrast may be defined as the difference in perceived brightness.

(1) **Contrast Sensitivity :** The response of the human eye to changes in the intensity of illumination is known to be non-linear.

Consider the simple experiment :

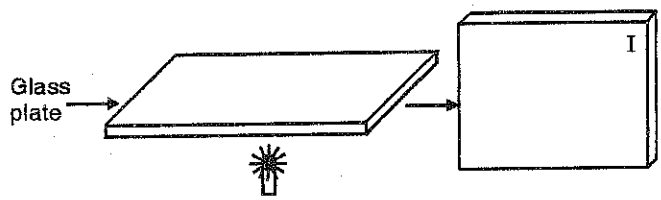


Fig. 1.6.3

We take a diffuser (opaque) glass plate and illuminate it from the bottom with constant illumination I.

The observer is asked to look at this glass plate, from the top. At the centre of this glass plate, the intensity of illumination is increased from I to I + ΔI. The observer is now asked to observe whether he or she can detect this increase.

If the observer cannot detect the change, the intensity is further increased by another increment of ΔI. This procedure is continued till the observer detects the difference. This is known as the Just Noticeable Difference (JND).

This ratio of $\frac{\Delta I_c}{I}$ is called the Weber ratio

$$\text{Weber ratio} = \frac{\Delta I_c}{I}$$

Here I is a constant and I_c is the incremented value.

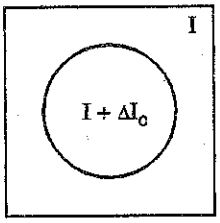


Fig. 1.6.4

A low Weber ratio implies that even a small variation (ΔI) was detected by the observer. A high Weber ratio implies that large variations (ΔI + ΔI + ...) were required for the observer to notice the change.

Hence a low Weber ratio means that the observer has good discernible vision. At proper illuminations, the Weber ratio of a group of people is more or less constant at 0.02. Weber ratio also depends on the illumination I.

As can be seen from the graph, the weber ratio is large at very low as well as very high levels of illumination. This should not come as a surprise. Our discrimination quality reduces when we are in a room that is not well lit. At the same time our discrimination quality also reduces when there is too much light.

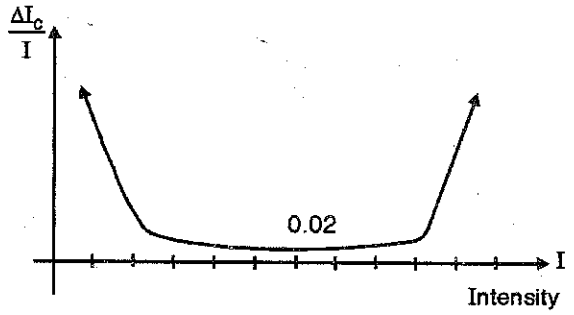


Fig. 1.6.5

(2) Brightness Adaptation : The range of light intensity levels to which the human visual system can adapt is enormous, of the order of 10^{10} from the scotopic threshold to the glare limit. It simply means we can see things in the dark and also when there is a lot of illumination. It has been shown that the intensity of light perceived by the human visual system (subjective brightness) is a logarithmic function of the light intensity incident on the human eye.

The curve in the Fig. 1.6.6 represents the range of intensities that the human visual system can adapt. When illumination is low, it is the scotopic vision that plays a dominant role while for high intensities of illumination, it is the photopic vision which is dominant.

As can be seen from the figure, the transition from scotopic vision to photopic vision is gradual and at certain levels of illuminations, both of them play a role. To cut a long story short, the dynamic range of the human eye is enormous. But there is a catch here. The eye cannot operate over the entire range simultaneously i.e., at a given point of time, the eye can only observe a small range of illuminations. This phenomena is known as Brightness Adaptation. The fact that our eyes can operate only on a small range, can be proved by a simple experiment on ourselves.

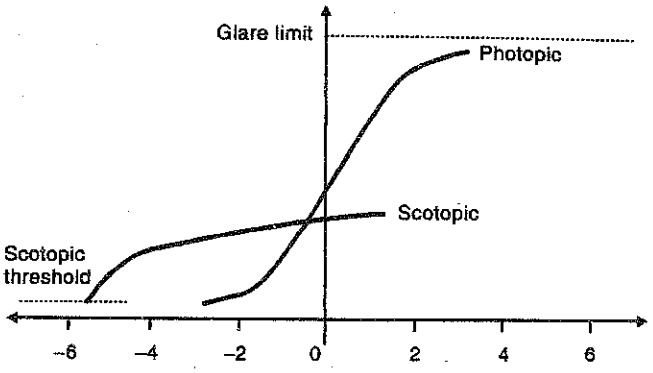


Fig. 1.6.6

Stare at the sun for a couple of seconds, the eye adapts itself to the upper range of the intensity values. Now look away from the sun and you will find that you cannot see anything for sometime. This is because the eye takes a finite time to adapt itself to this new range. A similar phenomenon is observed when the power supply of our homes is cut-off in the night. Everything seems to be pitch dark and nothing can be seen for sometime. But gradually our eyes adjust to this low level of illumination and then things start getting visible even in the dark.

(3) Acuity and Contour (Mach Bands) : Mach bands are named after the Austrian physicst Ernst Mach (1838 - 1916).

Consider a set of grey scale strips shown in Fig. 1.6.7.

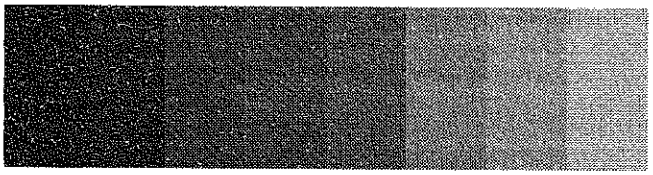
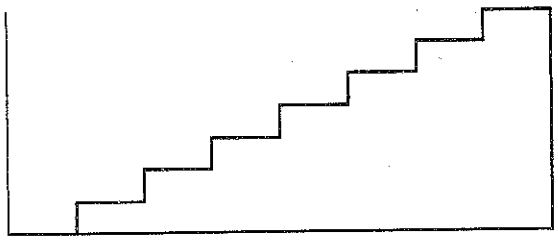


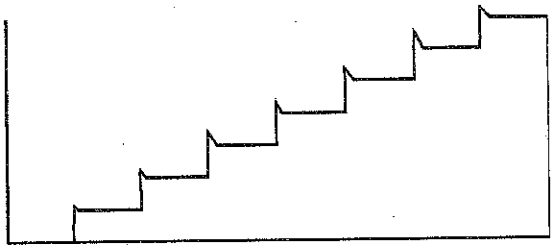
Fig. 1.6.7

Each of the eight strips have uniform intensities within the strip. This can be seen if we place our hand on all the other strips and observe only one strip at a time. But when we look at all the eight strips together, visual appearance is that each strip looks darker at its right side than its left. This is called the Mach band effect.

The actual and the perceived intensity charts are shown in Fig. 1.6.8 (a) and (b).



(a) Actual intensity



(b) Perceived intensity

Fig. 1.6.8

The intensities tend to overshoot at the right hand edges. The overshoot is a consequence of the spatial frequency response of the eye. The human eye possesses a lower sensitivity to high and low spatial frequencies than to mid range frequencies (spatial frequencies will be covered in chapter of Image Enhancement in Spatial Domain). The implication of this is that perfect edge contours in an image can be sacrificed to a certain extent as the human eye has an imperfect response to high spatial frequency brightness transitions.

(4) **Simultaneous Contrast :** Consider the image shown in Fig. 1.6.9.

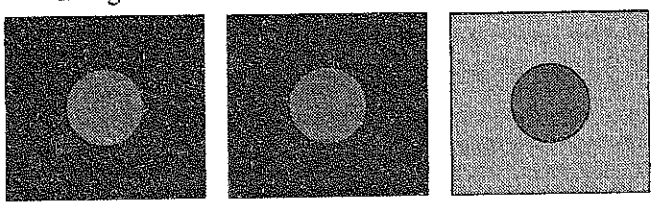


Fig. 1.6.9

Each of the small circles have the same intensity, but because the surrounding grey level of each of the circles is different, the circles do not appear equally bright. Hence the intensity that we perceive are not actually the absolute values.

(5) **Integration :** Another important property of our eye is that it integrates the scene as a whole. This is a property that facilitates programming. When we look at a picture, we do not look at each point, but the image as a whole.

Let us take an example, read the following lines as fast as you can.

"Accodring to a recheearch at an Elingsh Unvertisy, it deosn't mttaer in what oreder the ltters in a word are, the only iprmoetnt thing is that the frist and the lsat ltteer is at the rghit place. The rset can be a total mses and you can still raed it wouthit a porbelm. This is bcusease we do not raed ervey lteter by itself but the word as a wlohe."

You could read it, inspite of the spelling mistakes only because of the integration property of the eye.

Before we end, one issue needs to be sorted out. A very common question that people ask is what is the difference between image processing, computer graphics and computer vision ?

Image processing deals with manipulation of images. A input image is modified into a new image. Computer graphics deals with creation of images. In computer graphics, models (2D or 3D) are created using mathematical functions (Descriptors). Computer vision which is also known as Machine vision deals with the analysis of image content. Computer vision is used to automate a process.

Table 1.6.1

Input \ Output	Image	Description (Mathematical function)
Image	I.P.	C.V.
Description (Mathematical function)	C. G.	-

From the Table 1.6.1,

Image processing → Input (Image) – Output (Image)

Computer graphics → Input (Description) – Output (Image)

Computer vision → Input (Image) – Output (Description)

The material provided in this chapter is primarily basic information which would be required in subsequent discussions. Our study of the human visual system, though not exhaustive, provides a basic idea of the capabilities of the eye in perceiving pictorial information.

Summary

In this chapter, preliminary concepts of digital image processing are presented. Difference between one-dimensional and two-dimensional signals is explained. Topics such as electromagnetic spectrum and inverse square law are discussed with examples. Elements of the human visual system are presented. Basic anatomy of the human eye is explained with a few illustrations. Perceptual characteristics such as brightness adaptation and logarithmic response to incident intensity in the form of Weber's ratio are also introduced. The concepts explained here will be found useful in understanding image processing algorithms in subsequent chapters. This chapter forms the fundamental base required to understand image processing.

Review Questions

Q. 1 What do we mean by image processing ?

Q. 2 Why do we say that an image is a 2D-function ?

Q. 3 Calculate the frequency of oscillation of red light.

Q. 4 Explain the structure of the human eye.

Q. 5 Explain the term scotopic vision, photopic vision and mesopic vision.

Q. 6 Brightness discrimination is poor at low levels of illumination. Explain.

Q. 7 Images are processed either for human perception or for machine perception. Explain.

Q. 8 What does the Weber ratio imply ?

Q. 9 What do Mach bands imply ?

Q. 10 If an observer is looking at an object 12 m high and a distance of 2 m. Find out the size of the retinal image.

Q. 11 Distinguish between image processing and graphics.

Q. 12 Explain the term brightness adaptation.

Chapter Ends...

❑❑❑

CHAPTER 2

Image Sensing and Acquisition

2.1 Introduction

☞ Basic elements of an image processing system

Digital image processing is basically modification of images on a computer. The basic components of an image processing system are shown below.

(1) Image Acquisition.

(2) Image Storage.

(3) Image Processing.

(4) Display.

(5) Transmission (if required).

We shall discuss each one in detail.

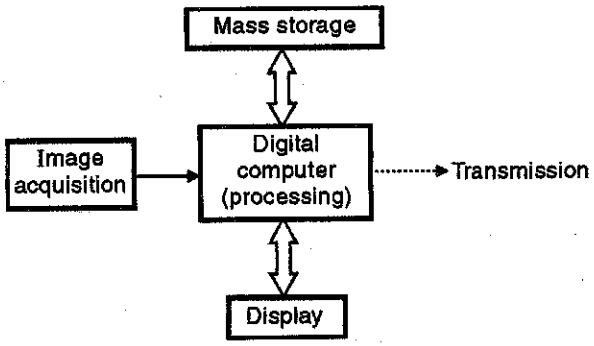


Fig. 2.1.1

(1) Image Acquisition : Image acquisition is the first step in any image processing system. The general aim of image acquisition is to transform an optical image (real world data) into an array of numerical data which could be later manipulated on a computer.

Image acquisition is achieved by suitable cameras. We use different cameras for different applications. If we need an X-ray image, we use a camera (film) that is sensitive to X-rays. If we want an infra-red image, we use cameras which are sensitive to infra-red radiations. For normal images (family pictures etc.) we use cameras which are sensitive to the visual spectrum. In this book we shall discuss cameras (sensors) which are sensitive only to the visual range.

Quantum detectors

Quantum detection is the most important mechanism of image sensing and acquisition. It relies upon the energy of absorbed photons being used to promote electrons from their stable state to a higher state above an energy threshold. Whenever this occurs, the properties of that material get altered in some measurable way. Planck/Einstein came up with a relationship between the wavelength of the incident photon and the energy that it carries.

$$e = \frac{h c}{\lambda} \qquad \ldots(2.1.1)$$

Where, e = Energy carried by a photon

h = Planck's constant, 6.626×10^{-34} Js

c = Speed of light, 3×10^{8} m/sec

λ = Wavelength of the incident radiation.

On collision, the photon transfers all or none of this quantum of energy to the electron.

The Equation (2.1.1) is very important as it tells us that the maximum wavelength to which the quantum detector will respond is determined by the energy threshold and hence by the material selection.

Of the several modes of operation of quantum detectors, the most important ones are

(a) Photoconductive (b) Photovoltaic.

(a) **Photoconductive :** The resistance of photoconductive materials drop in the presence of light due to the generation of free charge carriers. An external bias is applied across the material to measure this change. This principle of photoconductivity is used in Vidicon imaging tubes.

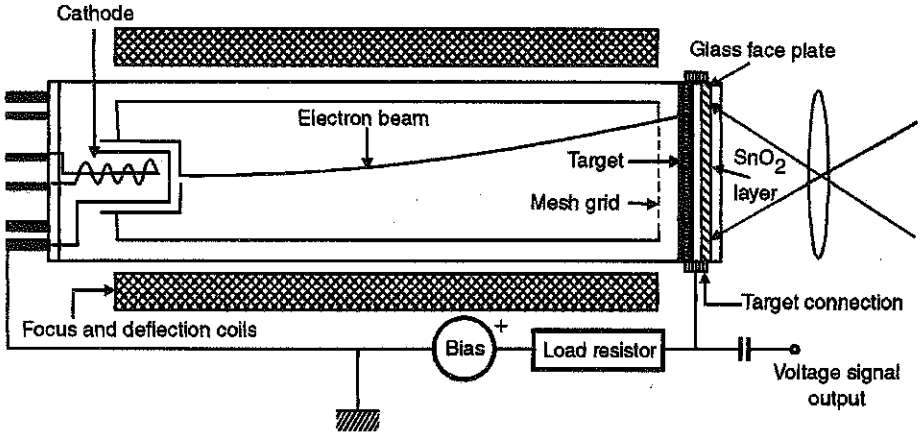


Fig. 2.1.2 : Vidicon imaging tubes

(b) **Photovoltaic :** Photovoltaic devices consist of semiconductor junctions. They are solid state arrays composed of discrete silicon imaging elements known as *Photosites*. Photovoltaic devices give a voltage output signal that is proportional to the intensity of the incident light. No external bias is required as was in the case of photoconductive devices.

The technology used in solid-state imaging sensors is based principally on charge-coupled devices, commonly known as Charged Coupled Devices CCDs. Hence the imaging sensors are called CCD sensors.

The solid state array (CCD) can be arranged in two different configurations.

(b.1) Line array CCD (b.2) Area array CCD.

(b.1) Line Arrays : The line array represents the simplest form of CCD imager and has been employed since the early 1970s. Line arrays consist of a one-dimensional array of photosites.

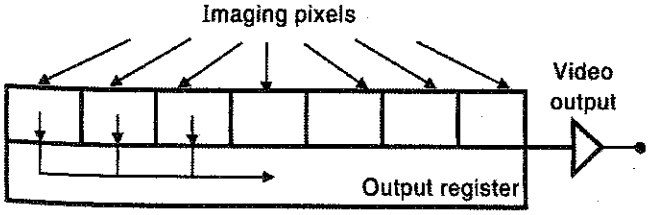


Fig. 2.1.3

A single line of CCD pixels are clocked out into the parallel output register as shown in Fig. 2.1.3. The amplifier outputs a voltage signal proportional to the contents of the row of photosites. One thing to note is that line array CCD scans only one line (hence is one-dimensional). In order to produce a two-dimensional image, the line array CCD imager has to be used as a scanning device by moving this array over the object by some mechanical activity.

This technique is used in flat bed scanners (the scanners that you come across in your laboratory or in a cyber cafe). A line array CCD can have anything from a few elements to upto 6000 or more.

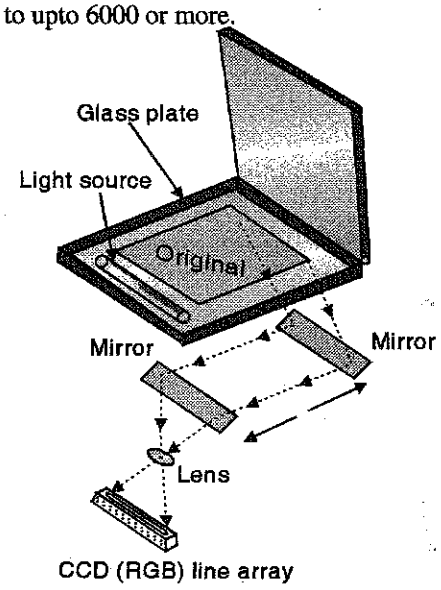


Fig. 2.1.4

(b.2) Area Arrays : The problem with line arrays is that it scans only one line. To get a two-dimensional image, we need to mechanically move the array over the entire image.

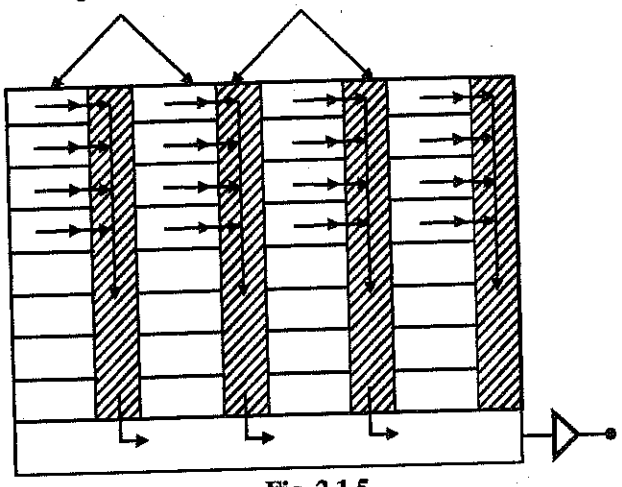


Fig. 2.1.5

Area arrays or matrix arrays consist of a two-dimensional array of photosites. They make it possible to investigate static real world scenes without any mechanical scanning. Thus much more information can be deduced from a single real time glance than would be possible with line arrays. Area arrays can be seen in all the digital cameras that we use for video imaging. The area arrays are more versatile than the line arrays, but there is a price to be paid for this. Area arrays are higher on cost and complexity.

Area sensors come in different ranges. i.e. 256 × 256, 490 × 380, 640 × 480, 780 × 575. CCD arrays are typically packaged as TV cameras. A significant advantage of solid state array sensors is that they can be shuttered at very high speeds (1/10,000 secs). This makes them ideal for applications in which freezing motion is required.

(2) Image Storage : All video signals are essentially in analog form i.e. electrical signals convey luminance and colour with continuously variable voltage. The cameras are interfaced to a computer where the processing algorithms are written. This is done by a frame grabber card. Usually a frame grabber card is a printed circuit board (PCB) fitted to the host computer with its analog entrance port matching the impedance of the incoming video signal. The A/D converter translates the video signals into digital values and a digital image is constructed. Frame grabber cards usually have a A/D card with resolution of 8 - 12-bits (256 to 4096 gray levels). Hence a frame grabber card is an interface between the camera and the computer.

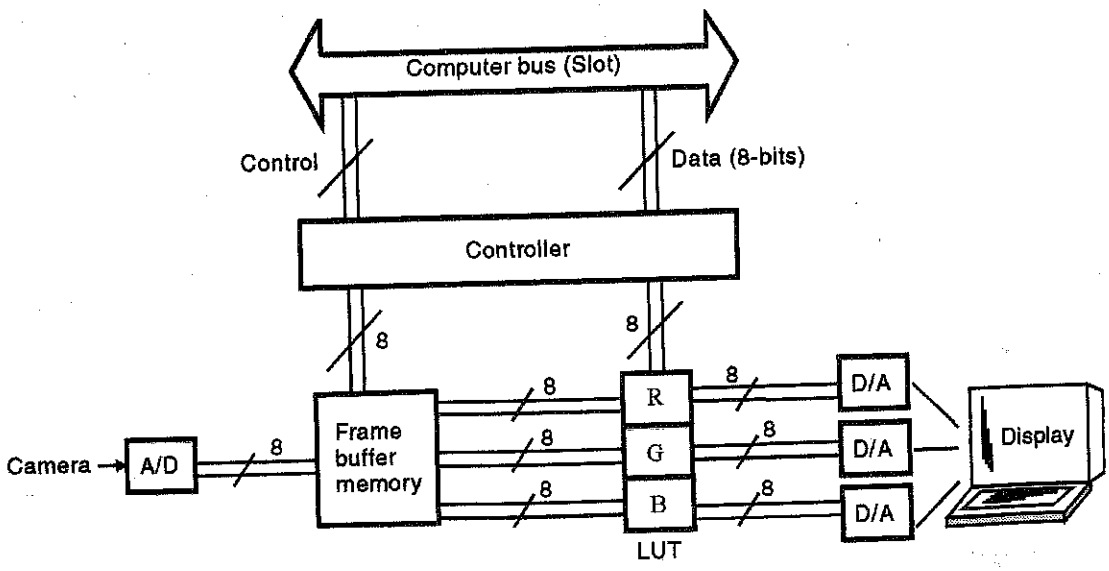


Fig. 2.1.6 : Simplified diagram of a frame grabber card

Frame grabber card has a block of memory, separate from the computers own memory, large enough to hold any image. This is known as the *frame buffer memory*. Image data, usually in the form of bytes (8-bits), is written into the frame grabber memory under the computer control, usually by DMA transfers. The contents of the memory are continuously read out at video rate (30 frames/sec), passed through the D/A converter and displayed on the monitor.

The output has a colour map or a colour Look Up Table (LUT) to permit pseudo colouring. The table consists of 8-bit values for each of the red, green and blue guns of the monitor. The commercially available frame grabber cards have additional features such as ability to zoom regions of the image and pan the images. The frame buffer memory can be upto 5 MB. Images being 2-dimensional functions, occupy a lot of space and hence the storage space on the host computer has to be large. Let us try to find out as to how much space is actually taken by images.

Each image is stored as a matrix, where every value of the matrix represents the grey level at that point. Suppose the image (matrix) is of size $A \times B$ and suppose we require C bits to represent each element of the matrix. Memory space required to store this image is approximately equal to $A \times B \times C$ bits.

Suppose we have D such images then total number of bits $\approx A \times B \times C \times D$ ...(2.1.2)

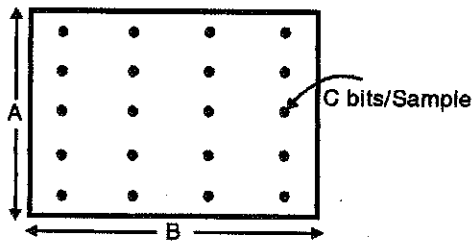


Fig. 2.1.7 (a)

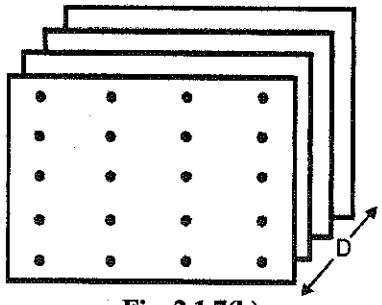


Fig. 2.1.7(b)

Thus the formula $\approx$ ABCD gives us a fairly good idea of the amount of space required.

	A	B	C	D	Number of bits
Low quality video phone	64	64	8	6	$\approx 0.2 \times 10^6$ bits
Digital broadcast TV	720	576	25	8	$\approx 83 \times 10^6$ bits
HDTV	1920	1150	50	8	$\approx 883 \times 10^6$ bits

With the advent of the PCI bus, the host computers memory could now be used as the frame buffer memory. This is because the PCI bus facilitates fast communication. Hence the image could now directly be stored on the computer RAM. Due to this, the frame buffer memory has become more or less obsolete. Only the very high end frame grabber cards still have a small frame grabber memory embedded on them.

Images being two-dimensional functions, occupy a lot of space and hence it is advisable to have a computer with a sizeable hard disk and also a good RAM. Processed images could be stored on magnetic floppies or CDs. Archival data are stored on magnetic tapes.

(3) Image Processing : Systems ranging from microcomputers to general purpose large computers are used in image processing. Dedicated image processing systems connected to host computers are very popular now-a-days. Processing of digital images involve procedures that are usually expressed in algorithmic form due to which most image processing functions are implemented in software. The only reason for specialized image processing hardware is the need for speed in some applications or to overcome some fundamental computer limitations. The trend though is to merge general purpose small computers with image processing hardware. As stated in the earlier section, frame grabber cards play the important role of merging the image processing hardware with

the host computer. One thing that we should remember is, image processing is characterized by specific solutions. Hence there is no one way to process images. A technique that works well in one area can be totally useless in some other applications.

The image processing software that is used in this book is MATLAB.

(4) **Display :** A display device produces and shows a visual form of numerical values stored in a computer as an image array. Principal display devices are printers, TV monitors and CRTs. Any erasable raster graphics display can be used as a display unit with an image processing system. However monochrome and colour TV monitors are the principal display devices used in modern image processing systems.

These raster devices convert image data into a video frame. One major problem is, it must refresh the screen at a rate of about 30 frames per second to avoid flicker. Since some computers are unable to transfer data at such high speeds, it is a good idea to buy a high end frame grabber card which has frame buffer memory on it.

If you want to build a image processing the system a home, you should have a Video Graphics Card (VGA) attached to a computer and a monitor that supports that VGA card. One could then use a simple camera that comes along with the system.

(5) **Transmission :** There are a lot of applications where we need to transmit images. The stages in the transmission of an image over a channel or network are shown in Fig. 2.1.8.

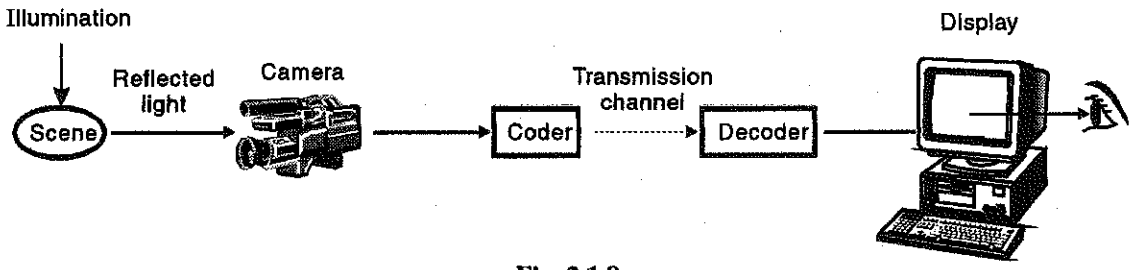


Fig. 2.1.8

The image sequence from the camera is coded into as concise a representation as possible for transmission over the channel.

Most transmission in broadcast television are still in analog form and analog coding methods are used to make it as efficient as possible. NTSE, PAL and SECAM are the 3 major coding systems used in various parts of the world (USA uses NTSE, while India uses PAL). Digital image coding is a high activity area. In image processing; it is concerned with efficient transmission of images over digital communication channels. A variety of indigenous ideas have been developed in recent years. Coding is influenced by the type of the channel used to carry the image signals. Several different types of transmission channels are encountered in practice including cables, terrestrial radio, satellites, and optical fibres.

Gamma : In the image transmission chain, there are many elements which exhibit a non-linear behaviour. If x is the input and y is the output, then

$$y = cx^{\gamma}$$

c = Constant

γ = Gamma of the device

The camera, the display device and eye all have non-unity gammas (all are non-linear). Hence to make sure that the perceived grey scale in the displayed image is correct, it is necessary to insert an additional compensating, non-linear device called the gamma corrector.

We have discussed a lot of things so far and it is advisable at this stage to have a branch diagram of these topics.

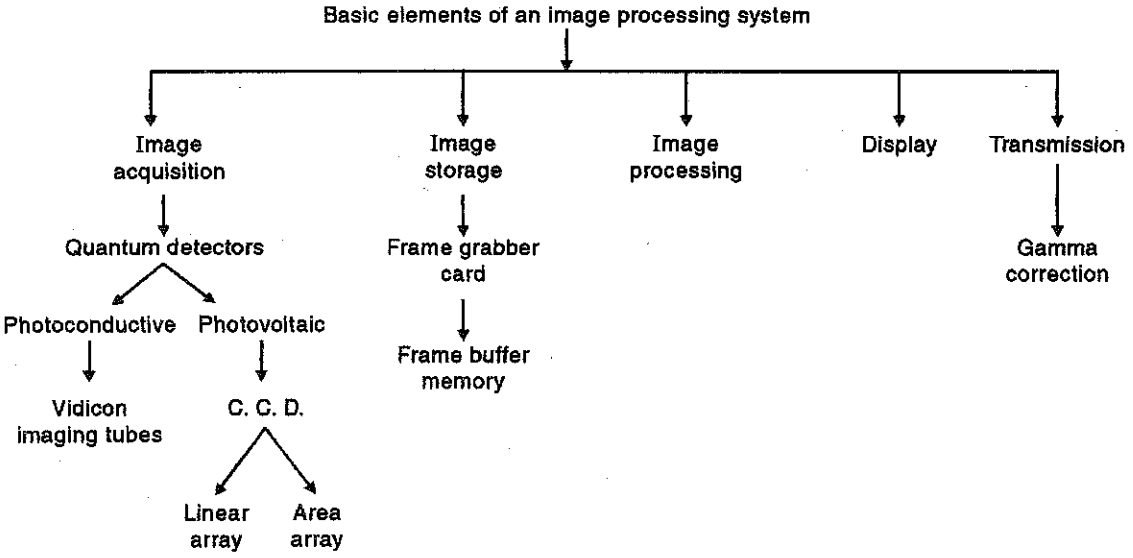


Fig. 2.1.9

2.2 Image Types

It was stated earlier that images are 2-dimensional functions. Refer Fig. 2.2.1.

Images can be classified as follows :

(1) Monochrome images (Binary images).

(2) Grey scale images.

(3) Colour (24-bit) images.

(4) Half-toned images.

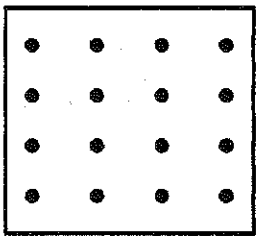


Fig. 2.2.1

(1) Monochrome Image : In this, each pixel is stored as a single bit (0 or 1). Here, 0 represents black while 1 represents white. It is a black and white image in the strictest sense. These images are also called bit mapped images. In such images, we have only black and white pixels and no other shades of grey. Refer Fig. 2.2.2.

Fig. 2.2.2

(2) Grey Scale Image : Here each pixel is usually stored as a byte (8-bits). Due to this, each pixel can have values ranging from 0 (black) to 255(white). Grey scale images, as the name suggests have black, white and various shades of grey present in the image. Refer Fig. 2.2.3.

Fig. 2.2.3

(3) **Colour Image (24-bit) :** Colour images are based on the fact that a variety of colours can be generated by mixing the three primary colours viz, Red, Green and Blue, in proper proportions. In colour images, each pixel is composed of RGB values and each of these colours require 8-bits (one byte) for its representation. Hence each pixel is represented by 24-bits [R(8-bits), G(8-bits), B(8-bits)].

A 24-bit colour image supports 16, 777, 216 different combination of colours.

Colour images can be easily converted to grey scale images using the equation

$$X = 0.30\,R + 0.59\,G + 0.11\,B \qquad ...(2.2.1)$$

An easier formula that could achieve similar results is

$$X = \frac{R + G + B}{3} \qquad ...(2.2.2)$$

MATLAB code for converting a colour image to a grey scale image is shown below.

```
%% Converting a colour image to a grey level image %%
clear
clc
im = imread('lily.tif');
[row col byt]=size(im);
a = im(:,:,1);        % Red plane
b = im(:,:,2);        % Green plane
c = im(:,:,3);        % Blue plane
a = double(a);
b = double(b);
c = double(c);
for x = 1:1:row
for y = 1:1:col
            new(x,y) = (a(x,y)+b(x,y)+c(x,y))/3;
new1(x,y) = 0.3*a(x,y)+0.59*b(x,y)+0.11*c(x,y);
end
end
figure(1)
imshow(uint 8(im))
figure(2)
imshow(uint 8(new))
figure(3)
imshow(uint8(new1))
```

Matlab has an inbuilt command for conversion *rgb2gray*.

(4) **Half Toning :** It is obvious that a grey scale image definitely looks better than the monochrome image as it utilizes more grey levels. But there is a problem in hand. Most of the printers that we use (inkjet, lasers, dot matrix) are all bi-level devices. i.e., they have only a black cartridge and can only produce two levels (black on a white back-ground). In fact, most of the printing jobs are done using bi-level devices.

You have all read newspapers at some point of time (hopefully). The images do look like grey level images. But if you look closely, all the images generated are basically using black colour. Refer Fig. 2.2.4.

Fig. 2.2.4

Even the images that you see in most of the books (including this one) are generated using black colour on a white background. In spite of this we do get an illusion of

seeing grey levels. The technique to achieve an illusion of grey levels from only black and white levels is called half-toning.

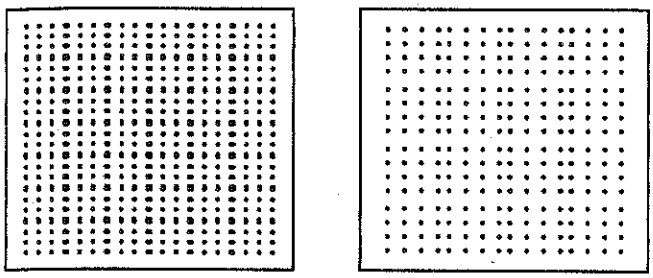


Fig. 2.2.5

The human eye integrates the scene that it sees. Consider a simple example. Consider two squares of say 0.03 × 0.03 sq. inch. One of these squares contains a lot of black dots while the other square contains fewer black dots. When we look at these squares from a distance, the two squares give us a perception of 2 different grey levels. This integration property of the eye is the basis for half toning. In this, we take a matrix of a fixed size and depending on the grey level required, we fill the matrix with black pixels.

Let us take an example.

Consider a 3 × 3 matrix. This matrix can generate an illusion of 10 different grey levels when viewed from a distance.

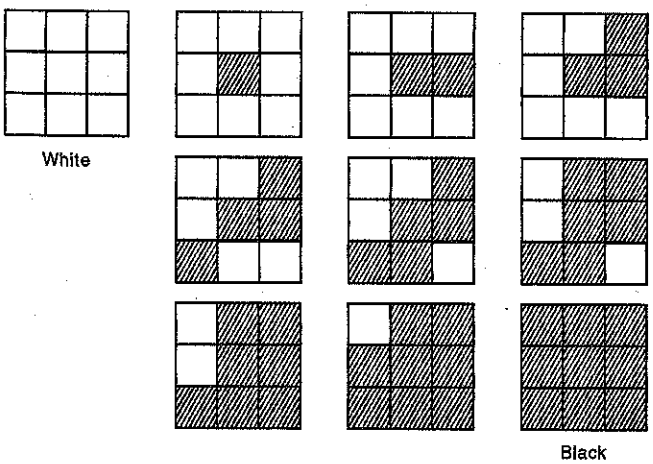


Fig. 2.2.6

Here, the first block represents white, the last block represents black and all the other blocks represent intermediate values of grey. So in this case, while printing, if we encounter grey level 0, we plot 9 pixels which are all white. If we encounter grey level 1, we plot 9 pixels of which only one pixel is black and so on.

As is evident a 3 × 3 matrix will generate 10 different grey levels.

The dots that are placed in the 3 × 3 matrix example can be in any order. But we need to follow two rules.

(1) Dots should be arranged in such a manner so that they don't form straight lines. Lines are very obvious to the viewer and hence should be avoided.

Example, suppose in an image we have 4 consecutive grey levels of value 3, and if we use a code as shown in Fig. 2.2.7, the half-toned image would look like Fig. 2.2.8.

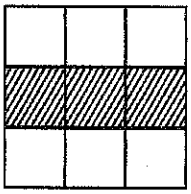


Fig. 2.2.7

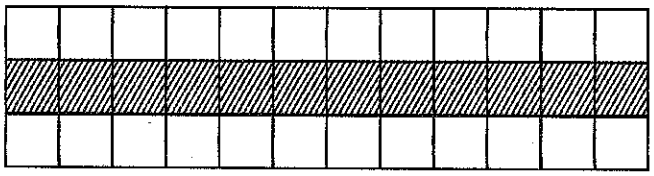


Fig. 2.2.8

It can be seen that it does not look like a grey level. It looks like a straight black line. Hence lines should be avoided while defining the matrices.

(2) If a pixel in a particular matrix is black for grey level i, it should be black for all further levels j > i. This reduces false contouring.

Half-toning gives excellent results and we do perceive a grey level image just by using black pixels on a white background.

The logic to implement a half-toned image from a grey level image is given below.

(1) Define the size of the half-toned matrices based on the number of grey levels the original image has.

(2) Generate the matrices starting from all white pixels to all black pixels.

(3) Read the original image. For every grey level value read, plot the corresponding matrix.

Remember, the physical size of the half-toned image will always be bigger than the original image as for every single grey level value, we output an entire matrix.

2.3 Image File Formats

Images obtained from the camera are stored on the host computer using different formats. A file format is a structure which defines how information is stored in the file and how that information is displayed on the monitor. There are numerous image file formats which are available. Of these only a few of them can be used universally. By universally it means, they can be understood by different operating systems.

Some of the image formats that can be used on either Macintosh or PC platforms are;

BMP (Bit Mapped Graphic Image)	JPEG (Joint Photographic Expert Group)
TIFF (Tagged Image File Format)	EPS (Encapsulated Post Script)
GIF (Graphic Interchange Format)	PICT (Macintosh Supported)

TIFF, which is one of the most well known formats was developed in 1986 and in its many versions is a standard image format for a bit-mapped graphics image. The TIFF format is also a data compression technique for monochrome as well as colour images. Images seen on the Internet sites are normally TIFF/GIF images simply because they occupy less space. GIF uses a form of Huffman coding.

BMP images developed by Microsoft can save both monochrome as well as colour images. All the wall papers that your computer has are BMP images. Similarly when we work in Paint Brush, we can save our work as a BMP image. The quality of BMP files is very good but they occupy a lot of memory space. PICT is a general purpose format supported by Macintosh and used for storing bit-mapped images. EPS is file format of the post script page description language and is device independent. This simply means that images can readily be transferred from one application to another. However EPS images can only be printed on post script compatible printers.

JPEG (Joint Photographic Expert Group) is the name of the committee that developed an image format which used compression algorithms.

JPEG images are compressed images which occupy very little space. It is based on the Discrete Cosine Transform-DCT (explained in chapter of Image Transforms). JPEG images use lossy compression algorithms which result in some loss of original data and hence the quality of JPEG images is not as good as BMP images.

Although these formats differ in technical details, they share structural similarities.

The Fig. 2.3.1 shows the typical organisation of information encoded in an image file.

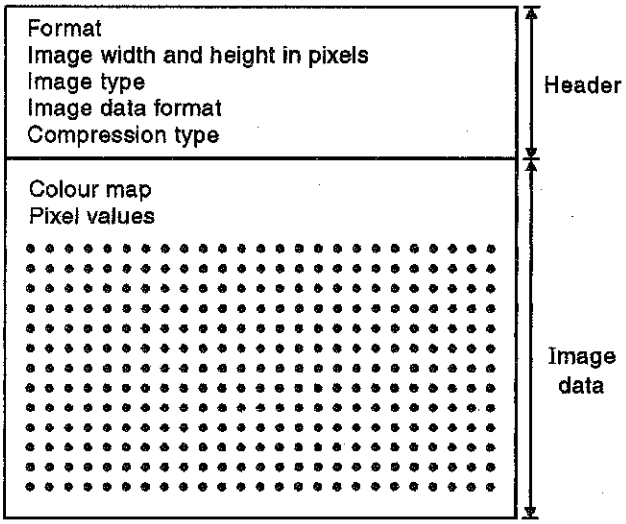


Fig. 2.3.1

The image file consists of two parts;

(1) Header (2) Image data

The header file gives us the information about the kind of image. The header file, begins with a binary code or ASCII string which identifies the format being used. The width and height of the image are given in number of pixels. Common image types include binary images, 8-bit grey scale images, 8-bit colour and 24-bit colour images. The image data format specifies the order in which pixel values

are stored in the image data section. A commonly used order is left to right and top to bottom. Image data format also specifies if the RGB values in the image are interlaced. By interlaced we mean that the RGB values of each pixel stay together consecutively followed by the three colour components for the next pixel i.e.,

R G B R G B R G B .....

1st Pixel 2nd Pixel .....

If the RGB values are not interlaced, the values of one primary colour for all pixels appear first, then the values of the next primary colour followed by the values of the third primary colour. Thus the image data is in the form.

R R R R..... G G G G..... B B B B.....

The compression type in the header indicates whether the image data is compressed using algorithms such as Run length encoding . Apart from these, there are a few more formats which we come across while dealing with images.

PGM (Portable Grey Map) - PGM is used to represent grey level images.

PBM (Portable Bit Map) - PBM is used to represent binary images.

PPM (Portable Pixel Map) - PPM is used to represent RGB colour images.

These formats are distinguished by 2 character signatures as shown below.

Signature	Image type	Storage type
P1	Binary (PBM)	ASCII
P2	Grey scale (PGM)	ASCII
P3	RGB (PPM)	ASCII
P4	Binary (PBM)	Raw bytes
P5	Grey scale (PGM)	Raw bytes
P6	RGB (PPM)	Raw bytes

Most often the storage type is ASCII.

The header of a PGM, PBM or PPM file begins with the appropriate signature followed by a blank line. There is a comment line after the signature which starts with the # character. Next in the header are the width and height of the image as ASCII decimal values. PBM files have no further header information. PGM and PPM files contain a third integer value, again in ASCII decimal form, representing the maximum allowable pixel value. From this value, we could find out the number of bits allocated for each pixel. For example, if the maximum allowable pixel value is 255 then we know that each bit is represented by 8 bits ($2^8 = 256$).

Fig. 2.3.2 shows a 8 × 8 grey scale image and its representation as a ASCII PGM file.

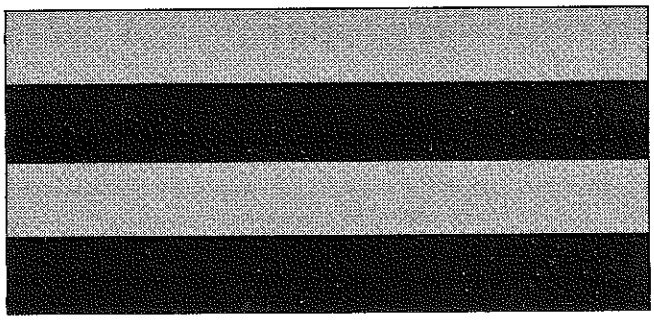


Fig. 2.3.2

Header

```
P2
#A PGM image
 8   8   255
120 120 120 120 120 120 120 120
120 120 120 120 120 120 120 120
 33  33  33  33  33  33  33  33
 33  33  33  33  33  33  33  33
120 120 120 120 120 120 120 120
120 120 120 120 120 120 120 120
 33  33  33  33  33  33  33  33
 33  33  33  33  33  33  33  33
```

Just by observing the header, we know, that it is a PGM file. P2 indicates that it is a grey level image stored is ASCII. The 8 × 8 255 below the comment line indicates that the size of the image is 8 × 8 and can have 256 grey levels i.e., each value is represented by 8-bits.

Summary

Image acquisition forms the first step in any image processing system. In this chapter, a basic block diagram of the image processing system is introduced. Each block in the system is explained in detail. Important devices such as

Vidicon cameras, CCD cameras and the frame grabber card are explained using simple illustrations. Importance of each is stated. Applications of line array CCD's and area array CCD's are presented. Basic concepts of image formats are also explained.

Review Questions

Q. 1 List the basic elements of an image processing system.

Q. 2 Explain working of a Vidicon camera.

Q. 3 Explain line array and area array CCD cameras.

Q. 4 Write a short note on frame grabber card.

Q. 5 Explain the concept of half toning.

Q. 6 Explain image formats.

Q. 7 Advantages of CCD over Vidicon.

Q. 8 Explain the basics of quantum detector.

Q. 9 Explain gamma of a camera.

Q. 10 How many grey levels will a half toned image have ? Explain in detail.

Chapter Ends...

❑❑❑

CHAPTER 3

Sampling and Quantization

3.1 Introduction

We know that an image is basically a 2-dimensional representation of the 3-dimensional world. We have also studied that images can be acquired using a Vidicon or a CCD camera or using scanners. The basic requirement for image processing is that images obtained be in the digital format. For example, one cannot work with or process photographs on identity cards unless he/she scans it using a scanner.

The scanner digitizes the photograph and stores it on the hard disk of the computer. Once this is done, one can use image-processing techniques to modify the image as per requirement. In a Vidicon too, the output which is in analog form needs to be digitized in order to work with the images. To cut a long story short, to perform image processing, we need to have the images on the computer. This will only be possible when we digitize the analog pictures.

Now that we have understood the importance of digitization, let us see what this term actually means.

The process of digitization involves two steps :

(1) Sampling

(2) Quantization

In other words, Digitization = Sampling + Quantization.

3.2 Sampling

Sampling is the process of converting a continuous scene into a discrete set of numbers. A scene in the real world consists of infinite points. When we click a picture of this scene using a camera, these infinite points get mapped on to a finite set of points based on the number of pixels on the sensor of the camera. Hence if our camera is of 12 Mega pixels, then the real world scene will be stored as 12×10^6 samples in the memory. This is known as sampling. Sampling determines the spatial resolution of the digitized image and depends on the Mega pixel of the camera.

3.3 Quantization

The values obtained by sampling a continuous function usually comprise of an infinite set of real numbers ranging from a minimum to a maximum depending upon the sensors calibration. These values must be represented by a finite number of bits usually used by a computer to store or process any data. In practice, the sampled signal values are represented by a finite set of integer values. This is known as quantization. Rounding of a number is a simple example of quantization.

With these concepts of sampling and quantization, we now need to understand what these terms mean when we look at an image on the computer monitor.

Higher the spatial resolution of the image, greater is the sampling rate. Similarly, higher the grey level resolution (tonal resolution), more are the number of quantized levels.

Hence spatial resolution gives us an indication of the sampling while grey level resolution (tonal resolution) gives us an indication of the quantization.

Spatial resolution ⟶ Sampling

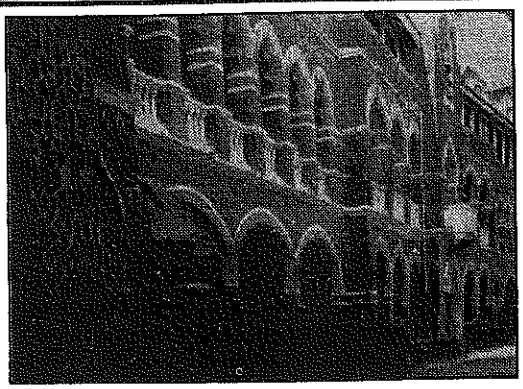
Grey level resolution⟶ Quantization

We have already stated that an image can be considered as a 2-D array. Image f(x,y) is arranged in the form of N × M array

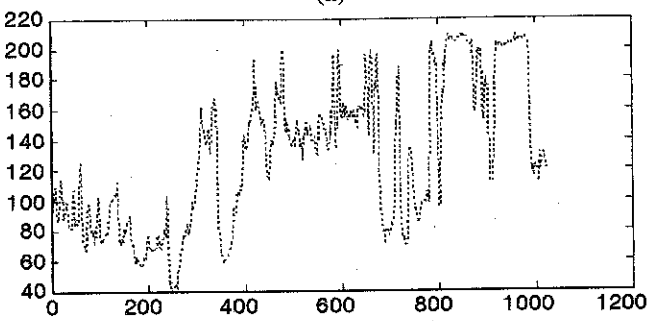
$$f(x,y) = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,M) \\ f(2,1) & f(2,2) & \cdots & f(2,M) \\ f(3,1) & f(3,2) & \cdots & f(3,M) \\ \vdots & & & \vdots \\ f(N,1) & f(N,2) & \cdots & f(N,M) \end{bmatrix}_{N \times M}$$

Hence every image that is seen on the monitor, is actually this matrix. Each element of the matrix is called a *pixel*. Never forget this. Whenever we see an image on the screen of the computer it is actually a matrix which consists of N × M pixels and each pixel is considered to be a sample. Hence more the pixels, more the samples, higher the sampling rate and hence better the spatial resolution. The value of each pixel is known as the grey level.

The computer understands only ones and zeros. Hence these grey levels need to be represented in terms of zeros and ones. If we have two bits to represent the grey levels, only 4 different grey levels (2^2) can be identified viz. 00, 01 10, 11, where 00 is black, 11 is white and the other two are different shades of grey. Similarly, if we have 8 bits to represent the grey levels, we will have 256 grey levels (2^8). Hence more the bits, more are the grey levels and better is the tonal clarity (quantization). The total size of the image is N × M × m, where m is the number of bits used.



(a)



(b)

Fig. 3.3.1 : The first row of the image is plotted. The x-axis is the number of samples (pixels) in the row (sampling) while the y-axis is the grey level of each sample (quantization)

Now, comes the obvious question. As we know, more the samples and the bits, better is the image. What then should be the ideal values of sampling and quantization. This answer will vary from image to image. Given below is a table of sampling and the quantization values. As the sampling and the quantization increase, the number of bits required to store the image increases tremendously.

The clarity increases, but storage space required increases too. We hence need to get a trade-off between the two. For simplicity, we consider a square matrix of size N × N.

Table 3.3.1 : Number of storage bits for various values of N and m

N \ m	1	2	3	4	5	6	7	8
32	1,024	2,048	3,072	4,098	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,385	32,768	49,152	65,536	81,290	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1,024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608

We shall see the effects of reducing quantization levels and reducing spatial resolution separately.

MATLAB code for reducing quantization levels

```
%% Effects of reducing the quantization values %%
clear all
clc
a=imread('zebra.tif');
a=double(a);
a=a+1;
b=max(max(a)); %% gives us the maximum value in the image
i=input ('how many bits do you want 1 2 4 8');
j=b/(2^i); %% since total number of bits is equal to 2^i
F=floor(a/(j+1));
F1=(F*255)/max(max(F)); % normalizing %
figure(1)
imshow(uint8(a))
figure(2)
imshow(uint8(F1))
```

(a) Original image

(b) Image using 4-bits

(c) Image using 2-bits

(d) Image using 1-bit

Fig. 3.3.2

Comparing the images, we see that "false contouring" takes place as we reduce the number of grey levels.

MATLAB code for reducing spatial resolution.

```
%% Down sampling
%% To see the effects of reducing the number of samples %%
clear all
```

```
clc
a= imread('deepa.tif')
[row,col]=size(a);
i=1; j=1;
for x=1:2:row
        for    y=1:2:col
                c(i,j)=a(x,y);
                        j=j+1;
                end
j= 1; %% This needs to be done else the value of j goes on
increasing %%
i= i+1;
end
figure(1),imshow(a)
figure(2),imshow(c)
figure(3),
imagesc(a),colormap(gray)
figure(4)
imagesc(c),colormap(gray)
```

(a) Original image (b) Down sampled
Fig. 3.3.3 Contd..

(c) Down sampled image displayed after zooming to match the size of the original image
Fig. 3.3.3

It is clear from the images that the resolution reduces as number of samples reduce. To compare and understand the actual effects, we plot them together. To make sure that they appear to be of the same size, we upsample the second image. Comparing Fig. 3.3.3(a) and Fig. 3.3.3(c) we see that the second image has a "Checker board" pattern due to the reduction of samples.

3.4 Isopreference Curves

We have seen the effects of reducing N and m in the preceding topic. We still do not know what is the ideal value of N (sampling resolution) and m (tonal resolution) for images.

An early study by T.S. Huang in 1965 attempted to quantify experimentally the effects on image quality produced by varying N and m simultaneously. There, different types of images were shown to a group of people. The first image was that which did not have a lot of details for example a woman's face. The second image was one which had intermediate amount of information for example a small group standing together and the third image was one which had a lot of details example an image of a crowd. In these images, N and m were varied. Observers were then asked to rank them according to their subjective quality. These results were summarised in the form of isopreference *curves* in the N-m plane.

Points lying on an isopreference curve corresponded to image of equal subjective quality. From the isopreference curves Huang concluded that images with large amount of details require few grey levels. Since the isopreference curve of the crowd is near vertical, it means that for a fixed value of N, the image is nearly independent of m.

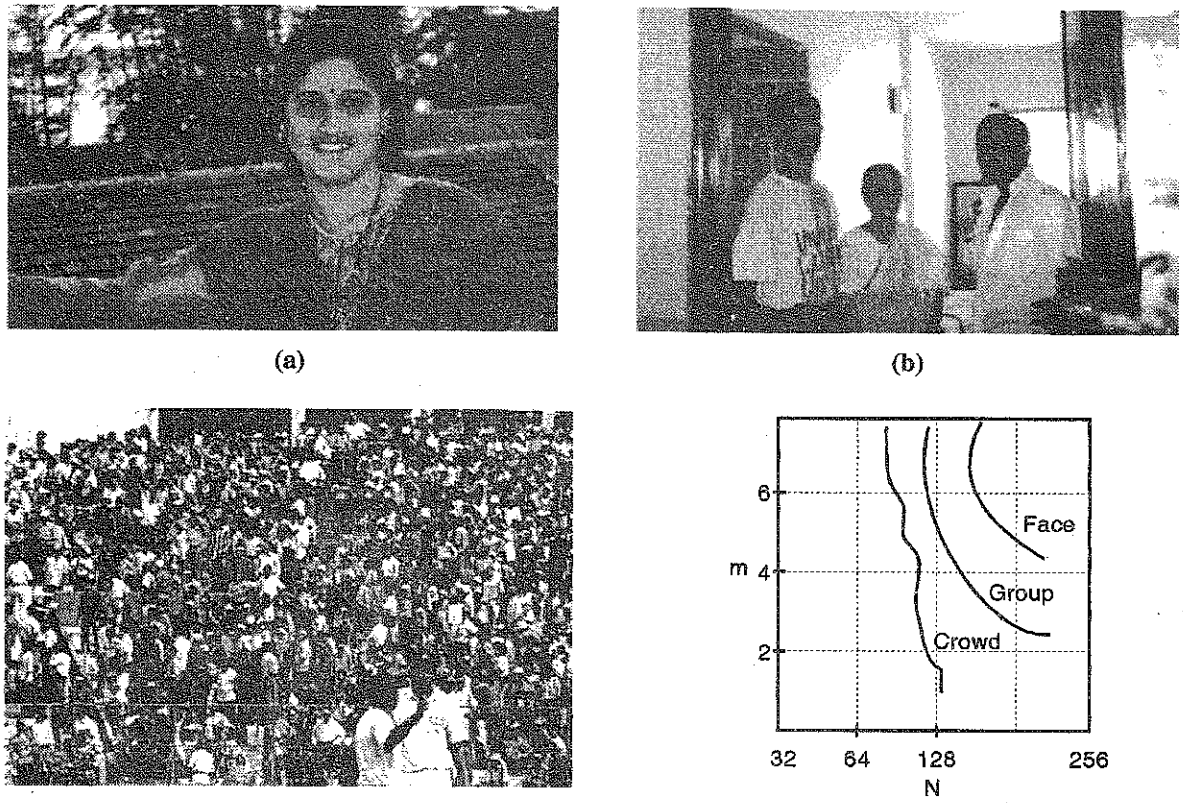


Fig. 3.4.1

3.5 Physical Resolution

By now we know that a digital image, or image for short, is composed of discrete pixels. These pixels are arranged in a row and column fashion to form a rectangular picture area. Clearly, the total number of pixels in an image is a function of size of the image and the number of pixels per unit area (example: inch) in the horizontal as well as the vertical direction. The number of pixels per unit length is referred to as the resolution of the displaying device (most of the deskjet printers have 670 dpi i.e., 670 dots per inch.).

Thus a 3 × 2 inch image at a resolution of 300 pixels per inch would have a total of 540,000 pixels!!

In most books as well as in this book, image size is given as the total number of pixels in the horizontal direction times the total number of pixels in the vertical direction (example : 128 × 128, 512 × 512, 640 × 480 ....). Although this convention makes it relatively straight forward to gauge the total number of pixels in an image, it does not specify the physical size of the image or its resolution as defined in the paragraph above.

A 640 × 480 image would measure 6.66 inches by 5 inches when displayed or printed at 96 pixels per inch. On the other hand, it would measure only 1.6 inch by 1.2 inch when displayed or printed at 400 pixels per inch.

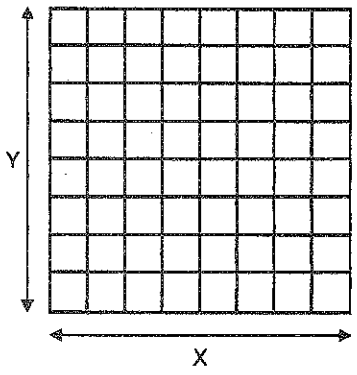


Fig. 3.5.1

Another term that we need to understand is the Aspect ratio.

The ratio of the image's width to its height, measured in unit length or number of pixels is referred to as its aspect ratio. Both, a 3×3 inch image and a 128×128 image have the same aspect ratio of 1.

$$\text{Aspect ratio} = \frac{X}{Y}$$

Ex. 3.5.1

Compute the physical size of a 640×480 image when printed by a printer at 240 pixels per inch.

Soln. :

Since we have 240 pixels per inch, the physical size of the image is $\frac{640}{240}$ by $\frac{480}{240}$.

Ex. 3.5.2

If we want to resize a 1024×768 image to one that is 600 pixels wide with the same aspect ratio as the original image, what should be the height of the resized image?

Soln. :

We know

$$\text{Aspect ratio} = \frac{\text{Width}}{\text{Height}}$$

For the original image the Aspect ratio is $= 1024/768 = 1.33$

Now for the resized image, we want the same aspect ratio but a width of 600 pixels.

$$\text{Height} = \frac{\text{Width}}{\text{Aspect ratio}} = \frac{600}{1.33}$$

$$= 451$$

Hence the resized image will be 600×451.

Ex. 3.5.3

A common measure of transmission for digital data is the baud rate, defined as the number of bits transmitted per second. Transmission is accomplished in packets consisting of a start bit, a byte (8-bits) of information and a stop bit.

(a) How many minutes would it take to transmit a 1024×1024 image with 256 grey levels if we use a 56 k baud modem ?

(b) What would be the time required if we use a 750 k band transmission line ?

Soln. :

(a) Since we have 256 grey levels, we need 8-bits for representing each pixel.

Along with these 8-bits, we also have a start bit and a stop bit.

Hence we have (8 + 2) bits per pixel.

$\therefore$ Total number of bits for transmission is

$$N = 1024 \times 1024 \times 10$$

$$N = 10485760\text{-bits}$$

These bits are transmitted at 56 k baud.

$$\therefore \text{Time taken} = \frac{N}{56 \times 10^3}$$

$$\therefore \text{Time taken} = 187.25 \text{ sec} \approx 3.1 \text{ minutes}$$

(b) Now if we use a faster transmission line of 750 k baud rate, then

$$\text{Time taken to transmit the image } \frac{N}{750 \times 10^3} = \frac{10485760}{750 \times 10^3}$$

$$= 13.98 \text{ sec} \approx 14 \text{ sec}$$

Ex. 3.5.4

We have a C. T. image which we want to transmit through a voice grade telephone line. The CT image is of size 512×512 and has 256 grey levels. During transmission each byte is preceded by a start bit and succeeded by a stop bit. How many minutes will it take to transmit the image ?

(Voice grade telephone lines can transmit 9600-bits/sec)

Soln. :

256 grey level required mean 8-bits.

$\therefore$ one packet = [8 + 02] bits = 10 bits

$\therefore$ image size $= 512 \times 512 \times 10$

$\therefore$ image size = 2621440

$\therefore$ time taken = $\frac{512 \times 512 \times 10}{9600}$

$\therefore$ time taken = 27-3.06 sec $\approx$ 4.5 min

Ex. 3.5.5

A medical image has a size of 8 × 8 inches. The sampling resolution is 5 cycles/mm. How many pixels are required ? Will an image of size 256 × 256 be enough ?

Soln. :

1 cycle/mm represents 1 line pair/mm. (1 black line and 1 white line). To represent this line pair, we need atleast 2 pixels (more the better).

$\therefore$ 1 cycle/mm = 1 line pair/mm = 2 pixels/mm

Since the sampling resolution of the given example is 5 cycles/mm, it implies that the spatial resolution is atleast 10 pixels/mm.

The given medical image is of size 8 inch × 8 inch i.e., 203.2 mm × 203.2 mm (since 1 inch = 25.4 mm).

In each mm there are 10 pixels. Hence the spatial resolution of the medical image will be 2032 × 2032 pixels.

Therefore to represent this image, we would require 2032 × 2032 pixels.

256 × 256 pixels will not be enough to represent the given image.

Summary

This chapter deals with converting a scene that is continuous in time and space into an image that can be processed by computers. The technique of converting a continuous signal into a discrete one is called digitization and is explained here. Digitisation comprises of two steps viz. sampling and quantization. Concepts of sampling are explained in the one-dimensional as well as in the two dimensional domain. Relationship between the time domain and the frequency domain i.e., the Fourier domain are presented here. Aliasing, which forms an important pitfall of sampling, has been introduced and the importance of the Nyquist criteria has been explained. The concept of an image being stored as a matrix has been introduced here. Visualising an image as a matrix is of utmost importance as all the image processing algorithms are based on this concept. Spatial resolution and grey level resolution are also explained using MATLAB codes.

Review Questions

Q. 1 Explain sampling and quantization.

Q. 2 Explain the effects of reducing sampling and quantization.

Q. 3 Explain isopreference curves.

Q. 4 Explain non-uniform sampling.

Q. 5 Compute the physical size of a 480 × 360 image when printed by a printer at 320 dpi.

Chapter Ends...

❑❑❑

CHAPTER

4

Image Enhancement in Spatial Domain

4.1 Introduction

Image enhancement is one of the first steps in image processing. As the name suggests, in this technique, the original image is processed so that the resultant image is more suitable than the original for specific applications i.e. the image is enhanced. Image enhancement is a purely subjective processing technique. By subjective we mean that the desired result varies from person to person. An image enhancement technique used to process images might be excellent for a person, but the same result might not be good enough for another. It is also important to know at the outset that image enhancement is a cosmetic procedure i.e. it does not add any extra information to the original image. It merely improves the subjective quality of the images by working with the existing data.

Image enhancement can be done in two domains :

(1) The spatial domain

(2) The frequency domain.

Let us start with explaining what is meant by the spatial domain, discuss the various methods of spatial domain enhancement and then move on to discuss the frequency domain technique in chapter of Image Enhancement in Frequency Domain.

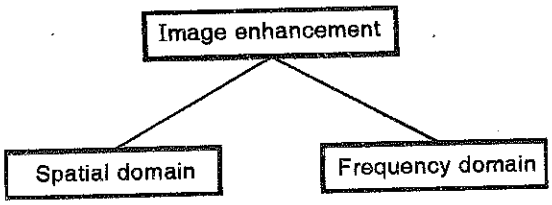


Fig. 4.1.1

4.2 Spatial Domain Methods

The term spatial domain means working in the given space, in this case, the image. It implies working with the pixel values or in other words, working directly with the raw data.

Let f(x, y) be the original image where f is the grey level value and (x, y) are the image coordinates. For a 8-bit image, f can take values from 0 - 255 where 0 represents black, 255 represents white and all the intermediate values represent shades of grey.

In a image of size 256 × 256 as shown in the Fig. 4.2.1.

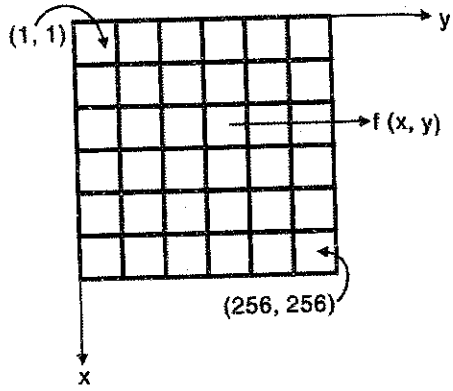


Fig. 4.2.1

Note : We always consider the axis as shown in Fig 4.2.1. The x-axis is taken in the downward vertical direction, while the y-axis is taken in the horizontal direction. The reason for taking this inverted coordinate system is to make sure that the image uses the standard matrix coordinates that we have been using right from our school. You will remember the first value of the matrix is always at the top left corner while the last value of the matrix is at the right bottom corner.

The modified image can be expressed as

$$g(x, y) = T[f(x, y)] \quad \ldots(4.2.1)$$

Here f(x, y) is the original image and T is the transformation applied to it to get a new modified image g(x, y). For all spatial domain techniques it is simply T that changes. The general equation remains the same.

To summarize, spatial domain techniques are those, which directly work with the pixel values to get a new image based on Equation (4.2.1).

Spatial domain enhancement can be carried out in two different ways :

(1) Point processing

(2) Neighbourhood processing

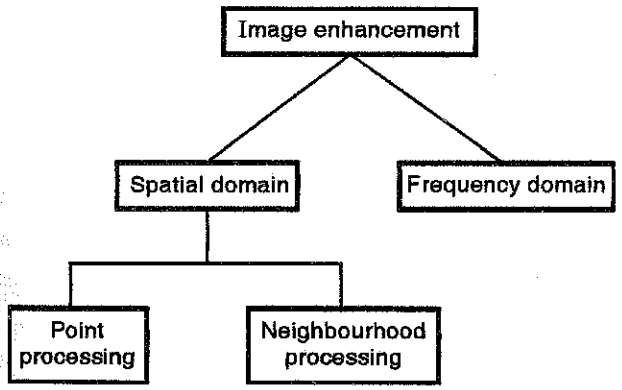


Fig. 4.2.2

4.3 Point Processing

In point processing, we work with single pixels i.e. T is 1×1 operator. It means that the new value f(x, y) depends on the operator T and the present f(x, y). This statement will be clear as we start giving some examples.

Some of the common examples of point processing are

(1) Digital negative

(2) Contrast stretching

(3) Thresholding

(4) Grey level slicing

(5) Bit plane slicing

(6) Dynamic range compression

(7) Power law transformation

Before we proceed to explain the following examples, it is important to understand the identity transformation. The identity transformation is given in the Fig. 4.3.1.

In the Fig. 4.3.1, the solid line is the transformation T. The horizontal axis represents the grey scale of the input image (r) and the vertical axis represents the grey scale of the output image (s). It is called an identity transformation because it does not modify the new image at all !!. As seen, the grey level 10 is modified to 10, 125 to 125 and finally 255 to 255. In general $s_1 = r_1$. Fig. 4.3.1 will help us understand the point processing techniques better.

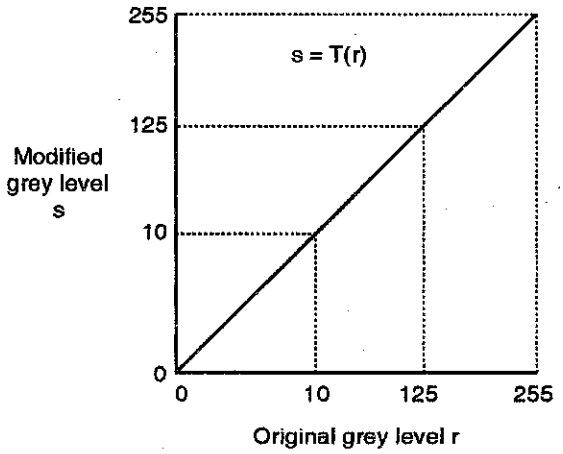


Fig. 4.3.1

(1) **Digital negative :** Digital negatives are useful in a lot of applications. A common example of digital negative is the displaying of an X-ray image. As the name suggests, negative simply means inverting the grey levels i.e. black in the original image will now look white and vice versa. The Fig. 4.3.2 is the digital negative transformation for a 8-bit image.

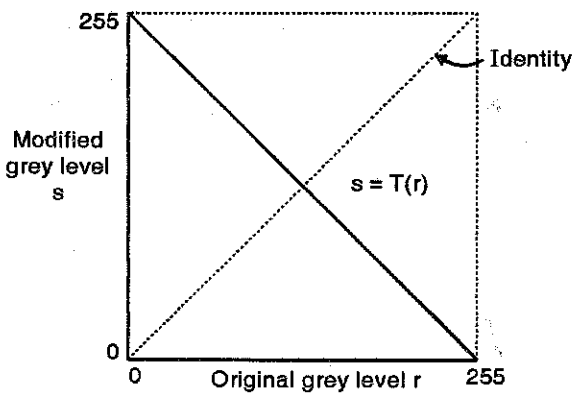


Fig. 4.3.2

The digital negative image can be obtained by using a simple transformation given by,

$$s = 255 - r \ (r_{max} = 255)$$

Hence when

$$r = 0, s = 255 \text{ and when } r = 255, s = 0.$$

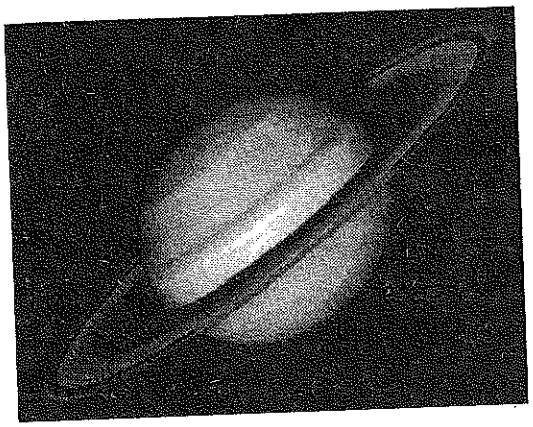
In general,

$$s = (L - 1) - r \qquad \ldots(4.3.1)$$

Here L is the number of grey levels. (256 in this case)

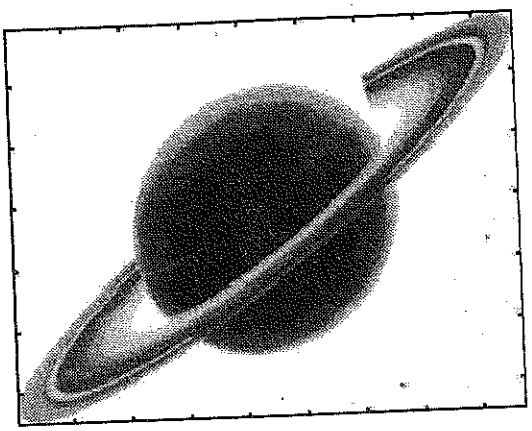
MATLAB program for finding the digital negative

```
%% MATLAB code to calculate negative %%
clear all
clc
aa=imread ('saturn.tif');
a=double (aa)
c=255; % for a 8-bit image %
b=c-a;
figure(1)
colormap(gray)
imagesc(a)
figure(2)
colormap(gray)
imagesc(b)
```



(a) Original image

Fig. 4.3.3 Continued...



(b) Digital negative

Fig. 4.3.3

(2) **Contrast stretching :** Many times we obtain low contrast images due to poor illuminations or due to wrong setting of the lens aperture. The idea behind contrast stretching is to increase the contrast of the images by making the dark portions darker and the bright portions brighter.

Fig. 4.3.4 shows the transformation used to achieve contrast stretching. In the Fig. 4.3.4, the dotted line indicates the identity transformation and the solid line is the contrast stretching transformation. As is evident from the Fig. 4.3.4, we make the dark grey levels darker by assigning a slope of less than one and make the bright grey levels brighter by assigning a slope greater than one. One can assign different slopes depending on the input image and the application. As was mentioned, image enhancement is a subjective technique and hence there is no one set of slope values that would yield the desired result.

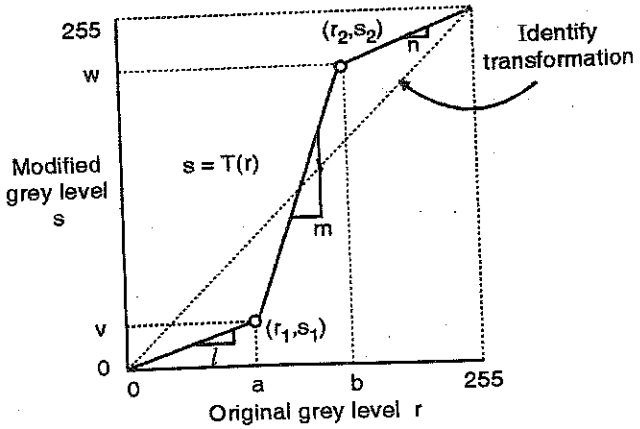


Fig. 4.3.4 : Original grey level r

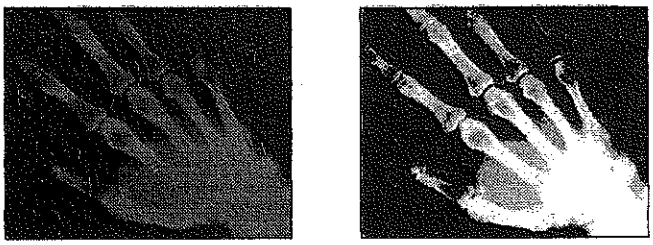
The formulation of the contrast-stretching algorithm is given below.

$$s = \begin{cases} l \cdot r & 0 \le r < a \\ m \cdot (r-a) + v & a \le r < b \\ n \cdot (r-b) + w & b \le r < L-1 \end{cases} \quad \ldots(4.3.2)$$

Where l, m and n are the slopes. It is clear from the figure that l and n are less than one while m is greater than one. The contrast stretching transformation increases the dynamic range of the modified image.

MATLAB program for contrast stretching

```
%% Contrast stretching of an image %%
% Slopes taken are 0.5, 2 and 0.5 %
clear all;
clc;
a=imread('xray1.tif')
a=double(a);
[row col]=size(a);
LT=input('Enter the lower threshold value:');
UT = input('Enter the upper threshold value:');
for x=1:1:row
for y=1:1:col
 if a(x,y)<=LT
   b(x,y)=0.5*a(x,y);
 else if a(x,y)<=UT
        b(x,y)=2*(a(x,y)-LT)+0.5*LT;
 else b(x,y)=0.5*(a(x,y)-UT)+0.5*LT+2*(UT-LT);
  end
end
  end
end'
subplot(2,1,1)
imshow(uint8(a))
title('Original Image');
subplot(2,1,2)
imshow(uint8(b))
title('Image after Contrast Stretching')
```



(a) Original image **(b) Contrast stretched image**

Fig. 4.3.5

(3) Thresholding : Extreme contrast stretching yields thresholding. If we observe the contrast stretching diagram closely we notice that if the first and the last slope are made zero, and the centre slope is increased, we would get a thresholding transformation i.e. if $r_1 = r_2$, $s_1 = 0$ and $s_2 = L - 1$, we get a thresholding function. It is shown in Fig. 4.3.6.

The formula for achieving thresholding is as follows.

$$\left.\begin{array}{ll} s = 0; & \text{if } r \le a \\ s = L-1; & \text{if } r > a \end{array}\right\} \quad \ldots(4.3.3)$$

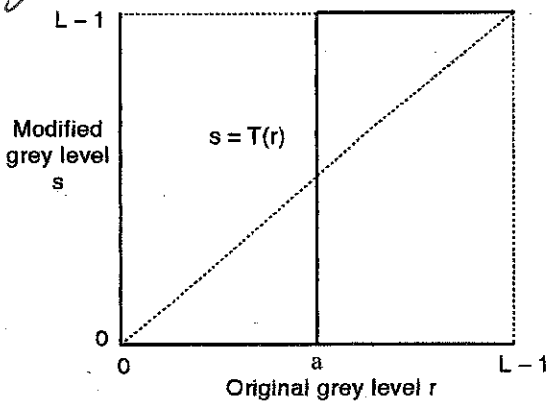


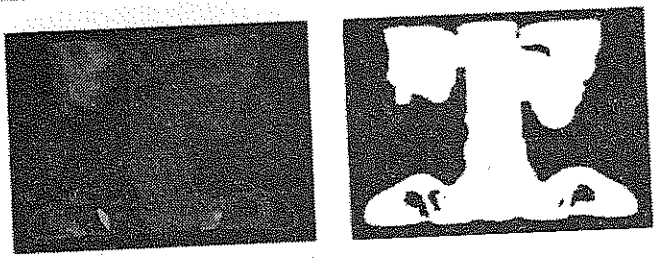
Fig. 4.3.6 : Original grey level r

Where L is the number of grey levels.

As mentioned earlier, image enhancement being a subjective phenomena, the value of a will vary from image to image and from person to person. The objective is to identify the region that he or she is interested in. An important thing to note is that a thresholded image has the maximum contrast as it has only black and white grey values.

MATLAB program for thresholding

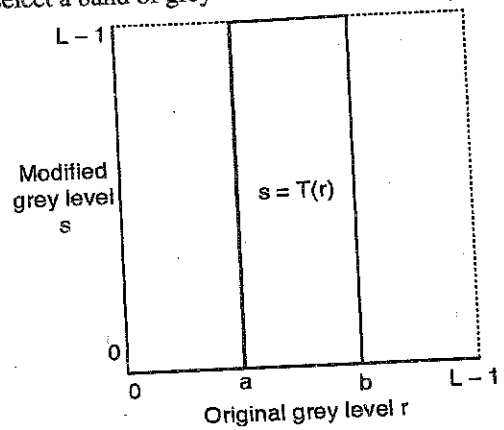
```
%%% Thresholding %%%
clear all
clc
p=imread('spine.tif');
a=p;
[row col]=size(a);
T=input('Enter value of Threshold :')
for i=1:1:row
    for j=1:1:col
        if(p(i,j)<T)
            a(i,j)=0;
        else
            a(i,j)=255;
    end
end
end
figure(1),imshow(p),
figure(2),imshow(a)
```



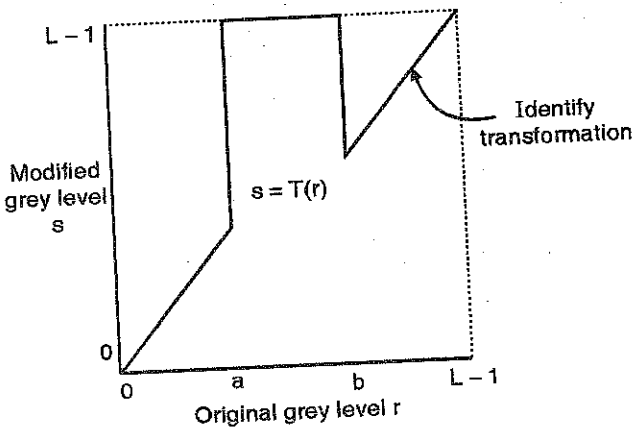
(a) Original image of spine **(b) Image obtained using threshold**

Fig. 4.3.7

(4) Grey level slicing (Intensity slicing) : What thresholding does is it splits the grey level into two parts. At times, we need to highlight a specific range of grey values like for example enhancing the flaws in an X-ray or a CT image. In such circumstances, we use a transformation known as grey level slicing. The transformation is shown in the Fig. 4.3.8(a). It looks similar to the thresholding function except that here we select a band of grey level values.



(a) Slicing without background



(b) Slicing with background

Fig. 4.3.8

This can be implemented using the formulation

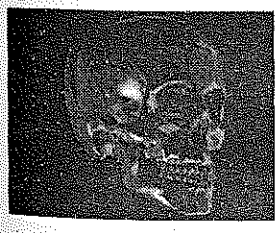
$$\left.\begin{array}{ll} s = L-1; & \text{if } a \le r \le b \\ s = 0; & \text{otherwise} \end{array}\right\} \quad \ldots(4.3.4)$$

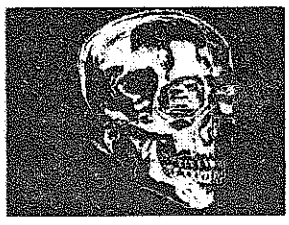
This method is known as Grey level slicing without background. This is because in this process, we have completely lost the background. In some applications, we not only need to enhance a band of grey levels but also need to retain the background. This technique of retaining the background is called as Grey-level slicing with background. The transformation is as shown in the Fig. 4.3.8(b). The formulation for this is

$$\left.\begin{array}{ll} s = L-1; & \text{if } a \le r \le b \\ s = r; & \text{otherwise} \end{array}\right\} \quad \ldots(4.3.5)$$

MATLAB program for grey level slicing with and without background is given below,

```
%% Grey level slicing without background %%
clear all
clc
p=imread ('skull.tif');
z=double (p);
[row,col]=size(z)
for i=1:1:row
        for j=1:1:col
            if((z(i,j)>50))&&(z(i,j)<150)
            z(i,j)=255;
            else
                z(i,j)=0;
            end
        end
end
figure (1); % .....original image.
imahow (p)
figure (2);  % .....gray level slicing without background
imshow (uint8(z))
```

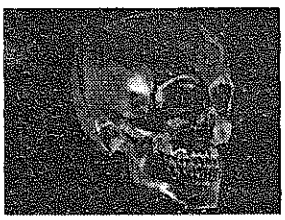


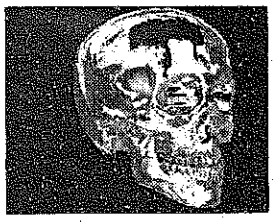


(a) Original image **(b) Gray level slicing without background**

Fig. 4.3.9

```
%% Grey level slicing with backgound %%
clear all
clc
p=imread('skull.tif');
z=double(p);
[row col]=size(p);
for i=1:1:row
    for j=1:1:col
        if(z(i,j)>50))&&(z(i,j)<150)
            z(i,j)=255;
        else
            z(i,j)=p(i,j);
        end
    end
end
figure (1); %------- original image.
imshow(p)
figure (2); %------- grey level slicing with background
imshow (uint8(z))
```





(a) Original image **(b) Grey level slicing with background**

Fig. 4.3.10

(5) **Bit plane slicing :** In this technique, we find out the contribution made by each bit to the final image. As mentioned earlier, an image is defined as say a 256 × 256 × 8 image. In this, 256 × 256 is the number of pixels present in the image and 8 is the number of bits required to represent each pixel. 8-bits simply means 28 or 256 grey levels.

Now each pixel will be represented by 8-bits. For example black is represented as 00000000 and white is

represented as 11111111 and between them, 254 grey levels are accommodated. In bit plane slicing, we see the importance of each bit in the final image. This can be done as follows. Consider the LSB value of each pixel and plot the image using only the LSBs. Continue doing this for each bit till we come to the MSB. Note that we will get 8 different images and all the 8 images will be binary.

Ex. 4.3.1

Given a 3 × 3 image, plot its bit planes.

1	2	0
4	3	2
7	5	2

Soln. :

Since 7 is the maximum grey level, we need only 3-bits to represent the grey levels.

Hence we will have 3-bit planes. Converting the image to binary we get,

001	010	000	1	0	0	0	1	0	0	0	0
100	011	010	0	1	0	0	1	1	1	0	0
111	101	010	1	1	0	1	0	1	1	1	0

Binary image LSB plane Middle bit plane MSB plane

```
%% MATLAB code for bit extraction %%
clear all
clc
a=imread('warne.tif');
a=double(a);
r=input('which bit image do you want to see 1=MSB 8=LSB');
[row col]=size(a);
for x=1:1:row
    for y=1:1:col
    c=dec2bin(a(x,y),8); % converts decimal to binary
    d=c(r);
    w(x,y)=double(d); %% since w is a char and cannot be plotted
if w(x,y)==49 %% since double of d will be either 49 or 48
            w(x,y)=255;
        else
            w(x,y)=0;
    end
end
end
figure(1)
imshow(uint8(a))
figure(2)
imshow(uint8(w))
```

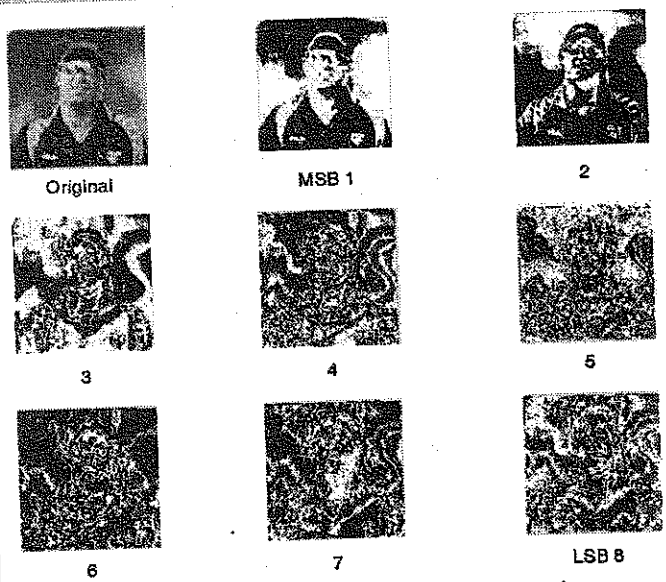


Fig. 4.3.11 : Eight images, each representing contribution of a single bit

Observing the images we come to the conclusion that the higher order bits contain majority of the visually significant data, while the lower bits contain the suitable details in the image. Bit plane slicing can hence be used in

matlab R2017a

image compression. We can transmit only the higher order bits and remove the lower order bits. Bit plane slicing is also used in steganography.

(6) **Dynamic range compression (Log transformation)**

At times, the dynamic range of the image exceeds the capability of the display device. What happens is that some pixel values are so large that the other low value pixels get obscured. A simple day-to-day example of such a phenomena is that during daytime, we cannot see the stars. The reason behind this is that the intensity of the sun is so large and that of the stars is so low that the eye cannot adjust to such a large dynamic range. In image processing, a classic example of such large differences in grey levels is the Fourier spectrum (will be discussed in detail in the frequency domain enhancement technique). In the Fourier spectrum only some of the values are very large while most of the values are too small. The dynamic range of pixels is of the order of 10^6. Hence, when we plot the Fourier spectrum, we see only small dots, which represent the large values.

Something needs to be done to be able to see the small values as well. This technique of compressing the dynamic range is known as dynamic range compression. We all know that the log operator is an excellent compressing function. Hence the dynamic range compression is achieved by using a log operator. C is the normalisation constant.

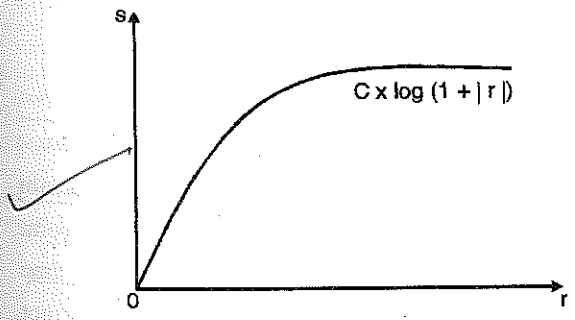
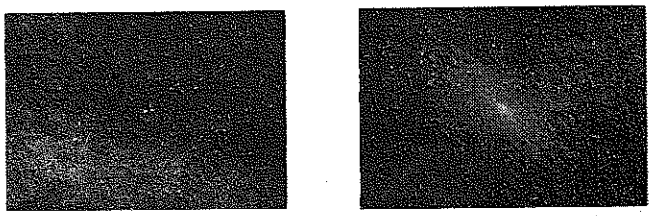


Fig. 4.3.15

MATLAB program for dynamic range compression

```
%% Dynamic range compression %%
clear all
clc
aa=imread('saturn.tif');
a=double(aa) ;
[row,col]=size(a);
for x=1:1:row
for y=1:1:col
c(x,y)=a(x,y)*((-1) ^ (x+y)); %% Needed to center the transform
  end
end
d=abs(fft2(c));
d_log=log(1+d);
%%% Plotting
figure(1)
colormap(gray)
imagesc(d)
figure(2)
colormap(gray)
imagesc(d_log)
```



(a) (b)

Fig. 4.3.16 : Dynamic range compression

(7) **Power law transformation**

The basic formula for power-law transformation is

$$g(x, y) = c \times f(x, y)^{\gamma}$$

It can also be written as $s = c\, r^{\gamma}$ ...(4.3.6)

Here c and γ are positive constants. The transformation is shown Fig. 4.3.17 for different values of γ which is also called the gamma correction factor. We observe that by changing the value of gamma, we obtain a family of transformation curves.

Non-Linearities encountered during image capturing, printing and displaying can be corrected using gamma correction. Hence gamma correction is important if the image needs to be displayed on the computer. The power law transformation can also be used to improve the dynamic range of an image. Given below is the MATLAB code for power transformation. The final image has been normalized to 0 - 255 range.

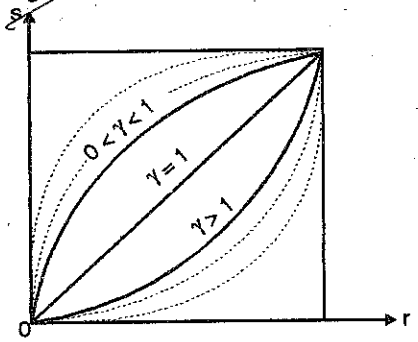


Fig. 4.3.17 : Gamma correction factor

```
%% Power transformation %%
clear all
clc
img1=imread('test.tif');
[row,col]=size(img1);
gamma=input('Enter the correction factor: ');
img=double(img1);

for i=1:row
    for j=1:col
        nuimg(i,j)=img(i,j) ^ gamma
    end
end
numax=max(max(nuimg));
numin=min(min(nuimg));
n=255/(numax-numin);
for i=1:row
    for j=1:col
        nuimg1(i,j)=n*(nuimg(i,j)-numin); % Normalisation
    end
end
nuimg2=uint8(nuimg1);
subplot(2,1,1)
imshow(img1)
title('Original image')
subplot(2,1,2)
imshow(nuimg2)
title('Image after power transformation')
```

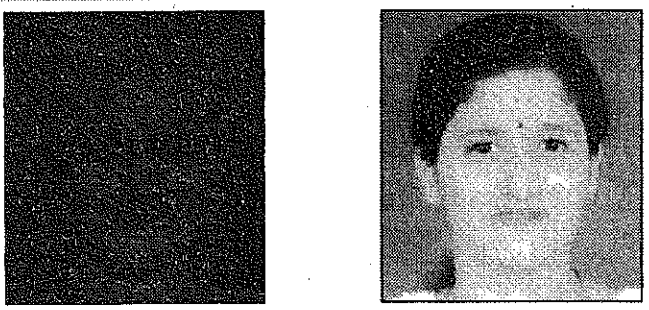


Fig. 4.3.18

4.4 Neighbourhood Processing

This, as mentioned before, is also a spatial domain technique in image enhancement. Unlike the point processing techniques where we consider one pixel at a time and modify it depending on our requirement, here we not only consider a pixel but also its immediate neighbours.

To cut a long story short, we change the value of the pixel f(x, y) based on the values of its 8 neighbours as shown in Fig. 4.4.1. Instead of a 3 × 3 neighbourhood, we could also use a 5 × 5 or a 7 × 7..... neighbourhood.

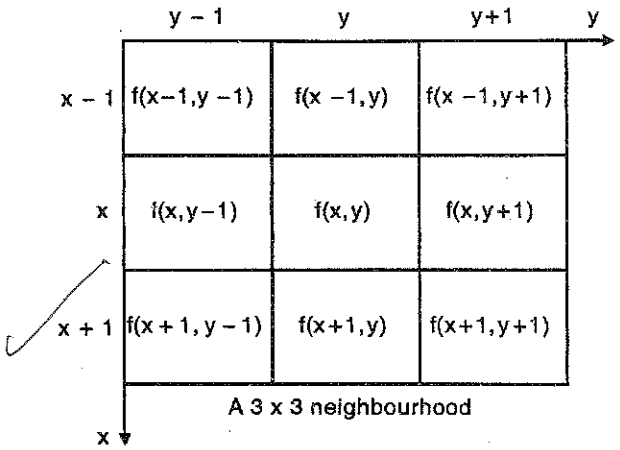


Fig. 4.4.1 : A 3 × 3 neighbourhood

w1	w2	w3
w4	w5	w6
w7	w8	w9

Fig. 4.4.2 : 3 × 3 mask

There are a lot of things that can be achieved by neighbourhood processing which are not possible with point processing. Fig. 4.4.2 shown is called a mask or a window or a template. To achieve neighbourhood processing, we place this 3 × 3 (it could also be a 5 × 5 or a 7 × 7...) mask on the image, multiply each component of the mask with the corresponding value of the image, add them up and place the value that we get, at the center.

This operation is the same as convolution. In the convolution operation, we have two signals. Of the two signals, we take one, flip it and then move it across the other signal step by step. The same thing is done here. We don't need to flip the mask as it is symmetric.

If f is the original image and g is the modified image, then

$$g(x, y) = f(x-1, y-1) \times w1 + f(x-1, y) \times w2 + f(x-1, y+1) \times w3 + f(x, y-1) \times w4 + f(x, y) \times w5 + \ldots\ldots\ldots + f(x+1, y+1) \times w9 \quad \ldots(4.4.1)$$

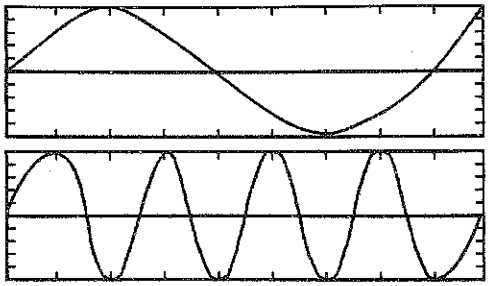


Fig. 4.4.3

Once g(x, y) is calculated, we shift the mask by one step towards the right to the next pixel. Now w5 coincides with f(x, y + 1). One of the important operations that can be achieved using neighbourhood processing is that of image filtering. We can perform low pass, high pass and band pass filtering using neighbourhood operations. Before explaining the procedure of performing filtering on images, it is imperative to understand what we mean by frequencies in an image !! We are all well versed with frequencies in 1-dimensional signals. Given two signals, we can easily distinguish between the lower frequency signal and the higher frequency signal. Refer Fig. 4.4.3.

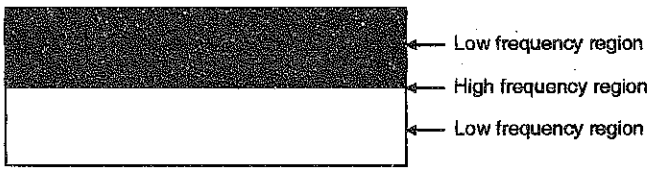


Fig. 4.4.4

We can conclude that the lower signal is of a much higher frequency, by checking the number of oscillations. That is, higher the frequency, greater are the number of oscillations. If the two signals represent voltages, then, how fast the voltages change is an indication of the frequency. The same concepts can be applied to images. In images, instead of voltages, we have grey levels. If the grey levels change slowly over a region, it is considered to have low frequency, whereas, if the grey levels change very rapidly, that region is considered to have high frequencies. Hence in images, regions where the grey levels change slowly are low frequency areas and regions which have abrupt grey level changes, are high frequency areas. Always remember this.

In most of the images, the background is considered to be a low frequency region, whereas the edges are considered to be high frequency regions. Hence low pass filtering implies removing (bluring) the edges while high pass filtering implies removing the background.

4.4.1 Low Pass Filtering (Smoothing)

Low pass filtering as the name suggests removes the high frequency content from the image. It is used to remove noise present in the image. Noise, is normally a high frequency signal and low pass filtering eliminates the noise. Before proceeding to explain the low-pass filtering technique, we shall spend some time discussing the types of noise that are fairly common in images.

4.4.2 Noise

The principal sources of noise in a digital image arise during image acquisition and during transmission. No matter how much care one takes, some amount of noise always creeps in. Based on the shapes (Probability Density Functions) of the noise, they are classified as :

(1) Gaussian noise (2) Salt and pepper noise

(3) Rayleigh noise (4) Gamma noise

(5) Exponential noise (6) Uniform noise

Of these, the first two are more common than the others. We shall explain the Gaussian and salt and pepper noise in this chapter.

(1) Gaussian noise : The Probability Density Function (PDF) of Gaussian noise is given by the formula,

$$p(z) = \frac{1 \cdot e^{-(z-\mu)^2 / 2\sigma^2}}{\sqrt{2\pi}\,\sigma}$$

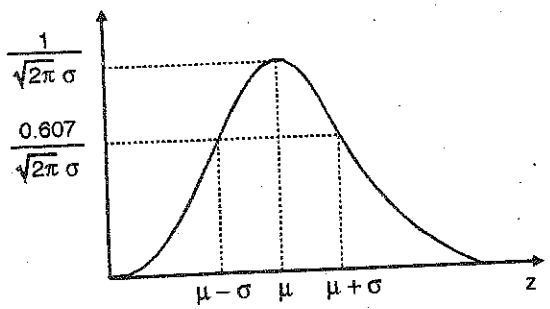


Fig. 4.4.5

$z \rightarrow$ Grey level

$\mu \rightarrow$ Mean of average value of z

$\sigma \rightarrow$ Standard deviation

$\sigma^2 \rightarrow$ Variance

If we plot this function, we notice that

70% of its value lies in the range $[(\mu - \sigma), (\mu + \sigma)]$ and

95% of its value lies in the range $[(\mu - 2\sigma), (\mu + 2\sigma)]$

Gaussian noise has a maximum value at μ and then it starts falling off.

Consider the image shown in Fig. 4.4.6.

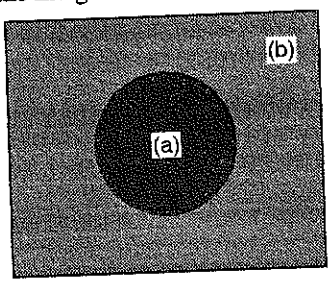


Fig. 4.4.6

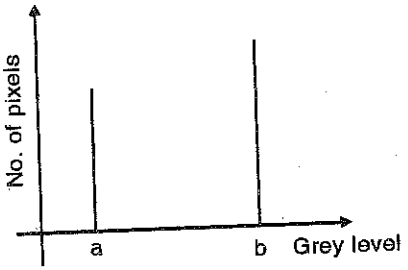


Fig. 4.4.7

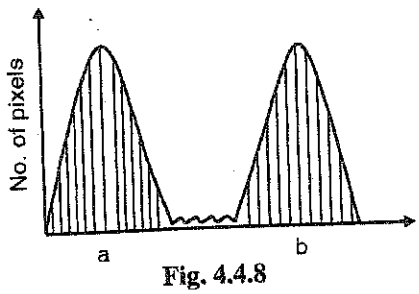


Fig. 4.4.8

There are two constant regions in the image. Hence, the histogram of the image is as shown in Fig. 4.4.7. If in this image, Gaussian noise creeps in, the histogram gets modified as shown in Fig. 4.4.8. Gaussian noise arise in an image due to factors such as circuit noise, sensor noise poor illumination and high temperature.

4.4.3 Salt and Pepper Noise

The PDF of the salt and pepper noise (bipolar noise) is

$$p(z) = \begin{cases} P_a; & \text{for } z = a \\ P_b; & \text{for } z = b \\ 0; & \text{otherwise} \end{cases}$$

If P_a or P_b is zero, this noise is called unipolar noise. The PDF of salt and pepper is shown in Fig. 4.4.9.

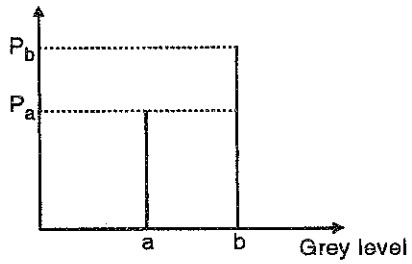


Fig. 4.4.9

Generally a and b are black and white grey levels respectively. Hence for a 8-bit image, a = 0, b = 255 because of which the noise is called salt (white) and pepper (black). Some books refer to it as speckle noise.

Take the same image as the one taken for the Gaussian example.

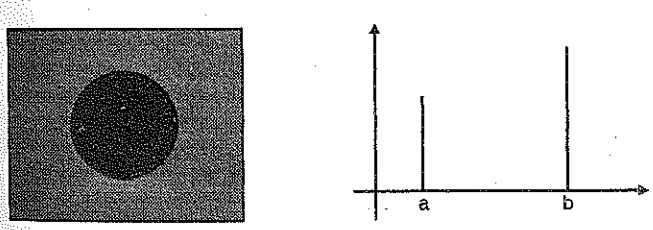


Fig. 4.4.10 (a)

When salt and pepper creeps in, the image looks like

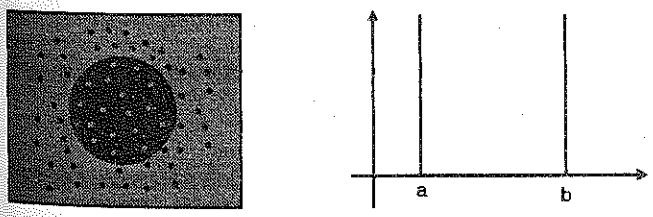


Fig. 4.4.10 (b)

Salt and pepper noise creeps into images in situations where quick transients, such as faulty switching take place.

4.4.4 Low Pass Averaging Filter

If an image has Gaussian noise present in it, we use a low pass averaging filter to eliminate the noise. The frequency response of the low-pass filter along with its spatial response is shown in Fig. 4.4.11. This relationship will be proved in chapter of Image Enhancement in Frequency Domain.

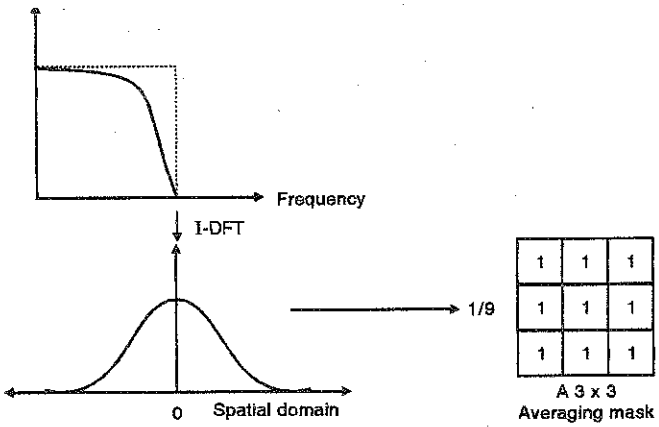


Fig. 4.4.11

From the spatial response we generate the mask that would give us the low pass filtering operation. One important thing to note from the spatial response is that all the coefficients are positive. The standard low pass averaging mask (3 × 3) is given above. As the name suggests, each element of the mask is the average value.

We could also use a 5 × 5 or a 7 × 7 mask as per our requirement.

1/25

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

A 5 x 5 Averaging mask

Fig. 4.4.12

Let us take an example of a 3 × 3 mask on a pseudo-image and see how it eliminates the edges. Consider a 8 × 8 size image. It is clear that the image has a single

edge between 10 and 50. To get rid of this edge (high frequency) we use a 3×3 averaging mask.

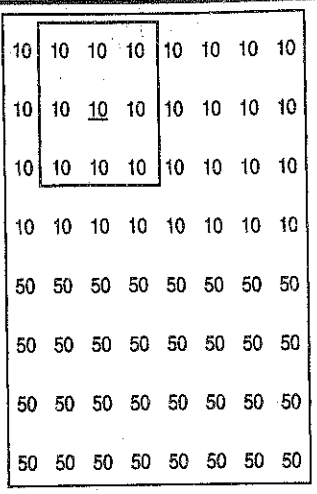
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

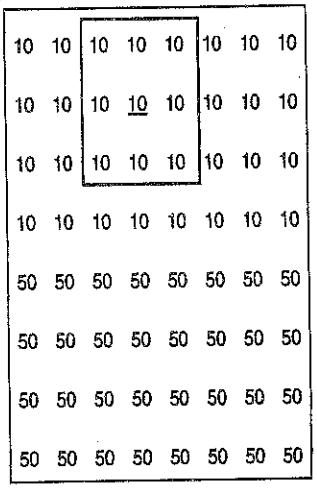
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

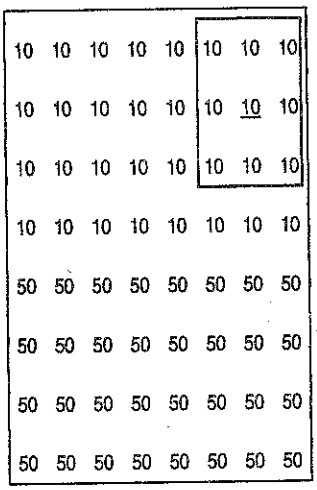
We place a 3×3 mask on this image. We start from the left hand top corner. We cannot work with the borders and hence are normally left as they are.

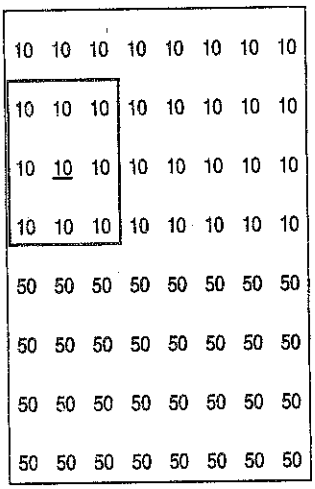
We then multiply each component of the image with the corresponding value of the mask. Since all the nine values of the image are 10, the average is also ten and the centre pixel (the underlined pixel) remains ten. We now shift the mask towards the right till we reach the end of the line and then move it downwards. Some of the mask positions are shown here.

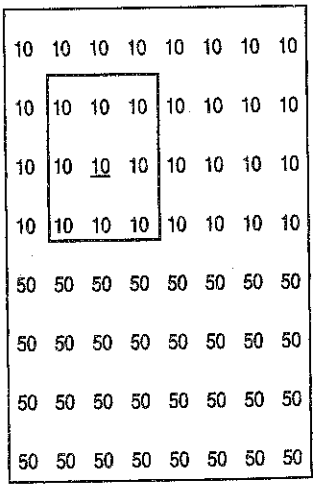
Note : The resultant should be written in a new matrix (image).

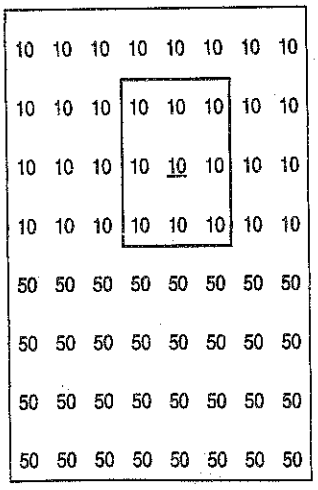












10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

The last one in the sequence being

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

The result of convolving the image with the 3 × 3 averaging mask is shown.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
23.3	23.3	23.3	23.3	23.3	23.3	23.3	23.3
36.6	36.6	36.6	36.6	36.6	36.6	36.6	36.6
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

As we notice, the low frequency regions have remained unchanged, but the sharp edge between 10 and 50 has become blurred. The transition has reduced. Now from 10, the grey level changes to 23.3 (23 after rounding off), from 23 it changes to 36.6 (36 after rounding off) and finally from 36 it changes to 50. Hence we can say that the sharp edge has become blurred. Check for yourself, what would happen if you took a bigger mask, say, 5 × 5. (You will have to take a bigger image to test your results).

These kinds of averaging filters are excellent when the image contains Gaussian noise. It achieves filtering by blurring the noise. Some of the other low pass averaging masks are shown.

$\frac{1}{6}\times$

0	1	0
1	2	1
0	1	0

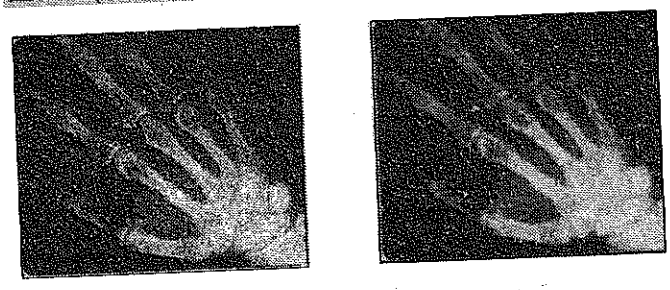
$\frac{1}{10}\times$

1	1	1
1	2	1
1	1	1

MATLAB program for low pass filtering along with the images with Gaussian noise

```
%% Low pass filter used on an image with Gaussian noise
clear all
clc
aa=imread('xray.tif'); % size 600 × 800
f=double(aa);
ab=imnoise(aa,'gaussian'); % adding noise
a=double(ab);
w=[1 1 1; 1 1 1; 1 1 1]/9
    [row col]=size(a);
for x=2:1:row-1
    for y=2:1:col-1
        a1(x,y)= w(1)*a(x-1,y-1)+w(2)*a(x-1,y)+w(3)*...
        a(x-1,y+1)+w(4)*a(x,y-1)+w(5)*a(x,y)+w(6)*...
        a(x,y+1)+w(7)*a(x+1,y-1)+w(8)*a(x+1,y)+w(9)*...
```

```
            a(x+1,y+1);
      end
end
figure (1)
imshow (uint8 (a))
figure (2)
imshow (uint8 (a1))
```



(a) Original image with Gaussian noise **(b) Low passed image**

Fig. 4.4.13

Generalised MATLAB program for low pass averaging using any mask size

```
%% Low pass filtering of an image using averaging technique
clear all;
clc;
a=imread('blood.tif');
a=double(a);
[row col]=size(a)
m=input('Enter the mask size:');
n=m ^ 2;
 for i=1:1:m
    for j=1:1:m
 w(i,j)=1/n;
    end
end
s=(m+1)/2;
for x=1:1:row
   for y=1:1:col
 b(x,y)=a(x,y);
        end
end
for x=s:1:row-s
     for y=s:1:col-s
 b(x,y)=0;
     end
end
for x=s:1:row-s
   for y=s:1:col-s
for i=1:1:m
     for j=1:1:m
          b(x,y)=a(x-s+i,y-s+j)*w(i,j)+b(x,y);
     end
end
end
end
subplot (3,1,1)
imshow (uint8(a))
title ('Original Image');
subplot (3,1,2)
imshow (uint8(b))
title ('Low Pass Filtered Image');
c=conv2(a,w);
```

```
subplot(3,1,3)
imshow(uint8(e))
title ('Low Pass Filtered Image using MATLAB');
```

4.4.5 Low Pass Median Filtering

The averaging filter removes the noise by blurring it till it is no longer seen. But in the process, it also blurs the edges. Bigger the averaging mask more is the blurring. There are times when the image contains salt and pepper noise. If we use an averaging filter to remove the same, it will blur the noise but it would also ruin the edges. Hence when we need to eliminate salt and pepper noise, we work with a non-linear filter known as the Median filter. They are also called order-statistic filters because their response is based on the ordering or ranking of the pixels contained within the mask.

In this case, we use a mask similar to the averaging filter except that the mask has no values. So it's like working directly with the 8 neighbours of the centre pixel.

The steps to perform median filtering are as follows :

(1) Assume a 3 × 3 empty mask.

(2) Place the empty mask at the left hand corner.

(3) Arrange the 9 pixels in ascending or descending order.

(4) Choose the median from these nine values.

(5) Place this median at the centre.

(6) Move the mask in a similar fashion to the averaging filter.

In median filtering, the grey level of the centre pixel is replaced by the median value of the neighbourhood. Always write the resultant in a new matrix (image). We shall explain median filtering with an example. In the image shown below let 250 be the salt and pepper noise. If we use an averaging filter, the noise would spread out and the edges would also end up getting blurred. If we only want to remove the noise without disturbing the edges, we use a median filter.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	250	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	250	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

Just like the earlier case, we start by placing the mask at the left hand corner. We again ignore the borders.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	250	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	250	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	250	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	250	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

Arranging the 9 pixels in ascending or descending order we get,

(10, 10, 10, 10, 10, 10, 10, 10, 250)

The median is the 5^{th} value since the mask is 3×3 mask. This value is placed at the center. We now move the mask to the new location and continue the procedure. Performing this operation on the entire image, we get the final image :

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

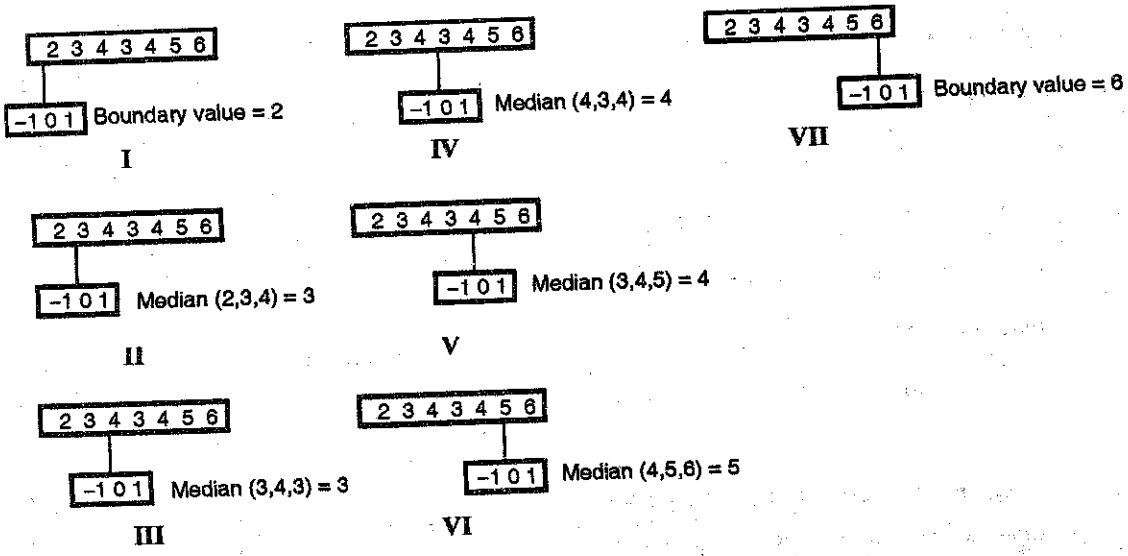
As can be seen, the salt and pepper noise gets eliminated without distorting the edges. Median filters give astonishing results even when the amount of noise is relatively large.

Ex. 4.4.1

If x = {2 3 4 3 4 5 6} and w = {–1 0 1}, perform median filtering.

Soln. :

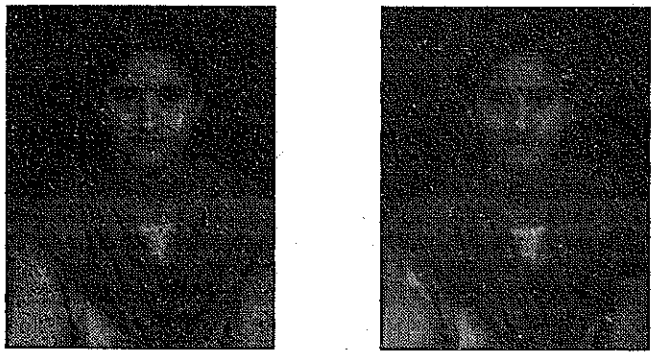
w = {–1, 0, 1} simply means that the size of the mask is 1×3 and the term 0 indicates the position from where the filtering starts.



Hence the final answer is y = {2 3 3 4 4 5 6}. The principal function of median filtering is to force points with distinct intensities to be more like their neighbours.

MATLAB program for median filtering along with the images with salt and pepper noise

```
%% Median filter on image with salt and pepper noise %%
clear all
clc
I=imread ('deepa.tif');
J=imnoise (I, 'salt & pepper',0.02); % Adding noise
a=double(J);
b=a;
[row col]=size(a);
for x=2:1:row-1;
for y=2:1:col-1;
%% To make a 3 × 3 mask into a 1 × 9 mask
a1=[a(x-1,y-1) a(x-1,y) a(x-1,y+1) a(x,y-1) a(x,y)...
a(x,y+1) a(x+1,y-1) a(x+1,y) a(x+1,y+1)];
a2=sort(a1);
med=a2(5); % the fifth value is the median
b(x,y)=med
end
end
figure(1)
Imshow(uint8(J))
figure(2)
imshow(uint8(b))
```

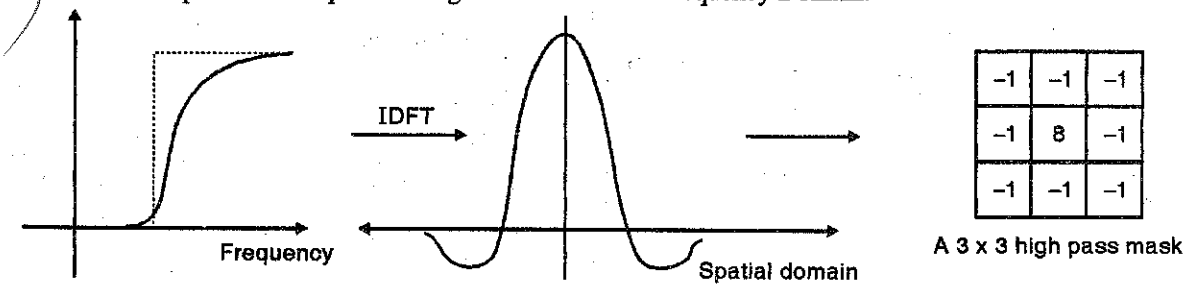


(a) Original image with salt and pepper noise **(b) Median filtered image**

Fig. 4.4.14

4.5 Highpass Filtering

Highpass filtering eliminates the low frequency regions while retaining or enhancing the high frequency components. An image, which is high-passed, would have no background (as background are low frequency regions) and would have enhanced edges. Hence high pass filters are used to sharpen blurred images. Once we have understood as to how a mask moves over the entire image for the averaging filter, the same applies to the high pass filter as well. All that needs to be changed are the mask coefficients. The frequency response and the spatial response of a high pass filter are shown in Fig. 4.5.1. This will be proved in chapter of Image Enhancement in Frequency Domain.



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

A 3 x 3 high pass mask

Fig. 4.5.1

As seen from the spatial response, it is clear that the mask coefficients have to be such that they have a positive value at the centre and negative values at the periphery.

One of the high pass masks is shown in Fig. 4.5.1.

The important thing to note is that the sum of the coefficients of the high pass mask has to be equal to zero. This is because, when we place this mask over the low frequency regions, the result should be zero.

Let us take an example of a pseudo-image.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100

By now we know what to expect when we do a high pass operation. The background, which is low frequency, gets eliminated while the edges get enhanced.

We move the mask in a similar fashion as in the low pass filter example. The output that we get is shown below. Note that the background has been eliminated and we get all zero values. This is because the sum of the mask coefficients is zero.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
–270	–270	–270	–270	–270	–270	–270	–270
270	270	270	270	270	270	270	270
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

There are two issues to be dealt with, when working with high pass masks.

(1) As we can see, there are negative values in the output image. Pixel values *cannot be negative*. They always start with zero. Hence we need to get rid of the negative values. If we consider the mod operation, do you think it would solve the problem ? NO. The reason for that is –270 is a value that is lower than zero, that is, it is supposed to be darker than the darkest value i.e. zero. If we now take the mod value i.e. MOD [–270], we get +270, which is definitely not darker than the zero value. What would happen is that all large negative values would be shown as bright spots. Hence taking the mod of negative values will distort the image grey levels. A simple way to get rid of this problem is to let all negative values be zero. Hence –270 would be written as 0.

(2) The values of the original image at the edge are very large and there is a tendency of them going out of range due to the centre weight of +8. We always encounter this problem in practice. To eliminate this problem, we use a mask with a scaling function.

Hence for practical purposes we use a mask shown below.

$\frac{1}{9}$

–1	–1	–1
–1	8	–1
–1	–1	–1

3×3

High pass mask

Note that 1/9 is simply a scaling function unlike the low pass mask in which the 1/9 term is a part of the averaging operation. In this case we could also work with 1/8 or 1/7 depending on the image. Some of the other high pass masks are also shown below.

0	–1	0
–1	4	–1
0	–1	0

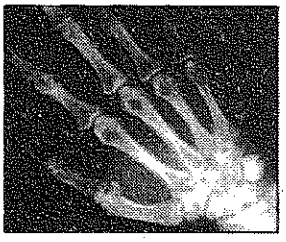
–1	–2	–1
–2	12	–2
–1	–2	–1

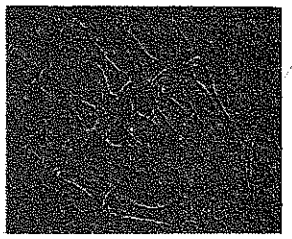
The result using the first mask and putting negative values as zeros is

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
30	30	30	30	30	30	30	30
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

MATLAB program for high pass filtering

```
%% High pass filtering %%
clear all
clc
aa=imread ('xray.tif');
a=double (aa);
[row col]=size(a);
w=[-1 -1 -1; -1 8 -1; -1 -1 -1]
for x=2:1:row-1
    for y=2:1:col-1
        a1(x,y)= w(1)*a(x-1,y-1)+w(2)*a(x-1,y)+w(3)* ...
        a(x-1,y+1)+w(4)*a(x,y-1)+w(5)*a(x,y)+w(6)* ...
    a(x,y+1)+w(7)*a(x+1,y-1)+w(8)*a(x+1,y)+w(9)* ...
        a(x+1,y+1);
    end
end
figure(1)
imshow(uint8(a))
figure(2)
imshow(uint8(a1))
```





(a) Original image **(b) High pass filtered image**

Fig. 4.5.2

4.6 High-Boost Filtering

As can be seen, the high pass filter gives great results. But there is one problem. It gets rid of the complete background. There are times, when we need to enhance the edges but also retain some of the background. To do that, we use a modified version of the high pass filter known as High-Boost filtering.

In High-Boost filtering, we pass some of the background along with the high frequency content.

We know that High pass = Original – Low pass

To pass some of the background, we multiply the original image with a multiplicative factor A. This gives us high boost filtering.

Hence,

High Boost = (A) original – Low pass

= (A –1) Original + Original – Low pass

High Boost = (A –1) Original + High pass

If A = 1, then

High Boost = High pass

If A > 1, then some of the original signal is added back to the high pass result. This process restores some of the background into the high passed image. This technique is also known as unsharp masking. The mask coefficients for high boost filtering are shown below.

Here $X = 9A - 1$

Hence if $A = 1$, $X = 8$ which is a high pass mask.

$\frac{1}{9}$

–1	–1	–1
–1	X	–1
–1	–1	–1

3×3 High boost mask

If $A = 1.1, X = 8.9$.

We have a mask which is as shown

$\frac{1}{9}$

–1	–1	–1
–1	8.9	–1
–1	–1	–1

3×3 High boost mask

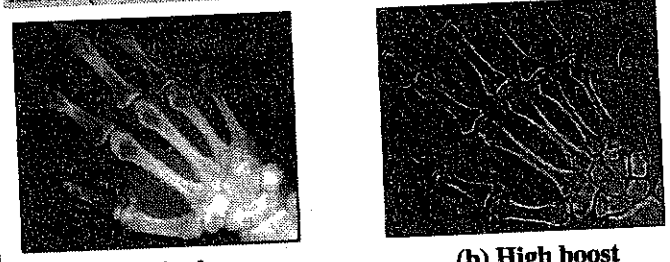
We can select different values of A and see the difference.

Use the same pseudo image as the one used in the high pass example to see the results. What you will notice is that places which had a zero in the high pass example will now have some positive values. Hence it does not eliminate the background completely. This technique is one of the basic tools that is used in the printing industry.

There are other neighbourhood techniques like Robert's filtering, Sobel's filtering and Prewitt's filtering, which are similar to the methods discussed above. They form a separate chapter called Segmentation and hence would be discussed later.

MATLAB program for high-boost filtering

```
%% High boost filtering
clear all
clc
aa=imread ('xray.tif');
a=double(aa);
[row col]=size (a);
w=[-1 -1 -1; -1 8.9 -1; -1 -1 -1]; %% High boost mask
for x=2:1:row-1
    for y=:1:col-1
a1(x,y)= w(1)*a(x-1,y-1)+w(2)*a(x-1,y)+w(3)* ...
a(x-1,y+1)+w(4)*a(x,y-1)+w(5)*a(x,y)+w(6)* ...
a(x,y+1)+w(7)*a(x+1,y-1)+w(8)*a(x+1,y)+w(9)* ...
a(x+1,y+1);
    end
end
figure(1)
imshow(uint8(a))
figure(2)
imshow(uint8(a1))
```



(a) Original (b) High boost

Fig. 4.6.1

The generalised MATLAB program for high pass filtering using any mask size is given next.

```
%% High pass filtering of an image %%
clear all;
clc;
a=imread('mri2.tif');
a=double(a);
[row col]=size(a);
m = input('Enter the mask size:');
for i=1:1:m
    for j=1:1:m
        w(i,j)=-1
    end
end
s=(m+1)/2;
w(s,s)=m^2-1;
```

```
for x=1:1:row
    for y=s:1:col
        b(x,y)=a(x,y);
    end
end
for x=s:1:row-s
    for y=s:1:col-s
        b(x,y)=0;
    end
end
for x=s:1:row-s
    for y=s:1:col-s
        for i=1:1:m
            for j=1:1:m
                b(x,y)=a(x-s+i,y-s+j)*w(i,j)+b(x,y);
            end
        end
        if b(x,y) < 0
            b(x,y)=0;
        end
    end
end
n=0
for x=s:1:row-s
    for y=s:1:col-s
        if b(x,y) > n
            n=b(x,y);
        end
    end
end
c=255/n;
for x=s:1:row-s
    for y=s:1:col-s
        b(x,y)=c*b(x,y);
    end
end
subplot(2,1,1)
imshow(uint8(a))
title('Original Image');
subplot(2,1,2)
imshow(uint8(b))
title('High Pass Filtered Image')
```

4.7 Zooming

Another important application in image enhancement where the spatial domain neighbourhood operation is used image zooming. Zooming of images is not something that is new to you. If you have used Microsoft paint or Photoshop editor, you would be aware that there is an option which allows us to zoom an image by 25%, 50%, and 100% .....Have you ever imagined as to how an image that looked small can suddenly look so big ? The technique is quite simple and after reading the next couple of pages, you would be able to do it yourself.

Zooming can be carried out using two different methods,

(1) Replication (2) Interpolation

We will discuss both these methods in detail. We shall first start with zooming using replication.

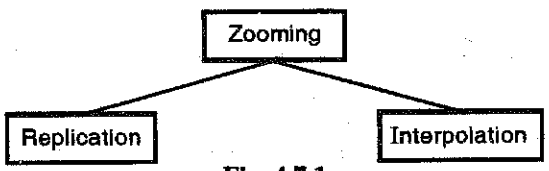


Fig. 4.7.1

4.7.1 Replication

In replication, we simply replicate each pixel and then replicate each row. Consider the pseudo-image shown below.

1	2	3	4
5	6	7	8
9	8	6	7
0	1	2	3

As stated earlier, we start from the first row. We replicate each pixel and then replicate each row. The first row now looks like

1 1 2 2 3 3 4 4

We now replicate this row to get

1 1 2 2 3 3 4 4

1 1 2 2 3 3 4 4

Performing this operation on the entire image we get

1	1	2	2	3	3	4	4
1	1	2	2	3	3	4	4
5	5	6	6	7	7	8	8
5	5	6	6	7	7	8	8
9	9	8	8	6	6	7	7
9	9	8	8	6	6	7	7
0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3

Hence a 4×4 image is zoomed to a 8×8 image. This method can be repeated to get bigger images. Remember, this is an image enhancement technique and hence no new data is added. In the zoomed pseudo-image, we observe that as we increase the size of the image, clusters of grey levels are formed which are disconcerting to the observer. Hence zooming increases the size of the image no doubt, but it also gives the image a patchy look.

Zooming by replication can be implemented on the computer by using a replication mask. The first step is to interlace the original image with zeros. This is known as zero interlacing. In this we add zeros after every pixel and then add a complete row of zeros. Adding zeros to every other pixel of the first row we get,

1 0 2 0 3 0 4 0

This is also known as zero interlacing the columns, as we add zeros at every other column. Now inserting a row full of zeros gives us. This is known as zero interlacing the rows as we add zeros at every other row.

1 0 2 0 3 0 4 0

0 0 0 0 0 0 0 0

1	0	2	0	3	0	4	0
0	0	0	0	0	0	0	0
5	0	6	0	7	0	8	0
0	0	0	0	0	0	0	0
9	0	8	0	6	0	7	0
0	0	0	0	0	0	0	0
0	0	1	0	2	0	3	0
0	0	0	0	0	0	0	0

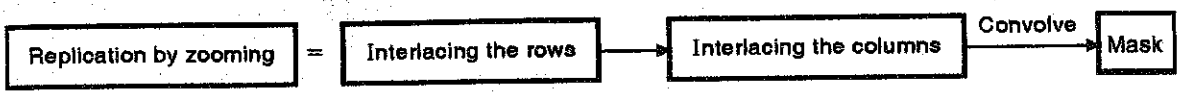
This image is known as the zero interlaced image. On this image, we run a replication mask given below to get the zoomed image.

1	1
1	1

The final zoomed image is shown below,

1	1	2	2	3	3	4	4
1	1	2	2	3	3	4	4
5	5	6	6	7	7	8	8
5	5	6	6	7	7	8	8
9	9	8	8	6	6	7	7
9	9	8	8	6	6	7	7
0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3

Note : To get the above result, we need to add a row of zeroes at the top and a column of zeroes at the beginning of the zero interlaced image making it 9×9.



As mentioned earlier, zooming by replication gives the final image a patchy look since clusters of grey levels are formed. This can be substantially reduced by using a better method of zooming known as interpolation.

4.7.2 Linear Interpolation

In this method, instead of replicating each pixel, average of the two adjacent pixels along the rows is taken and placed between the two pixels. The same operation is then performed along the columns. This operation is done on the interlaced image.

1	0	2	0	3	0	4	0
0	0	0	0	0	0	0	0
5	0	6	0	7	0	8	0
0	0	0	0	0	0	0	0
9	0	8	0	6	0	7	0
0	0	0	0	0	0	0	0
0	0	1	0	2	0	3	0
0	0	0	0	0	0	0	0

Interpolaiton along rows is given by,

$$v_1(m, 2n) = u(m, n); \quad 0 \le m \le M-1, 0 \le n \le N-1$$

$$v_1(m, 2n+1) = \frac{1}{2}[u(m, n) + u(m, n+1)];$$

$$0 \le m \le M-1; \quad 0 \le n \le N-1$$

Interpolation of the column is given by,

$$v(2m, n) = v_1(m, n);$$

$$v(2m+1, n) = \frac{1}{2}[v_1(m, n) + v_1(m+1, n)];$$

$$0 \le m \le M-1; 0 \le n \le N-1$$

Hence the first row now becomes the modified image is as shown

1	1.5	2	2.5	3	3.5	4	2

1	1.5	2	2.5	3	3.5	4	2
0	0	0	0	0	0	0	0
5	5.5	6	6.5	7	7.5	8	4
0	0	0	0	0	0	0	0
9	8.5	8	7	6	6.5	7	3.5
0	0	0	0	0	0	0	0
0	0.5	1	1.5	2	2.5	3	1.5
0	0	0	0	0	0	0	0

This is the first step. We now proceed to find the average values along the columns. Hence the final image is as shown. In practice, we need to round of the pixel values.

1	1.5	2	2.5	3	3.5	4	2
3	3.5	4	4.5	5	5.5	6	3
5	5.5	6	6.5	7	7.5	8	4
7	7	7	6.25	6.5	7	7.5	3.25
9	8.5	8	7	6	6.5	7	3.5
4.5	4.5	4.5	4.25	4	4.5	5	2.5
0	0.5	1	1.5	2	2.5	3	1.5
0	0.25	0.5	0.75	1	1.25	1.5	0.75

If we compare the results of replication and interpolation, it is clear that the patchiness that was present in the replicated image is much less in the interpolated image. Hence we can conclude that zooming by interpolation is more effective than zooming by replication.

To implement zooming by interpolation, we simply run an interpolation mask on the zero interlaced image.

1/4	1/2	1/4
1/2	1	1/2
1/4	1/2	1/4

MATLAB program for zooming by replication as well as interpolation

```
%% Zooming using replication as well as interpolation %%
clc
clear all
x=imread ('warne.tif');
s=size(x);
ss=s(1)+s(2);    %% To set the size of the output image %%
    y=zeros(ss);
     i=1;
     j=1;
for n=1:2:ss
          for m=1:2:ss
          y (m, n)= x (i, j);
          i=i+1;
          end
        i=1;  %% Else the value of i goes on increasing%%
          j=j+1;
end
h1=[1 1;1 1];                    % Mask for replication
h2=[1/4  1/2  1/4;  1/2  1  1/2;  1/4  1/2  1/4]; %% Mask for interpolation %%
A1=conv2 (y,h1); %% We can use the formula that is used in Low pass
                    filtering %%
A2=conv2 (y,h2);
figure(1),imshow(x)
figure(2),imshow(uint8 (A1))
figure(3),imshow(uint8 (A2))
```

(a) Original (b) Zooming by replication (c) Zooming by interpolation

Fig. 4.7.2

4.8 Solved Examples

Ex. 4.8.1

Obtain the digital negative of the following 8 Bits Per Pixel - BPP image.

121	205	217	156	151
139	127	157	117	125
252	117	236	138	142
227	182	178	197	242
201	106	119	251	240

Soln. :

It is known that it is a 8-bit image. Hence the number of grey levels that this image can hold is $2^8 = 256$

$\therefore$ L = 256.

Hence the minimum grey level is 0

while the maximum grey level is 255

$$s(x,y) = (L-1) - r(x,y)$$

$$= (256-1) - r(x,y)$$

$$s(x,y) = 255 - r(x,y)$$

We get the digital negative using the above equation.

134	50	38	99	104
116	128	98	138	130
3	138	19	117	113
28	73	77	58	13
54	149	136	4	15

Ex. 4.8.2

For a given image find -

(i) Digital negative of an image.

(ii) Bit plane slicing.

4	3	2	1
3	1	2	4
5	1	6	2
2	3	5	6

Soln. :

We assume the image to be 3-bit

(i) Digital negative

$$s = (L-1) - r$$

$$\text{or } g(x, y) = (L-1) - f(x, y)$$

$$\text{Here } L = 2^3 = 8$$

$$\therefore s = 7 - r$$

$$\text{i.e., } g(x, y) = 7 - f(x, y)$$

$\therefore$ The digital negative of the image is

3	4	5	6
4	6	5	3
2	6	1	5
5	4	2	1

(ii) Bit plane slicing

In this, we convert the given image into binary and then separate the planes.

We assume the image to be 3 bit.

100	011	010	001
011	001	010	100
101	001	110	010
010	011	101	110

Separating the bit planes we get

1	0	0	0
0	0	0	1
1	0	1	0
0	0	1	1

MSB plane

0	1	1	0
1	0	1	0
0	0	1	1
1	1	0	1

Center bit plane

0	1	0	1
1	1	0	0
1	1	0	0
0	1	1	0

LSB plane

Ex. 4.8.3

For following image find :

(i) Contrast stretching $r_2 = 5, r_1 = 3, s_2 = 6, s_1 = 2$.

$f(x, y) =$

4	3	2	1
3	1	2	4
5	1	6	2
2	3	5	6

Soln. :

(i) Contrast stretching $r_2 = 5, r_1 = 3, s_2 = 6, s_1 = 2$:

$f(x, y) =$

4	3	2	1
3	1	2	4
5	1	6	2
2	3	5	6

The contrast stretching transformation is shown in Fig. P. 4.8.3.

From the values of r_1 and s_1 we can find the slope of α. In a similar manner, we can find the value of β using the values of r_1, r_2 and s_1, s_2.

From the values of r_2 and s_2, we can find the value of γ.

$$\alpha = \frac{s_1}{r_1} = \frac{2}{3} = 0.66$$

$$\beta = \frac{y_2 - y_1}{x_2 - x_1} = \frac{6-2}{5-3} = 2$$

Finally, $$\gamma = \frac{y_2 - y_1}{x_2 - x_1} = \frac{7-6}{7-5} = 0.5$$

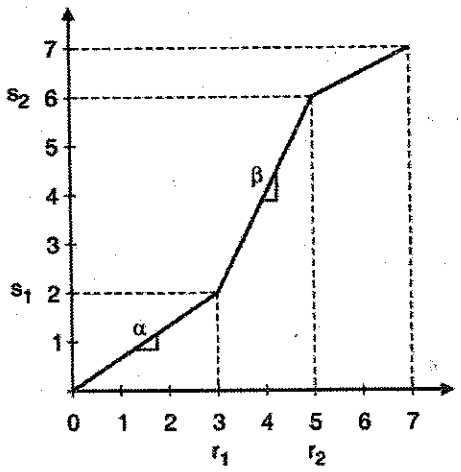


Fig. P. 4.8.3

Hence we have the required slopes to compute contrast stretching. The contrast stretching formula is given below.

$$s = \begin{cases} \alpha r & 0 \le r < 3 \\ \beta \cdot (r - r_1) + s_1 & 3 \le r < 5 \\ \gamma (r - r_2) + s_2 & 5 \le r < 7 \end{cases}$$

Here $r_1 = 3,\ r_2 = 5,\ s_1 = 2,\ s_2 = 6$

$\alpha = 0.66,\ \beta = 2,\ \gamma = 0.5$

We make a table of values for r and s using the contrast stretching formula.

r	s
0	$s = \alpha r = 0.66 \times 0 = 0$
1	$s = \alpha r = 0.66 \times 1 = 0.66$
2	$s = \alpha \cdot r = 0.66 \times 2 = 1.32$
3	$s = \beta (r - r_1) + s_1 = 2(3-3) + 2 = 2$
4	$s = \beta (r - r_1) + s_1 = 2(4-3) + 2 = 4$
5	$s = \gamma (r - r_2) + s_2 = 0.5(5-5) + 6 = 6$
6	$s = \gamma (r - r_2) + s_2 = 0.5(6-5) + 6 = 6.5$
7	$s = \gamma (r - r_2) + s_2 = 0.5(7-5) + 6 = 7$

Hence the contrast stretched image is

4	2	1.32	0.66
2	0.66	1.32	4
6	0.66	6.5	1.32
1.32	2	6	6.5

If we round off, the final image ⇒ would be

4	2	1	1
2	1	1	4
6	1	7	1
1	2	6	7

Ex. 4.8.4

For the 3-bit 4 × 4 size image, perform the following operations :

(i) Negation

(ii) Thresholding with T = 4

(iii) Intensity level slicing with background $r_1 = 2$ and $r_2 = 5$

(iv) Bit plane slicing for MSB and LSB planes

(v) Clipping with $r_1 = 2$ and $r_2 = 5$

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

Soln. :

Let the given image be f(x, y)

(i) Negation

$$s = (L-1) - r$$

OR

$$g(x, y) = (L-1) - f(x, y)$$

Since the image is 3-bit, $L = 2^3 = 8$

$\therefore L - 1 = 7$

$\therefore g(x, y) = 7 - f(x, y)$

Hence the digital negative of the image is

$$g(x, y) = \begin{array}{|c|c|c|c|} \hline 6 & 5 & 4 & 7 \\ \hline 5 & 3 & 1 & 0 \\ \hline 2 & 5 & 3 & 4 \\ \hline 4 & 5 & 1 & 6 \\ \hline \end{array}$$

(ii) Thresholding with T = 4

The thresholding function is shown below along with thresholding formula.

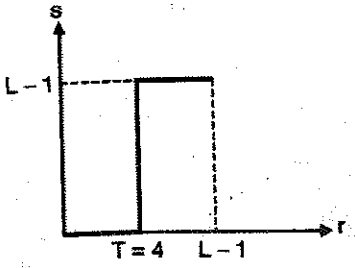


Fig. P. 4.8.4

$$g(x, y) = \begin{cases} 0 & \text{if } f(x, y) \leq 4 \\ L-1 & \text{if } f(x, y) \geq 4 \end{cases}$$

Here

$L - 1 = 7$

∴ The thresholded image is shown below

7	7	7	7
7	0	0	0
0	7	0	7
7	7	0	7

(iii) Bit plane slicing

We convert the given image into binary and then separate the plane. Since the given image is 3-bit, the binary image would be as shown.

001	010	011	000
010	100	110	111
101	010	100	011
011	010	110	001

Separating the bit planes we obtain.

0	0	0	0
0	1	1	1
1	0	1	0
0	0	1	0

MSB plane

0	1	1	0
1	0	1	1
0	1	0	1
1	1	1	1

Center bit plane

1	0	1	0
0	0	0	1
1	0	0	1
1	0	0	1

LSB plane

(iv) Clipping with $r_1 = 2$, $r_2 = 5$

The clipping transformation is shown in Fig. P. 4.8.4(a).

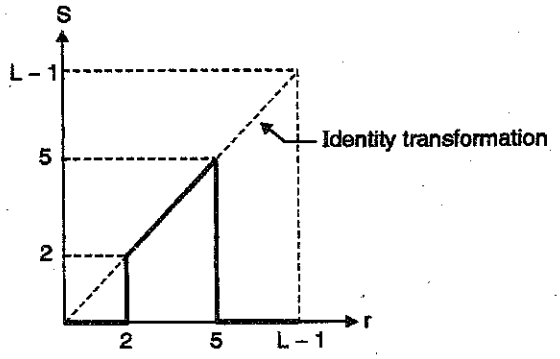


Fig. P. 4.8.4(a)

$$s = \begin{cases} r & 2 \leq r \leq 5 \\ 0 & \text{otherwise} \end{cases}$$

Hence the final clipped image is shown below

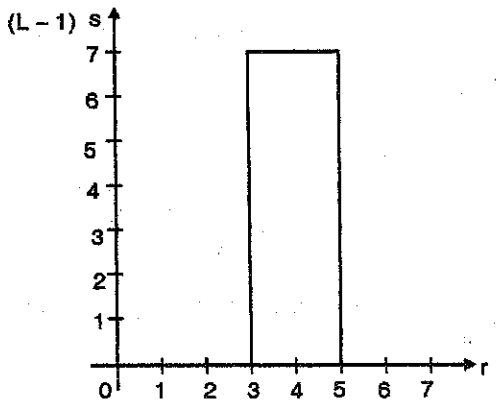
0	2	3	0
2	4	0	0
5	2	4	3
3	2	0	0

Ex. 4.8.5

Perform intensity level (grey level) slicing on the 3 BPP image. Let $r_1 = 3$ and $r_2 = 5$. Draw the modified image using with background and without background transformations.

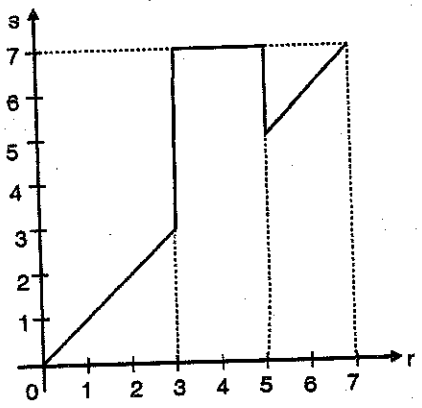
Soln. : Let us draw the grey level transformation.

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



(a)

Fig. P. 4.8.5 Continued...



(b)

Fig. P. 4.8.5

$s = L - 1 \quad r_1 \le r \le r_2$ $\quad$ $s = 0$ otherwise

0	0	0	0	0
0	7	7	7	0
0	0	0	0	0
0	0	0	7	0
0	7	0	0	0

Without background

$s = L - 1 \quad r_1 \le r \le r_2$ $\quad$ $s = r$ otherwise

2	1	2	2	1
2	7	7	7	2
6	2	7	6	0
2	6	6	7	1
0	7	2	2	1

With background

Ex. 4.8.6

The grey levels in an image range from 10 to 50. We need to display the image on a device that has a grey level range of 0 to 255. Give a transformation which would accomplish this.

Soln. :

The transformation that we use is known as the Range normalization.

It is given by,

$$s(x, y) = \frac{d - c}{b - a}[r(x, y) - a] + c$$

This transformation converts the original range [a, b] to the new range [c, d].

In this sum a = 10, b = 50, c = 0, d = 255.

$$s(x, y) = \frac{255}{40}[r(x, y) - 10] + 0$$

$$s(x, y) = \frac{51}{8} r(x, y) - \frac{255}{4}$$

When $r = 10, s = 0$

$r = 50, s = 255$

Ex. 4.8.7

The image shown below has 8 different grey levels. Plot this image using only 4 grey levels.

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

Soln. :

This can be achieved using the Grey level reduction transformation which is given by,

$$s(x, y) = \left[\frac{Gr(x, y)}{K}\right]\frac{K}{G}$$

Here $K \rightarrow$ Original number of grey levels

$G \rightarrow$ Modified number of grey levels

$[.] \rightarrow$ Choosing the floor value.

$$s(x, y) = \left[\frac{4r(x, y)}{8}\right]\frac{8}{4}$$

$\therefore$ r(x, y)	s(x, y)	
0	0	1
1	0	
2	2	2
3	2	
4	4	3
5	4	
6	6	4
7	6	

∴ The modified image is

0	0	0	0	0	4
0	0	2	2	2	2
0	0	2	2	2	2
0	2	4	6	2	2
0	2	4	2	4	4
0	2	2	6	2	4

Ex. 4.8.8

Show that a high pass filtered image can be obtained in the spatial domain as High pass = Original – Low pass.

Soln. :

Original image

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Low pass filter

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

High pass filter

–1/9	–1/9	–1/9
–1/9	8/9	–1/9
–1/9	–1/9	–1/9

When we apply the LPF on the image, the center pixel z_5 changes to

$$\frac{1}{9}[z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7 + z_8 + z_9]$$

$$\text{Original} - \text{Low pass} = z_5 - \frac{1}{9}[z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7 + z_8 + z_9]$$

$$= z_5 - \frac{z_1}{9} - \frac{z_2}{9} - \frac{z_3}{9} - \frac{z_4}{9} - \frac{z_5}{9} - \frac{z_6}{9} - \frac{z_7}{9} - \frac{z_8}{9} - \frac{z_9}{9}$$

$$= \frac{8z_5}{9} - \frac{1}{9}[z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7 + z_8 + z_9]$$

This is nothing but a high pass mask

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Note : We can also prove this by taking a pseudo-image. We move a low pass mask on the original image and subtract it from the original. We now move a high pass mask on the original image. The results of both these operations will be the same.

Ex. 4.8.9

Filter the following image using 3 × 3 neighbourhood averaging by assuming (i) Zero padding (ii) Pixel replication

$$f(x, y) = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 2 \\ \hline 4 & 2 & 5 & 1 \\ \hline 1 & 2 & 6 & 3 \\ \hline 2 & 4 & 6 & 7 \\ \hline \end{array}$$

Soln. :

A 3 × 3 averaging mask is shown below :

$$w(x, y) = \frac{1}{9} \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Filtering is performed using the convolution operation,

i.e., $g(x, y) = f(x, y) * w(x, y)$.

(i) Zero padding

In this, we zero pad the image before performing the filtering operation. This gives us a 6 × 6 image.

$$f(x, y) = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 2 & 3 & 2 & 0 \\ \hline 0 & 4 & 2 & 5 & 1 & 0 \\ \hline 0 & 1 & 2 & 6 & 3 & 0 \\ \hline 0 & 2 & 4 & 6 & 7 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

We move the averaging mask over this image starting from the top left corner as shown.

∴ $g(x, y) = f(x, y) * w(x, y)$

0	0	0	0	0	0
0	1	1.88	1.66	1.22	0
0	1.33	2.88	2.88	2.22	0
0	1.66	3.55	4	3.11	0
0	1	2.33	3.11	2.44	0
0	0	0	0	0	0

(ii) Pixel replication

In this we replicate the border pixels to generate a 6×6 image.

$f(x, y) =$

1	1	2	3	2	2
1	1	2	3	2	2
4	4	2	5	1	1
1	1	2	6	3	3
2	2	4	6	7	7
2	2	4	6	7	7

We move the averaging mask over this image starting from the top left corner as shown.

$\therefore\ g(x, y) = f(x, y) * w(x, y)$

$g(x, y) =$

1	1	2	3	2	2
1	2	2.55	2.44	2.33	2
4	2	2.88	2.88	2.88	1
1	2.44	3.55	4	4.33	3
2	2.22	3.66	5	5.77	7
2	2	4	6	7	7

Ex. 4.8.10

Perform discrete convolution on the following image arrays. Assume that the left bottom corner elements are at the origin in both cases. f_1 and f_2 are two images.

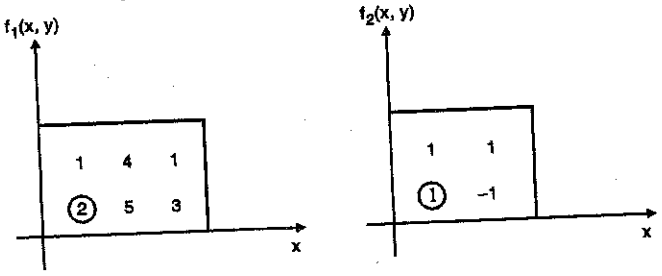


Fig. P. 4.8.10

Soln. :

$$f_1(x, y) * f_2(x, y) = \sum_m \sum_n f_1(m, n)\, f_2(x - m, y - n)$$

Let $f_1(x, y) * f_2(x, y) = g(x, y)$

$$g(x, y) = \sum_m \sum_n f_1(m, n)\, f_2(x - m, y - n)$$

$$g(0, 0) = \sum_m \sum_n f_1(m, n)\, f_2(-m, -n)$$

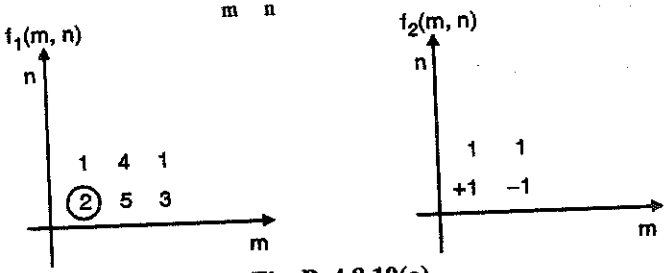


Fig. P. 4.8.10(a)

The basic shift operations are

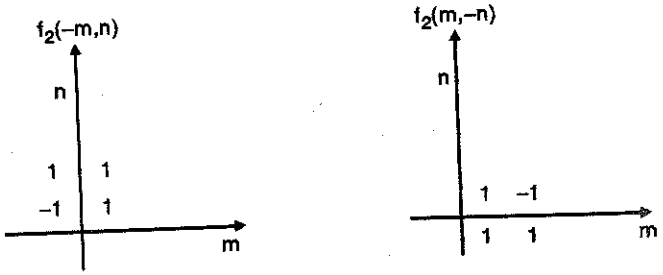
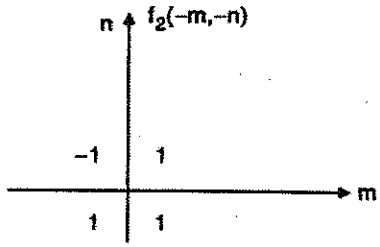


Fig. P. 4.8.10(b)

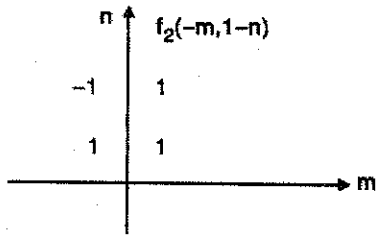
$f_2(-m, n) \Rightarrow$ fliping about m-axis



$g(0,0) = \sum_m \sum_n f_1(m, n)\, f_2(-m, -n)$

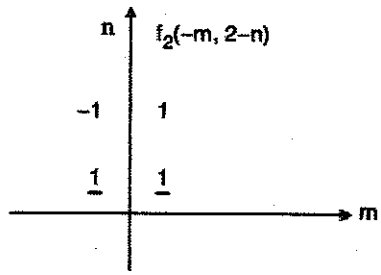
$g(0,0) = 2 \times 1 = 2$

$f_2(m, -n) \Rightarrow$ fliping about n-axis



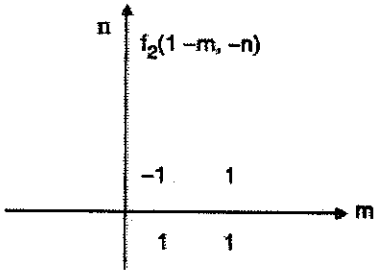
$g(0,1) = \sum_m \sum_n f_1(m, n)\, f_2(-m, 1-n)$

$g(0,1) = 1 \times 1 + 2 \times 1 = 3$



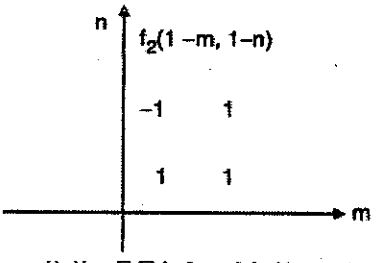
$g(0,2) = \sum_m \sum_n f_1(m, n)\, f_2(-m, 2-n)$

$g(0,2) = 1 \times 1 = 1$



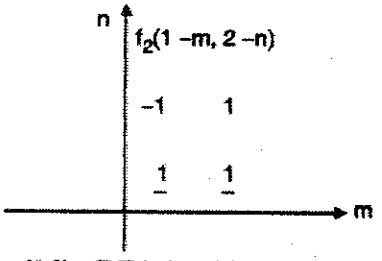
$g(1,0) = \sum \sum f_1(m, n)\, f_2(1-m, -n)$

$g(1,0) = 2 \times -1 + 5 \times 1 = 3$



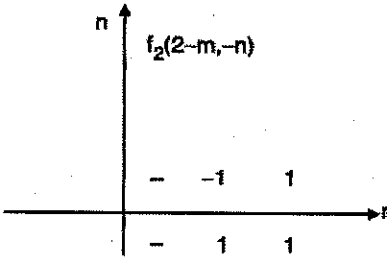
$g(1,1) = \sum_m \sum_n f_1(m, n)\, f_2(1-m, 1-n)$

$g(1,1) = 1 \times -1 + 4 \times 1 + 2 \times 1 + 5 \times 1 = 10$



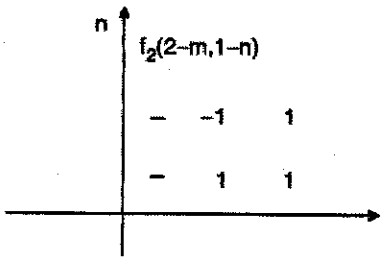
$g(1,2) = \sum_m \sum_n f_1(m, n)\, f_2(1-m, 2-n)$

$g(1,2) = 1 \times 1 + 4 \times 1 = 5$



$g(2,0) = \sum_m \sum_n f_1(m, n)\, f_2(2-m, -n)$

$g(2,0) = 5 \times -1 + 3 \times 1 = -2$



$g(2,1) = \sum_m \sum_n f_1(m, n)\, f_2(2-m, 1-n)$

$g(2,1) = 4 \times -1 + 1 \times 1 + 5 \times 1 + 3 \times 1 = 5$

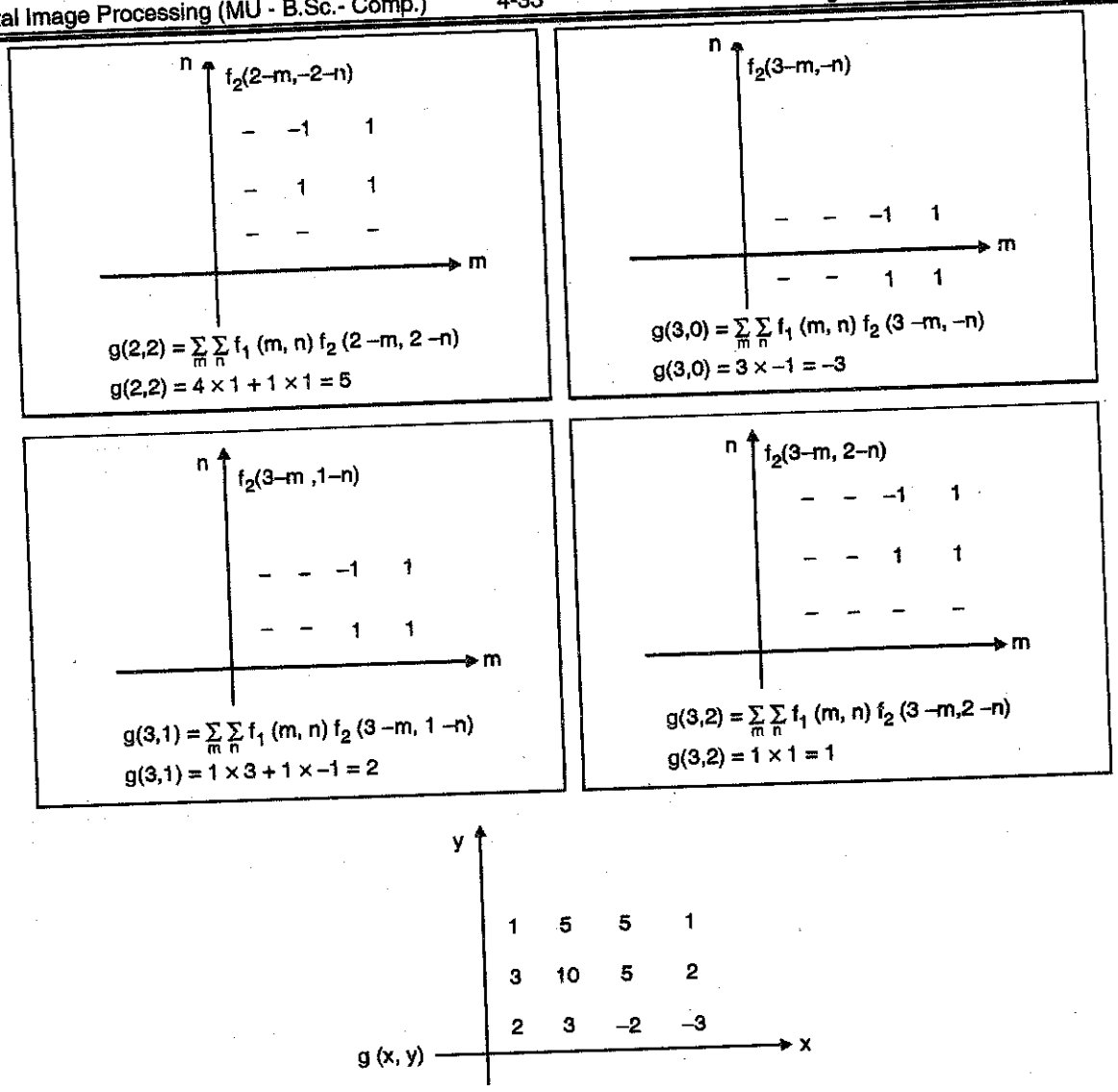


Fig. P. 4.8.10(c)

Another method : It is convenient to represent digital images as matrices and exploit the benefits of linear algebra. We modify the matrix so as to suit convolution.

Let F_1 and F_2 be the two images that need to be convolved :

Let F_1 be a $m_1 \times n_1$ image and F_2 be a $m_2 \times n_2$ image. We zero pad F_1 and F_2 so that both of them are of size $(m_1 + m_2 - 1) \times (n_1 + n_2 - 1)$.

Ex. 4.8.11

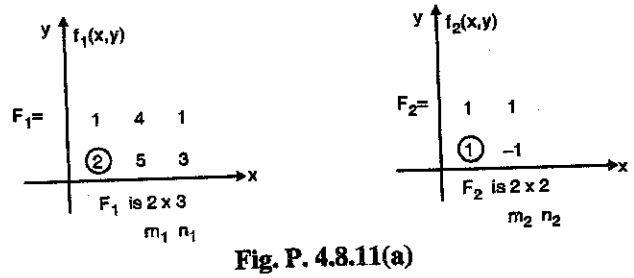


Fig. P. 4.8.11(a)

Soln. :

Zero padding F_1 and F_2 so that both are of size

$$(m_1 + m_2 - 1) \times (n_1 + n_2 - 1) = 3 \times 4$$

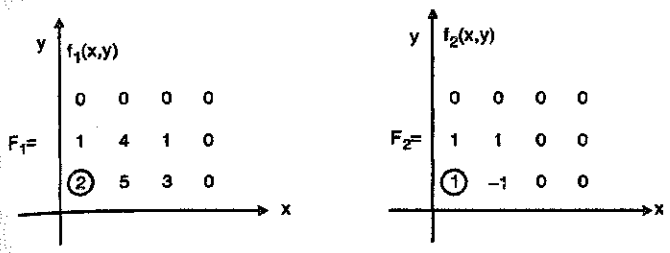


Fig. P. 4.8.11(b)

Now, of these two, arrange one as a circular matrix and the other one as a row stacking matrix. Let the first element of these be their respective origins, then

$$G(x, y) = F_1(x, y)\, F_2(x, y)$$

F_1 (circular matrix, origin = 2):

(2)	0	0	0	0	0	1	4	1	0	3	5
5	(2)	0	0	0	0	0	1	4	1	0	3
3	5	(2)	0	0	0	0	0	1	4	1	0
0	3	5	(2)	0	0	0	0	0	1	4	1
1	0	3	5	(2)	0	0	0	0	0	1	4
4	1	0	3	5	(2)	0	0	0	0	0	1
1	4	1	0	3	5	(2)	0	0	0	0	0
0	1	4	1	0	3	5	(2)	0	0	0	0
0	0	1	4	1	0	3	5	(2)	0	0	0
0	0	0	1	4	1	0	3	5	(2)	0	0
0	0	0	0	1	4	1	0	3	5	(2)	0
0	0	0	0	0	1	4	1	0	3	5	(2)

$\times$ F_2 (row stacking, origin = 1):

(1)
-1
0
0
1
1
0
0
0
0
0
0

$=$ G:

(2)
3
-2
-3
3
10
5
2
1
5
5
1

$\therefore$ G (x, y)

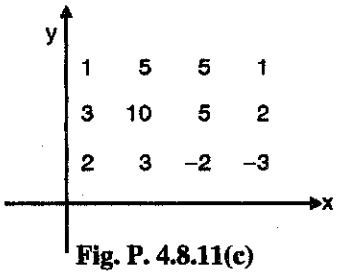


Fig. P. 4.8.11(c)

We see that the results are the same as obtained from the previous method.

Ex. 4.8.12

Apply Low and High Pass Spatial masks on the following Fig. P. 4.8.12 matrix. Prove that High pass = Original – Low pass. Assume virtual Rows and Columns.

$f(x,y) =$

30	31	32
33	120	30
32	32	31

Fig. P. 4.8.12

Soln. :

We zero padd the image to make it 5×5

$f(x,y) =$

0	0	0	0	0
0	30	31	32	0
0	33	120	30	0
0	32	32	31	0
0	0	0	0	0

We convolve this image with a standard 3×3 Low pass and a High pass mask.

(i) Low Pass

$$\text{Low pass mask } h_1(x,y) = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$g_1(x,y) = f(x,y) * h_1(x,y) = \begin{bmatrix} 23.77 & 30.66 & 23.66 \\ 30.88 & 41.22 & 30.66 \\ 24.11 & 30.88 & 23.667 \end{bmatrix}$$

(ii) High Pass

$$\text{High pass mask } h_2(x,y) = \frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$g_2(x,y) = f(x,y) * h_2(x,y) = \begin{bmatrix} 6.22 & 0.33 & 8.33 \\ 2.11 & 78.77 & -0.667 \\ 7.889 & 1.11 & 7.33 \end{bmatrix}$$

We now show that High pass = Original – Low pass

$$\text{Original} - \text{Lowpass} = \begin{bmatrix} 30 & 31 & 32 \\ 33 & 120 & 30 \\ 32 & 32 & 31 \end{bmatrix} - \begin{bmatrix} 23.77 & 30.66 & 23.66 \\ 30.88 & 41.22 & 30.66 \\ 24.11 & 30.88 & 23.667 \end{bmatrix}$$

$$= \begin{bmatrix} 6.22 & 0.33 & 8.33 \\ 2.11 & 78.77 & -0.667 \\ 7.889 & 1.11 & 7.33 \end{bmatrix}$$

This is the same as the High pass result. Therefore,

High pass = Original – Low pass

Ex. 4.8.13

Given $f(x, y) = \begin{bmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \end{bmatrix}$, $h(x, y) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

Find linear convolution of input image f(x, y) with filter of h(x, y).

Soln. :

Let the output image be g (x, y)

Here, $g(x, y) = f(x, y) * h(x, y)$

We shall solve this using the circular matrix method. (The other method is the graphical technique).

We zero pad f(x, y) and h(x, y) in such a way that both of them are of size.

$(m_1 + m_2 - 1) \times (n_1 + n_2 - 1)$

Here $m_1 \times n_1$ is the size of f(x, y) [2×3] and $m_2 \times n_2$ is the size of h(x, y) [2×2], Therefore $m_1 = 2$, $m_2 = 2$, $n_1 = 3$, $n_2 = 2$. Hence zero padding should be done in such a manner so that f_1 (x, y) and h(x, y) are of size $(2 + 2 - 1) \times (3 + 2 - 1)$.

$$= 3 \times 4$$

$$\therefore \quad f(x, y) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 5 & 6 & 7 & 0 \\ \textcircled{8} & 9 & 10 & 0 \end{bmatrix}; h(x, y)$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ \textcircled{3} & 4 & 0 & 0 \end{bmatrix}$$

We now arrange one of these as a circular matrix and the other one as a row slacking matrix. Let us convert f(x, y) to a circular matrix. Let the encircled terms be the starting points.

$$\begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 0 & 7 & 6 & 5 & 0 & 10 & 9 \\ 9 & 8 & 0 & 0 & 0 & 0 & 0 & 7 & 6 & 5 & 0 & 10 \\ 10 & 9 & 8 & 0 & 0 & 0 & 0 & 0 & 7 & 6 & 5 & 0 \\ 0 & 10 & 9 & 8 & 0 & 0 & 0 & 0 & 0 & 7 & 6 & 5 \\ 5 & 0 & 10 & 9 & 8 & 0 & 0 & 0 & 0 & 0 & 7 & 6 \\ 6 & 5 & 0 & 10 & 9 & 8 & 0 & 0 & 0 & 0 & 0 & 7 \\ 7 & 6 & 5 & 0 & 10 & 9 & 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 6 & 5 & 0 & 10 & 9 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 6 & 5 & 0 & 10 & 9 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 6 & 5 & 0 & 10 & 9 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 6 & 5 & 0 & 10 & 9 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 6 & 5 & 0 & 10 & 9 & 8 \end{bmatrix} \times \begin{bmatrix} 3 \\ 4 \\ 0 \\ 0 \\ 1 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Multiplying the two matrices gives us

$$\begin{bmatrix} (24) \\ 59 \\ 66 \\ 40 \\ \hline 23 \\ 63 \\ 73 \\ 48 \\ \hline 5 \\ 16 \\ 19 \\ 14 \end{bmatrix}$$

We re-arrange this matrix to generate a filtered image.

$$\therefore \quad g(x, y) = \begin{bmatrix} 5 & 16 & 19 & 14 \\ 23 & 63 & 73 & 48 \\ 24 & 59 & 66 & 40 \end{bmatrix}$$

We would get the same answer if we make h(x, y) into a circular matrix instead of f(x, y). Go ahead and try it out yourself.

Ex. 4.8.14

For the given 3 bit, 4 × 4 size image perform the following operations :

(i) Thresholding (T = 4)

(ii) Intensity level slicing with background for $r_1 = 2, r_2 = 5$

(iii) Bit plane slicing for LSB and MSB planes

(iv) Negation.

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

Soln. :

(i) Thresholding (T = 4)

Since the given image is 3-bit, $L = 2^3 = 8$

The transformation for thresholding is shown in Fig. P. 4.8.14.

Here,

$$s = L - 1 = 7 \quad r \geq 4$$

$$s = 0 \quad r < 4$$

$\therefore$ The final image would be,

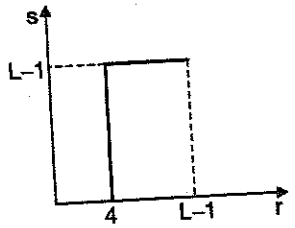


Fig. P. 4.8.14

7	0	0	0
0	0	7	7
7	0	0	0
0	7	7	7

(ii) Intensity level slicing with background for $r_1 = 2, r_2 = 5$

The transformation for intensity level slicing is shown in Fig. P. 4.8.14(a).

Here,

$$s = L - 1 = 7 \quad 2 \leq r \leq 5$$

$$s = r \quad \text{otherwise}$$

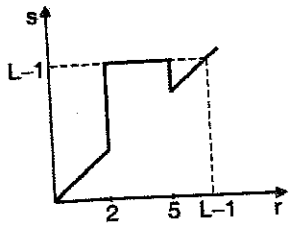


Fig. P. 4.8.14(a)

Hence all values between 2 and 5 are made equal to 7 and the remaining values remain unchanged. Hence the final result is,

7	7	7	0
1	7	7	7
7	7	7	1
7	7	6	7

(iii) Bit plane slicing

We convert the image to binary. Since there are 8 grey levels (0 to 7), we require 3 bits to represent each pixel.

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

Binary ⟶

100	010	011	000
001	011	101	111
101	011	010	001
010	100	110	111

The LSB and MSB planes are shown below.

0	0	1	0
1	1	1	1
1	1	0	1
0	0	0	1

LSB plane

1	0	0	0
0	0	1	1
1	0	0	0
0	1	1	1

MSB plane

(iv) Negation

The transformation for negation is shown in Fig. P. 4.8.14(b).

Here,

$$s = (L-1) - r$$

$$s = 7 - r$$

Here the grey levels get inverted.

∴ The final image is as shown below.

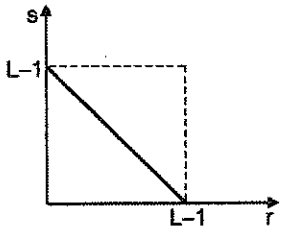
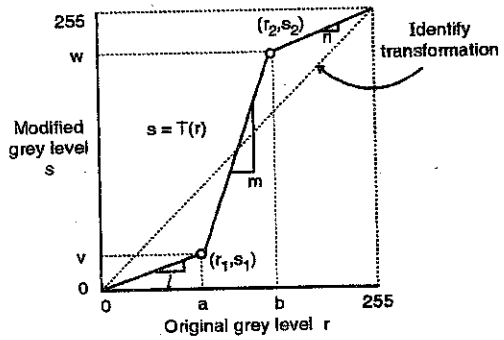
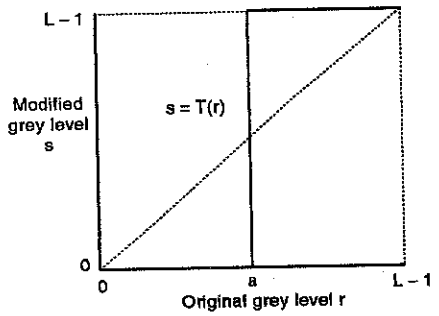


Fig. P. 4.8.14(b)

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

4.9 Comparison of Contrast Stretching and Thresholding

Sr. No.	Contrast Stretching	Thresholding
1.	Contrast stretching increases the dynamic range by making the dark pixels darker and bright pixels brighter.	Thresholding also increase that dynamic range by setting a threshold. All values less than the threshold are made black while values greater than the threshold are made equal to white.
2.	The formula of the contrast-stretching algorithm is given below : $s = \begin{cases} l \cdot r & 0 \le r < a \\ m \cdot (r-a) + v & a \le r < b \\ n \cdot (r-b) + w & b \le r < L-1 \end{cases}$	The formula for achieving thresholding is as follows : $s = 0$; if $r \le a$ $s = L-1$; if $r > a$

Sr. No.	Contrast Stretching	Thresholding
3.	The transformation for contrast stretching is shown below : 	The transformation for thresholding is shown below : 
4.	We can adjust the contrast in an image by changing the values of the slope.	Thresholding is extreme contrast stretching. In other words, a thresholded image has maximum contrast.
5.	The final result of contrast stretching gives us a grey scale image. The output image has different shades of grey.	The final result of thresholding gives us a binary image. Which means that the thresholded image has only black and white values.

Summary

In this chapter, we introduce the term spatial domain and image enhancement in the spatial domain. Different image enhancement techniques that improve the subjective quality of the image are explained. The methods described here can be classified into two broad groups : point processing and neighborhood processing. The concept of frequencies in an image is explained using simple illustrations. Basic filtering operations such low pass filtering and high pass filtering are presented. Advantages of each of the operations are stated using MATLAB codes. Any particular technique that may be applied depends on the subjective quality of the image as well as the purpose behind the processing. It must be noted that more than one technique can be used to get the desired result.

Review Questions

Q. 1 Distinguish between point operations and neighbourhood operations.

Q. 2 Compare between enhancement and restoration.

Q. 3 Explain spatial domain processing.

Q. 4 Compare and contrast average filtering and median filtering.

Q. 5 Explain the difference between operations involving 3×3 mask for median filtering and average filtering.

Q. 6 Develop a procedure for computing the median in $n \times n$ neighbourhood.

Q. 7 State and explain the features of median filtering. Compute the output of the median filter in the following cases.

A. Y(m) = {2, 4, 8, 3, 2}

w = {–1, 0, 1, 2}

B. Y(m) = {8, 2, 4, 3, 4}

w = {–1, 0, 1}

where, Y(m) is an array and w the window.

Q. 8 What do we mean by Gaussian noise and why is an averaging filter used to eliminate it ?

Q. 9 What is salt and pepper noise (binary noise) and how does a median filter remove it ?

Q. 10 Explain the output and application of the following zero memory enhancement techniques.

(1) Contrast stretching

(2) Thresholding

(3) Range compression

(4) Bit extraction.

Q. 11 Compare and contrast the low pass and high pass spatial filters.

Q. 12 Show that a high pass filter can be obtained as

HP = ORIGINAL – LP (assume a 3 × 3 mask)

Q. 13 Explain why a median filter is called a salt and pepper filter.

Q. 14 Develop the procedure for computing low pass filtering using a 3 × 3 mask.

Q. 15 Compare and contrast the smoothing and sharpening filters.

Q. 16 Enhancement does not add any information. Explain.

Q. 17 Low pass filter is a smoothing filter. Explain.

Q. 18 Give a brief account of enhancement filters in the spatial domain.

Q. 19 Explain why median filter is not suitable for gaussian noise. What is the limitation of median filter ?

Q. 20 What do we mean by frequencies in an image ?

Q. 21 What is meant by unsharp masking and crisping ? Explain with relevant figures.

Q. 22 Given a image, what is the output using a 3 × 3 averaging filter and a medain filter ? Explain the result.

3	3	3	3	3	3
3	3	3	3	3	3
3	3	10	10	3	3
3	10	3	3	3	3
3	3	3	3	8	3
3	3	3	3	3	3
3	3	3	3	3	3
3	3	3	3	3	3

Q. 23 How is image subtraction carried out ? Give an example of image subtraction in the field of medicine.

Q. 24 If in a sample of blood (say) the RBC's are supposed to be counted. What kind of operation would you apply ? The RBC's are visible but are not very clear. Also note that apart from the RBC's, there are no other cells present in the sample. Justify your answer.

Q. 25 What is the effect of using a larger mask in case of averaging filters e.g. 5 × 5 mask ? Explain using an example.

Q. 26 Explain the major difference between high pass and high boost filtering with an example.

Q. 27 Discuss the limiting effect of repeatedly applying a 3 × 3 low pass spatial filter to a digital image. You may ignore the border effects.

Q. 28 The figure below shows the cross sections of the transfer function of a high pass filter and its corresponding impulse response function. What would the transfer function and the impulse response look like for a high boost filter ?

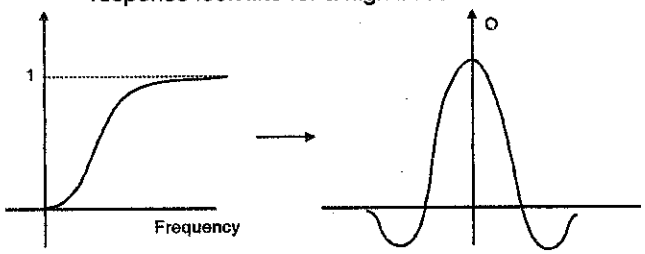


Fig. Q. 28

Q. 29 Explain the procedure of zooming an image using replication and interpolation.

Q. 30 The grey levels in an image range from 10 to 50 and we wish to display the image on a device that has a grey level range of 0 to 255. What transformation would we use ?

Q. 31 Following figure is an image with 8 grey levels. Transform it to an image with 4 grey levels without changing the brightness and the contrast.

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

Q. 32 Why should the sum of any high pass mask be zero ?

Q. 33 Perform discrete convolution on the following image arrays.

1	3	2
②	5	6

2	4
①	3

Fig. Q. 33

Q. 34 Draw the probability density functions of :

(a) Rayleigh noise

(b) Exponential noise.

Q. 35 Explain the power-law transformation.

Chapter Ends...

❑❑❑

CHAPTER 5

Histogram Modeling

5.1 Introduction

Histogram of images provide a global description of the appearance of an image. The information obtained from histograms is enormous and hence histogram modelling; though a spatial domain technique is introduced as a separate chapter.

By definition, histogram of an image represents the relative frequency of occurrence of the various grey levels in an image. Histogram of an image can be plotted in two ways.

In the first method, the x-axis has the grey levels and the y-axis has the number of pixels in each grey level, while in the second method, the x-axis represents the grey levels, while the y-axis represents the probability of the occurrence of that grey level.

☞ **Method 1**

Grey level	Number of pixels (n_k)
0	40
1	20
2	10
3	15
4	10
5	3
6	2

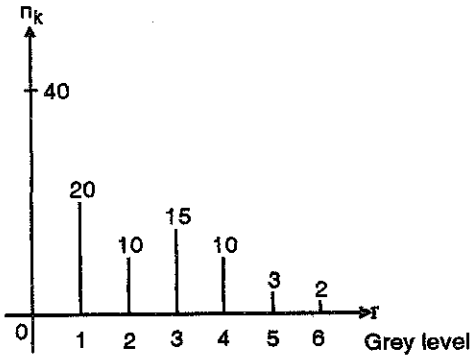


Fig. 5.1.1

☞ **Method 2**

In the second method, instead of plotting the number of pixels directly, we plot their probability of occurrence i.e.,

$$p(r_k) = \frac{n_k}{n}$$

$r_k \rightarrow k^{th}$ grey level

$n_k \rightarrow$ Number of pixels in the k^{th} grey level

$n \rightarrow$ Total number of pixels in an image

Grey level	Number of pixels (n_k)	$p(r_k) = \frac{n_k}{n}$
0	40	0.4
1	20	0.2
2	10	0.1
3	15	0.15
4	10	0.1
5	3	0.03
6	2	0.02
	n = 100	

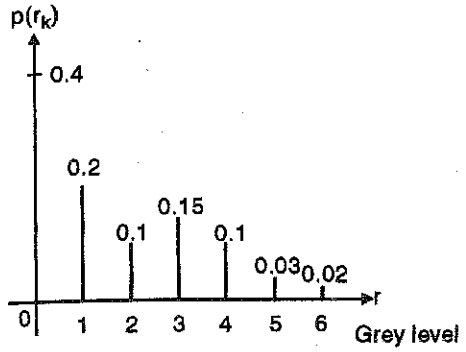
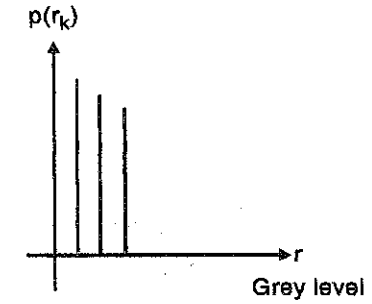
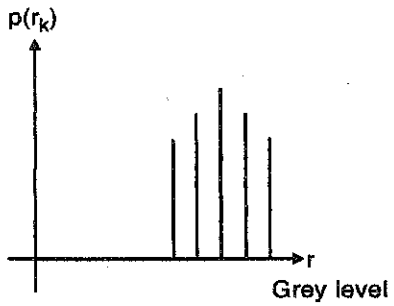


Fig. 5.1.2

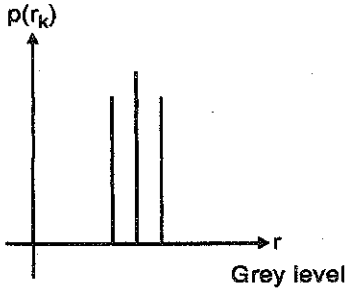
This is known as a normalised histogram. The advantage of the second method is that the maximum value to be plotted will always be 1. Generally black is considered as grey level 0 and white as the maximum. Just by looking at the histogram of the image, a great deal of information can be obtained. Some of the typical histograms are shown in Fig. 5.1.3.



(a)

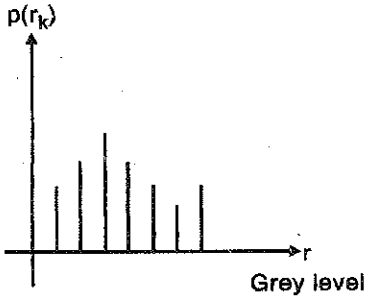


(b)



(c)

Fig. 5.1.3 Continued...



(d)

Fig. 5.1.3

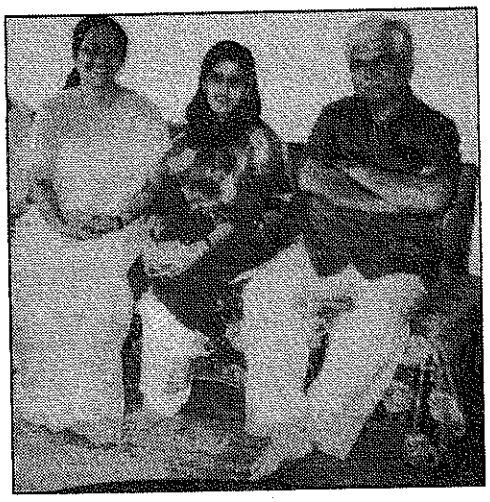
In Fig. 5.1.3(a) all the pixels belong to the lower grey levels 0, 1, ... and hence we can be sure that the image, represented by this histogram, is a dark image. Similarly, Fig. 5.1.3(b) is the histogram of a bright image. Fig. 5.1.3(c) is a low contrast image since only a small range of grey levels are present. Fig. 5.1.3(d) is a high contrast image. Of all the 4 histograms shown, the last histogram represents the best image.

Our aim in this chapter would be to transform the first three histograms into the fourth histogram. In other words, we try to increase the dynamic range of the image. Though MATLAB has a inbuilt function to plot the histogram (hist) we shall write our own code.

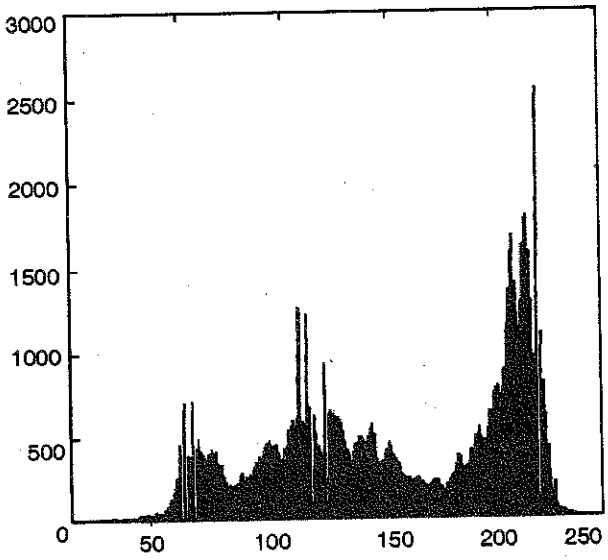
MATLAB code for plotting the histogram of an image.

```
%% Program for histogram %%
clear all
clc
a=imread('test.tif');
a=double(a);
[row col]=size(a);
h=zeros(1,300);
for n=1:1:row
for m=1:1:col
    if a(n,m)==0
```

```
                %% To ensure that the values of a are not zero
            a(n,m)=1;
        end
    end
end
for n=1:1:row
for m=1:1:col
        t=a(n,m);          %% Takes the value of the pixel ex. 12
        h(t)=h(t)+1;       %% Incrementing each counter h(12)
    end
end
%% PLOTTING %%
figure (1)
imshow(uint8(a))
figure(2)
bar(h)
```



(a) Original image



(b) Histogram of a

Fig. 5.1.4

5.2 Linear Stretching

One way to increase the dynamic range is by using a technique known as *histogram stretching*.

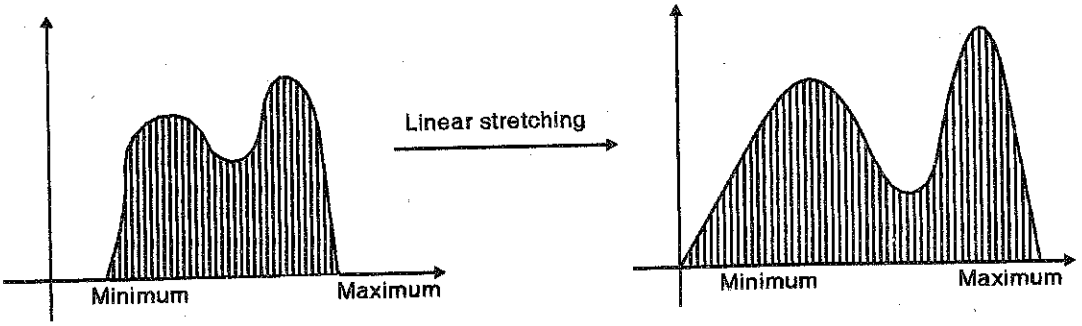


Fig. 5.2.1

In this method, we do not alter the basic shape of the histogram, but we spread it so as to cover the entire dynamic range. We do this by using a straight line equation having slope $(s_{max} - s_{min}) / (r_{max} - r_{min})$

Where s_{max} → Maximum grey level of output image

s_{min} → Minimum grey level of output image

r_{max} → Maximum grey level of input image

r_{min} → Minimum grey level of input image

$$s = T(r) = \frac{s_{max} - s_{min}}{r_{max} - r_{min}} (r - r_{min}) + s_{min} \ldots (5.2.1)$$

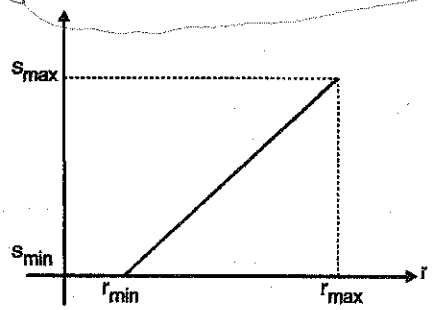


Fig. 5.2.2

This transformation function shifts and stretches the grey level range of the input image to occupy the entire dynamic range (s_{min}, s_{max})

Let us take an example

Ex. 5.2.1

Grey level	0	1	2	3	4	5	6	7
Number of pixels	0	0	50	60	50	20	10	0

Perform histogram stretching so that the new image has a dynamic range of [0, 7].

Soln. :

Here $r_{min} = 2$ $r_{max} = 6$

$s_{min} = 0$ $s_{max} = 7$

$$\therefore s = \frac{7}{4}(r - 2) + 0$$

When $r = 2$, $s = 0$

$r = 3$, $s = 1.75 \approx 2$

$r = 4$, $s = 3.5 \approx 4$

$r = 5$, $s = 5.2 \approx 5$

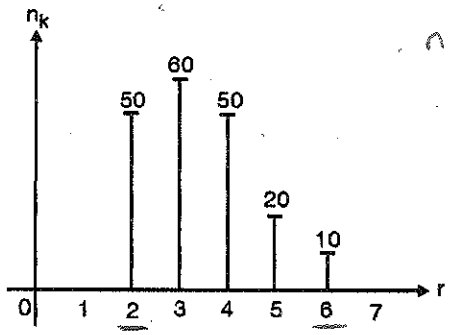
$r = 6$, $s = 7$

∴ We have

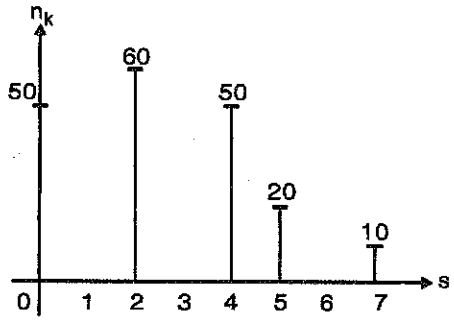
r = 2	s = 0
r = 3	s = 2
r = 4	s = 4
r = 5	s = 5
r = 6	s = 7

∴ Modified histogram is

Grey level	0	1	2	3	4	5	6	7
Number of pixels	50	0	60	0	50	20	0	10



Original histogram



Histogram after linear stretching

Fig. P. 5.2.1

We see that the shape of the histogram is retained but it is spread out or stretched.

Let us take another example of histogram linear stretching.

Ex. 5.2.2

Grey level	0	1	2	3	4	5	6	7
Number of pixels	100	90	85	70	0	0	0	0

Perform histogram stretching so that the new image has a dynamic range of [0, 7].

Soln. :

This histogram of the image is shown in Fig. P. 5.2.2. As can be concluded, it is a dark image.

Here, $r_{min} = 0, \quad r_{max} = 3,$

$s_{min} = 0, \quad s_{max} = 7$

$$s = \frac{s_{max} - s_{min}}{r_{max} - r_{min}} (r - r_{min}) + s_{min}$$

$$\therefore s = \frac{7}{3}(r - 0) + 0$$

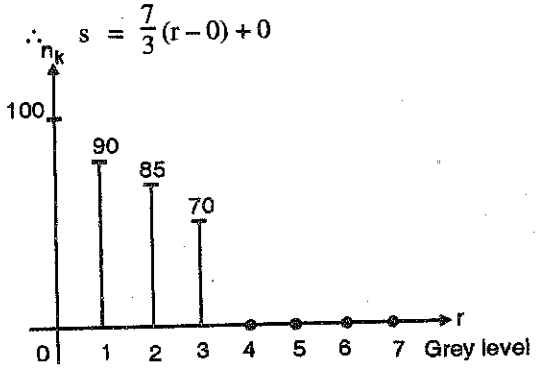


Fig. P. 5.2.2

When $r = 0, s = 0$

$$r = 1, s = \frac{7}{3}(1 - 0) = 2.3 \approx 2$$

$$r = 2, s = \frac{7}{3}(2 - 0) = 4.67 \approx 5$$

$$r = 3, s = \frac{7}{3}(3 - 0) = 7$$

$\therefore$ We have

r = 0	s = 0
r = 1	s = 2
r = 2	s = 5
r = 3	s = 7

Hence the histogram tends to spread out as shown in Fig. P. 5.2.2(a).

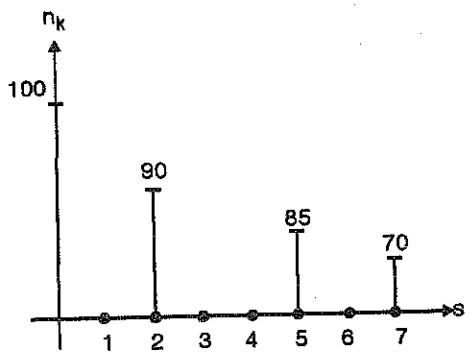


Fig. P. 5.2.2(a)

MATLAB code for histogram stretching %%

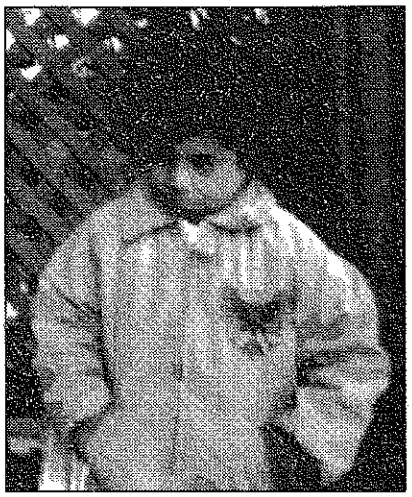
```
%% Histogram stretching %%
clear all
clc
a=imread('pout.tif');
a=double(a);
[row col]=size(a);
a=a+1;
w=min(min(a));
constant=255/(max(max(a))-min(min(a)));
cmin=0;
for x1=1:1:row
for y1=1:1:col
    c(x1,y1)=constant*(a(x1,y1)-w)+cmin;
 end
end
%% Plotting histogram of the original image %%
h=zeros(1,300);
for n=1:1:row
```

```
for m=1:1:col
t=a(n,m);      %% Takes the value of the pixel for example 12
h(t)=h(t)+1; %% Incrementing the corresponding counter h(12)
end
end
figure(1), bar(h)
```

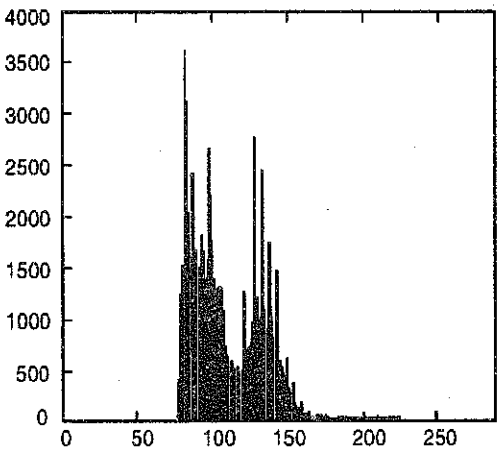
```
% Plotting histogram of the modified image %
c=c+1;
h=zeros(1,400);
for n=1:1:row
  for m=1:1:col
    t=round(c(n,m));
    h(t)=h(t)+1;
  end
end
figure(2),bar(h)
figure(3),imshow(uint8(a))
figure(4),imshow(uint8(c))
```

(a) Original image

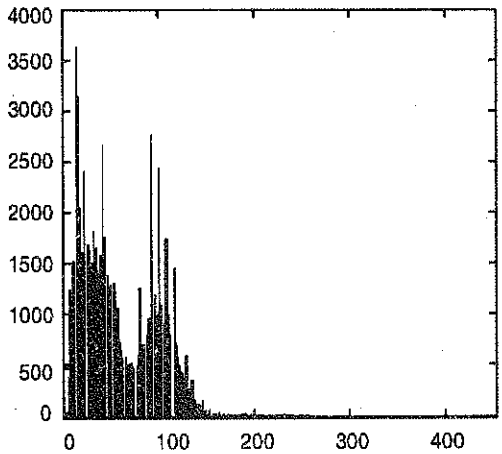
Fig. 5.2.3 Continued...



(b) Image after histogram stretching



(c) Original histogram



(d) Linear stretched histogram

Fig. 5.2.3

5.3 Histogram Equalization

Linear stretching is a good technique but as mentioned earlier, the shape remains the same.

There are many applications, wherein we need a flat histogram. This cannot be achieved by histogram stretching.

We now introduce a new technique known as histogram equalization. A perfect image is one which has equal number of pixels in all its grey levels. Hence our objective is not only to spread the dynamic range, but also to have equal pixels in all the grey levels.

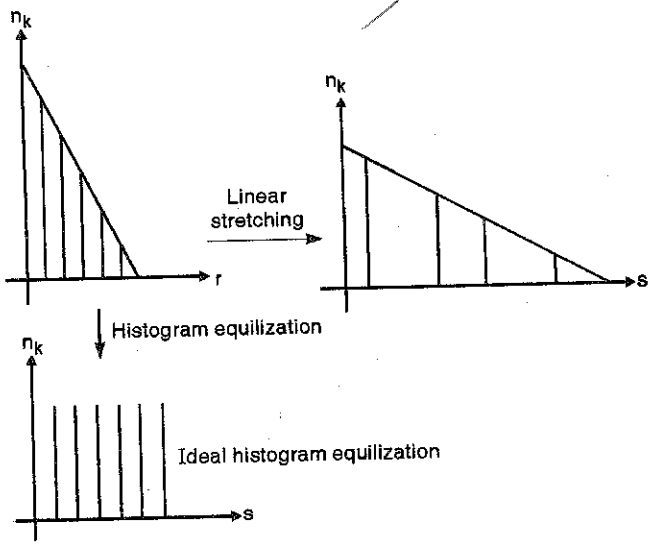


Fig. 5.3.1

This technique is known as histogram equalization. We now search for a transformation that would transform a bad histogram to a flat histogram

We know $s = T(r)$

What is this T which produces equal values in each grey level ?

The transformation that we are looking for must satisfy the following two conditions.

(a) T(r) must be single valued and monotonically increasing in the interval $0 \le r \le 1$ and

(b) $0 \le T(r) \le 1$ for $0 \le r \le 1$ i.e., $0 \le s \le 1$ for $0 \le r \le 1$

Here the range of r is taken as [0, 1]. This is called the normalized range. This range is taken for simplicity. So instead of r being in the range [0, 255] we take [0, 1]

Let us see what these two conditions mean

(1) If T (r) is not single valued. Refer Fig. 5.3.2(a). Here r_1 and r_2 are mapped as a single grey level s_1 i.e., two different grey levels look the same in the modified image. This is a big drawback; hence the transformation has to be single valued. This preserves the order from black to white.

A grey level transformation that is both single valued and monotonically increasing is shown in Fig. 5.3.2(b).

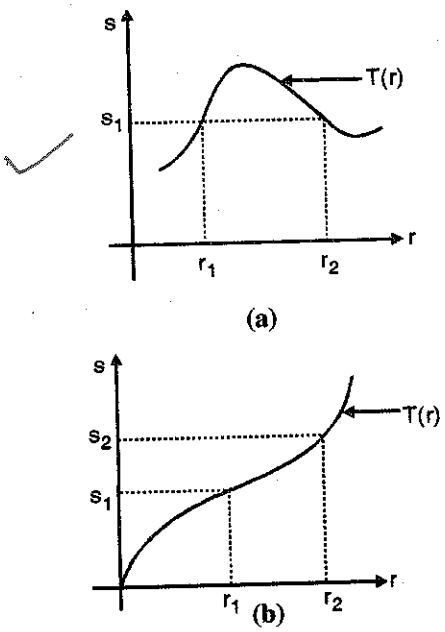


Fig. 5.3.2

(2) If condition (b) is not satisfied, then the mapping will not be consistent with the allowed range of pixel values. In other words, s will go beyond the valid range.

Since the transformation is single valued and monotonically increasing, the inverse transformation exists.

$$\therefore r = T^{-1}(s) ; 0 \le s \le 1$$

The grey levels for continuous variables can be characterized by their probability density functions $p_r(r)$ and $p_s(s)$.

From probability theory we know that if $p_r(r)$ and T(r) are known and if $T^{-1}(s)$ satisfies condition (a), then the probability density of the transformed grey level is

$$p_s(s) = \left[p_r(r)\frac{dr}{ds}\right]_{r=T^{-1}(s)} \quad \ldots(5.3.1)$$

i.e., the probability density of the transformed image is equal to the probability density of the original image multiplied by the inverse slope of the transformation.

We now need to find a transformation which would give us a flat histogram. Let us consider the Cumulative Density Function (CDF). Cumulative density function is obtained by simply adding up all the Probability Density Functions (PDF).

$$s = T(r)$$

$$s = \int_0^r p_r(r)\,dr; \quad 0 \le r \le 1 \quad \ldots(5.3.2)$$

Differentiating with respect to r, we get

$$\frac{ds}{dr} = p_r(r) \quad \ldots(5.3.3)$$

Using Equation (5.3.3) in Equation (5.3.1)

$$p_s(s) = [1] = 1; \quad 0 \le s \le 1$$

i.e., $p_s(s) = 1$

This is nothing but a uniform density function !!

Let us see what we have got here.

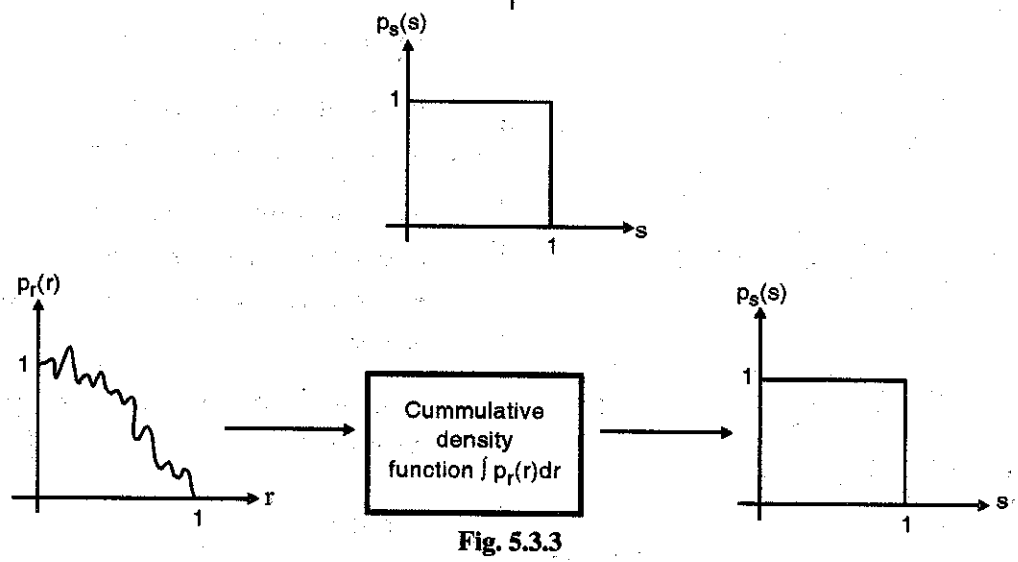


Fig. 5.3.3

A bad histogram becomes a flat histogram when we find the cumulative density function!! Let us cement this concept by taking an example.

Ex. 5.3.1

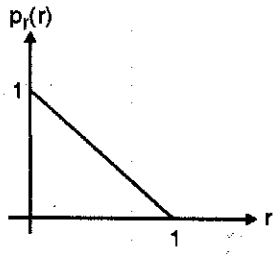


Fig. P. 5.3.1

Note : $p_r(r)$ is the PDF and hence cannot be greater than 1.

Given the above histogram, equalize it.

Soln. : From the diagram, we know that the above histogram represents a dark image.

$$p_r(r) = \begin{cases} -r+1 & ; 0 \le r \le 1 \\ 0 & ; \text{otherwise} \end{cases}$$

We shall use the CDF that we just discovered.

$$s = T(r) = \int_0^r p_r(r)dr = \int_0^r (-r+1)\,dr$$

$$s = \frac{-r^2}{2} + r$$

$$2s = -r^2 + 2r$$

$$2s - 1 = -r^2 + 2r - 1$$

$$(1-2s) = (r^2 - 2r + 1)$$

$$(1-2s) = (r-1)^2$$

$$(r-1) = \pm\sqrt{1-2s}$$

$$r = 1 \pm \sqrt{1-2s}$$

$$\because 0 \le r \le 1 \qquad \therefore r = 1 - \sqrt{1-2s}$$

We know $p_s(s) = \left[p_r(r)\frac{dr}{ds}\right] = \left[(-r+1)\frac{dr}{ds}\right]$

Substituting the value of r

$$p_s(s) = (-1 + \sqrt{1-2s} + 1)\frac{d}{ds}\left\{1 - (1-2s)^{1/2}\right\}$$

$$= \sqrt{1-2s} \cdot \frac{d}{ds}\left\{-(1-2s)^{1/2}\right\}$$

$$\left(\because \frac{d}{ds}(1) = 0\right)$$

$$= \sqrt{1-2s} \cdot \left\{\frac{-1}{2}(1-2s)^{-1/2}(-2)\right\}$$

$$= \sqrt{1-2s} \cdot \frac{1}{\sqrt{1-2s}}$$

$\therefore p_s(s) = 1$;

Hence $p_s(s) = 1; \quad 0 \le r \le 1.$

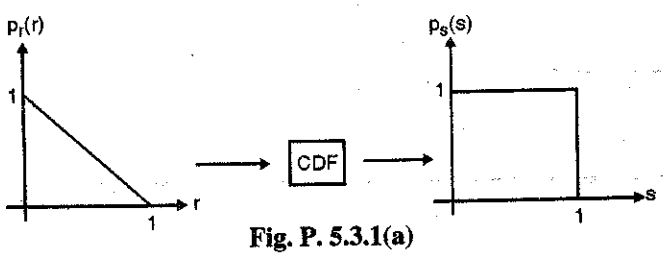


Fig. P. 5.3.1(a)

We see that $p_r(r)$ which represented a dark histogram has been transformed to an equalized histogram.

Ex. 5.3.2

Though $p_r(r)$ cannot be greater than 1, you could just consider it as another problem and solve it

$$p_r(s) = \begin{cases} -2r+2 & ; 0 \le r \le 1 \\ 0 & ; \text{otherwise} \end{cases}$$

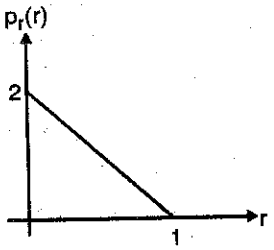


Fig. P. 5.3.2

Soln. :

Try out this problem yourself and see what you get.

We see that CDF gives us a flat response no matter what the shape of the original histogram is.

So far, we have developed the technique of histogram equalization using continuous domain mathematics. For image processing, we need to work in the discrete domain.

Hence if n is the total number of pixels in an image, then the PDF is

$$p_r(r_k) = \frac{n_k}{n}$$

We know,

$$s = T(r) = \int_0^r p_r(r)$$

In the discrete domain,

$$s_k = T(r_k) = \sum_0^r p_r(r).$$ This is the C. D. F

Let us take up an example to see how this works in the discrete domain. Some additional steps need to be taken into account while working in the discrete domain.

Ex. 5.3.3

Equalize the given histogram

Grey level	0	1	2	3	4	5	6	7
Number of pixels	790	1023	850	656	329	245	122	81

Soln. :

L = 8 (Number of grey levels)

We first plot the original histogram

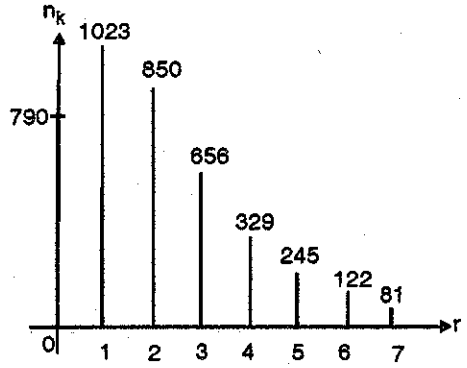


Fig. P. 5.3.3 : Original dark histogram

Grey level	n_k	PDF $p_r(r_k) = \frac{n_k}{n}$	CDF $s_k = \sum p_r(r_j)$	$(L-1) \times s_k$ i.e. $7 \times s_k$	Rounding off
0	790	0.19	0.19	1.33	1
1	1023	0.25	0.44	3.08	3
2	850	0.21	0.65	4.55	5
3	656	0.16	0.81	5.67	6
4	329	0.08	0.89	6.23	6
5	245	0.06	0.95	6.65	7
6	122	0.03	0.98	6.86	7
7	81	0.02	1	7	7
	N = 4096				

We take 1st, 2nd and the last column

Old grey level	Equalized grey level	New grey level
0	790	→ 1
1	1023	→ 3
2	850	→ 5
3	656	→ 6
4	329	→ 6
5	245	→ 7
6	122	→ 7
7	81	→ 7

We notice that the new grey levels have pixels only at 1, 3, 5, 6, 7. There are no pixels in grey levels 0, 2 and 4.

Equalized grey level	Number of pixels
0	0
1	790
2	0
3	1023
4	0
5	850
6	656 + 329 = 985
7	245 + 122 + 81 = 448

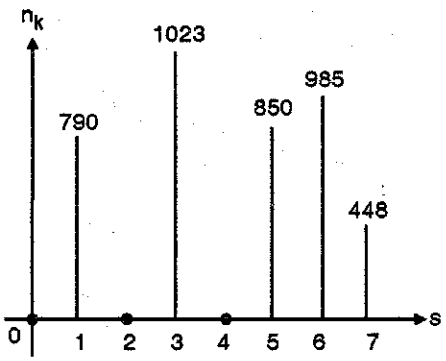


Fig. P. 5.3.3(a) : Equalized histogram

So here we are. A dark histogram becomes an evenly spaced histogram. But wait a minute, we had started out to get a flat histogram as in the continuous case but the histogram obtained is not flat; why?

Discrete domain is an approximation of the continuous domain i.e., values between 0 and 1, 1 and 2 and so on are not known. Due to this reason, perfectly flat results are never obtained.

Ex. 5.3.4

Grey level	n_k
0	100
1	90
2	50
3	20
4	0
5	0
6	0
7	0

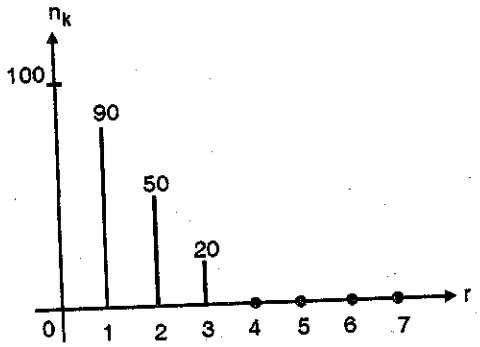


Fig. P. 5.3.4

Equalize the above histogram (Here L = 8)

Soln. :

Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$	$s_k = \sum p_r(r_k)$	$s_k \times (L-1)$ i.e. $s_k \times 7$	Rounding off
0	100	0.384	0.384	2.688	3
1	90	0.346	0.73	5.11	5
2	50	0.1923	0.9223	6.456	6
3	20	0.0769	1	7	7
4	0	0	1	7	7
5	0	0	1	7	7
6	0	0	1	7	7
7	0	0	1	7	7
	n = 260				

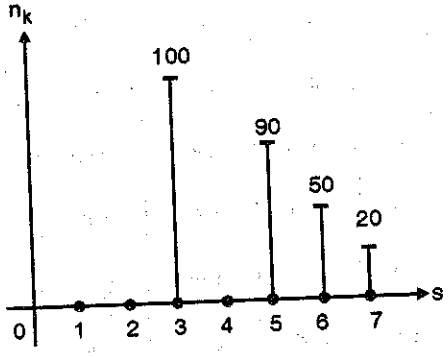
We take the 1st, 2nd and the last column

Old grey levels	n_k	New grey levels	New n_k	Modified grey levels
0	100 →	3 →	100	0 → 0
1	90 →	5 →	90	1 → 0
2	50 →	6 →	50	2 → 0
3	20	7		3 → 100
4	0	7		4 → 0
5	0	7	20	5 → 90
6	0	7		6 → 50
7	0	7		7 → 20

20 + 0 + 0 + 0 + 0 = 20

The original histogram showed that the image had only 0-3 grey levels. After histogram equalization we see that the dynamic range has increased. Hence histogram equalization does increase the dynamic range.

Next question that is usually asked is what happens when we equalize an already equalized histogram. How many times can we equalize an image ?



Equalized histogram

Fig. P. 5.3.4(a)

i.e. Dark historgram → Equalize → Flat historgram

Equalize again ?

We shall see what happens by considering an example.

Ex. 5.3.5

Given a histogram, see what happens when we equalize it twice.

Grey level	0	1	2	3
n_k	70	20	7	3

Soln. :

Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$	$s_k = \sum p_r(r_k)$	$L = 4$ $s_k \times (L-1)$	Rounding off	New n_k
0	70	0.7	0.7	2.1	2 →	70
1	20	0.2	0.9	2.7	3	
2	7	0.07	0.97	2.91	3 →	20 + 7 + 3
3	3	0.03	1	3	3	= 30
L = 4		100				

∴ The modified grey levels are

Grey level	0	1	2	3
n_k	0	0	70	30

We now equalize this again.

Grey level	n_k	$p_r(r_k)$	s_k	$L = 4$ i.e. $s_k \times (L-1)$	Rounded off	Modified grey level
0	0	0	0	0	0	
1	0	0	0	0	0	
2	70	0.7	0.7	2.1	2 →	70
3	30	0.3	1	3	3 →	30
	100					

∴ We get,

Grey level	0	1	2	3
n_k	0	0	70	30

which is the same as what we had got after the first equalization.

Hence, equalizing again gives us the same result i.e., equalizing an already equalized histogram causes no change in the histogram. Go ahead; try out your own examples. They will all give similar results.

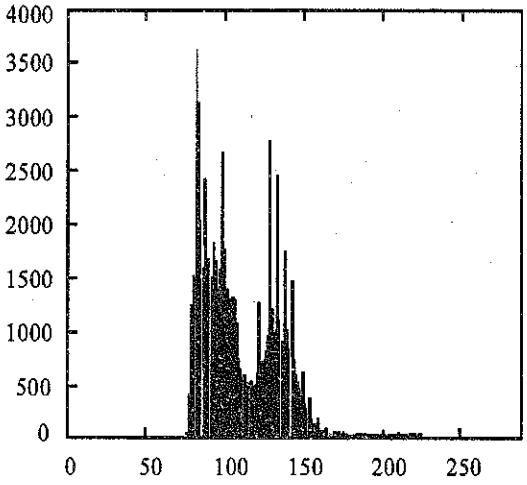
Histogram equalization is guaranteed to yield exact results only in the continuous case. As the number of levels decrease, the error between the specified and resulting histogram increase and hence we do not get a flat histogram.

MATLAB code for histogram equalization

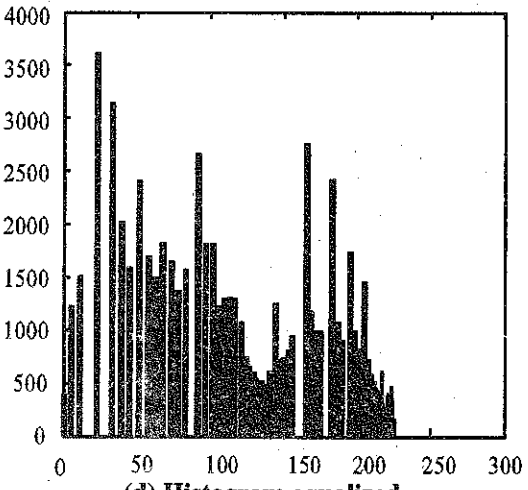
```
%% Histogram equalization %%
clear all
clc
a=imread('kids2.tif');
                    % You can try the program with 'pout.tif'%
a=double(a);
big=max((max(a)));
                    %% Gives maximum value in the image
[row col]=size(a);
C=row*col; %% Gives the total number of pixels required
h=zeros(1,300);
                    %% Defining two arrays (h and z) to store
                    the histogram values
z=zeros(1,300);
for n=1:1:row
    for m=1:1:col
        if a(n,m)==0
            %% To ensure that the values of 'a' are not zero
    a(n,m)=1;
    end
 end
end
%% Plotting the histogram of the original image %%
for n=1:1:row
    for m=1:1:col
        t=a(n,m);
            %% Takes the value of the pixel example 12 %%
```

```
        h(t)=h(t)+1;
        %% Incrementing the corresponding counter h(12)%%
    end
end
pdf=h/C;                    %% Probability function %%
cdf(1)=pdf(1);
                    %% So that we don't have the condition x(0)
                                        in the for loop below %
for x=2:1:big
cdf(x)=pdf(x)+cdf(x-1);     %% Calculating the cdf %%
end
new=round(cdf*big);         %% New is the new gray level
new=new+1;
                %% if new = 0, it is not to be taken in the loop
for p=1:1:row
for q=1:1:col
    temp=a(p,q);
    b(p,q)=new(temp);  %% b has the equalized image %%
%% Plotting the equalised histogram %%
    t=b(p,q);     %% Takes the value of the pixel example
                        12 %%
    z(t)=z(t)+1;
        %% Incrementing the corresponding counter z(12)%%
end
end
b=b-1;
    %% Since we had initially incremented the value of n %%
%% Plotting %%
figure (1), imshow(uint8(a))
figure (2), bar(h)
figure (3), imshow(uint8(b))
figure (4), bar(z)
```

(a) Original image **(b) Image after histogram equalization**



(c) Original histogram



(d) Histogram equalized

Fig. 5.3.4

5.4 Additional Examples on Histogram Modelling

Ex. 5.4.1

What effect would setting to zero the lower order bit planes have on the histogram of an image in general ?

Soln. :

The number of pixels having different grey level values would decrease, thus causing the number of components in the histogram to decrease. Since the number of pixels is constant, this would cause the height of the remaining histogram peaks to increase. Typically, less variability in grey level values will reduce the contrast.

Let us check this out with an example.

Given below is a 4×4 image having grey levels 0-15.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Histogram of the original image is

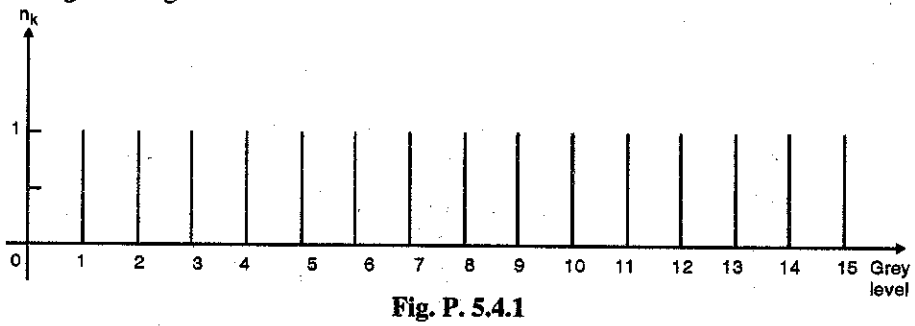


Fig. P. 5.4.1

We convert the original image to a binary image. Since the maximum grey level is 15, we need 4-bits for its representation.

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Suppose we make the last two bits equal to zero, we get

0000	0000	0000	0000
0100	0100	0100	0100
1000	1000	1000	1000
1100	1100	1100	1100

Converting back to the grey level format we get,

0	0	0	0
4	4	4	4
8	8	8	8
12	12	12	12

∴ Its histogram is

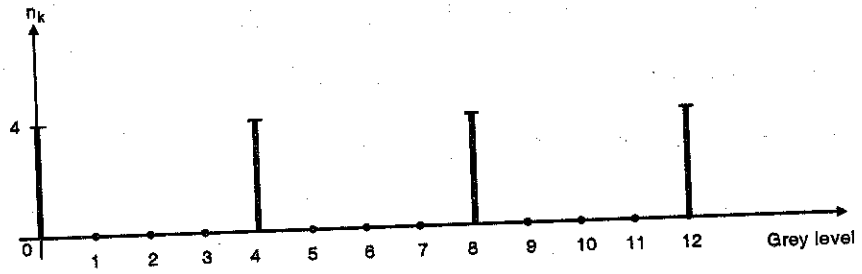


Fig. P. 5.4.1(a)

Comparing this histogram with the original histogram, we see that the variability is reduced, i.e. number of grey levels is reduced and the height of these grey levels has increased.

Ex. 5.4.2

What would be the effect on the histogram if we set to zero, the higher order bit planes.

Soln. :

Proceeding in a similar manner as in Ex. 5.4.1 and working with the same image, we get

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

(a)

Converting to binary

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

(b)

Converting the upper two bits to zero

0000	0000	0000	0000
0100	0100	0100	0100
1000	1000	1000	1000
1100	1100	1100	1100

(c)

Converting back to grey levels we get

0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3

(d)

Fig. P. 5.4.2

Plotting the histogram.

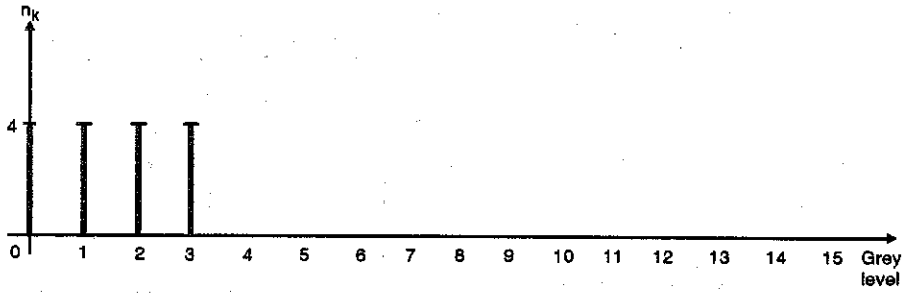
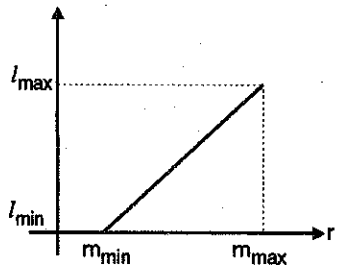


Fig. P. 5.4.2 (e)

In this case too, the grey levels reduce. But one important thing that happens is that the image becomes much darker.

Ex. 5.4.3

Given below is the slope transformation and the image. Draw the frequency tables for original and new image. Assume $l_{min} = 0$ and $l_{max} = 7$.



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

Fig. P. 5.4.3

Soln. :

This is a sum of histogram linear stretching. By frequency tables, we mean histogram. From the image, it is clear that $m_{min} = 2$, $m_{max} = 6$, $l_{min} = 0$ and $l_{max} = 7$.

Using the formula

$$l = \frac{l_{max} - l_{min}}{m_{max} - m_{min}} (m - m_{min}) + l_{min}$$

$$\therefore \quad l = \frac{7-0}{6-2} (m-2) + 0$$

$$\therefore \quad l = \frac{7}{4} (m-2)$$

From the above formulation

When, $m = 2$, $l = 0$

$m = 3$, $l = 1.75 \approx 2$

$m = 4$, $l = 3.5 \approx 4$

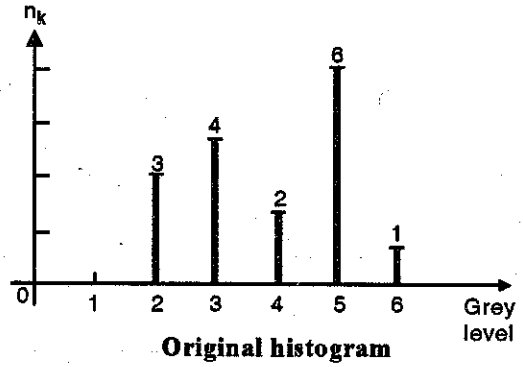
$m = 5$, $l = 5.25 \approx 5$

$m = 6$, $l = 7$

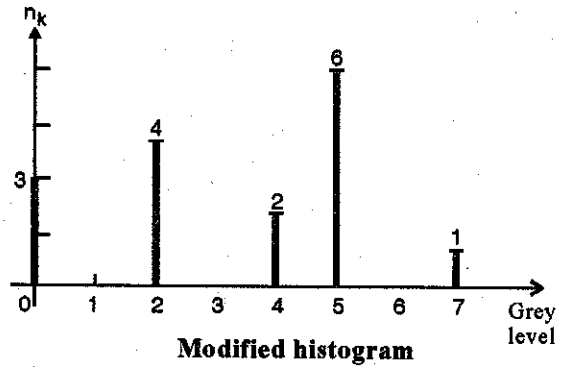
$\therefore$ The modified image is

0	2	4	0
5	5	0	4
2	7	2	5
5	2	5	5

Histogram of the original and modified images is shown in Fig. P. 5.4.3(a).



Original histogram



Modified histogram

Fig. P. 5.4.3(a)

Ex. 5.4.4

A popular procedure for image enhancement combines high frequency emphasis and histogram equalization to achieve edge

sharpening and contrast enhancement. Does it matter which of the two processes are applied first ?

Soln. :

High frequency emphasis is the same as a high boost filter. Let us see what we get when we apply a high frequency emphasis filter.

$$f(x,y) \longrightarrow \boxed{\begin{matrix} h(x,y) \\ \text{High boost} \end{matrix}} \longrightarrow f(x,y) * h(x,y) = g(x,y)$$

i.e., the output g(x, y) is a convolution of the input image and the impulse response of the high boost filter.

If we now perform histogram equalization we get

$$s = T[g(x, y)]$$

$$s = T[f(x, y) * h(x, y)] \quad \ldots(1)$$

If we decide to apply histogram equalization prior to high boost filter, we get

$$T[f(x,y)] \longrightarrow \boxed{\begin{matrix} h(x,y) \\ \text{High boost} \end{matrix}} \longrightarrow T[f(x,y)] * h(x,y)$$

$$\ldots(2)$$

We see that Equation (1) ≠ Equation (2).

∴The order of the operation does matter i.e., the final result varies if we interchange the two operations.

Ex. 5.4.5

A 8 level image is given below :

f(x, y) =

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

Image

Prepare the histogram of the given image.

Perform Histogram equalization and draw new Histogram.

Soln. :

A 8-level image implies a 3-bit image (0 to 7). We form a table of grey levels and the number of pixels in each grey level.

Grey level	0	1	2	3	4	5	6	7
No. of pixels	4	3	2	2	5	3	2	4

We now draw the original histogram.

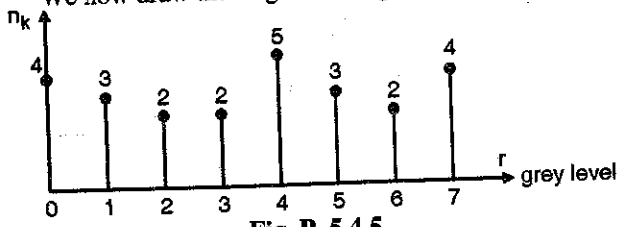


Fig. P. 5.4.5

We now perform histogram equalization.

Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$	$s_k = \sum p_r(r_k)$	$s_k \times (L-1)$ i.e. $s_k \times 7$	(New grey levels) Rounding off
0	4	0.16	0.16	1.12	1
1	3	0.12	0.28	1.96	2
2	2	0.08	0.36	2.52	3
3	2	0.08	0.44	3.08	3
4	5	0.2	0.64	4.48	4
5	3	0.12	0.76	5.32	5
6	2	0.08	0.84	5.88	6
7	4	0.16	1	7	7
	n = 25				

We take the 1st, 2nd and the last column

Old grey levels	n_k	New grey levels	New n_k
0	4	1 →	4
1	3	2 →	3
2	2	3 →	2 + 2 = 4
3	2	3	
4	5	4 →	5
5	3	5 →	5
6	2	6 →	2
7	4	7 →	4

Hence we obtain the new histogram table

Grey levels	0	1	2	3	4	5	6	7
No. of pixels	0	4	3	4	5	5	2	4

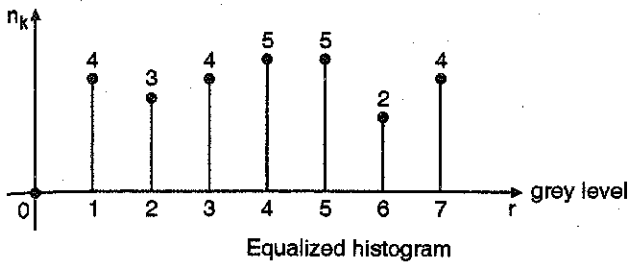


Fig. P. 5.4.5(a)

Ex. 5.4.6

Justify the statement : The entropy of an image is maximized by histogram equalization.

Soln. : Histogram equalization gives us a flat histogram in the continuous domain. As a result of this, the probability of occurrence of each grey level in the image is equal.

If all grey levels are equi-probable, the entropy is maximized.

e.g. given a 256 grey level image with equal probability of occurrence for each grey level, the maximum word length is given below.

$$H = -\sum_{i=0}^{L-1} p_i \log_2 p_i = -\sum_{i=0}^{L-1} \left(\frac{1}{256}\right) \log_2 \left(\frac{1}{256}\right)$$

$H = 8$ bits/pixel

This simply means that an equal-length code can be used on an image that has uniform probability density function.

Ex. 5.4.7

Histogram of a digital image with eight quantization level is given below. Perform histogram equalisation. Derive the transformation function and new histogram.

Grey level = r	0	1	2	3	4	5	6	7
No. of grey levels	220	140	50	60	70	170	130	160

Soln. :

Here L = 8 (8 grey levels)

The original histogram is shown in Fig. P. 5.4.7.

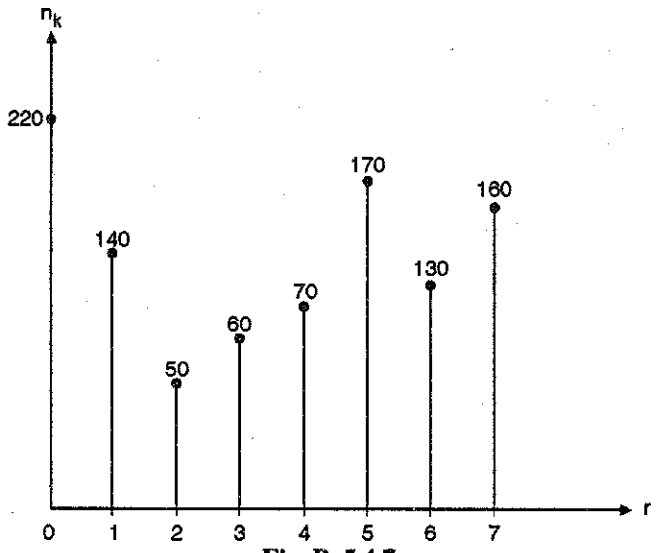


Fig. P. 5.4.7

We create a table to perform histogram equalization

Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$	$s_k = \sum p_r(r_k)$	$s_k \times (L-1)$	Rounding off
0	220	0.22	0.22	1.54	2
1	140	0.14	0.36	2.52	3
2	50	0.05	0.41	2.87	3
3	60	0.06	0.47	3.29	3
4	70	0.07	0.54	3.78	4
5	170	0.17	0.71	4.97	5
6	130	0.13	0.84	5.88	6
7	160	0.16	1	7	7
	n = 1000				

We take the 1st, 2nd and the last column

Old grey level	n_k	New grey levels	New n_k	Modified grey levels
0	220	2	220	2
1	140	3		
2	50	3 } 140 + 50 + 60	250	3
3	60	3		
4	70	4	70	4
5	170	5	170	5
6	130	6	130	6
7	160	7	160	7

Hence the equalized histogram table is shown below

Modified grey level	0	1	2	3	4	5	6	7
n_k	0	0	220	250	70	170	130	160

The equalized histogram is as shown in Fig. P. 5.4.7(a).

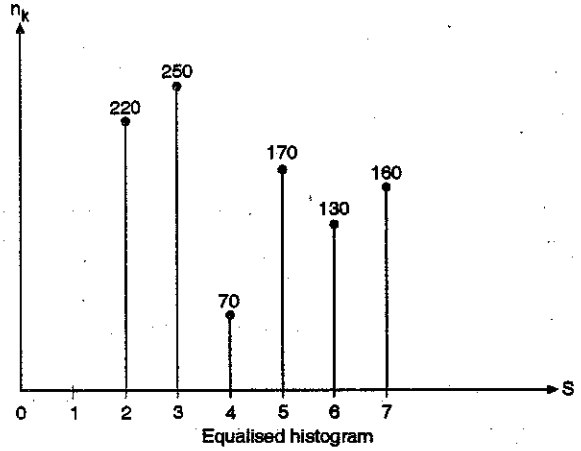


Fig. P. 5.4.7(a)

Ex. 5.4.8

Perform histogram equalization for the following image. Plot the original and equalized histograms.

Intensity	0	1	2	3	4	5	6	7
No. of Pixels	70	100	40	60	0	80	10	40

Soln. :

We begin with drawing the original histogram.

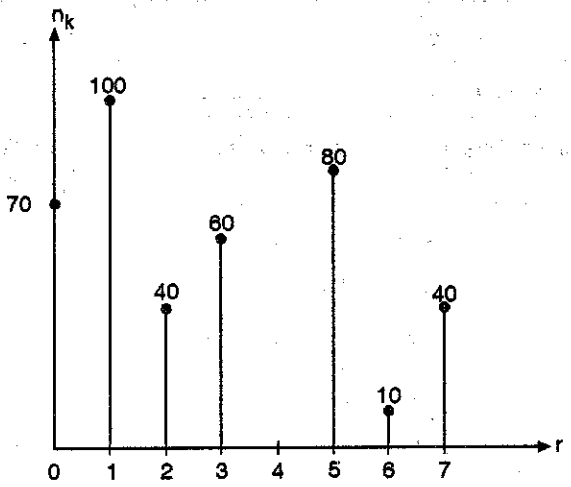


Fig. P. 5.4.8

We now perform histogram equalization

Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$ (PDF)	$s_k = \sum p_r(r_k)$ (CDF)	CDF × (L − 1) i.e. CDF × 7	Rounding off (New grey level)
0	70	0.175	0.175	1.22	1
1	100	0.25	0.425	2.97	3
2	40	0.1	0.525	3.67	4
3	60	0.15	0.675	4.72	5
4	0	0	0.675	4.72	5
5	80	0.2	0.875	6.12	6
6	10	0.025	0.9	6.3	6
7	40	0.1	1	7	7
	n = 400				

We take the 1st, 2nd and the last column.

Old grey level	n_k	New grey levels	New n_k
0	70	1 →	70
1	100	3 →	100
2	40	4 →	40
3	60	5 } →	60 + 0 = 60
4	0	5 }	
5	80	6 } →	80 + 10 = 90
6	10	6 }	
7	40	7 →	40

Hence the new histogram table is,

Grey level	0	1	2	3	4	5	6	7
n_k	0	70	0	100	40	60	90	40

The final equalized histogram is shown in Fig. P. 5.4.8(a)

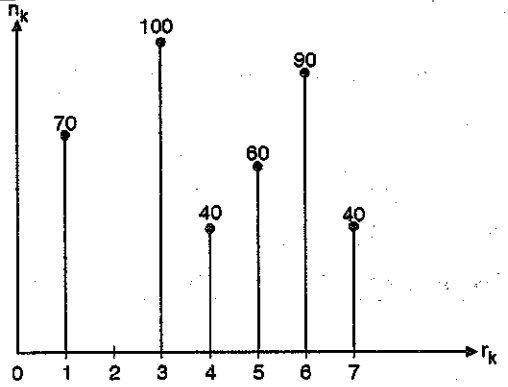


Fig. P. 5.4.8(a)

Ex. 5.4.9

Perform histogram equalization for 8 × 8 image shown in table.

Image grey level distribution

Grey Level	0	1	2	3	4	5	6	7
Number of Pixel	8	10	10	2	12	16	4	2

Soln. :

We first draw the original histogram.

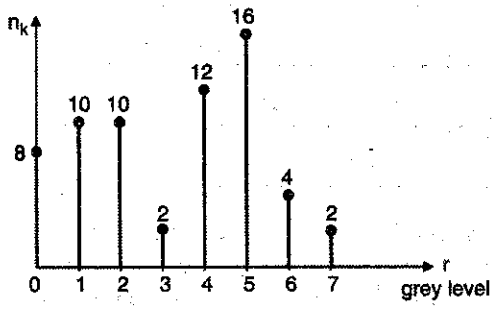


Fig. P. 5.4.9

We now perform histogram equalization by computing the PDF and the CDF.

Grey level	n_k	PDF $p_r(r_k) = \frac{n_k}{n}$	CDF $s_k = \sum p_r(r_k)$	$s_k \times (L-1)$ i.e. $s_k \times 7$	Round off	New n_k
0	8	0.125	0.125	0.875	1	8
1	10	0.156	0.281	1.967	2	10
2	10	0.156	0.437	3.059	3	10 + 2 = 12
3	2	0.031	0.468	3.276	3	
4	12	0.187	0.655	4.585	5	12
5	16	0.25	0.905	6.335	6	16
6	4	0.0625	0.967	6.769	7	4 + 2 = 6
7	2	0.031	1	7	7	
	$\sum n_k = 64$					

Hence the new histogram table is shown below.

Grey level	0	1	2	3	4	5	6	7
Number of pixels	0	8	10	12	0	12	16	6

The equalized histogram is shown in Fig. P. 5.4.9(a).

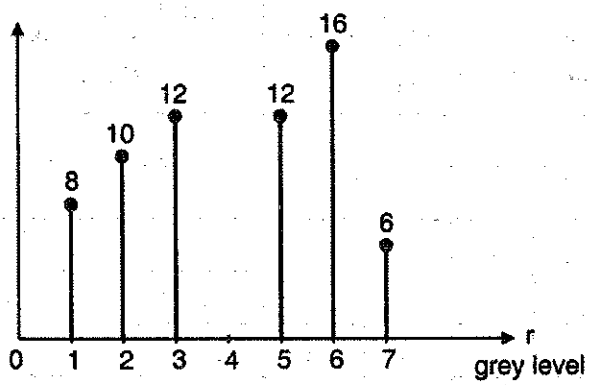


Fig. P. 5.4.9(a)

Ex. 5.4.10

Consider the following image :

$$f(x, y) = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 4 & 5 & 4 & 3 \\ 3 & 5 & 5 & 5 & 3 \\ 3 & 4 & 5 & 4 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix}$$

Write procedure for histogram equalization.

Soln. :

We start by creating the frequency table of the given image,

Grey level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	06	14	05	0	0

The original histogram is shown in Fig. P. 5.4.10.

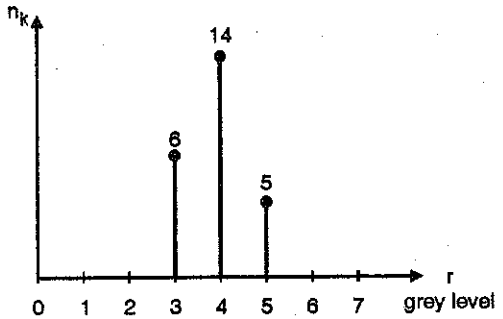


Fig. P. 5.4.10

We now perform histogram equalization by computing PDF and CDF.

Grey level	n_k	PDF $P_r(r_k) = \frac{n_k}{\sum n_k}$	CDF $s_k = \sum$ PDF	$s_k \times (L-1)$ i.e. $s_k \times 7$	(New grey level) Round off	New n_k
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	6	0.24	0.24	1.68	2	6
4	14	0.56	0.8	5.6	6	14
5	5	0.20	1	7	7	5 + 0 + 0
6	0	00	1	7	7	
7	0	00	1	7	7	
	$\sum n_k = 25$					

Hence the equalized frequency table is,

New grey level	0	1	2	3	4	5	6	7
n_k	0	0	6	0	0	0	14	5

The equalized histogram is shown in Fig. 5.4.10(a).

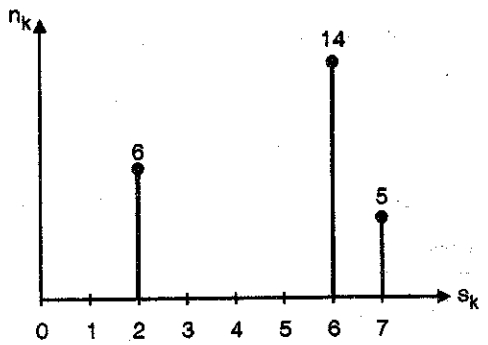


Fig. P. 5.4.10(a)

Ex. 5.4.11

A particular digital image with eight quantization levels has following histogram.

Perform histogram equalization and give new equalized histogram.

Grey level	0	1	2	3	4	5	6	7
No. of pixels	200	270	130	60	60	80	140	160

Soln. :

Here L = 8; L – 1 = 7

Grey level	n_k	PDF	CDF	CDF × (L – 1)	Round off
0	200	0.18	0.18	1.23	1
1	270	0.245	0.425	2.975	3
2	130	0.118	0.543	3.801	4
3	60	0.054	0.597	4.179	4
4	60	0.054	0.651	4.557	5
5	80	0.072	0.723	5.061	5
6	140	0.1272	0.85	5.95	6
7	160	0.145	1	7	7
	$\sum n_k = 1100$				

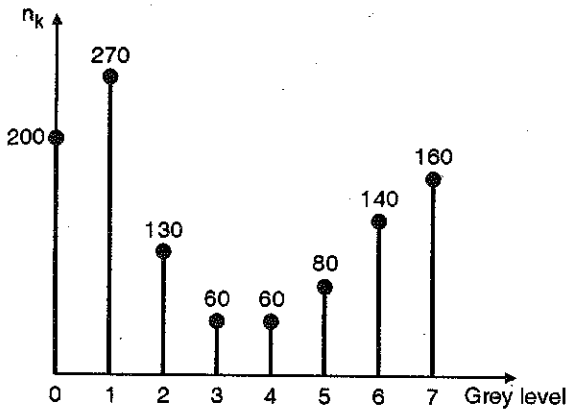
Consider the 1st, 2nd and the last column.

Grey level	n_k	New grey level Round off	
0	200	1	
1	270	3	
2	130	4	130 + 60
3	60	4	
4	60	5	60 + 80
5	80	5	
6	140	6	
7	160	7	

Hence the equalised frequency table is shown below

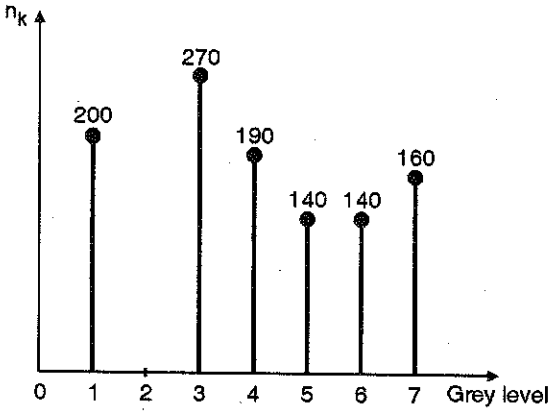
Grey level	0	1	2	3	4	5	6	7
n_k	0	200	0	270	190	140	140	160

We draw the original and modified histograms.



Original histogram

(a)



Equalized histogram

(b)

Fig. P. 5.4.11

Ex. 5.4.12

Perform Histogram Equalization and draw the Histogram for the given grey levels of an image shown :

Grey level	0	1	2	3	4	5	6	7
Frequency	123	78	281	417	639	1054	816	688

Soln. :

Here L = 8

We first plot the original histogram is shown in Fig. P. 5.4.12.

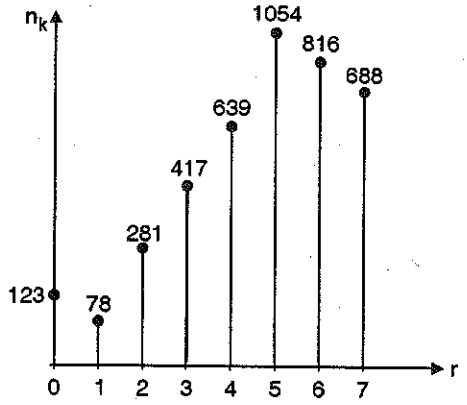


Fig. P. 5.4.12

Grey level	n_k	PDF = $\frac{n_k}{n}$ $p_r(r_k)$	CDF $s_k = \Sigma p_r(r_k)$	$(L-1) \times s_k$ i.e. $7 \times s_k$	Round off
0	123	0.03	0.03	0.21	0
1	78	0.019	0.049	0.34	0
2	281	0.068	0.117	0.819	1
3	417	0.101	0.218	1.526	2
4	639	0.156	0.374	2.618	3
5	1054	0.257	0.631	4.417	4
6	816	0.199	0.83	5.81	6
7	688	0.167	1	7	7
	N = 4096				

We consider the 1st, 2nd and last column.

Old grey level	n_k	New grey levels	New n_k
0	123	0 } 123 + 78 = 201	201
1	78	0 }	
2	281	1	281
3	417	2	417
4	639	3	639
5	1054	4	1054
6	816	6	816
7	688	7	688

Hence the equalized frequency table is shown below.

Grey level	0	1	2	3	4	5	6	7
Frequency	201	281	417	639	1054	-	816	688

The equalized histogram is shown in Fig. P. 5.4.12(a).

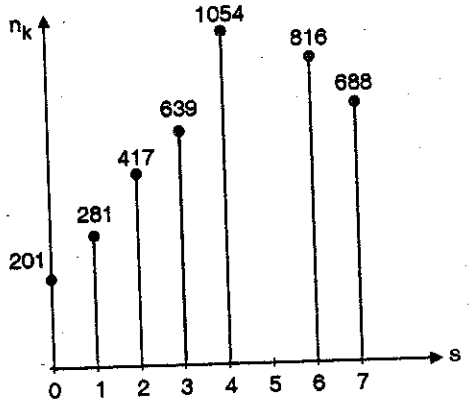


Fig. P. 5.4.12(a)

Ex. 5.4.13

Perform histogram stretching so that the new image has a dynamic range of [0, 7]

Grey level	0	1	2	3	4	5	6	7
No. of pixels	80	90	75	100	0	0	0	0

Soln. :

The original histogram is shown in Fig. P. 5.4.13.

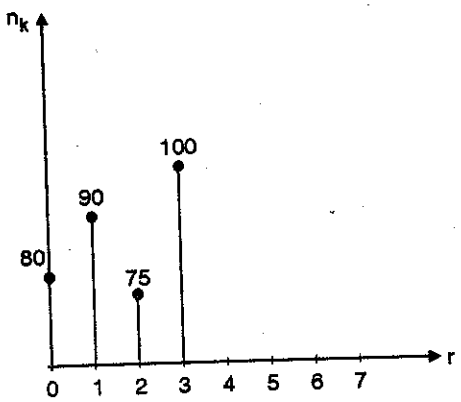


Fig. P. 5.4.13

Here $r_{min} = 0$, $r_{max} = 3$

$r_{min} = 0$, $r_{max} = 7$ since dynamic range is [0, 7].

Histogram stretching is given by the equation.

$$s = \frac{r_{max} - r_{min}}{r_{max} - r_{min}} (r - r_{min}) + r_{min}$$

$$\therefore \quad s = \frac{7}{3} (r - 0) + 0$$

When

$r = 0$; $s = 0$

$r = 1$; $s = \frac{7}{3}(1 - 0) = 2.3 \approx 2$

$r = 2$; $s = \frac{7}{3}(2 - 0) = 4.67 \approx 5$

$r = 3$; $s = \frac{7}{3}(3 - 0) = 7$

$\therefore$ We have

r = 0	s = 0
r = 1	s = 2
r = 2	s = 5
r = 3	s = 7

Hence the stretched frequency table is given below.

Grey level	0	1	2	3	4	5	6	7
Frequency	80	0	90	0	0	75	0	100

The stretched histogram is shown in Fig. P. 5.4.13(a).

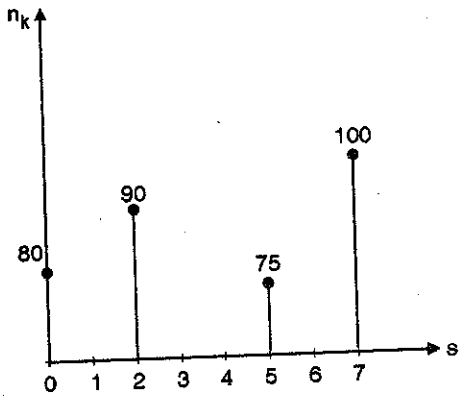


Fig. P. 5.4.13(a)

Summary

In this chapter, the concept of histograms is introduced. Though histogram processing is a spatial domain technique, it is presented here as a separate chapter because of its wide

spread applications. Histograms give us an idea about the quality of an image. One of the most common defects found in recorded images is poor contrast. This degradation may be caused by inadequate lighting, aperture size and/or shutter speed. Procedures of modifying the histogram to get a high contrast image are discussed. Emphasis has been laid on the two important techniques: linear stretching and histogram equalization.

Review Questions

Q. 1 Plot the graph of a cumulative density function (CDF).

Q. 2 Explain how histogram can aid in the process of thresholding and contrast stretching.

Q. 3 Equalize the given histogram.

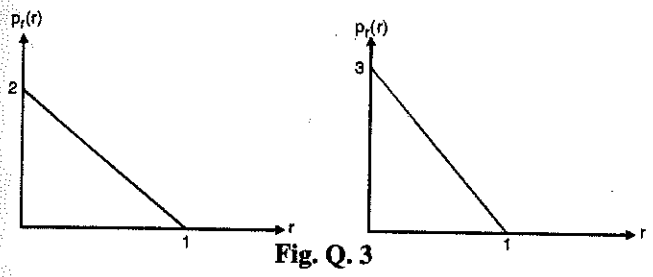


Fig. Q. 3

Note : $p_r(r)$ cannot be greater than 1 but solve this as any other mathematical problem.

Q. 4 Explain why a discrete histogram technique does not, in general, yield a flat histogram.

Q. 5 Suppose that a image is subjected to a histogram equalization. Show that a second pass of histogram equalization will produce exactly the same result as the first pass.

Q. 6 What are the two conditions that a transformation has to satisfy ?

Q. 7 Write down an algorithm to plot the histogram of an image.

Q. 8 Equalize the given histogram.

Grey	0	1	2	3	4	5	6	7
Number of pixels	50	0	50	0	50	0	50	0

Q. 9 Compare between contrast stretching and histogram equalization

Q. 10 Suppose we have a dark image which needs to be compressed and also equalized. Which operation would we use first ? Will the results be the same if the order of operations is reserved ?

Q. 11 Equalize the given image.

0	1	2
3	4	5
6	7	8

Q. 12 Given the frequency table,

Grey	0	1	2	3	4	5	6	7
Number of pixels	0	0	a	b	c	d	e	0

Perform linear stretching.

Chapter Ends...

❑❑❑

CHAPTER 6

Image Enhancement in Frequency Domain

6.1 Introduction

In the last two chapters we discussed in detail what we mean by image enhancement and how it is achieved in the spatial domain. In this chapter, we will discuss what we mean by the frequency domain and how enhancement is performed in the frequency domain. By the time we reach the end of this chapter, you would understand the relationship between the spatial and the frequency domain. You would realise that it is fairly simple to move from one domain to the other.

Frequency domain techniques are all about working in the Fourier domain. We will first discuss the Fourier transform and then proceed to understand image enhancement in the frequency domain.

6.2 The Fourier Transform

Fourier transform, named after the French mathematician Jean Baptiste Joseph Fourier who was born in 1768 in Auxerre, is one of the most important transforms that is used in Digital Signal Processing and Image Processing. These subjects would not be what they are had the Fourier transform not been discovered. Fourier transform gave a new direction and understanding to Digital Signal Processing and Image Processing.

To understand what the Fourier transform does, take a simple example that we all must have observed. Consider Fig. 6.2.1.

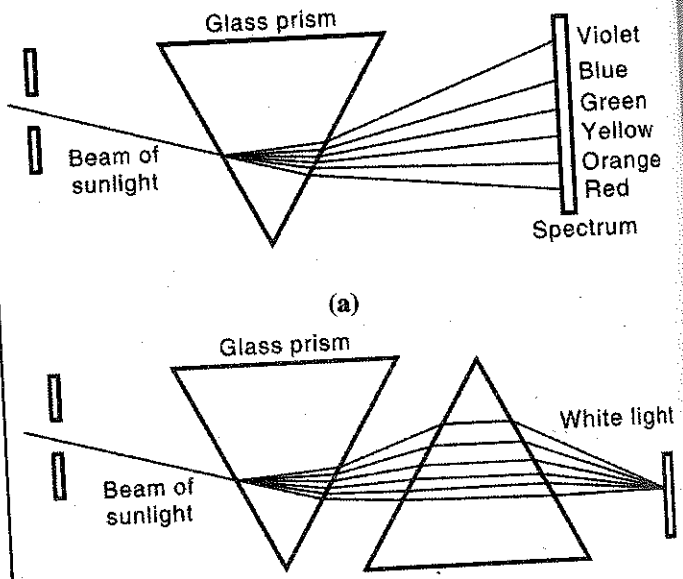


Fig. 6.2.1

We are all familiar with what is shown here. White light when passed through a prism, gets split into different colours known as spectrum. The colours that we see are VIBGYOR. It means that white light is actually made up of these 7 colours. In a paper submitted in 1672 to the Royal Society, Isaac Newton used the term spectrum to describe the continuous band of colours produced by this apparatus. From physics we know that each colour corresponds to a specific frequency of the visible spectrum. Hence analysis of light into colour is actually a form of frequency analysis.

The prism splits white light into separate colours (wave lengths). What the prism does to light, the Fourier transform does to signals. This is what Fourier transform is all about : Splitting signals into its component signals. The second prism that gives back the white light is equivalent to the

inverse Fourier transform. Fourier transform states that any function that periodically repeats itself can be expressed as the sum of sines and cosines of different frequencies and different amplitudes.

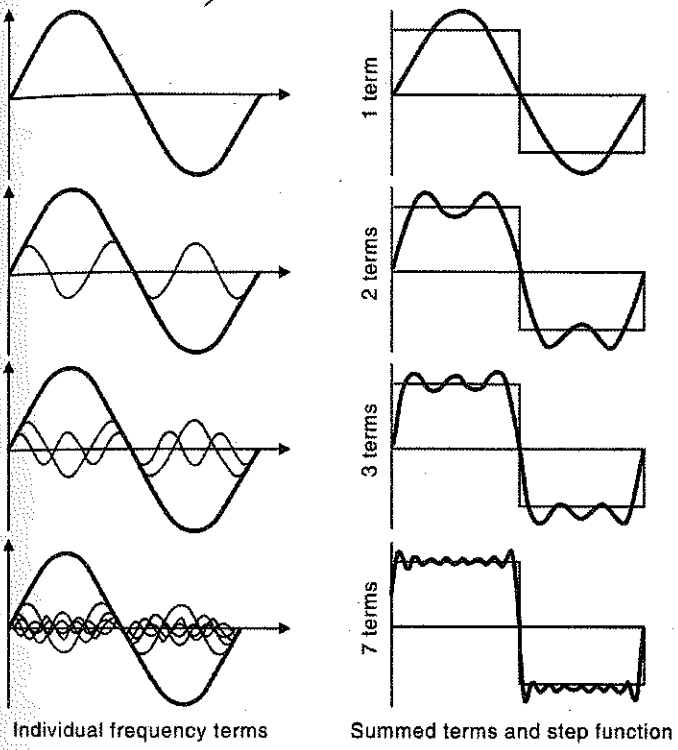


Fig. 6.2.2 : Summation of Fourier frequency terms to fit a simple step function

Hence the Fourier transform is nothing but a mathematical prism. It can be said that the original signal is composed of infinite frequencies of varying amplitudes and the Fourier transform tells us what these frequencies are. Fourier series is used for periodic signals while the Fourier transform is used for aperiodic signals. Since most signals and images that we encounter are studied over a finite duration, they are all aperiodic.

Hence the utility of Fourier transform is far greater than the Fourier series. We will discuss only the Fourier transform in this book. Now that we know what the Fourier transform does, we shall dwell on the mathematics of this wonderful transform.

6.3 1-Dimensional Fourier Transform

Let f(x) be a continuous function of x. The Fourier transform of f(x) is

$$F(u) = \int_{-\infty}^{\infty} f(x)\, e^{-j2\pi ux}\, dx \qquad \ldots(6.3.1)$$

$$F(u) = \int_{-\infty}^{\infty} f(x)\, [\cos 2\pi ux - j \sin 2\pi ux]\, dx$$

Similarly, the inverse Fourier transform is

$$f(x) = \int_{-\infty}^{\infty} F(u)\, e^{+j2\pi ux}\, du$$

Because F(u) is complex

$$F(u) = R(u) + j\, I(u)$$

where R is real and I is imaginary.

Fig. 6.3.1

F(u) has a magnitude plot as well as a phase plot.

$$\text{Mag} \rightarrow |F(u)| = [R^2(u) + I^2(u)]^{1/2}$$

$$\text{Phase} \rightarrow \angle F(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right]$$

$|F(u)|$, when plotted is called the magnitude plot or the Fourier spectrum.

At times, we also plot the power spectrum.

$$P(u) = |F(u)|^2 = R^2(u) + I^2(u)$$

It is the magnitude plot that gives us maximum information and hence we normally plot only the magnitude.

Ex. 6.3.1

Find the Fourier transform of the signal shown in Fig. P. 6.3.1(a).

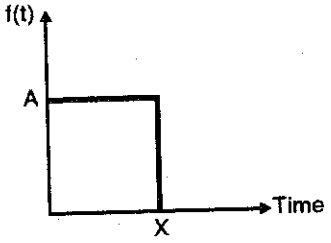


Fig. P. 6.3.1(a)

Soln. :

The signal is continuous and aperiodic and hence we use the Fourier transform.

$$F(u) = \int_{-\infty}^{\infty} f(t)\, e^{-j2\pi ut}\, dt = \int_{0}^{x} A\, e^{-j2\pi ut}\, dt$$

$$= \frac{A}{-j2\pi u}\left[e^{-j2\pi ut}\right]_0^x$$

$$F(u) = \frac{-A}{j2\pi u}\left[e^{-j2\pi ux} - 1\right] = \frac{A}{j2\pi u}\left[1 - e^{-j2\pi ux}\right]$$

$$= \frac{A}{j2\pi u}\left[e^{+j\pi ux} - e^{-j\pi ux}\right] e^{-j\pi ux}$$

$$= \frac{A}{\pi u}\left[\frac{e^{+j\pi ux} - e^{-j\pi ux}}{2j}\right] e^{-j\pi ux}$$

$$F(u) = \frac{A}{\pi u}\left[\sin(\pi ux) e^{-j\pi ux}\right]$$

$$F(u) = A\frac{\sin(\pi ux)}{\pi u}\left[e^{-j\pi ux}\right]$$

Multiplying and dividing by x, we get

$$F(u) = Ax\frac{\sin(\pi ux)}{\pi ux}\left[e^{-j\pi ux}\right]$$

As mentioned F(u) is complex. The magnitude values are

$$|F(u)| = |Ax|\left|\frac{\sin(\pi ux)}{\pi ux}\right| |e^{-j\pi ux}|$$

$$\because \quad |e^{-j\pi ux}| = |\cos \pi ux - j \sin \pi ux|$$

$$= \sqrt{\cos^2 \pi ux + \sin^2 \pi ux} = 1$$

$$\therefore \quad |F(u)| = Ax\left|\frac{\sin(\pi ux)}{\pi ux}\right| 1$$

Hence for different values of u, we get a magnitude plot as shown in Fig. P. 6.3.1(b)

How did we get F(u) = Ax at u = 0 ?

$$\text{At } u = 0;\ |F(u)| = Ax\left|\frac{\sin 0}{0}\right|$$

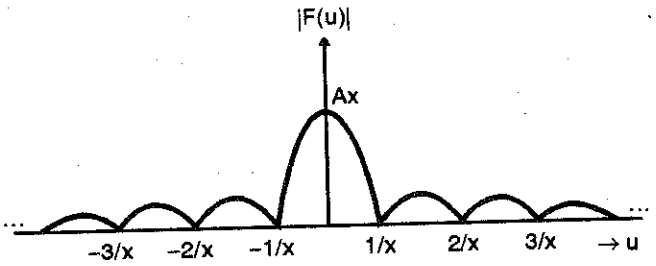


Fig. P. 6.3.1(b)

Now, $\frac{\sin 0}{0}$ is indeterminate. In such cases, we use the L'Hospital's rule, i.e., we take the derivative of the numerator and the denominator with its limit tending to zero.

$$\text{i.e., } \frac{\sin x}{x}\Big|_{x=0} \qquad \lim_{x \to 0}\left[\frac{\frac{d}{dx}\sin x}{\frac{dx}{dx}}\right] = \lim_{x \to 0}\frac{\cos x}{1} = 1$$

$$\therefore |F(u)| \text{ at } u = 0 \text{ is } Ax \cdot 1 = Ax$$

Note that since the original sequence was aperiodic, the spectrum will be continuous. You can verify this property by referring to any standard book on Digital Signal Processing. Let us interpret the results that we have just obtained.

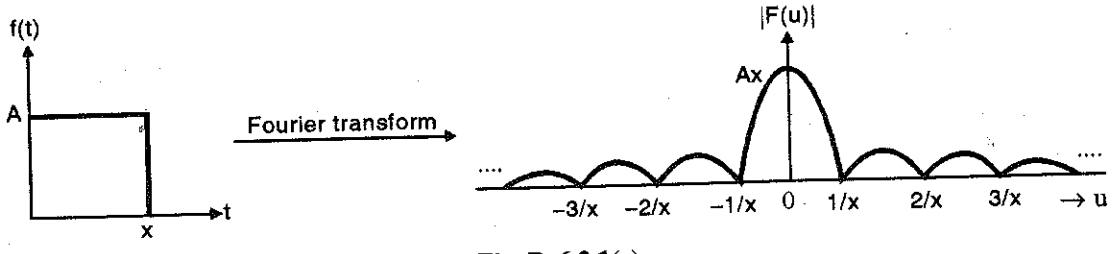


Fig. P. 6.3.1(c)

The Fourier spectrum tells us that a pulse of the kind shown is made up of infinite frequencies. The frequencies and their amplitudes are what is seen from the Fourier spectrum. On observing the spectrum, we see that the frequency component at 0 has the maximum amplitude and at higher frequencies, amplitude goes on decreasing. If we add up all these frequencies (u), we get back the original signal.

Given below is the MATLAB code for generating the spectrum.

```
%% Program for plotting the 1-D Fourier transform %%
clear
clc
A = input ( 'enter the value of A (amplitude) : ' ) ;
X = input ( 'enter the value of X (length) : ' ) ;
u = – 5: 05 : 5 ;
a = sin ( pi * X * u ) ;
b = pi *u * X ;
F = ( A * X ) * ( a. /b) ;
%% Plotting %%
plot (u, ( F ) )
%% we have not considered F(u) at u = 0 separately %
%% as that would have complicated the program %
```

6.4 2-Dimensional Fourier Transform

Images being 2-dimensional functions, we need to define a 2-D Fourier transform. The good thing is that the 1-D Fourier transform can be easily extended to a function, f(x, y), of two variables.

$$F\{f(x, y)\} = F(u, v)$$

$$= \int_{-\infty}^{\infty}\int f(x, y)\, e^{-j2\pi(ux + vy)}\, dx\, dy \quad \text{and} \qquad \ldots(6.4.1)$$

$$F^{-1}\{F(u, v)\} = f(x, y) = \int_{-\infty}^{\infty}\int F(u, v)\, e^{j2\pi(ux + vy)}\, du\, dv$$

Just as in the case of the 1-D Fourier transform, the magnitude plot is given by,

$$|F(u, v)| = [R^2(u, v) + I^2(u, v)]^{1/2}$$

and the phase plot is given by,

$$F(u, v) = \tan^{-1}\left[\frac{I(u, v)}{R(u, v)}\right]$$

The power spectrum is given by

$$P(u, v) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v)$$

Let's consider an example.

Ex. 6.4.1

Find the Fourier transform of the given function.

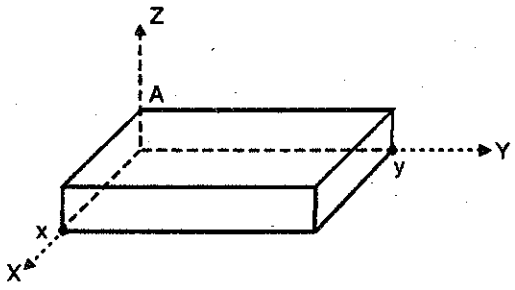


Fig. P. 6.4.1

Soln. :

$$F(u, v) = \int_{-\infty}^{\infty}\int f(x, y)\, e^{-j2\pi(ux + vy)}\, dx\, dy$$

$$F(u, v) = \int_{0}^{x}\int_{0}^{y} A\, e^{-j2\pi(ux + yv)}\, dx\, dy$$

$$= A\left[\int_{0}^{x} e^{-j2\pi ux}\, dx \int_{0}^{y} e^{-j2\pi vy}\, dy\right]$$

$$F(u, v) = A\left\{\left[\frac{e^{-j2\pi ux}}{-j2\pi u}\right]_0^x \left[\frac{e^{-j2\pi vy}}{-j2\pi v}\right]_0^y\right\}$$

$$= A\left\{\left[\frac{1 - e^{-j2\pi ux}}{j2\pi u}\right]\left[\frac{1 - e^{-j2\pi vy}}{j2\pi v}\right]\right\}$$

$$= A \times \left\{\left[\frac{e^{+j\pi ux} - e^{-j\pi ux}}{j2\pi u}\right] e^{-j\pi ux} \left[\frac{e^{+j2\pi vy} - e^{-j\pi vy}}{j2\pi v}\right] e^{-j\pi vy}\right\}$$

$$F(u, v) = Axy \left[\frac{\sin(\pi ux)}{\pi ux}\right] e^{-j\pi ux} \left[\frac{\sin(\pi vy)}{\pi vy}\right] e^{-j\pi vy}$$

$\therefore$ Magnitude

$$|F(u, v)| = Axy \left|\frac{\sin(\pi ux)}{\pi ux}\right| \left|\frac{\sin(\pi vy)}{\pi vy}\right|$$

$\because |e^{-j\pi ux}| = \sqrt{\cos^2 \pi ux + \sin^2 \pi ux} = 1$

$|e^{-j\pi vy}| = \sqrt{\cos^2 \pi vy + \sin^2 \pi vy} = 1$

MATLAB code for plotting 2-D Fourier transform

```
%% Program for plotting the 2-D Fourier transform %%
clear
clc
A = input('enter the value of A(amplitude)      :   ');
X = input('enter the value of X(length)         :   ');
Y = input('enter the value of Y(length)         :   ');
    i=1;
    j=1;
    for u = -5:0.1:5;
            for v  = -5:0.1:5;
                a = sin(pi*X*u);
                b = pi*u*X;
                a1 = sin (pi*Y*v);
                b1 = pi*v*Y;
                F(i,j) = (A*X*Y)*((a./b)*(a1./b1));
                  i=i+1
                end
                  i=1
                  j=j+1
                end
                %% Plotting %%
                mesh (abs(F))
```

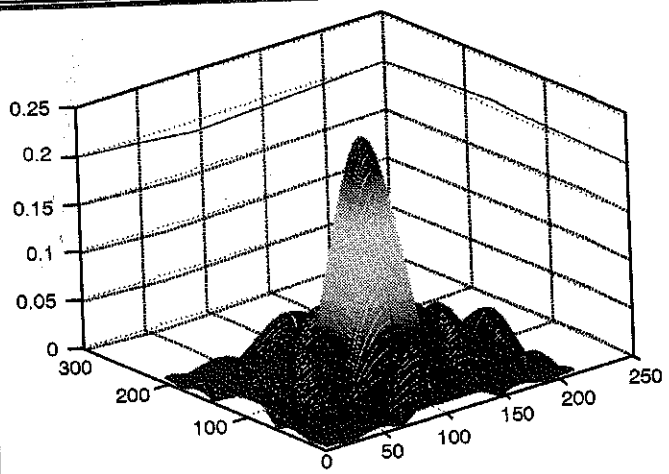


Fig. 6.4.1 : 2-D Fourier transform

As mentioned earlier we only plot the magnitude response.

We ought to know that since both the sequences i.e., 1-D and 2-D signals, are aperiodic the Fourier spectrum is continuous and that is what is seen in the figure (if this statement is not clear, the reader is requested to read any standard book on Digital Signal Processing).

6.4.1 Discrete Fourier Transform (DFT)

Continuous spectrums cannot be of use unless we discretize it. Discretization ensures that we get specific frequencies. This is known as the Discrete Fourier Transform (DFT). Whenever we work on a computer, we need to use the DFT.

Note : We cannot study a continuous signal on a computer. Whenever we feed a analog signal to it, it has to be done through an ADC (Analog to Digital Converter) card. In short we discretize the input signal.

The DFT is a slight modification of the Fourier transform just studied.

The DFT is given by,

$$F(u) = \sum_{x=0}^{N-1} f(x)\, e^{-j2\pi ux/N} \; ; u = 0, 1, ..., N-1 \qquad ...(6.4.2)$$

Where N is the number of samples of the input signal.

The Inverse Discrete Fourier Transform (IDFT) is given by

$$f(x) = \frac{1}{N} \sum_{u=0}^{N-1} F(u)\, e^{+j2\pi ux/N} \quad ; \text{ for } x = 0, 1, 2, \ldots\ldots, N-1$$

...(6.4.3)

Note : $\frac{1}{N}$ is a scaling function. This book uses $\frac{1}{N}$ in the IDFT formula some books prefer to write it in the forward DFT formula. The formula used here is what MATLAB uses in its library function. The reason for using the scaling function will be clear in chapter of Image Transforms.

The values u = 0, 1, ..., N – 1 in the DFT correspond to samples of the continuous spectrum.

In a similar fashion, a 2-D discrete Fourier transform is

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

and

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

Ex. 6.4.2

Find the DFT of f(x) = {0, 1, 2, 1}

Soln. :

$$F(u) = \sum_{x=0}^{N-1} f(x)\, e^{-j2\pi\left(\frac{ux}{N}\right)} \qquad \text{Here } N = 4$$

$$\therefore F(u) = \sum_{x=0}^{3} f(x)\, e^{-j2\pi\left(\frac{ux}{4}\right)} \quad u = 0,1, \ldots\ldots, N-1$$

$$\therefore F(0) = \left[f(0)\, e^{-j2\pi\frac{0\times0}{4}} + f(1)\, e^{-j2\pi\frac{0\times1}{4}} + f(2)\, e^{-j2\pi\frac{0\times2}{4}} + f(3)\, e^{-j2\pi\frac{0\times3}{4}}\right]$$

$$= [f(0) + f(1) + f(2) + f(3)] = [0 + 1 + 2 + 1]$$

$$\mathbf{F(0) = 4}$$

$$F(1) = \left[f(0)\, e^{-j2\pi\frac{1\times0}{4}} + f(1)\, e^{-j2\pi\frac{1\times1}{4}} + f(2)\, e^{-j2\pi\frac{1\times2}{4}} + f(3)\, e^{-j2\pi\frac{1\times3}{4}}\right]$$

$$= \left[f(0) + f(1)\, e^{-j\frac{2\pi}{4}} + f(2)\, e^{-j\frac{4\pi}{4}} + f(3)\, e^{-j\frac{6\pi}{4}}\right]$$

$$= \left\{0 + 1\left[\cos\frac{\pi}{2} - j\sin\frac{\pi}{2}\right] + 2\,[\cos\pi - j\sin\pi] + \left[1\cos\frac{6\pi}{4} - j\sin\frac{6\pi}{4}\right]\right\}$$

$$F(1) = 0 - j - 2 + j$$

$$\mathbf{F(1) = -2}$$

$$F(2) = \left[f(0)\, e^{-j2\pi\frac{2\times0}{4}} + f(1)\, e^{-j2\pi\frac{2\times1}{4}} + f(2)\, e^{-j2\pi\frac{2\times2}{4}} + f(3)\, e^{-j2\pi\frac{2\times3}{4}}\right]$$

$$F(2) = \{0 + 1\,[\cos\pi - j\sin\pi] + 2\,[\cos 2\pi - j\sin 2\pi] + [\cos 3\pi - j\sin 3\pi]\}$$

$$= [0 - 1 + 2 - 1]$$

$$\mathbf{F(2) = 0}$$

$$F(3) = \left[f(0)\, e^{-j2\pi\frac{3\times0}{4}} + f(1)\, e^{-j2\pi\frac{3\times1}{4}} + f(2)\, e^{-j2\pi\frac{3\times2}{4}} + f(3)\, e^{-j2\pi\frac{3\times3}{4}}\right]$$

$$= \left\{0 + 1\left[\cos\frac{6\pi}{4} - j\sin\frac{6\pi}{4}\right] + 2\,[\cos 3\pi - j\sin 3\pi] + \left[1\cos\frac{9\pi}{2} - j\sin\frac{9\pi}{2}\right]\right\}$$

$$= 0 - j - 2 + j$$

$$\mathbf{F(3) = -2}$$

$$\therefore F(u) = \{4, -2, 0, -2\}$$

The DFT spectrum is obtained by taking the magnitude

$$\therefore |F(0)| = 4$$

$$|F(1)| = 2$$

$$|F(2)| = 0$$

$$|F(3)| = 2$$

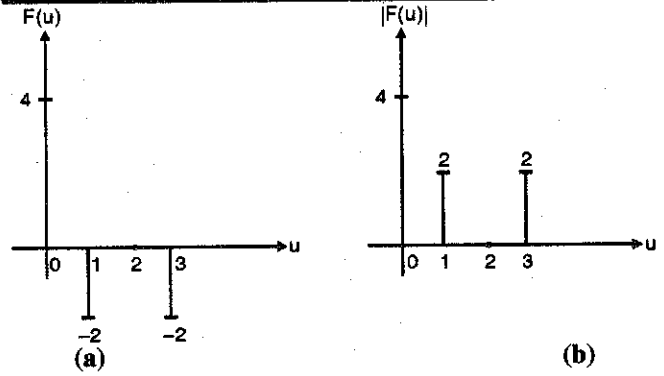


Fig. P. 6.4.2

There is another much easier way to calculate the DFT. This method is known as the DFT matrix method. We shall now explain this method.

If x(n) is the input sequence and X[k] is the DFT, then we have

$$X[k] = \sum_{n=0}^{N-1} x(n)\, e^{\frac{-j2\pi nk}{N}}$$

We now define a new factor, W_N, called the Twiddle factor, where

$$W_N = e^{-j2\pi/N}$$

$\therefore$ The DFT equation reduces to

$$X[k] = \sum_{n=0}^{N-1} x(n)\, W_N^{nk} \qquad \ldots(6.4.4)$$

To solve this equation, we form a square matrix W_N of size $N \times N$.

Equation (6.4.4) reduces to

$$X = W \cdot x \qquad \ldots..(6.4.5)$$

Let us consider the example that was solved previously,

$x(n) = \{0, 1, 2, 1\}$

$N = 4 \therefore W_N$ will be a 4×4 matrix

$$W_4 = \begin{matrix} \\ 0 \\ 1 \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} k\,n \rightarrow 0 & 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} \end{matrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \rightarrow \text{DFT matrix}$$

Similarly depending on the number of samples (N) of the input we form a matrix W_N of size $N \times N$. Multiplying the input sequence with this matrix gives us the DFT sequence X(k). This matrix method makes the whole procedure of calculating the DFT much easier.

Ex. 6.4.3

Find the DFT of the sequence,

$x(n) = \{0, 1, 2, 1\}$

Soln. :

In this case number of samples is $N = 4$

$$\therefore W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Multiplying W_4 with x(n) gives X[k]

$$\begin{matrix} X[k] & & W_4 & & x(n) \end{matrix}$$

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

$$\therefore X[k] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

$$\therefore X[k] = \{4, -2, 0, -2\}$$

This answer agrees with the conventional method. Remember, the size of the DFT matrix depends on the number of points in the input sequence.

We use this matrix method to compute the DFT on the computer.

Though there is an inbuilt command in MATLAB for calculating the 1-D DFT, we shall write our own small code to calculate the DFT based on the matrix method.

```
%% 1-D Fourier transform using the DFT matrix %%

clear all

clc

a = [0 1 2 1]
```

```
s = size(a);

aa = fft(a);  %% Solving it using the inbuilt function of fft for

                    %% .. Comparing the results %%

%% To find the Twiddle factor %%

for k = 0:1:s( 2 )-1

    for n = 0:1:s(2)-1

    p = exp ( -i*2*pi*k*n/s( 2 ) );

    x ( k+1,n+1 ) = p;          %% x is the DFT matrix %%

end

end

%% To find the 1-D Fourier Transform %%

    result = x*a';

end

aa, result
```

6.4.2 2-D- Discrete Fourier Transform (DFT)

Image processing is all about 2-dimensional functions, hence 1-D DFT is not applicable here. In case of images we need to work with the 2-D DFT.

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

There are some properties which are invaluable in calculating the 2D DFT and they make the task easier.

6.4.3 Properties of Discrete Fourier Transform

(1) The Separability Property

This property states that a 2-D DFT can be separated into two 1-D DFTs.

We know (assume a square image)

$$F(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi\left(\frac{ux}{N} + \frac{vy}{N}\right)} \quad \ldots(6.4.6)$$

This can be split up as

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

$$\text{Let } F(x, v) = \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi\frac{vy}{N}}$$

$$\therefore F(u, v) = \sum_{y=0}^{N-1} f(x, v)\, e^{-j2\pi\frac{ux}{N}} \quad \ldots(6.4.7)$$

As stated, the principle advantage of the separability property is that the 2-D DFT can be obtained in two steps by successive application of 1-D DFT.

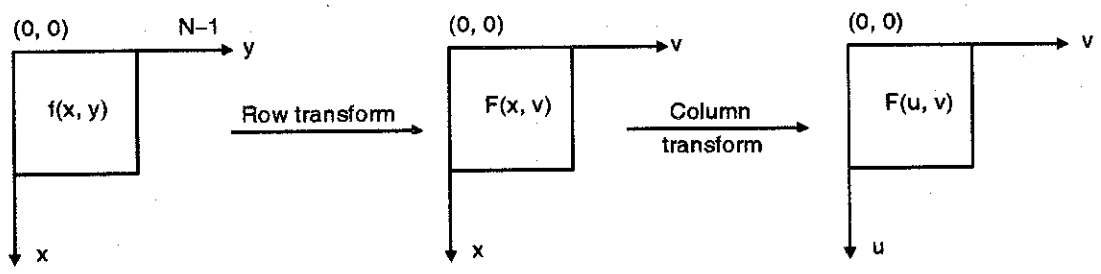


Fig. 6.4.2

F(u, v) can be obtained by applying 1-D DFT along the rows and then along the columns. One could also take the 1-D DFT first along the columns and then along the rows.

Let us take an example

Ex. 6.4.4

Find the DFT of the image

$$\begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 2 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 2 \end{bmatrix}$$

Soln. :

We have studied the 1-D DFT matrix. We shall use the DFT along the rows and then along the columns.

Length of each row is N = 4 ∴ we need a 4×4 DFT matrix.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \\ 0 \\ -2 \end{bmatrix} \rightarrow \text{DFT of } 1^{st} \text{ row}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 8 \\ -2 \\ 0 \\ -2 \end{bmatrix} \rightarrow \text{DFT of } 2^{nd} \text{ row}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 12 \\ -2 \\ 0 \\ -2 \end{bmatrix} \rightarrow \text{DFT of } 3^{rd} \text{ row}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 8 \\ -2 \\ 0 \\ -2 \end{bmatrix} \rightarrow \text{DFT of } 4^{th} \text{ row}$$

Hence we have an intermediate stage

$$\begin{bmatrix} 4 & -2 & 0 & -2 \\ 8 & -2 & 0 & -2 \\ 12 & -2 & 0 & -2 \\ 8 & -2 & 0 & 2 \end{bmatrix}$$

Now using the 1-D DFT along the columns of this intermediate image we get

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 4 \\ 8 \\ 12 \\ 8 \end{bmatrix} = \begin{bmatrix} 32 \\ -8 \\ 0 \\ -8 \end{bmatrix} \rightarrow \text{DFT of } 1^{st} \text{ column}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} -2 \\ -2 \\ -2 \\ -2 \end{bmatrix} = \begin{bmatrix} -8 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \text{DFT of } 2^{nd} \text{ column}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \text{DFT of } 3^{rd} \text{ column}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -i & 1 & -i \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} -2 \\ -2 \\ -2 \\ -2 \end{bmatrix} = \begin{bmatrix} -8 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \text{DFT of } 4^{th} \text{ column}$$

∴ The final DFT of the entire image is

$$\begin{bmatrix} 32 & -8 & 0 & -8 \\ -8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -8 & 0 & 0 & 0 \end{bmatrix}$$

Though MATLAB has an inbuilt command for calculating the 2-D DFT, we shall write our own program. MATLAB program for calculating 2-D DFT of the sum solved is given below. In the program we check the size of the image and create the DFT matrix accordingly.

```
%% 2-D Fourier transform using 1-D Fourier transform %%
clear all
clc
a = [0 1 2 1; 1 2 3 2; 2 3 4 3; 1 2 3 2];
s = size(a);
aa = fft2(a);%% Solving it using the inbuilt function of fft for..
                %%.. comparing the results%%
%% To find the twiddle factor %%
    for k = 0:1:s(1)-1
    for n = 0:1:s(2)-1
    p=exp(-i*2*pi*k*n/s(2));
    x(k+1,n+1) = p;
    end
end
%% To find the 1-D Fourier Transform along the rows
```

```
for n = 1:1:s(2)
    d = x*a(n,:).';
    z(:,n) = d;
end
%% To find the 1D Fourier Transform again along the columns
%% (Transpose of columns)
for n1 = 1:1:s(2)
    d1 = x*z(n1,:).';
    z1(:,n1) = d1;
end
z1
%% 2-D Fourier transform using the 1-D transform
% (On actual square image)
clear all
clc
aa = imread('test.tif');
a = double( aa );
a1 = abs(fft2(a)); %% Using the inbuilt FFT function for
                   comparison %%
s = size(a);
%% To find the Twiddle factor
for k = 0:1:s(1)-1      % Image is square %
    for n = 0:1:s(2)-1
        p = exp(-i*2*pi*k*n/s(2));
        x(k+1,n+1) = p;   % x is the DFT matrix %%
    end
end
%% To find the 1-D Fourier transform again along the rows
                                                    %%
for n = 1:1:s(1)
    d = x*a(n,:).';
    z(:,n) = d;  %% Arranged in rows %%
end
    %% To find the 1-D Fourier transform again along the
columns %%
for n1 = 1:1:s(2)
    d1 = x*z(n1,:).';
    z1(:,n1) = d1;
end
%% Plotting
z = abs(z1);
figure(1)
colormap(gray)
imagesc(fftshift(log(1+z)))  % will be explained later %
figure(2)
colormap(gray)
imagesc(fftshift(log(1+a1)))
```

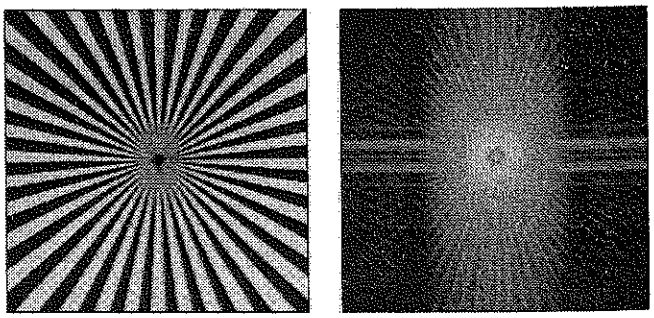


Fig. 6.4.3 : 2D Fourier transform of real image

(2) Translation Property (Shifting Property)

$$f(x, y)e^{+j2\pi \frac{(u_0x + v_0y)}{N}} \xrightarrow{FT} F(u - u_0, v - v_0) \qquad ...(6.4.8)$$

To put it in words, if f(x, y) is multiplied by an exponential, the original Fourier transform F (u, v) gets shifted in frequency by $F(u - u_0, v - v_0)$. Let us prove this property.

We know,

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

Multiplying f(x, y) by $e^{j2\pi\frac{(u_0x + v_0y)}{N}}$

$$F\left\{ f(x, y)\, e^{j2\pi\frac{(u_0x + v_0y)}{N}} \right\} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y) \times e^{-j2\pi\left(\frac{ux + vy}{N}\right)} e^{j2\pi\left(\frac{u_0x + v_0y}{N}\right)}$$

$$= \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y)\, e^{\left(\frac{-j2\pi ux}{N} + \frac{j2\pi u_0x}{N}\right)} e^{\left(\frac{-j2\pi vy}{N} + \frac{j2\pi v_0y}{N}\right)}$$

$$= \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y)\, e^{\frac{-j2\pi x(u - u_0)}{N}} e^{\frac{-j2\pi y(v - v_0)}{N}}$$

$$= \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y)\, e^{\frac{-j2\pi[(u - u_0)x + (v - v_0)y]}{N}}$$

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

Hence,

$$\text{if } F\{f(x, y)\} \longrightarrow F(u, v)$$

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

We make considerable use of this property while plotting the Fourier spectrum. Actually we always use this property. We shall soon see how.

Let us see what happens to the exponential when $u_0 = v_0 = \frac{N}{2}$

$$e^{j2\pi\frac{\frac{N}{2}x + \frac{N}{2}y}{N}} = e^{j\pi(x + y)} = (-1)^{x+y}$$

$$\therefore f(x, y) \cdot (-1)^{x+y} \longrightarrow F\left(u - \frac{N}{2}, v - \frac{N}{2}\right) \quad ...(6.4.9)$$

This will centre the transform. F(0,0) will be placed at F(N/2, N/2)

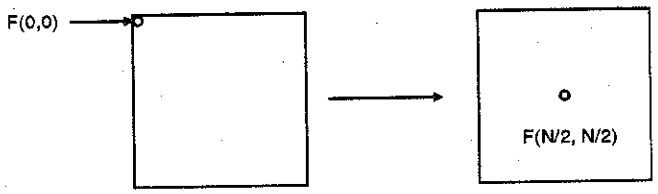


Fig. 6.4.4

(3) Periodicity and Conjugate Symmetry Property

The discrete Fourier transform is periodic i.e., a N point DFT is periodic such that

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

$$\therefore |F(u)| = |F(u + N)|$$

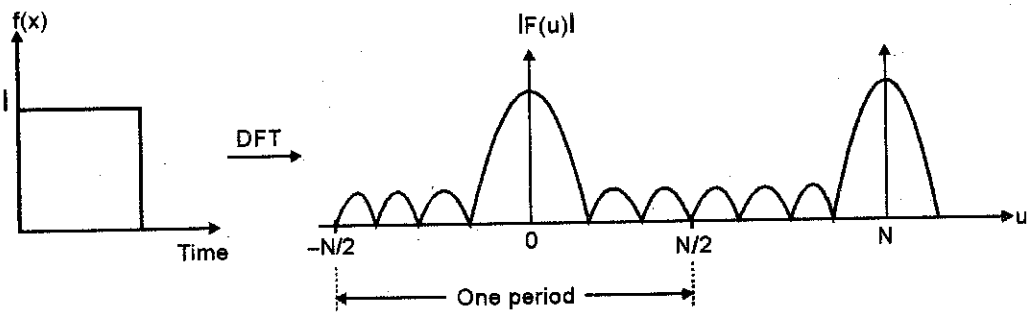


Fig. 6.4.5

The Fourier transform is also symmetric i.e.

$$F(u) = F^*(-u)$$

$$|F(u)| = |F(-u)|$$

Hence the Fourier spectrum is symmetric about 0 i.e., $-\frac{N}{2}$ to 0 is a reflection of 0 to $\frac{N}{2}$. To display one full period, all that is necessary is to move the origin of the transform to the point $u = \frac{N}{2}$.

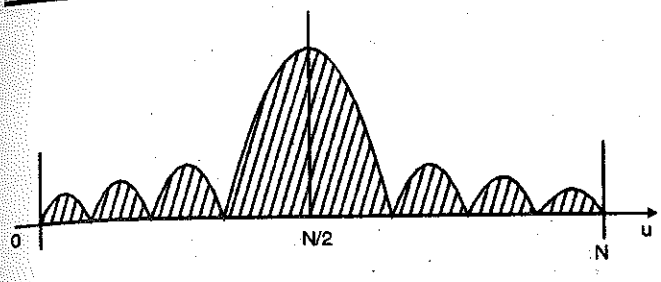


Fig. 6.4.6

This is done by multiplying f(x) by $(-1)^x$ before taking the transform. This was seen in the earlier property. Since this example is of a 1-D signal, we take only $(-1)^x$.

For a 2-D case, symmetric means

$$F(u, v) = F^*(-u, -v)$$

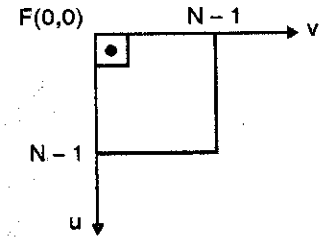
$$\therefore |F(u,v)| = |F(-u,-v)|$$

Hence to summarize, we list down the steps for plotting the 2-D DFT

(1) Multiplying f(x, y) with $(-1)^{x+y}$ so that the transform is centred.

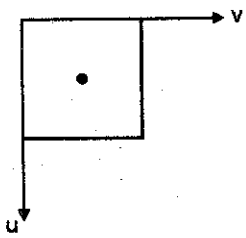
(2) Take the discrete Fourier transform.

Plotting the 2-D DFT is shown in Fig. 6.4.7.



(**Note :** F(0, 0) is the d.c. which is the maximum value. Value at the center will be very small because the higher frequencies lie there).

After using the shifting property, we get



Now the d.c. value will lie at the center i.e. F(N/2, N/2)and high frequencies will lie at exterior end.

Fig. 6.4.7

There is one more issue that needs to be addressed and that is the dynamic range.

When we calculate the DFT, the value of the d.c. component is very large compared to the higher frequencies i.e. the dynamic range of a Fourier spectrum can be as large as $[0, 2.5 \times 10^6]$ where 0 represents dark and 2.5×10^6 represents the brightest pattern.

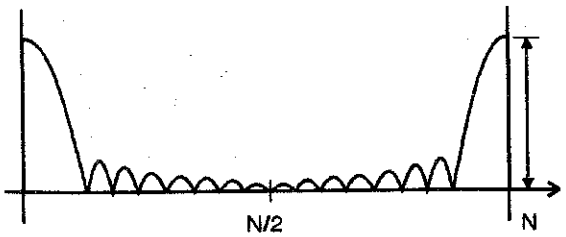


Fig. 6.4.8

No display device has such a large dynamic range. Hence when we plot the spectrum, we see only the maximum value i.e., only a small dot can be seen at the centre.

We now use dynamic range compression (Remember we did it in point processing ?) to rectify this problem. i.e., instead of plotting just F(u, v), we plot the log of F(u, v)

$$G(u, v) = c \cdot \log (1 + F(u, v))$$

The spectrum obtained after dynamic range compression would look like the one shown below

Hence the steps for plotting 2D-DFT are

(1) Multiplying f(x, y) by $(-1)^{x+y}$

(2) Take the 2-D DFT (it will be centred)

(3) Take log of the result obtained in step 2

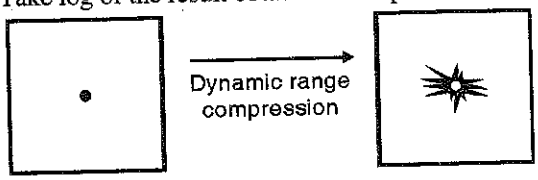
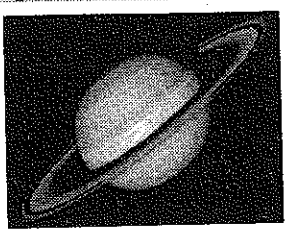


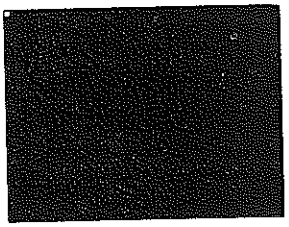
Fig. 6.4.9

Always remember, the spectrums that you see in Digital Image Processing books are obtained using the 3 steps just discussed. MATLAB has an inbuilt command known as "fftshift" to centre the transform, but we shall write our own program.

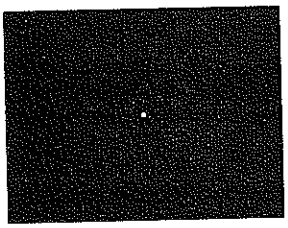
```
%% Centering the transform and using dynamic range
                    compression %%

clear all

clc

aa-imread('saturn.tif');

a = double( aa );

b = abs( fft2( a ) );   % Instead of using this, we could use the

[row col] = size (a);

                    % algorithm that we developed for 2D-FFT

for x = 1:1:row

for y = 1:1:col

   c(x,y)=a(x,y)*( ( - 1) ^ (x + y) ); ;

                              %% centring the transform %%

    end

end

d = abs(fft2(c));          % We could use our DFT program %

max_d = max(max( d) );

min_d = min(min( d ) );

dynamic_range = max_d-min_d

                          %% Displays the dynamic range %%

d_log = log( 1 + d ) ;  %% Dynamic range compression %%

% Plotting

figure(1) , colormap(gray) , imagesc(b)

figure(2) , colormap(gray) , imagesc(d)

figure(3) , colormap(gray) , imagesc(d_log)

figure(4) , colormap(gray) , imagesc(aa)
```



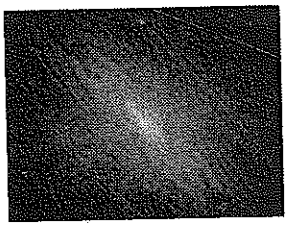
(a) Original



(b) Performing DFT



(c) After centring the transform



(d) Centring and taking the LOG

Fig. 6.4.10

There are other properties of the 2D-DFT that we shall discuss now. Though they are not used as often, it will be an added advantage to understand them.

(4) **Rotation property :** If we introduce polar coordinates

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

$$u = w\cos\phi \;;\; v = w\sin\phi$$

Then f(x, y) and F(u, v) become f(r, θ) and F(w, ϕ).

If we substitute this in the continuous or discrete Fourier transform pair, we get

$$f(r, \theta + \theta_0) \Leftrightarrow F(w, \phi + \theta_0) \qquad ...(6.4.10)$$

It simply means that if we rotate f(x, y) by an angle θ_0, the Fourier spectrum F(u, v) also rotates by the same angle.

(5) **Distributivity and scaling property**

According to this property

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

and

$$F\{f_1(x, y) \cdot f_2(x, y)\} \neq F\{f_1(x, y)\} \cdot F\{f_2(x, y)\} \quad ...(6.4.11)$$

To put it in words, the Fourier transform and its inverse are distributive over addition but not over multiplication. We shall encounter this property when we work on a special type of filter known as an homomorphic filter.

For two scalars a and b

$$a\, f(x, y) \Leftrightarrow a\, F(u, v)$$

and $$f(ax, by) \Leftrightarrow \frac{1}{|ab|} F(u/a, v/b)$$

(6) Average value property : This is given by,

$$\bar{f}(x, y) = \frac{1}{N^2} \sum_{y=0}^{N-1} f(x, y) \quad ...(6.4.12)$$

The standard 2-D DFT formula is

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

When u = 0, v = 0

$$F(u, v) = F(0, 0) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \quad ...(6.4.13)$$

Comparing Equations (6.4.12) and (6.4.13) we get

$$\bar{f}(x, y) = \frac{1}{N^2} F(0, 0) \quad ...(6.4.14)$$

(7) Laplacian property (Second derivative)

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

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

From definition of 2-D DFT,

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

We will encounter the Laplacian function in the chapter on Segmentation. The Laplacian function is used to detect edges in an image. It is a fairly simple task to prove the Laplacian result. We shall do so for the discrete case.

We know

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

.....(IDFT) {Taking N = M}

For simplicity we ignore the term $\frac{1}{N \cdot M}$

$$= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v)\, e^{\frac{+j2\pi ux}{N} + \frac{j2\pi vy}{N}}$$

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

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

$$= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} j2\pi u\, F(u, v)\, e^{\frac{+j2\pi ux}{N} + \frac{j2\pi vy}{N}}$$

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

$$\frac{\partial}{\partial x}\{f(x,y)\} = F^{-1}\{j2\pi u\, F(u, v)\}$$

$$\therefore F\left\{\frac{\partial}{\partial x} f(x, y)\right\} = j2\pi u\, F(u, v)$$

$$\therefore F\left\{\frac{\partial^2}{\partial x^2} f(x, y)\right\} = (j2\pi u)^2\, F(u, v)$$

Similarly

$$\therefore F\left\{\frac{\partial}{\partial y} f(x, y)\right\} = j2\pi v\, F(u, v)$$

and

$$\therefore F\left\{\frac{\partial^2}{\partial y^2} f(x, y)\right\} = (j2\pi v)^2\, F(u, v)$$

We know that Laplacian of a 2-D function f(x, y) is

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

$$\therefore F\{\nabla^2 f(x, y)\} = (j2\pi u)^2 F(u, v) + (j2\pi v)^2 F(u, v)$$

$$= -(2\pi)^2 (u^2 + v^2) F(u, v)$$

$$\therefore \quad F\{\nabla^2 f(x, y)\} = -(2\pi)^2 (u^2 + v^2) F(u, v) \qquad \ldots(6.4.15)$$

As stated earlier, we shall use the Laplacian operator in the chapter on segmentation to detect the edges.

(8) Convolution property

This is the most important property and is the basis to the frequency domain enhancement technique. This property forms the basic link between the spatial domain and the frequency domain. Let us see where we are.....

In previous chapters we discussed image enhancement in the spatial domain. In these chapters, we discussed various masks (low pass, high pass.....) and also learnt that filtering was achieved by moving these masks over the entire image. We also learnt that this moving of masks was nothing but the convolution operation. If f(x, y) was the image and h(x, y) was a 3×3 mask (or 5×5, 7×7 ....) then the modified output was $g(x, y) = f(x, y) * h(x, y)$

Now, in this chapter we are in a new domain - the Frequency domain (Fourier domain). If we want to do filtering in this domain, we need to know what is the equivalent of convolution in this new domain. Hence the convolution property is so important as it provides a link between the spatial and the frequency domain. This can be seen by studying a simple proof.

Proof : We have already studied the convolution formula

$$y[n] = \sum_k x[k]\, h[n-k]$$

Taking the Fourier transform (we could also use the DFT formula)

$$Y(\omega) = \sum y(n)\, e^{-j\omega n}$$

$$Y(\omega) = \sum_y \left(\sum x[k]\, h[n-k] \right) e^{-j\omega n}$$

Put $p = n - k \therefore n = p + k$

$$Y(\omega) = \sum\sum x[k]\, h[p]\, e^{-j\omega(p+k)}$$

$$= \sum\sum x[k]\, h[p]\, e^{-j\omega p}\, e^{-j\omega k}$$

$$= \sum x[k] e^{-j\omega k} \sum h[p]\, e^{-j\omega p} \qquad \ldots(6.4.16)$$

$$Y(\omega) = X(\omega)\, H(\omega)$$

Let us see what we have got.

Convolution in the time domain is equivalent to multiplication in the frequency domain; If we extend this to the 2-D domain, we get

$g(x, y) = f(x, y) * h(x, y)$ in the spatial domain is equivalent to

$G(u, v) = F(u, v) \times H(u, v)$ in the frequency domain

Hence to find G(u, v) we take the 2-D DFT of the input image, find the transfer function of the filter, H(u, v), and simply multiply the two. We finally take the inverse Fourier transform of G(u, v) to get the modified image i.e., g(x, y).

6.4.4 Image Enhancement in Frequency Domain

In this, we work with the Fourier transform of the image.

Let us draw a block diagram to simplify what we stated, in the earlier section.

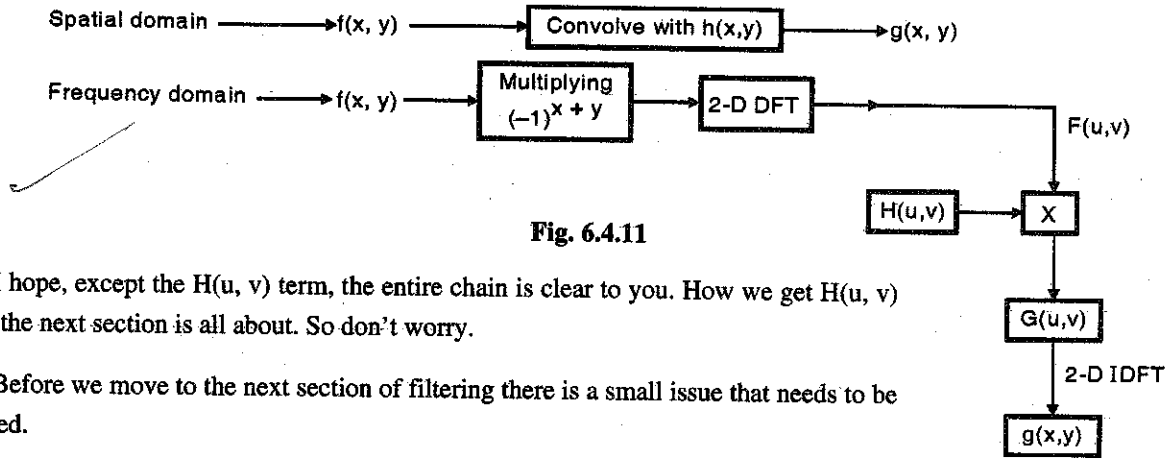


Fig. 6.4.11

I hope, except the H(u, v) term, the entire chain is clear to you. How we get H(u, v) is what the next section is all about. So don't worry.

Before we move to the next section of filtering there is a small issue that needs to be addressed.

As a student of image processing, you must be able to understand the frequency plot. To perform filtering (removal of certain frequencies) we need to know where the frequencies reside in the Fourier plot.

We have seen that for a 1-D signal

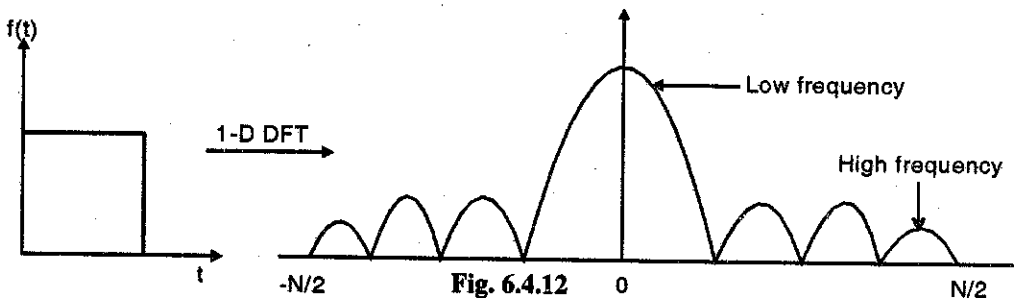


Fig. 6.4.12

Here, the 0 represents the d.c. term. As we move to the right, the frequency goes on increasing, the maximum being N/2. On using the translation property, we have

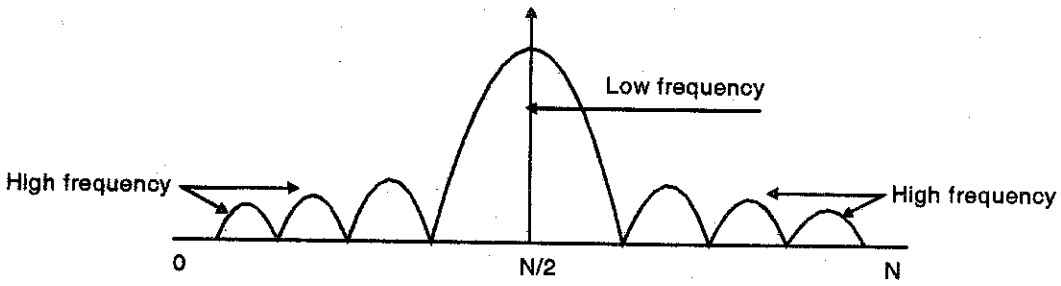


Fig. 6.4.13

Extending this to the 2-D domain, we get the figure shown. Hence we conclude that in the Fourier spectrum, the centre is where the low frequencies reside and as we go away from the centre, we encounter the high frequencies. This is true only after using the centring formula.

Remember this - centre → low frequencies,

peripheries → high frequencies

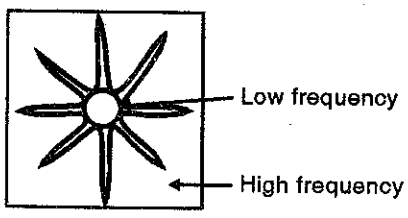


Fig. 6.4.14

Now if we want to perform low pass filtering, we need to eliminate the high frequencies. We shall remove the peripheries and retain only the central values as that is where the low frequencies exist.

6.5 Low Pass Frequency Domain Filters

By now we know, what we mean by low frequencies as well as high frequencies. We are also aware as to where they lie in the Fourier spectrum.

The basic formula for any kind of filtering is based on the convolution integral.

$$\text{i.e., } f(x, y) * h(x, y) \Leftrightarrow FT \Rightarrow F(u, v) \times H(u, v)$$

(* implies convolution)

Where
- $f(x, y) \rightarrow$ Original image
- $h(x, y) \rightarrow$ Filtering mask
- $F(u, v) \rightarrow$ Fourier transform of the original image
- $H(u, v) \rightarrow$ Fourier transform of the filtering mask

Hence for filtering we use the formula

$$G(u, v) = F(u, v) \times H(u, v)$$

Our objective is to choose H(u, v) so that we get a low pass filter.

We shall study three basic types of low pass filters

(a) Ideal **(b)** Butterworth **(c)** Gaussian

In all these 3 cases, its only H(u, v) that changes.

6.5.1 Ideal Low Pass Filter (ILPF)

These are the simplest of the three filters. This filter cuts off all high frequency components of the Fourier transform that are at a distance greater than a specified distance D_0.

$$H(u, v) = 1 \text{ ; if } D(u, v) \leq D_0$$

$$H(u, v) = 0 \text{ ; if } D(u, v) > D_0 \quad ...(6.5.1)$$

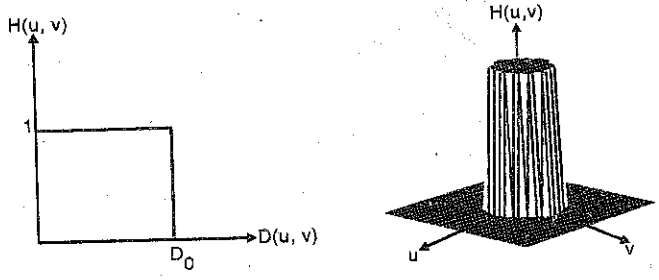


Fig. 6.5.1(a)

Here D_0 is the specified non-negative distance. What is D(u, v)? Remember, the Fourier transform that we plot is centred using the formula $f(x, y)(-1)^{x+y}$.

D(u, v) is the distance from the point (u, v) to the origin of the frequency rectangle for an M × N image.

$$D(u, v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2} \quad ...(6.5.2)$$

$\therefore$ For an image, when $u = \frac{M}{2}$, $v = \frac{N}{2}$

$$D(u, v) = 0$$

D(u, v) gives us concentric rings with each ring having a fixed value. (Study the program given on the next page). This formula centres our H(u, v).

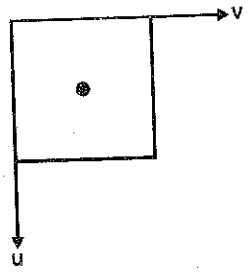
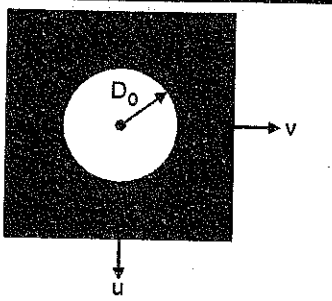


Fig. 6.5.1(b)

This formula is very important when we implement the LPF practically. Hence if we say $D_0 = 5$, then H(u, v) will be a bright circle of radius 5. This can be easily understood from the above formula.

Here the shaded portion has a value 0 and the white portion has a value 1. Depending on the value of D_0, the white circle becomes larger or smaller. Do is called the cut-off frequency. When we multiply F(u, v) and H(u, v) only that part of the spectrum is retained that coincides with the white part of H(u, v), everything else becomes zero as the shaded part of H(u, v) is equal to zero.



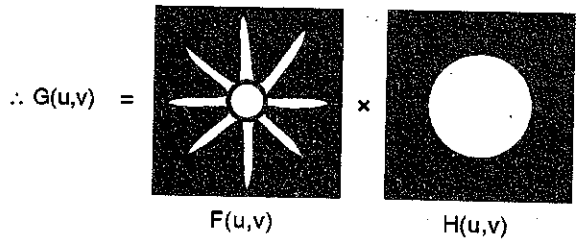


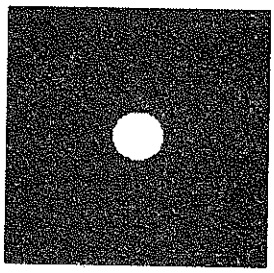
Fig. 6.5.1(c)

As stated earlier, it is called an ideal filter because it removes everything beyond D_0. Can ideal filters be implemented practically ? Though ideal filters cannot be realized using physical components like capacitors and resistors they can definitely be implemented on a computer. Hence ideal filters can be used to filter images.

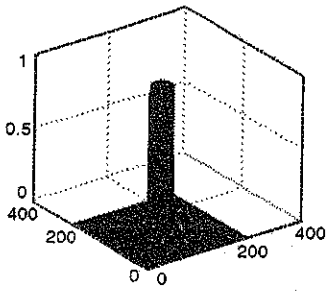
MATLAB program for implementing ideal LPF on square images

```
%% Ideal low pass filter %%
clear all
clc
a = imread('cameraman . tif');
a = double( a ) ;
c = size( a ) ;
N = c( 1 )
D0 = input('Enter the cut-off frequency' ) ;
for u = 1:1:c(1)
  for v = 1:1:c(2)
     D = ((u-(N/2)) ^ 2+(v-(N/2)) ^ 2) ^ 0.5;
          if D < D0;
    H(u, v) = 1;
          else
    H(u, v) = 0;
      end
   end
end
vv = fft2( a ) ;                % We could use our code %
vc = fftshift( vv ) ;           % We could use our code %
x = vc.*H;
X = abs( ifft2(x) ) ;
%% Plotting %%
figure( 1 ) , imshow(uint8(a) )
figure( 2 ) , mesh(H)
figure(3)   , imshow(uint8(X) )
figure(4)   , imagesc(H),colormap( gray )
```

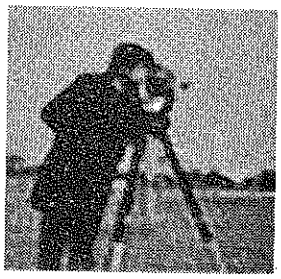
(a) Original image



(b) H(u,v) 2-D Ideal low pass filter



(c) Ideal low pass filter Frequency response



(d) Ideal low pass filtered image cut-off = 25

Fig. 6.5.2

What should be the ideal value of D_0 ? We know that H(u, v) is a circular structure based on the value of D_0. One way to establish a set of standard cut-off frequency loci is to compute circles that enclose specified amounts of total image power P_{total}.

We know, $P(u, v) = F(u,v)^2 = [R^2(u, v) + I^2(u, v)]$

$$P_{total} = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} P(u, v)$$

A circular radius of value r with the origin at the centre of the frequency rectangle encloses a percent of power where,

$$\alpha = 100 \times \left[\sum_{u=0}^{M-1} \sum_{y=0}^{N-1} P(u, v) / P_{total} \right]$$

We shall explain this with a diagram.

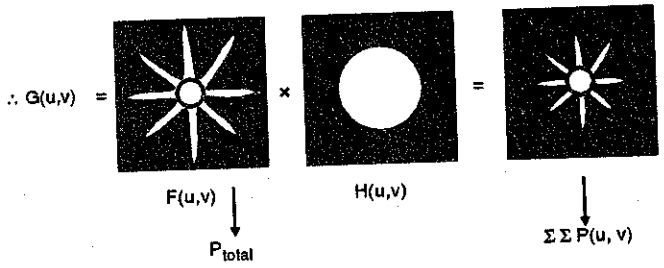


Fig. 6.5.3

We calculate P_{total} from the original Fourier spectrum and then calculate the power of the filtered spectrum and then use the above formula to calculate α. It has been observed that most of the power lies within a radius of 5-10. Though ideal filters can be implemented, they are not normally used. The reason being there is severe ringing and blurring of the image. Blurring is something that can be expected but ringing [see Fig. 6.5.2(d)] creeps in and is very disconcerting to the viewer.

We shall proceed to explain the reason behind the ringing effects. The ringing effects can be explained by referring to the great convolution theorem.

We know, $G(u, v) = F(u, v) \times H(u, v)$

↓ 2-D IDFT

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

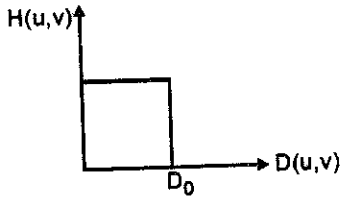


Fig. 6.5.4

Now H(u, v) has a shape (radial cross section) as shown in Fig. 6.5.4.

What happens when we take the inverse Fourier transform ?

Remember

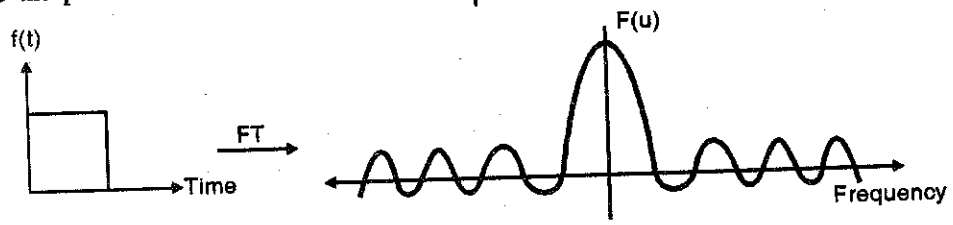


Fig. 6.5.5

In a similar manner

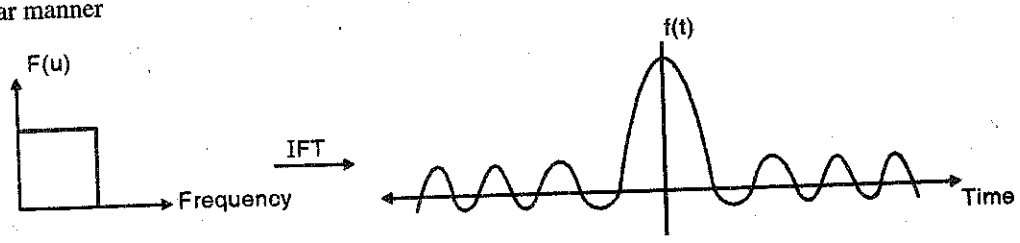


Fig. 6.5.6

This is the dual property of the Fourier transform.

Therefore for H(u, v)

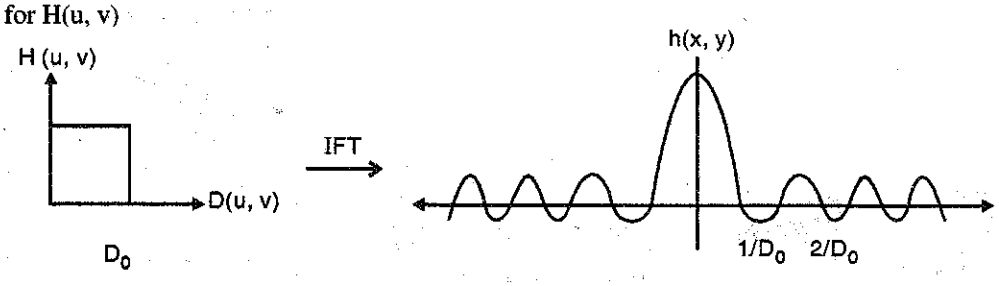


Fig. 6.5.7

Hence when h(x, y) is displayed as an image, we see that it has two distinctive characteristics : a dominant component at the origin and concentric components about the centre. When we convolve this h(x, y) with the image, we get the ringing effects since convolution is merely copying h(x, y) at every location. To be precise, the centre component is responsible for blurring while the concentric circles are responsible for the ringing effects.

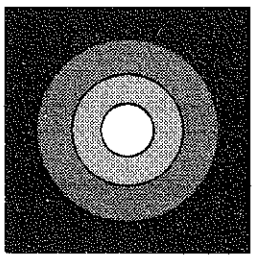


Fig. 6.5.8

Hence as long as there are sharp cut-offs, their spatial representations will be sinc functions and hence ringing effects will always be there. If D0 increases, the sinc function becomes narrow and hence ringing effects look more pronounced in the image.

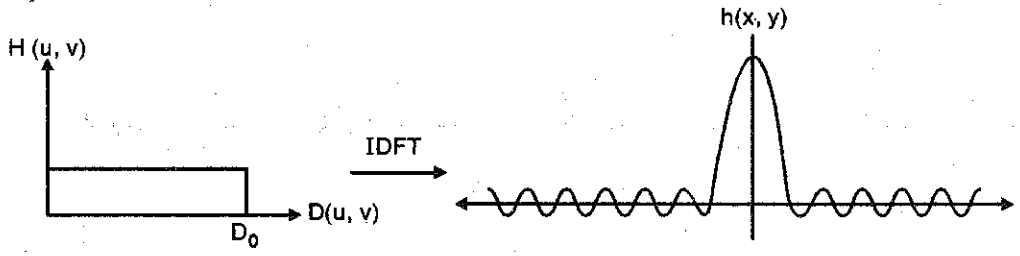


Fig. 6.5.9

To summarize, in order to eliminate the ringing effects, we have to eliminate the sharp cut-offs in the frequency domain. Hence ideal filters are rarely used for low pass filtering.

6.5.2 Butterworth Low Pass Filters (BLPF)

As stated in the earlier section, the ringing effects are due to the sharp cut-offs in the ideal filter and to get rid of the ringing effects, we need to eliminate these sharp cut-offs. This is exactly what we do in Butterworth low pass filters.

The transfer function of the Butterworth low pass filter of order n and cut-off frequency at a distance D0 from the origin is defined as

$$H(u, v) = \frac{1}{1 + [D(u, v)/ D_0]^{2n}} \qquad \ldots(6.5.3)$$

Where D(u, v) was already defined in the earlier section.

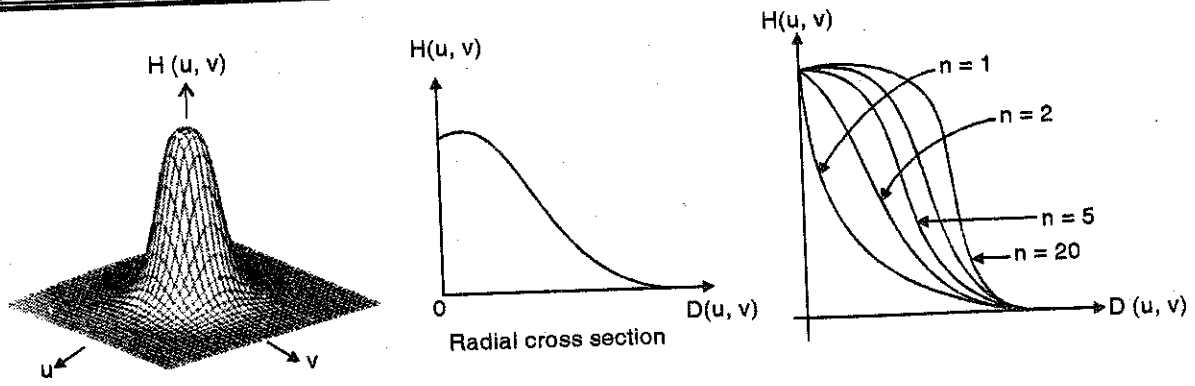


Fig. 6.5.10

Unlike the ILPF, the BLPF does not have sharp discontinuities and hence there are no ringing effects present when a BLPF is used. But as the order of the filter goes on increasing, a small amount of ringing effect does creep in because the Butterworth low pass filter tends to be an ideal filter.

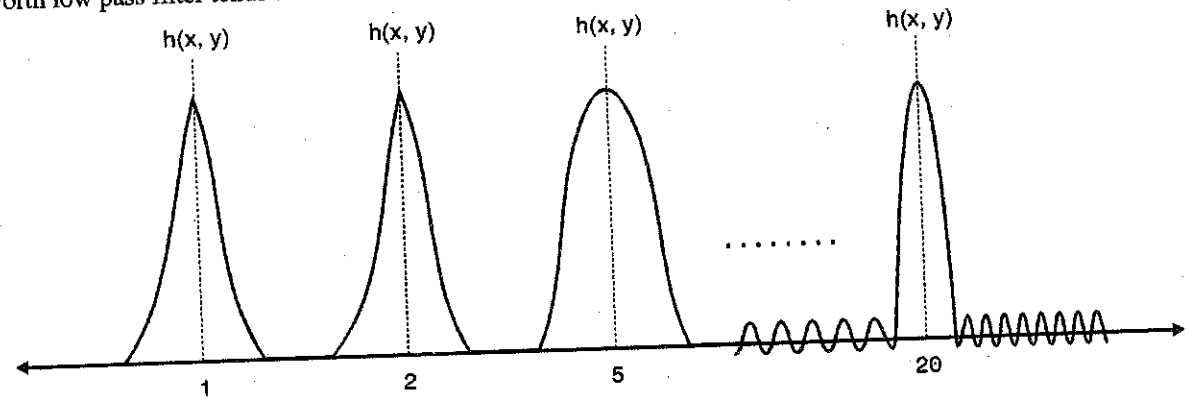


Fig. 6.5.11

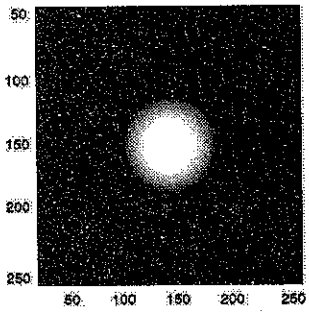
Hence to get good results, use an order ≤ 5.

MATLAB program for Butterworth low pass filter on a square image

```
%% Low pass Butterworth filter %%
clear all
clc
a = imread('cameraman.tif');
a = double(a);
c = size(a);
n = c(1);
n = input('Enter the order of the filter');
D0 = input('Enter the cut-off frequency');
for u = 1:1:c(1)
    for v = 1:1:c(2)
        D = ((u-(N/2)) ^ 2+(v-(N/2)) ^ 2) ^ 0.5 ;
        H(u,v) = 1/(1+((D/D0) ^ (2*n)));
    end
end
vv = fft2(a);% we could use our code %
vc = fftshift(vv);
x = vc.*H;
x = abs(ifft2(x));
%% Plotting
figure (1) , imshow(uint8(a))
figure (2) , mesh(H)
```

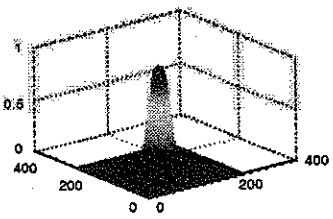
figure (3) , imshow(uint8(X))

figure (4) , imagesc(H),colormap(gray)



(a) **Original image** (b) **2-D Butterworth filter**

Fig. 6.5.12 Continued...



(c) Butterworth frequency response

(d) Butterworth low-pass filtered image order = 2, cut-off = 25

Fig. 6.5.12

6.5.3 Gaussian Low Pass Filter (GLPF)

Gaussian LPF is given by,

$$H(u, v) = e^{-D^2(u, v)/2\sigma^2} \quad ...(6.5.4)$$

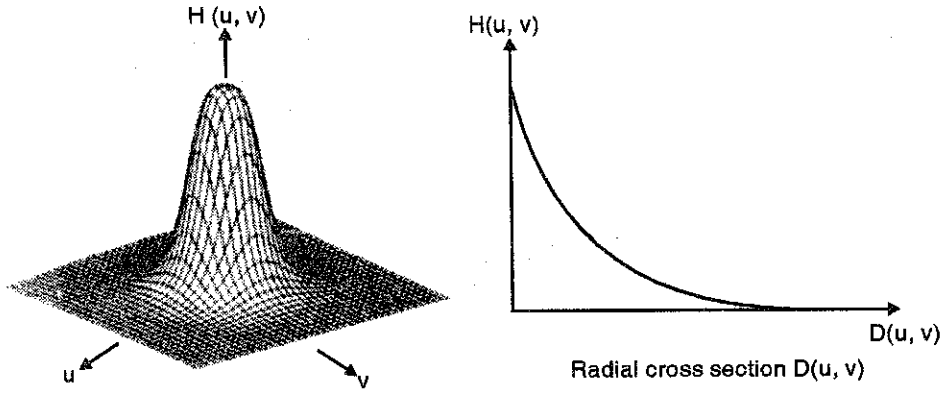


Fig. 6.5.13

Here σ is the standard deviation and is a measure of spread of the Gaussian curve.

If we put $\sigma = D_0$ we get,

$$H(u, v) = e^{-D^2(u, v)/2D_0^2}$$

The response of the Gaussian LPF is similar to that of BLPF but there are no ringing effects at all !!!!

The reason for studying the Gaussian low pass filter is that it has a very important property in the Fourier domain which can be used as an analysis tool. We shall explain that when we try to study the relationship between the spatial domain masks and their frequency response.

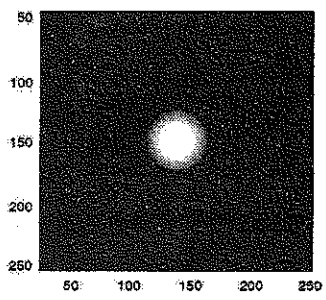
MATLAB program for Gaussian LPF on square images

```
%% Low pass Gaussian filter %%
clear all
clo
a = mread('cameraman.tif') ;
a = double(a) ;
c = size(a) ;
N = c(1)
```

```
D0 = input ('Enter the cut-off frequency / standard deviation');
for u = 1:1:c(1)
    for v = 1:1:c(2)
    Dx = ( (u – (N/2)) ^ 2 + (v – (N/2) ) ^ 2) ^ 0.5 ;
    D = Dx*Dx
            %% In the formula we need to take the square %%
    H(u,v) = exp(– D / (2*D0*D0) ) ; %% D0 is the standard deviation,.
                                        %%..sigma%%
    end
end
vv = fft2(a);
vc = fftshift(vv);
x = vc.*H;
X = abs(ifft2(x));
%% Plotting
figure (1) , imshow(uint8(a))
figure (2) , mesh(H)
figure (3) , imshow(uint8(X))
figure (4) , imagesc(H),colormap(gray)
```

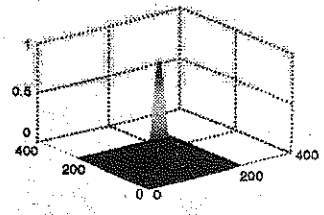
Main advantage of a Gaussian LPF over a Butterworth LPF is that we are assured that there will be no ringing effects no matter what filter order we choose to work with.

(a) Original image

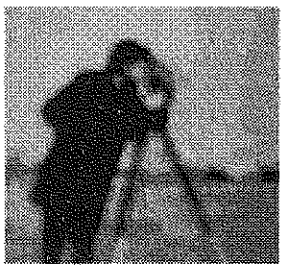


(b) 2-D Gaussian filter

Fig. 6.5.14 Continued...



(c) Gaussian filter response



(d) Gaussian low pass filtered image

Fig. 6.5.14

6.6 High Pass Frequency Domain Filters

Edges and other abrupt changes in the grey levels are associated with high frequency components while the slowly varying changes are the low frequency areas. Since high pass filters perform the reverse operation of the low pass filters, we can write.

$$H_{hp}(u, v) = 1 - H_{lp}(u, v) \quad \ldots(6.6.1)$$

Just as in the low pass filter case, we shall define three basic types of high pass filters

(a) Ideal High Pass

(b) Butterworth High Pass

(c) Gaussian High Pass

6.6.1 Ideal High Pass Filters (IHPF)

A 2-D ideal high pass filter is defined as

$$H(u, v) = 0 ; \text{ if } D(u, v) \le D_0 \quad \ldots(6.6.2)$$

$$= 1 ; \text{ if } D(u, v) > D_0$$

where D_0 is called the cut-off. D(u, v) is the same as defined in the low pass filtering section.

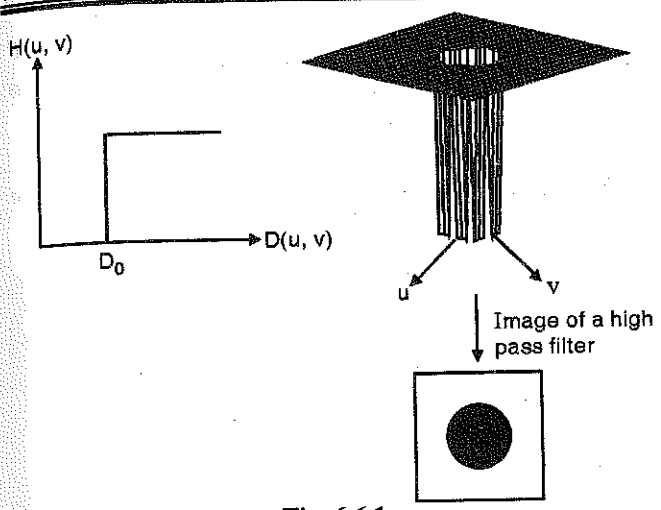


Fig. 6.6.1

In the Fig. 6.6.2, the shaded portion has a value 0 while, the white portion has a value 1. What happens when we multiply F(u, v) with this H(u, v) ?

$$G(u, v) = F(u, v) \times H(u, v)$$

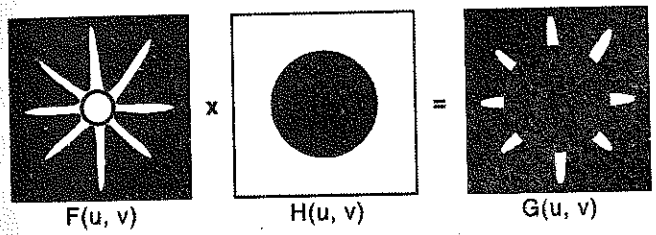


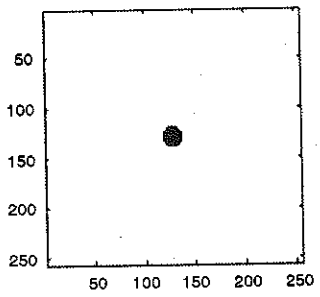
Fig. 6.6.2

As was guessed, the centre portion gets eliminated and only the high frequencies are retained. As in the case of ideal low pass filters, ideal high pass filters are not realizable using physical components like capacitors and resistors but can be implemented on the computer.

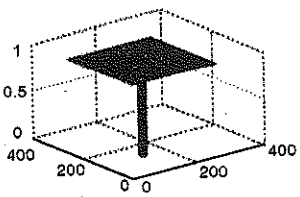
MATLAB code for ideal high pass filter on a square image

```
%% Ideal High pass filter %%
clear all
clc
a = imread('cameraman.tif');
a = double(a);
[row col] = size(a);
set = input( 'Enter the value of the cut-off frequency ') ;
for u = 1:1:row
for v = 1:1:col
D = ( (u – (row/2)) ^ 2 + (v –(col/2)) ^ 2) ^ 0.5
if D<set ;
H(u,v) = 0;
else
H(u,v) = 1;
end
end
end
vv = fft2(a) ;
vc = fftshift(vv) ;
x = (vc.*H) ;
X = abs(ifft2(x) ) ;
% Plotting %
figure (1) , imshow(uint8(a))
figure (2) , mesh(H)
figure (3) , imshow(uint8(X))
figure (4) , imagesc(H),colormap(gray)
```

(a) Original image



(b) 2-D ideal filter



(c) Ideal high pass frequency response

(d) Ideal high passed image

Fig. 6.6.3

Like the ideal low pass filters, ideal high pass filters are not suitable as there are ringing effects which cannot be eliminated. This is clear if we see the images. Hence ideal high pass filters are rarely used for image high pass filtering.

6.6.2 Butterworth High Pass Filters (BHPF)

The general formula for any high pass filter is

$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

$$\therefore H_{HBWF}(u, v) = 1 - H_{LBWF}(u, v)$$ (LBWF → Low pass Butterworth filter)

(HBWF → High pass Butterworth filter)

We know

$$H_{LBWF}(u, v) = \frac{1}{1 + \left[\frac{D(u, v)}{D_0}\right]^{2n}}$$

$$H_{HBWF}(u, v) = 1 - \frac{1}{1 + \left[\frac{D(u, v)}{D_0}\right]^{2n}}$$

Let $\left[\frac{D(u, v)}{D_0}\right]^{2n} = x$

$$\therefore H_{HBWF}(u, v) = 1 - \frac{1}{1 + [x]} = \frac{1 + [x] - 1}{1 + [x]} = \frac{[x]}{1 + [x]}$$

$$= \frac{1}{1 + \frac{1}{[x]}} = \frac{1}{1 + \frac{1}{\left[\frac{D(u, v)}{D_0}\right]^{2n}}}$$

$$= \frac{1}{1 + \left[\frac{D_0}{D(u, v)}\right]^{2n}}$$

Hence, Butterworth high pass filter is

$$\therefore H_{HBWF}(u, v) = \frac{1}{1 + \left[\frac{D_0}{D(u, v)}\right]^{2n}} \quad ...(6.6.3)$$

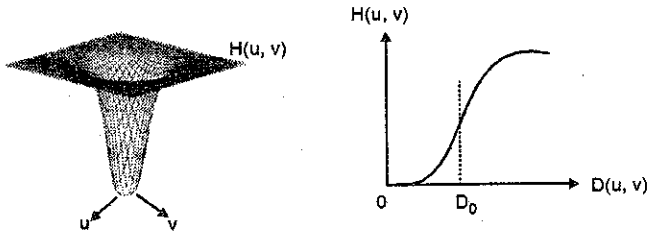


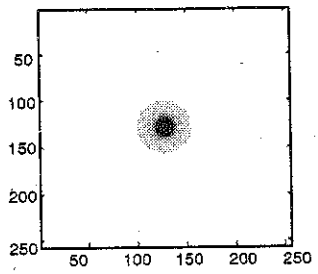
Fig. 6.6.4

The high pass Butterworth filter does not have a sharp cut-off i.e, the transition is smooth and hence there are no ringing effects.

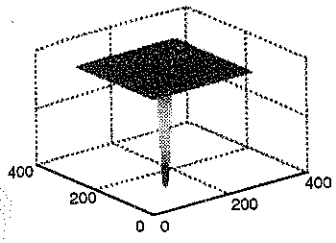
MATLAB code for Butterworth high pass filter on square image

```
%% Butterworth high pass filter %%
clear all
clc
a = imread('cameraman.tif');
a = double(a) ;
c = size(a) ;
N = c(1) ;
N = input('Enter the order of the filter ') ;
D0 = input('Enter the cut-off frequency ') ;
for u = 1:1: c(1)
    for v = 1:1:c(2)
    D = ((u-(N/2)) ^2+(v-(N/2)) ^ 2) ^ 0.5 ;
    H(u,v) = 1/((1+(D0/D) ^ (2*n)) ) ;
    end
end
vv = fft2(a) ;
vc = fftshift(vv) ;
x = vc.*H ;
x = abs(ifft2(x) ) ;
%%Plotting
figure (1) , imshow(uint8(x) )
figure (2) , mesh(H)
figure (3) , imshow(uint8(X) )
figure (4) , imagesc(H), colormap (gray)
```

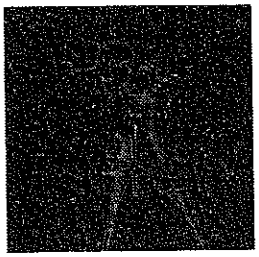
(a) Original image



(b) 2-D Butterworth high pass filter



(c) Butterworth high pass filter response



(d) Butterworth high pass image cut-off 10

Fig. 6.6.5

6.6.3 Gaussian High Pass Filters (GHPF)

We again use the basic formula

$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

$$\therefore H_{GaussianHP}(u, v) = 1 - H_{GaussianLP}(u, v)$$

$$H_{GHPF} = 1 - e^{-D^2(u, v) / 2D_0^2} \quad ...(6.6.4)$$

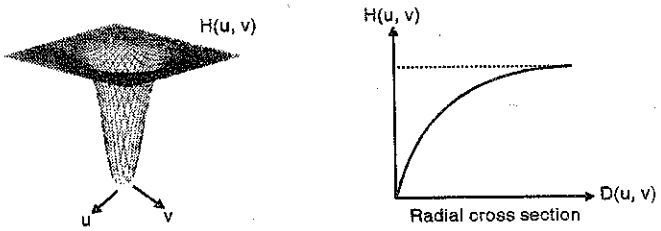


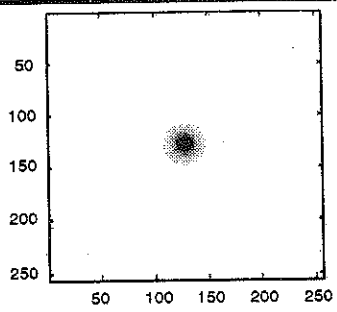
Fig. 6.6.6

The results of Gaussian high pass filter are shown in Fig. 6.6.7. The results are smoother than the ones discussed previously. The results are also cleaner.

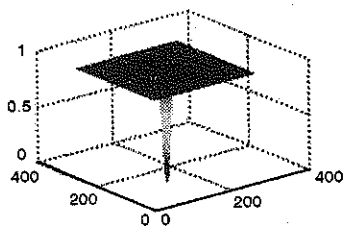
MATLAB code for Gaussian high pass filter on square images

```
%% Gaussian high pass filter %%
clear all
clc
a = imread('cameraman. tif ') ;
a = double(a) ;
c = size(a) ;
N = c(1)
D0 = input('Enter the cut-off frequency/standard deviation ');
for u = 1:1:c(1)
    for v = 1:1:c(2)
    Dx = ((u – (N/2)) ^ 2+ (v – (N/2)) ^ 2) ^ 0.5;
    D = Dx*Dx; %% In the formula we need to take the
                square %%
    H(u,v) = 1-exp(-D/(2*D0*D0));
  % Here D0 is taken as the standard % deviation sigma %%
    end
end
vv = fft2(a);
vc = fftshift(vv)
x = vc.*H;
X = abs(ifft2(x));
%%Plotting%%
figure (1) , imshow (uint8(a) )
figure (2) , mesh(H)
figure (3) , imshow(uint8(X) )
figure (4) , imagesc(H),colormap (gray)
```

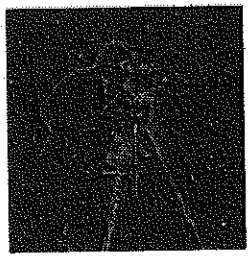
(a) Original image



(b) 2-D Gaussian high pass filter



(c) Gaussian high passed filter response



(d) Gaussian high passed image

Fig. 6.6.7

6.6.4 High Boost Filtering (Unsharp Masking / High Frequency Emphasis)

We have already seen what we mean by high boost filtering in the spatial domain. Since high pass filters completely remove the low frequency components, we get only the edges. There are applications where we need to preserve some of the low frequency components. This can be achieved by high boost filtering. We shall use the following abbreviated forms in the frequency domain.

$F_{HP} \rightarrow$ High pass filtered image

$F_{LP} \rightarrow$ Low pass filtered image

$H_{LP} \rightarrow$ Low pass filter

$H_{HP} \rightarrow$ High pass filter

$H_{HB} \rightarrow$ High boost filter

We have seen that in the spatial domain.

High boost = (A – 1) Original + High pass

i.e. $F_{HB}(x, y) = (A - 1) f(x, y) + F_{HP}(x, y)$

What happens to this in the frequency domain ?

We know, High pass = Original – Low pass

i.e., $F_{HP}(x, y) = f(x, y) - F_{LP}(x, y)$

Taking the Fourier transform

$$F_{HP}(u, v) = F(u, v) - F_{LP}(u, v)$$

$$\text{But } F_{LP}(u, v) = H_{LP}(u, v) \cdot F(u, v)$$

$$\therefore F_{HP}(u, v) = [1 - H_{LP}(u, v)] F(u, v)$$

$$\therefore H_{HP}(u, v) \cdot F(u, v) = [1 - H_{LP}(u, v)] F(u, v)$$

$$\therefore H_{HP}(u, v) = [1 - H_{LP}(u, v)]$$

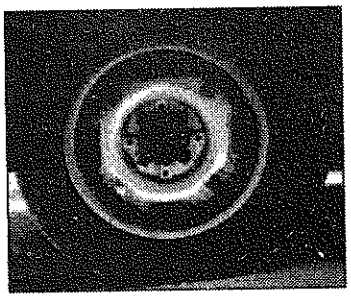
In a similar manner

$$H_{HB}(u, v) = (A - 1) + H_{HP}(u, v); \quad A > 1$$

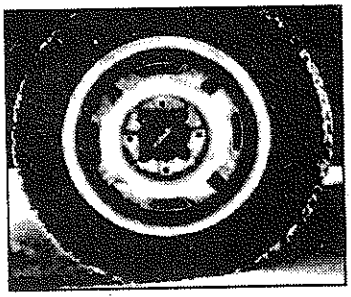
MATLAB code for high boost filtering on square images.

```
%% High boost filtering in the frequency domain %%
clear all
clc
a = imread('tire.tif');
a = double(a);
[row col]= size (a);
for u = 1:1:row
for v = 1:1:col
D = ((u-(row/2))^2+(v-(col/2))^2)^0.5;
  if D<10 ;%% Cut-off is taken as 10. It could be changed..
  H(u,v) = 0 ;          %% .. as per the requirement
  else
  H(u,v) = 1 ;
     end
end
end
A = 1.5 ;
        %% This is the high boost factor. It could be changed..
H1 = (A-1)+ H ;       %% .. as per requirement
```

```
vv = fft2(a) ;
vc = fftshift (vv);  %% Required to centre the transform %%
x = vc.*H ;    %% Multiplication by the high pass filter %%
x1 = vc.*H1;  %% Multiplication by the high boost filter %%
X = abs ( ifft2(x) ) ;
X1 = abs (ifft2(x1) ;
%%Plotting %%
figure ( 1 )
imshow (uint8(a) )
figure (2)
mesh (H)
figure (3)
imshow (uint8(X) )
figure (4)
mesh (H1)
figure (5)
imshow (uint8(X1) )
```



(a) Original image



(b) High boost filter image

Fig. 6.6.8

6.7 Homomorphic Filtering

Homomorphic filter uses a slightly different approach compared to the frequency domain filters studied so far. Before studying the homomorphic filter, we need to study the image formation model.

Image Formation Model

We have seen that an image is a 2-D function. A grey level (monochrome) image is given as f(x, y) where f is the grey level and x and y are the spatial coordinates.

Images that we see are formed due to the fact that there is some amount of light falling on the object and some amount of light reflecting from the object.

Light falling on the object is called illumination and light reflecting from the object is called reflectance.

Since images that we see fall in the electromagnetic spectrum,

$$0 < f(x, y) < \infty$$

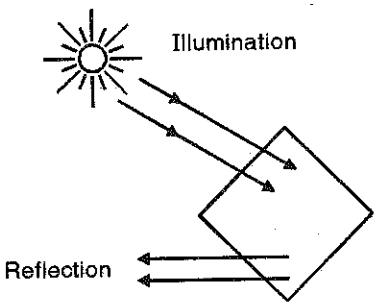


Fig. 6.7.1

f(x, y) is characterized by two components :

(1) The amount of light incident on the scene/object. (illumination)

(2) Amount of light reflected by the object. (reflection)

They are called illumination and reflectance components i(x, y) and r(x, y) respectively.

These two combine as a product to form f(x, y)

$$\therefore f(x, y) = i(x, y)\, r(x, y) \qquad ...(6.7.1)$$

where $0 < i(x, y) < \infty$

and $0 < r(x, y) < 1$; 0 indicates total absorption and 1 indicates total reflectance.

Homomorphic filtering uses this illumination-reflectance model to filter images in the frequency domain. In homomorphic filtering we seperately the illumination and reflectance components and work with them seperately.

$$f(x, y) = i(x, y) \cdot r(x, y)$$

We cannot directly take the Fourier transform since

$$F\{f(x, y)\} \neq F\{i(x, y)\}F\{r(x, y)\}$$

To separate i and the r components, we use the log function.

$$Z(x, y) = \ln f(x, y)$$

$$Z(x, y) = \ln i(x, y) + \ln r(x, y)$$

We now take the Fourier transform to move into the frequency domain

$$F\{Z(x, y)\} = F\{\ln i(x, y)\} + F\{\ln r(x, y)\}$$

$$Z(u, v) = F_i(u, v) + F_r(u, v)$$

where $F_i(u, v) = F\{\ln i(x, y)\}$

$$F_r(u, v) = F\{\ln r(x, y)\}$$

We now use a filter H(u, v)

$$\therefore S(u, v) = Z(u, v) \cdot H(u, v)$$

$$S(u, v) = F_i(u, v) \cdot H(u, v) + F_r(u, v) \cdot H(u, v)$$

The key approach in homomorphic filtering is to seperate the illumination and reflectance components. Between them i(x, y) contributes to the low frequency since illumination is more or less uniform and r(x, y) is the high frequency component since r(x, y) tends to vary abruptly at junctions of dissimilar objects.

A typical homomorphic filter H(u, v) is shown in Fig. 6.7.2.

Generally $Y_L < 1$ and $Y_H > 1$. H(u, v) tends to decrease the contribution made by low frequencies (illumination) and amplify the contribution made by high frequencies (reflectance).

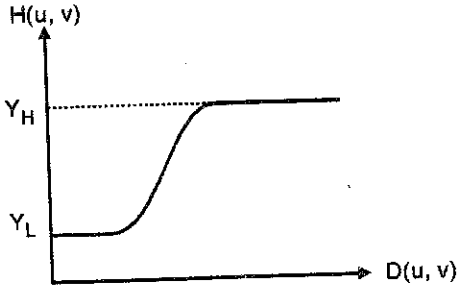


Fig. 6.7.2

How do we get back the modified image ?

$$S(u, v) = H(u, v) F_i(u, v) + H(u, v) F_r(u, v)$$

To come back to the spatial domain, we take the inverse Fourier transform.

$$s(x, y) = F^{-1}\{S(u, v)\}$$

$$= F^{-1}\{H(u, v) F_i(u, v)\} + F^{-1}\{H(u, v) F_r(u, v)\}$$

$$s(x, y) = i'(x, y) + r'(x, y)$$

Finally remember, the first step in homomorphic filtering was to take the log, hence here we need to take the inverse operation i.e., the exponential operation.

$$\therefore g(x, y) = e^{s(x, y)}$$

$$= e^{i'(x, y)} \cdot e^{r'(x, y)}$$

$$= i_0(x, y) \cdot r_0(x, y)$$

where $i_0(x, y) = e^{i'(x, y)}$

$$r_0(x, y) = e^{r'(x, y)}$$

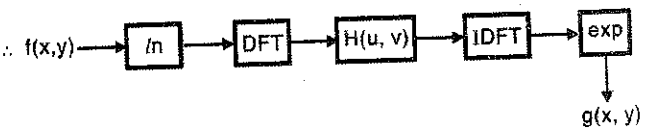


Fig. 6.7.3

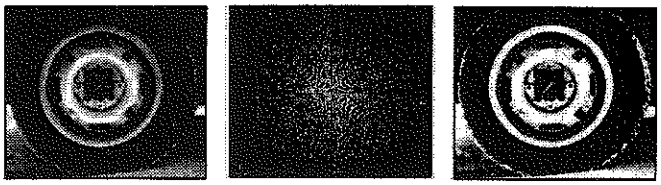
The filter H(u, v) can be implemented by a simple formula,

$$H(u, v) = (Y_H - Y_L)\ [1 - e^{-C(D^2(u, v)/D_0^2)}] + Y_L \quad ...(6.7.2)$$

C is a constant, used to control the sharpness.

MATLAB code for homomorphic filtering on square image

```
%% Program for homomorphic filtering %%
clear all
clc
a = imread('tire.tif') ;
a = double(a) ;
b = a ;
%% Constants required for the filter %%
D0 = 5 ;
Yl = 1 ;
Yh = 2 ;
[row col]= size(a) ;

%% To make sure that the coefficients are not zero %%
for x = 1:1:row
    for y = 1:1:col
    if a(x,y)= =0
     b(x,y) = 1;
    end
end
end
hi = log(b) ;        %% Taking log %%
c = fft2(hi) ;           %% Taking the fft %%
d = fftshift(c) ;        %% Centering the transform %%
for u = 1:1:row
for v = 1:1:col
D = ( (u- (row/2)) ^ 2+(v- (col/2)) ^ 2) ^ 0.5;
H(u,v) = (Yh-Yl)*(1-exp(-1*( (D*D)/(D0*D0) ) ) )+Yl ;
    end
end
x = d.*H;
fin = abs(ifft2 (x) ) ;
Final = exp (fin) ;
Figure (1)
imshow (uint8(a) )
figure (2)
imshow (uint8(x) )
figure (3)
imshow(uint8 (Final) )
```



(a) Original image **(b) Intermediate image in the transform domain** **(c) Homomorphic filtered image**

Fig. 6.7.4

6.8 Importance of Phase

We have discussed in detail about manipulating the magnitude plot of the 2-D DFT.

All the filtering operations were basically operating on the magnitude plot.

We shall now discuss the importance of phase of the Fourier transform.

There is a tendency to place more importance on the magnitude than the phase. However, we shall show that it is actually the phase that plays a more important role.

The Fourier transform F(u, v) can be expressed in polar co-ordinates as

$$F(u, v) = |F(u, v)| e^{j\phi(u, v)}$$

Where $$|F(u, v)| = \{R^2 [F, (u, v)] + I^2 [F (u, v)]\}^{1/2}$$

is called the magnitude spectrum of the Fourier transform and

$$\phi(u, v) = \tan^{-1} \left\{ \frac{I [F (u, v)]}{R [F (u, v)]} \right\}$$

is the phase spectrum.

Eliminating the phase information by setting the phase equal to zero and performing the Inverse Fourier transform gives us an image in the spatial domain that bears no resemblance to the original image.

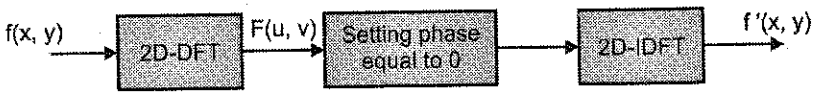


Fig. 6.8.1

On the other hand, eliminating the magnitude information by setting it to a constant and performing the inverse Fourier transform gives us an image which has a certain resemblance to the original image.

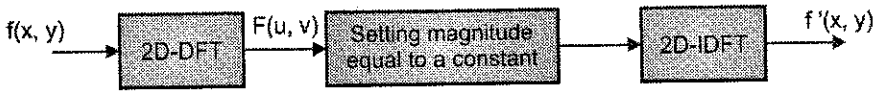


Fig. 6.8.2

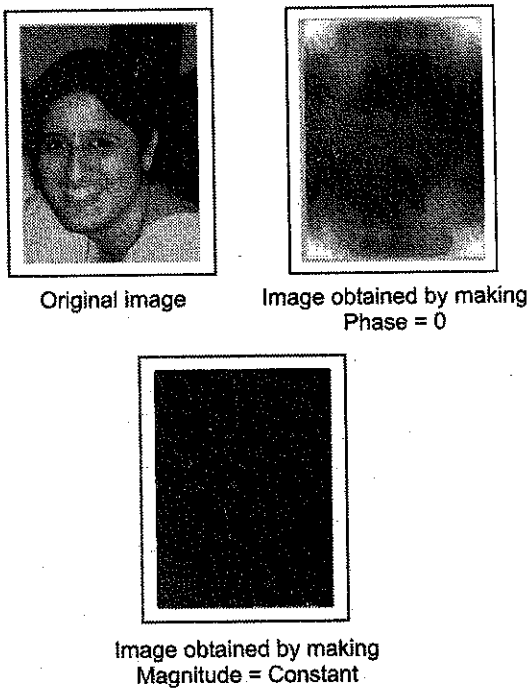


Fig. 6.8.3 : Importance of phase

Note : The Magnitude cannot be zero or else the entire image will become zero.

We shall perform one more experiment to understand the importance of phase.

We now read two different images and interchange their phases and see what happens.

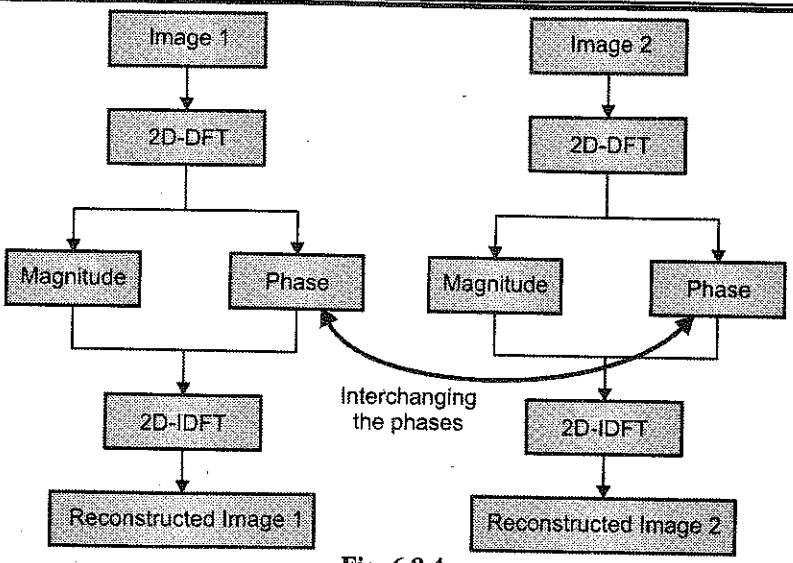


Fig. 6.8.4

Because of interchanging the phases, reconstructed Image 1 looks more like Image 2 while reconstructed Image 2 looks more like Image 1.

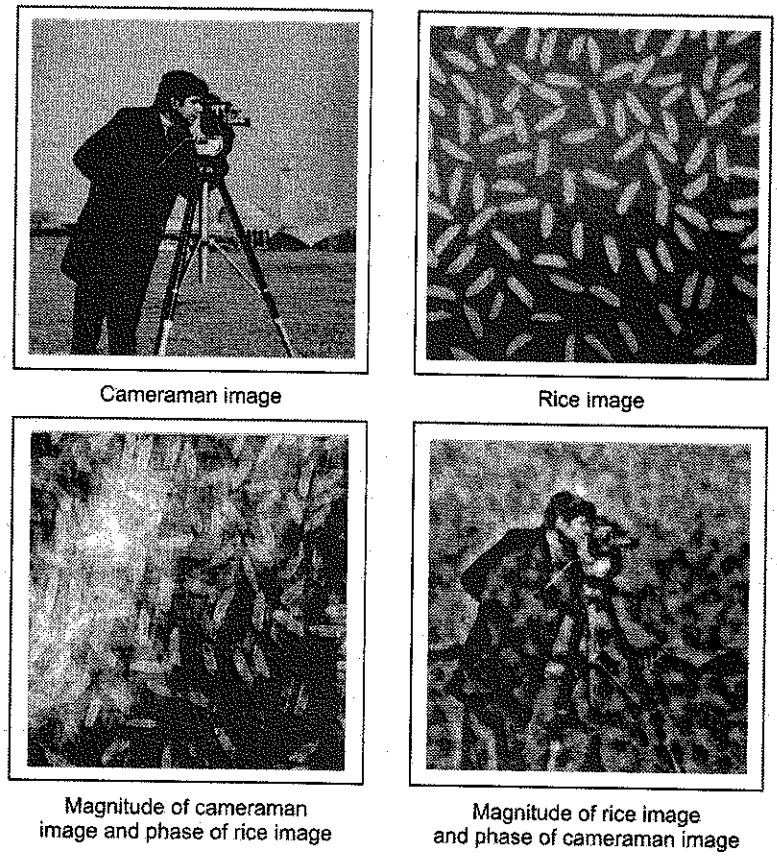


Fig. 6.8.5 : Interchanging phase

Summary

The concept of frequency domain analysis is explained in this chapter. It begins with the introduction to the Fourier transform and then moves on to discuss the various applications of frequency domain analysis. A whole gamut of frequency domain filters is discussed along with MATLAB codes. A simple method of modifying the Fourier spectrum is explained. An important filter known as the homomorphic filter is also discussed along with the code.

Review Questions

Q. 1 Explain the frequency domain concept.

Q. 2 What is advantage of processing an image in the frequency domain ?

Q. 3 Describe the basic principles of image enhancement by

(a) Spatial domain methods

(b) Frequency domain methods

Q. 4 Derive the Fourier transform of the given pulse. Explain the Fourier spectrum that you obtain.

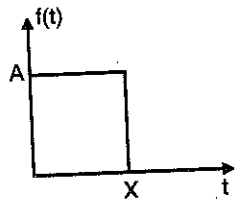


Fig. Q. 4

Q. 5 What are blurring and ringing effects ? How can they be avoided ?

Q. 6 Compare the basic frequency domain filters

(a) Ideal low pass

(b) Butterworth low pass

(c) Gaussian low pass.

Q. 7 Explain the concept of the power spectrum. What information does it yield ?

Q. 8 Explain high boost filter in the frequency domain.

Q. 9 Explain the image model. What do the illumination and reflectance components denote in terms of frequency ?

Q. 10 Explain the homomorphic filter.

Q. 11 Explain the various properties of the Fourier transform. Explain the relevance of each in image processing.

Q. 12 A 2-dimensional-DFT can be obtained using a 1-dimensional-DFT algorithm twice; Explain.

Q. 13 Explain convolution in the spatial and the frequency domain. Derive the relationship between the two domains.

Chapter Ends...

❑❑❑

CHAPTER 7

Image Segmentation

7.1 Introduction

Segmentation, as the name suggests, subdivides (segments) an image into its constituent regions or objects. Unlike image enhancement, were our primary concern was to improve the subjective quality of images for display, in segmentation, we address some aspects of analyzing the content of an image. This simply means we endeavour to find out what is in the picture. Segmentation forms a section of computer vision i.e., we use segmentation, when we want the computer to make decisions. Segmentation is used when we need to automate a particular activity. The ultimate aim in an automated system is to extract important features from image data, from which description, interpretation, or understanding of the scene can be provided by the machine.

Segmentation is not required when the images have to be shown to a human being. This is because a human visual system has an inherent quality to segment the image shownto it.

Let us take two simple examples where image segmentation is used.

(1) **Automated blood cell counting :** We all know blood is made up of RBC's, WBC's, platelets etc.

In manual machines, the technician looks at the slide and based on various parameters classifies them as RBC's, WBC's etc. As stated earlier, the human visual system automatically segments the image as RBC's, WBC's...

In an automated blood cell counting machine, this classification has to be made by the machine (computer). The computer scans the slide and based on certain algorithms classifies them as RBC's, WBC's....

We hence needed to do image segmentation to divide the image into various parts.

(2) **Finger print matching in forensic studies :** In this, there exists a data base of finger prints of people who have been convicted of crime at some point in time.

Now if there has been a crime at a place and some finger prints have been found, images are taken, and these images are run through an algorithm to see if this particular finger print matches with the one in the data base.

Here too, since the system is automated we need to use segmentation to divided the image into various parts.

We hence say that image segmentation is used for computer decision making (machine vision) and not for human interpretation and the main purpose of segmentation is to partition the image.

Image segmentation algorithms are generally based on one of the two basic properties i.e. image segmentation can be achieved in any one of the two ways viz.

(1) Segmentation based on discontinuities in intensity.

(2) Segmentation based on similarities in intensity.

In the first method, the approach is to partition an image based on abrupt changes in intensity, such as edges, while in the second method, we partition an image into regions that are similar according to a set of predefined criteria.

We shall explain segmentation based on discontinuities first and then proceed to explain segmentation by similarities later.

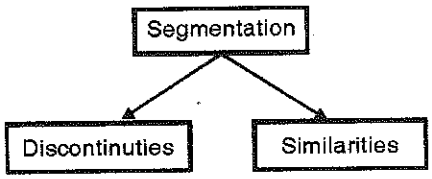


Fig. 7.1.1

7.2 Image Segmentation based on Discontinuties

When we look at an image, we immediately observe 3 basic types of discontinuities in the grey level viz. points, lines and edges. The easiest way is to use spatial masks which have properties to detect these discontinuities.

7.2.1 Point Detection

Remember one thing, points, lines and edges are all high frequency components and hence we need masks which are basically high pass. Hence the masks that we present here for point, line and edge detection have the properties of a high mask mainly, the sum of the coefficients of the mask has to be equal to zero in all the three cases.

Detecting points is fairly simple. We use the standard high pass mask for it. And so that this mask detects only points and not lines, we set a threshold value i.e., we say a point has been detected at the location on which the mask is centered only if

−1	−1	−1
−1	8	−1
−1	−1	−1

$$|R| \geq T \qquad \ldots(7.2.1)$$

Where R is derived from the standard convolution formula

$$R = w_1 z_1 + w_2 z_2 + \ldots\ldots\ldots + w_9 z_9$$

$$R = \sum_{i=1}^{9} w_i z_i$$

We take | R | because we want to detect both the kinds of points i.e. white points on a black background as well as black points on a white background.

T is a non negative threshold which is defined by the user.

7.2.2 Line Detection

Detection of lines can be done using the masks shown below. In an image, lines can be in any direction and detecting these lines would need different masks.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

+ 45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

2	−1	−1
−1	2	−1
−1	−1	2

− 45°

We notice that all these masks have a sum equal to zero, and hence all of them are high pass masks. The first mask would respond strongly to lines that are oriented horizontally. The second mask would respond to lines at an angle of + 45°, the third mask would respond strongly to vertical lines and the last mask would respond strongly to lines at an angle of − 45°.

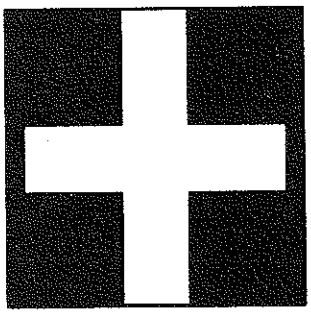


Fig. 7.2.1

Let us take a simple example of a 8×8 pseudo image.

0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0
0	10	10	10	10	10	10	0
0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0

It is quite evident that in this image we have two lines, one horizontal and the second vertical.

Let us see what happens when we move the first (Horizontal) mask over the image. By now you should be quite comfortable with moving masks on the entire image. We leave the borders and hence start from the encircled pixel.

The result that we obtain is as shown

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	– 20	– 20	– 20	– 20	– 30	– 20	0
0	+ 40	+ 40	+ 40	+ 40	+ 60	+ 40	0
0	– 20	– 20	– 20	– 20	– 30	– 20	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Since line detection masks are high pass masks, we encounter negative values. These values are made zero as was discussed in chapter of Image Enhancement in Spatial Domain. Hence the final image obtained is –

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	40	40	40	40	60	40	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

We see that the horizontal mask detects lines only in the horizontal direction and cleans up the other lines (vertical line in this case).

In a similar fashion we can show that vertical, + 45°, and – 45° masks respond strongly to lines in their respective directions. It would be a good idea to try this out yourself.

7.2.3 Edge Detection

More than isolated points and lines, it is the detection of edges that form an important part of image segmentation. An edge can be defined as a set of connected pixels that form a boundary between two disjoint regions.

A typical example of an edge is shown in Fig. 7.2.2.

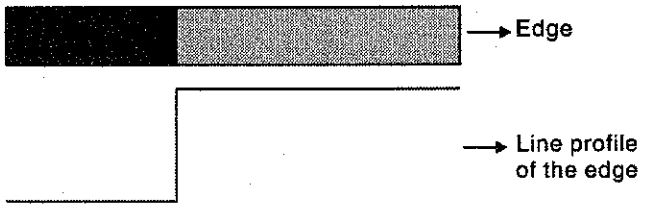


Fig. 7.2.2

Here the set of pixels that separate the black region from the grey region is called an *edge*. The line profile of such an edge is shown along with edge.

The image shown has a step edge. Such step edges occur only in artificially generated images such as test patterns. Images obtained from digitization of optical images of real scenes do not possess step edges. Recollect what happens when we digitize an image. We first sample the image using the Nyquist condition and then quantize it.

Real world signals are not band-limited i.e. F(u, v) ≠ 0 outside an interval and maximum frequency present in it is infinite. Hence the Nyquist criteria is not met. We therefore pass the original signal through an anti-aliasing filter (which is a low pass filter) to remove the very high frequencies.

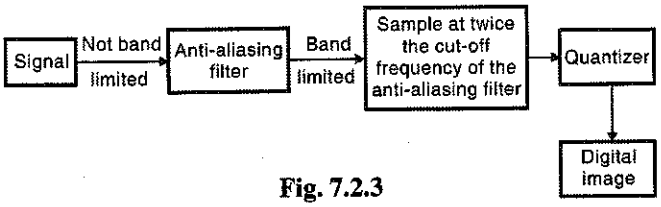


Fig. 7.2.3

It is this anti-aliasing filter which reduces the slope of the edge. Due to this, real world images when captured through a camera, will not have step edges. In practice, edges are modelled as having a ramp like profile.

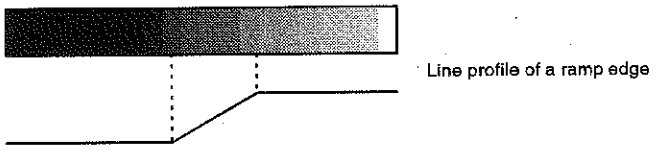


Fig. 7.2.4

Here the slope of the ramp is inversely proportional to the degree of blurring.

Given below are some of the two dimensional, discrete domain edge models.

a a a a a b b b b b a a a a a b b b b b a a a a a b b b b b a a a a a b b b b b a a a a a b b b b b	a a a b b b b b b b a a a a b b b b b b a a a a a b b b b b a a a a a a b b b b a a a a a a a b b b
Vertical step edge	Diagonal step edge
a a a a a a a a a a a a a a a a a a a a a a a a a b b b b b a a a a a b b b b b a a a a a b b b b b	a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b
Corner step edge	Vertical ramp edge

a a c b b b b b b b a a a c b b b b b b a a a a c b b b b b a a a a a c b b b b a a a a a a c b b b	a a a a a a a a a a a a a a c c c c c c a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b
Diagonal ramp edge Single pixel transition	Corner ramp edge
a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b	a a d e b b b b b b a a a d e b b b b b a a a a d e b b b b a a a a a d e b b b a a a a a a d e b b
Vertical ramp edge	Diagonal ramp edge Smoothed transition

a a a a a a a a a a a a a a d c c c c c a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b
Corner ramp edge

$$c = \frac{a+b}{2}, \quad d = \frac{3a+b}{4}, \quad e = \frac{a+3b}{4}, \quad b > a$$

Fig. 7.2.5

7.3 Detection of Edges

How do these edges come in an image ?

Variation of scene features, usually brightness, give rise to edges. In other words, edges are representations of the discontinuities of the scene intensity function. There could be various reasons such as type of material, surface texture, lighting conditions, etc., which play an important role in forming these discontinuities.

Our aim in this chapter is to detect these discontinuities as edges. The process of edge detection is carried out by the derivative approach.

The basic idea of the derivative approach is the computation of the local derivative operator.

Consider the image shown in Fig. 7.3.1.

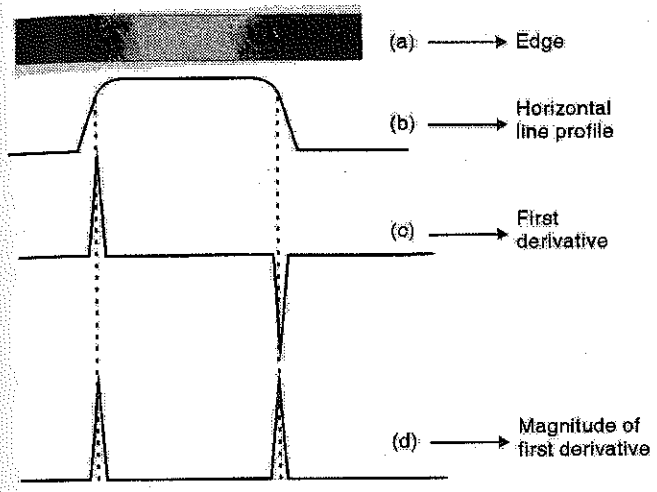


Fig. 7.3.1

Fig. 7.3.1 shows that the first derivative of the grey line profile is positive at the leading edge of a transition, negative at the trailing edge and zero in areas of constant grey levels.

If we now want to plot the derivative image, we need to take the magnitude of Fig. 7.3.1(c). This gives us Fig. 7.3.1(d). As can be seen, we get non-negative values only at the two edges of the image. We can thus conclude that computing the derivative helps us in detecting the edges.

Although this discussions has been limited to the 1-D horizontal profile, a similar argument applies to an edge of any orientation in an image.

The first derivative at any point in an image is obtained by using the magnitude of the gradient at that point.

7.3.1 Computing the Gradient

Let us spend some time understanding the basics of the gradient. This discussion will help us understand edge detection better.

Case 1 : Output is a function of one variable (**Example :** 1-dimensional signal) y = f(x).

We know that the derivative of y w.r.t. x is

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = f'(x)$$

The slope is $\frac{dy}{dx}$

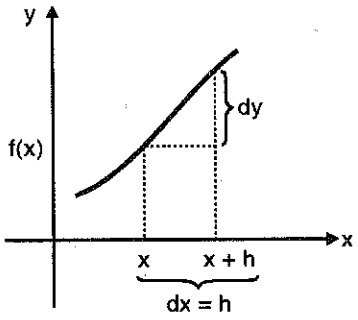


Fig. 7.3.2

Case 2 : Output is a function of two variables (2-dimensional example), f(x, y). In the two variable case, x and y are the variables. Both these variables change to say x + h and y + k. To find the gradient, we work with each separately i.e., we take partial derivatives.

Let us first keep y fixed and alter only x.

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h}$$

Here $\frac{\partial f}{\partial x}$ is the rate of change of f w.r.t. x keeping y fixed.

Similarly, keeping x fixed and changing y, we get

$$\frac{\partial f}{\partial y} = \lim_{k \to 0} \frac{f(x, y+k) - f(x, y)}{k}$$

$\frac{\partial f}{\partial y}$ is the rate of change of f w.r.t. y keeping x constant.

Hence the final gradient is

$$\nabla F = \hat{i} \frac{\partial f}{\partial x} + \hat{j} \frac{\partial f}{\partial y} \qquad ...(7.3.1)$$

Real world signals are not band-limited i.e. F(u, v) ≠ 0 outside an interval and maximum frequency present in it is infinite. Hence the Nyquist criteria is not met. We therefore pass the original signal through an anti-aliasing filter (which is a low pass filter) to remove the very high frequencies.

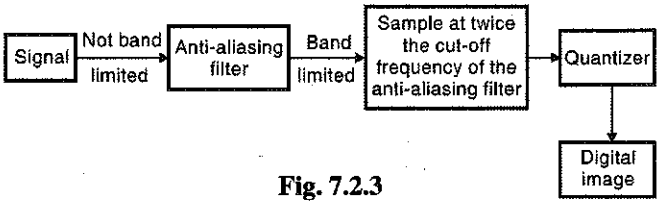


Fig. 7.2.3

It is this anti-aliasing filter which reduces the slope of the edge. Due to this, real world images when captured through a camera, will not have step edges. In practice, edges are modelled as having a ramp like profile.

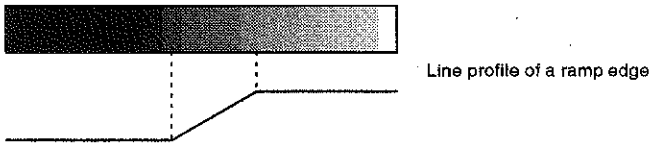


Fig. 7.2.4

Here the slope of the ramp is inversely proportional to the degree of blurring.

Given below are some of the two dimensional, discrete domain edge models.

a a a a a b b b b b a a a a a b b b b b a a a a a b b b b b a a a a a b b b b b a a a a a b b b b b	a a a b b b b b b b a a a a b b b b b b a a a a a b b b b b a a a a a a b b b b a a a a a a a b b b
Vertical step edge	Diagonal step edge
a a a a a a a a a a a a a a a a a a a a a a a a a b b b b b a a a a a b b b b b a a a a a b b b b b	a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b
Corner step edge	Vertical ramp edge

a a c b b b b b b b a a a c b b b b b b a a a a c b b b b b a a a a a c b b b b a a a a a a c b b b	a a a a a a a a a a a a a a c c c c c c a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b
Diagonal ramp edge Single pixel transition	Corner ramp edge
a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b	a a d e b b b b b b a a a d e b b b b b a a a a d e b b b b a a a a a d e b b b a a a a a a d e b b
Vertical ramp edge	Diagonal ramp edge Smoothed transition

a a a a a a a a a a a a a a d c c c c c a a a a c b b b b b a a a a c b b b b b a a a a c b b b b b
Corner ramp edge

$$c = \frac{a+b}{2}, \quad d = \frac{3a+b}{4}, \quad e = \frac{a+3b}{4}, \quad b > a$$

Fig. 7.2.5

7.3 Detection of Edges

How do these edges come in an image ?

Variation of scene features, usually brightness, give rise to edges. In other words, edges are representations of the discontinuities of the scene intensity function. There could be various reasons such as type of material, surface texture, lighting conditions, etc., which play an important role in forming these discontinuities.

Our aim in this chapter is to detect these discontinuities as edges. The process of edge detection is carried out by the derivative approach.

The basic idea of the derivative approach is the computation of the local derivative operator.

Consider the image shown in Fig. 7.3.1.

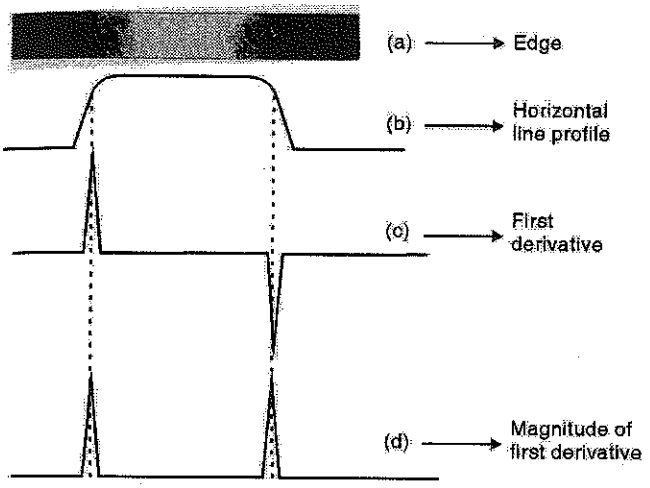


Fig. 7.3.1

Fig. 7.3.1 shows that the first derivative of the grey line profile is positive at the leading edge of a transition, negative at the trailing edge and zero in areas of constant grey levels.

If we now want to plot the derivative image, we need to take the magnitude of Fig. 7.3.1(c). This gives us Fig. 7.3.1(d). As can be seen, we get non-negative values only at the two edges of the image. We can thus conclude that computing the derivative helps us in detecting the edges.

Although this discussions has been limited to the 1-D horizontal profile, a similar argument applies to an edge of any orientation in an image.

The first derivative at any point in an image is obtained by using the magnitude of the gradient at that point.

7.3.1 Computing the Gradient

Let us spend some time understanding the basics of the gradient. This discussion will help us understand edge detection better.

Case 1 : Output is a function of one variable (**Example :** 1-dimensional signal) y = f(x).

We know that the derivative of y w.r.t. x is

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = f'(x)$$

The slope is $\frac{dy}{dx}$

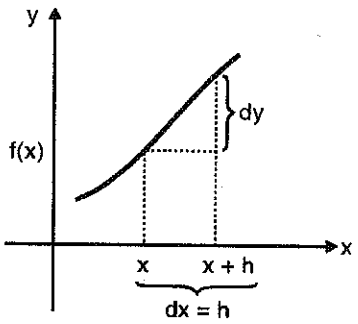


Fig. 7.3.2

Case 2 : Output is a function of two variables (2-dimensional example), f(x, y). In the two variable case, x and y are the variables. Both these variables change to say x + h and y + k. To find the gradient, we work with each separately i.e., we take partial derivatives.

Let us first keep y fixed and alter only x.

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h}$$

Here $\frac{\partial f}{\partial x}$ is the rate of change of f w.r.t. x keeping y fixed.

Similarly, keeping x fixed and changing y, we get

$$\frac{\partial f}{\partial y} = \lim_{k \to 0} \frac{f(x, y+k) - f(x, y)}{k}$$

$\frac{\partial f}{\partial y}$ is the rate of change of f w.r.t. y keeping x constant.

Hence the final gradient is

$$\nabla F = \hat{i} \frac{\partial f}{\partial x} + \hat{j} \frac{\partial f}{\partial y} \quad ...(7.3.1)$$