CHAPTER 5

Introduction to Image Processing

Syllabus

Introduction to Digital Image, Digital Image Processing System, Image File Formats: BMP, TIFF and JPEG.

5.1 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.

5.2 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. 5.2.1.
- 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. 5.2.1

- 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.

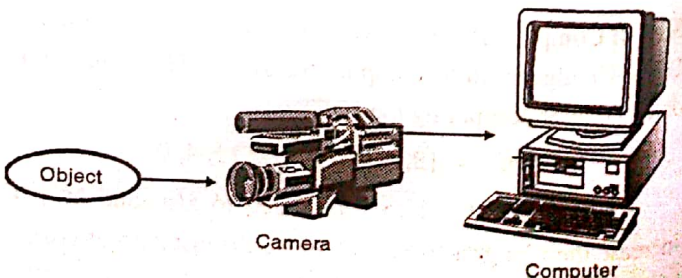


Fig. 5.2.2

5.3 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. 5.3.1.

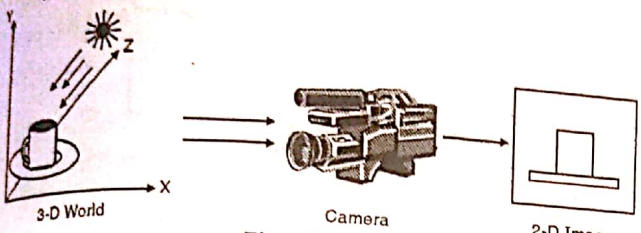


Fig. 5.3.1

- 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. 5.3.1, 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. 5.3.2. 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.

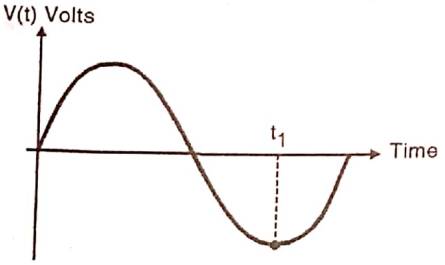


Fig. 5.3.2

- Let us see why images are 2-dimensional functions. Consider the image shown in Fig. 5.3.3.

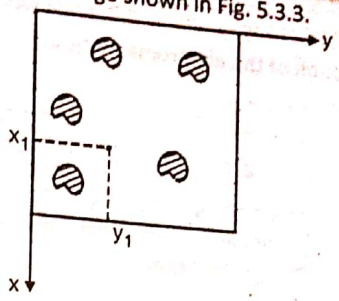


Fig. 5.3.3

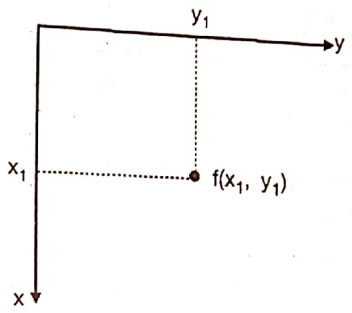


Fig. 5.3.4

- In this case, to locate the dot shown, we need to know its position in two directions (x and y) Fig. 5.3.4. Hence all images that we see are 2-dimensional functions. A typical image is represented as shown in Fig. 5.3.4. 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.

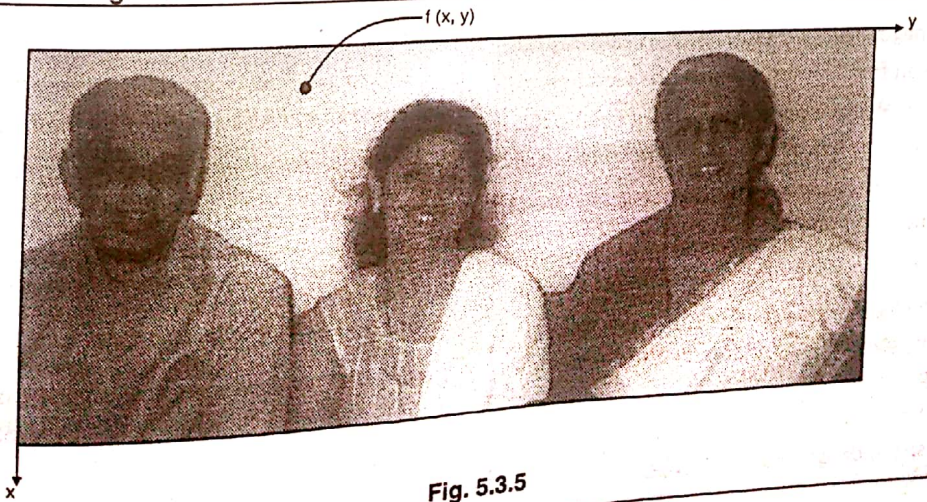


Fig. 5.3.5

5.4 The Electromagnetic Spectrum

- The apparatus shown in Fig. 5.2.2 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. 5.4.1.

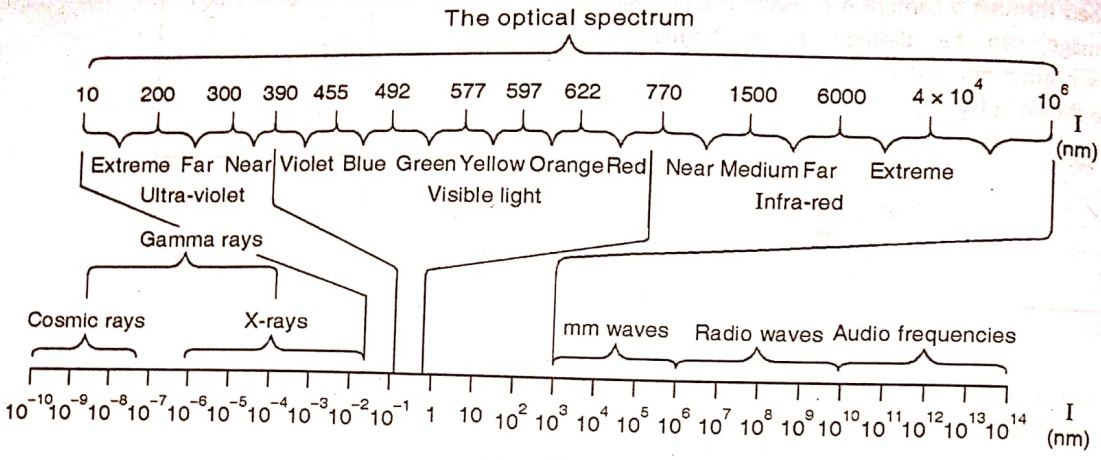


Fig. 5.4.1

- 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

$$\lambda = \frac{c}{\nu} \qquad \ldots(5.4.1)$$

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

Solved Example

Ex. 5.4.1 : Calculate the frequency of oscillation of green light.

Soln. :

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

$(500 \times 10^{-9}$ m)

Its frequency of oscillations can be calculated using Equation (5.4.1).

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

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

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

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).

5.5 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.

5.6 Basic Elements/Steps of an Image Processing System

MU - Dec. 2018

Q. Explain Fundamental steps in Image processing. **(Dec. 2018, 5 Marks)**

Digital image processing is basically modification of images on a computer. The basic components of an image processing system are as follows :

(1) Image Acquisition.
(2) Image Storage.
(3) Image Processing.
(4) Display.
(5) Transmission (if required).

We shall discuss each one in detail.

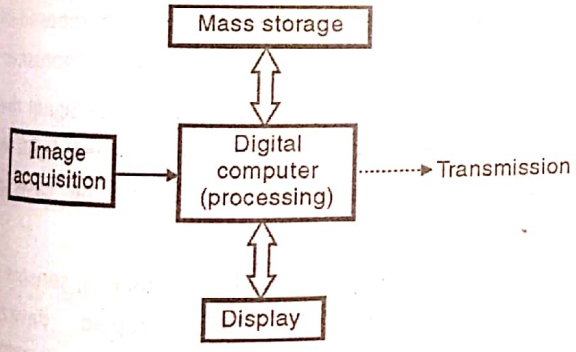


Fig. 5.6.1

(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.

$$e = \frac{h c}{\lambda} \qquad \ldots(5.6.1)$$

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 (5.6.1) 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.

Vidicon imaging tubes

- Vidicon is a vacuum tube used for image acquisition. Fig. 5.6.2 is the schematic diagram of the vidicon tube image sensor. The light image is focused on a transparent signal plate coated on the inside with a photoconductive target material.

- The front face of the signal plate is coated with SnO_2 (tin oxide) which forms the conducting layer. This plate is biased to a positive potential of about 50 V. The target coating is maintained at approximately 0 V by a mesh grid which is placed behind it.

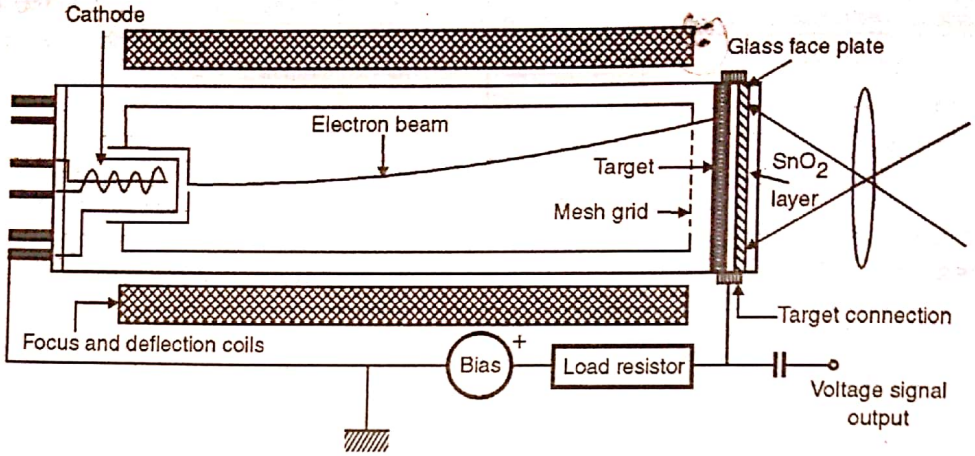


Fig. 5.6.2

- Due to this the signal plate and target behave as a continuous array of capacitors containing a photoconductive dielectric.
- When there is no light incident on the target, its resistance is very high due to which the charge across the capacitor is maintained. However when light strikes the target, the dielectric material becomes leaky and the target side voltage rises.
- The target is now scanned by a low velocity electron beam. When this beam strikes the discharged capacitor, current flows to restore the target side to its original voltage of 0 V.
- This current also flows in the external bias circuit, which produces a voltage drop in the load resistance.
- In order to produce the output necessary for broadcast video applications, the electron beam is scanned across the surface of the target by electromagnetic deflection coils. (Deflection coils were encountered when you studied CRO, (cathode ray oscilloscopes), in your lower semesters).
- The original vidicons use antimony-trisulphide, which is a photoconductive material, as their target coatings. Other forms of vidicon are made using more complex target materials.
- The silicon target vidicon achieves improved sensitivity and a spectral response similar to that of the human eye.
- The biggest drawback of the vidicon tube is that it uses the concept of deflection coils to position the electron beam. These deflection coils make the vidicon bulky and are responsible for major power drain. Since the coils are magnetic, the vidicon also becomes sensitive to external fields.

(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.

(i) Line array CCD.
(ii) Area array CCD.

(i) **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 photo sites.

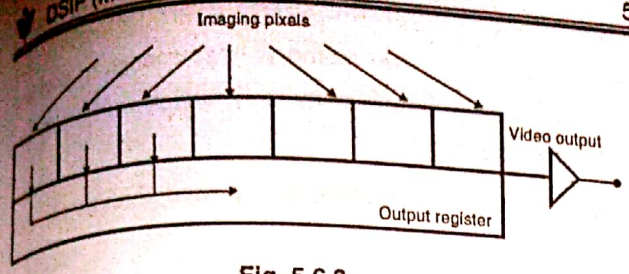


Fig. 5.6.3

- A single line of CCD pixels are clocked out into the parallel output register as shown in Fig. 5.6.3. 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.

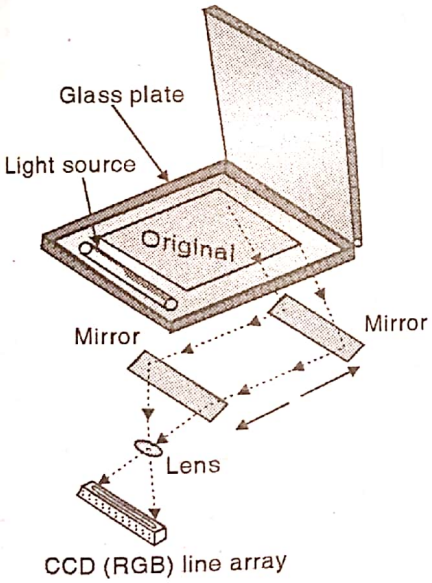


Fig. 5.6.4

- 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.

(ii) **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.

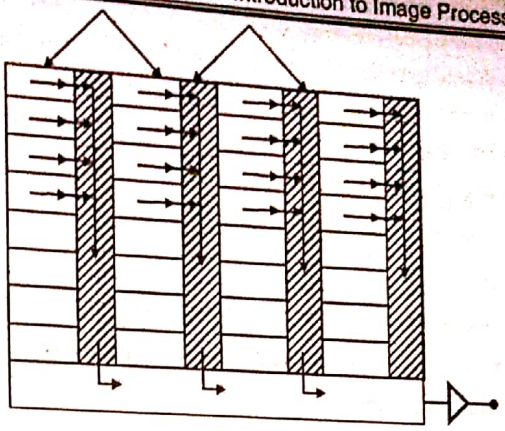


Fig. 5.6.5

- 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.

- 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.

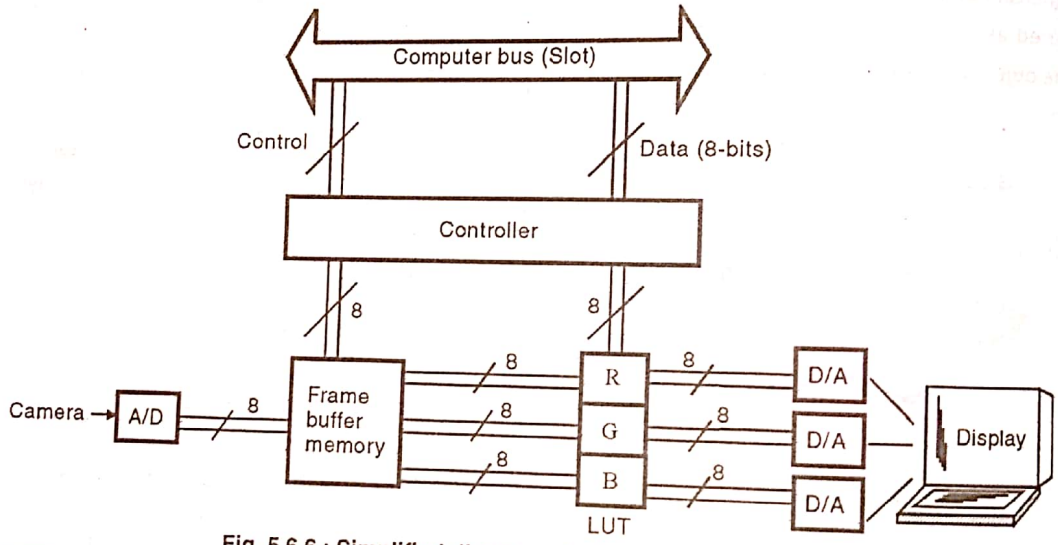


Fig. 5.6.6 : Simplified diagram of a frame grabber card

- 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 × 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 × B × C bits.
- Suppose we have D such images then total number of bits ≈ A × B × C × D ...(5.6.2)

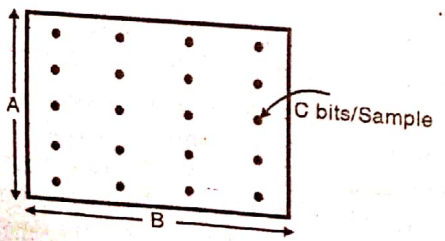


Fig. 5.6.7 (a)

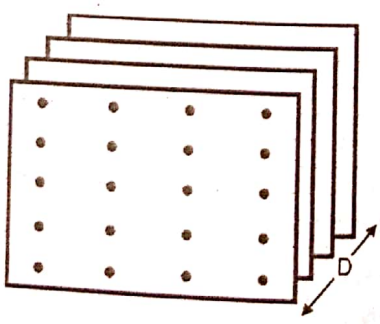


Fig. 5.6.7(b)

- Thus the formula ≈ ABCD gives us a fairly good idea of the amount of space required.

Table 5.6.1

	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. 5.6.8.

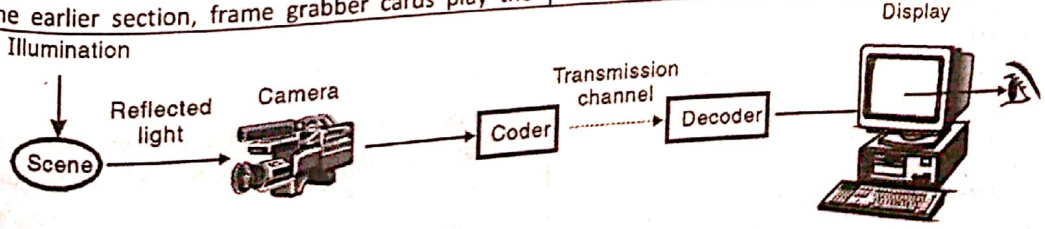


Fig. 5.6.8

- 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. After decoding at the receiver, the image sequence is
- **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

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

where, 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.

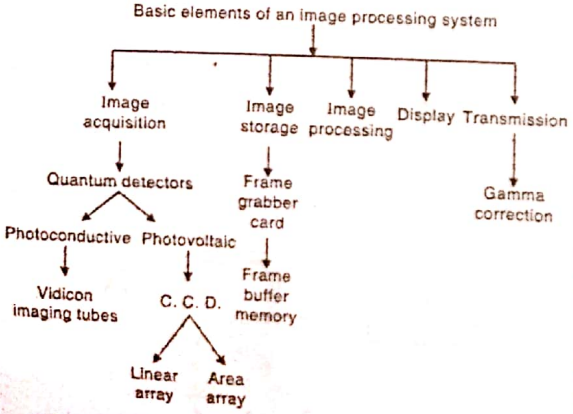


Fig. 5.6.9

5.7 Image Types

Images are 2-dimensional functions, as shown in Fig. 5.7.1.

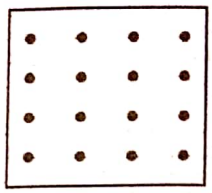


Fig. 5.7.1

Images can be classified as follows :

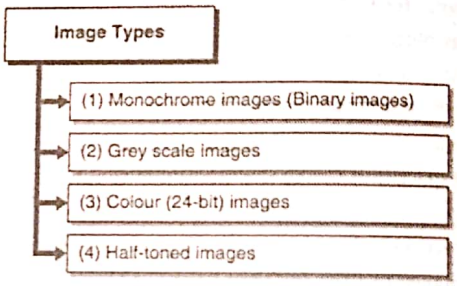


Fig. 5.7.2 : Image Types

(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. 5.7.3.

Fig. 5.7.3

(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. 5.7.4.

Fig. 5.7.4

(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

$$X = 0.30\,R + 0.59\,G + 0.11\,B \quad \ldots(5.7.1)$$

- An easier formula that could achieve similar results is

$$X = \frac{R + G + B}{3} \quad \ldots(5.7.2)$$

MATLAB code for converting a colour image to a grey scale image is as follows :

```
%% 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) = 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. 5.7.5.

Fig. 5.7.5

- 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.

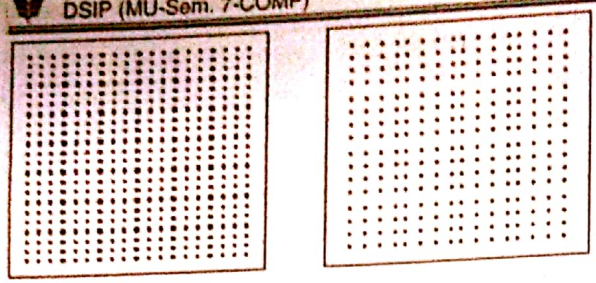


Fig. 5.7.6

- 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.

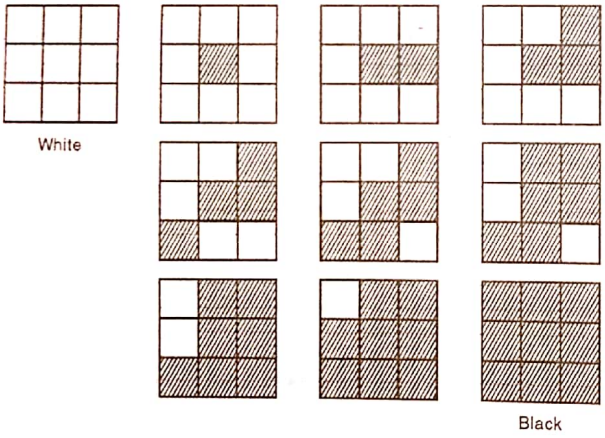


Fig. 5.7.7

- 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. 5.7.8, the half-toned image would look like Fig. 5.7.9.

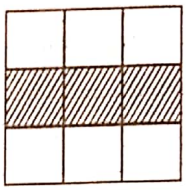


Fig. 5.7.8

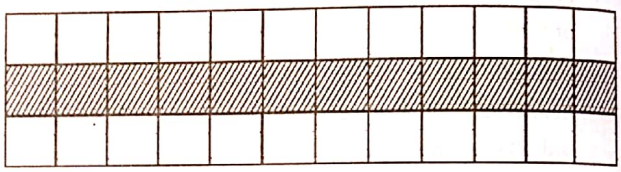


Fig. 5.7.9

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 as follows :

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

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

(iii) 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.

5.8 Image File Formats

MU : May 2016, Dec. 2016

Q. Explain in detail any two types of image file formats. **(May 2016, 8 Marks)**

Q. What are various file formats ? Explain each in brief. **(Dec. 2016, 8 Marks)**

- 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. 5.8.1 shows the typical organisation of information encoded in an image file.

The image file consists of two parts;

(1) Header

(2) Image data

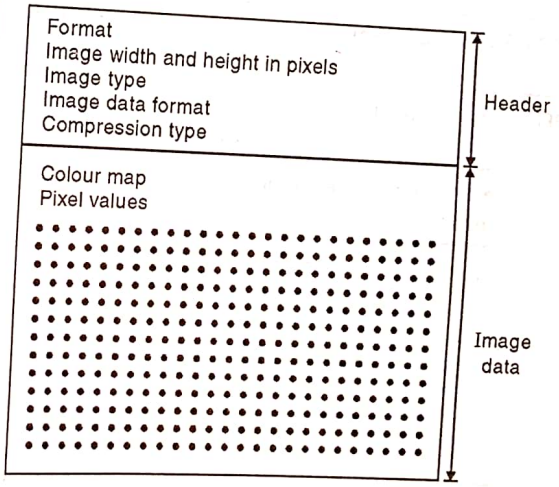


Fig. 5.8.1

- 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 follows :

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

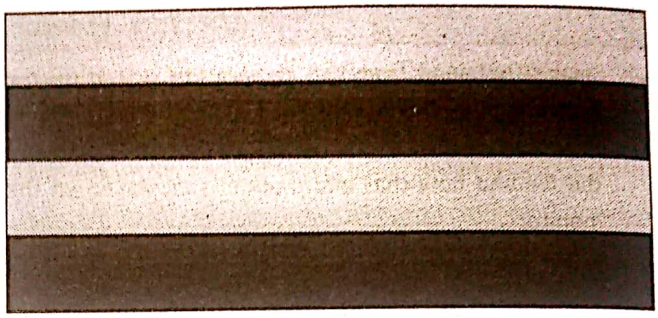


Fig. 5.8.2

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.

5.9 Image Processing, Computer Graphics and Computer Vision

- 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).

- 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 follows.

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

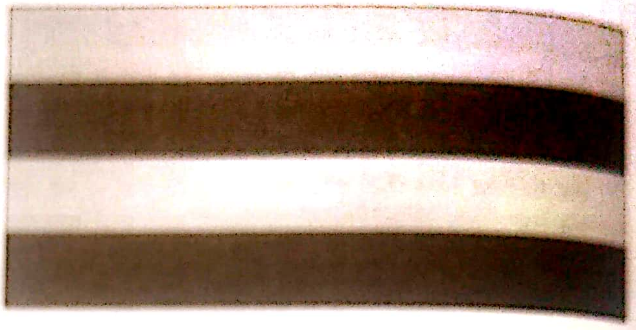


Fig. 5.8.2

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.

5.9 Image Processing, Computer Graphics and Computer Vision

- 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 5.9.1

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

From the Table 5.9.1,

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 are discussed with examples.

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. 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 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 Distinguish between image processing and graphics.

Q. 5 List the basic elements of an image processing system.

Q. 6 Explain working of a Vidicon camera.

Q. 7 Explain line array and area array CCD cameras.

Q. 8 Write a short note on frame grabber card.

Q. 9 Explain the concept of half toning.

Q. 10 Explain image formats.

Q. 11 Advantages of CCD over Vidicon.

Q. 12 Explain the basics of quantum detector.

Q. 13 Explain gamma of a camera.

Q. 14 How many grey levels will a half toned image have ? Explain in detail.

❑❑❑

CHAPTER 6

Sampling and Quantization

Syllabus :
Sampling and Quantization. Representation of Digital Image

6.1 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 :

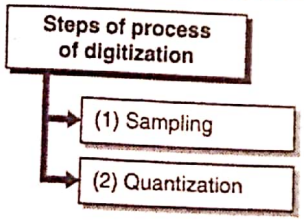


Fig. 6.1.1 : Steps of process of digitization

In other words, Digitization = Sampling + Quantization.

- We have had exposure to these terms in the lower semesters in subjects like Principles of Communication Engineering, Signals and Systems and Signal Processing which dealt with 1-dimensional signals. Let us take a brief look at these concepts and move ahead to the 2-dimensional domain.

6.2 Sampling

MU - May 2012, May 2013, Dec. 2013

Q. Write short note on : Sampling and Quantization.
(May 2012, May 2013, Dec. 2013, 5 Marks)

- Consider a band-limited signal. Although real world signals are rarely band limited, we can produce one by passing the signal first through a low-pass filter known as an anti-aliasing filter.

Note : A signal is said to be band limited, if its Fourier transform (FT), F(u), is zero outside a bounded region.

- (To understand sampling, we first need to understand convolution involving impulse functions). If f(t) is a step signal shown in Fig. 6.2.1 and g(t) is a train of impulses (g(t) is also known as a COMB function), then the convolution of the two is given by f(t) * g(t). Let us see what happens when we convolve f(t) with g(t).

Proof : We know that the convolution of f(t) and g(t) is

$$y(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau = \int_{-\infty}^{\infty} g(\tau)\, f(t-\tau)\, d\tau$$

$$y(t) = \int_{-\infty}^{\infty} [\delta(t-T) + \delta(t) + \delta(t+T)]\, f(t-\tau) d\tau$$

$$y(t) = \int_{-\infty}^{\infty} \delta(t-T)\, f(t-\tau)\, d\tau + \int_{-\infty}^{\infty} \delta(t)\, f(t-\tau)\, d\tau + \int_{-\infty}^{\infty} \delta(t+T)\, f(t-\tau)\, d\tau$$

but, $\int_{-\infty}^{\infty} \delta(t-T)\, f(t-\tau)\, d\tau = \delta(t-T) * f(t)$

[This is impulse at $t = -T$ convolved with $f(t)$]

and, $\int_{-\infty}^{\infty} \delta(t+T)\, f(t-\tau)\, d\tau = \delta(t+T) * f(t)$

[This is impulse at t = +T convolved with f(t)]

This gives us the figures as shown in Fig. 6.2.1.

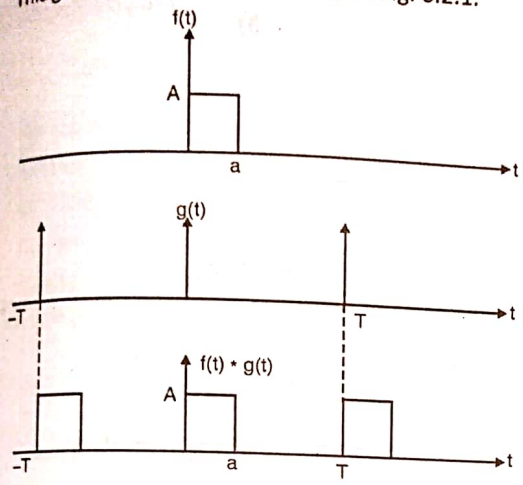


Fig. 6.2.1

- We see that convolution replicates the signal f(t) at every impulse g(t).
- Another important property that one needs to know is that convolution in one domain is equal to multiplication in the other i.e. convolution in the time domain is equal to multiplication in the frequency domain and similarly multiplication in the time domain is equal to convolution in the frequency domain.
- As is seen in Fig. 6.2.2, if we use Fig. 6.2.2(c) as the sampling period, we encounter overlapping or aliasing in the frequency domain [Fig. (f)]. This is due to undersampling. If we decrease the sampling period (increase the sampling frequency) as in Fig. 6.2.2(g), we can eliminate aliasing in the frequency domain [Fig. 6.2.2(h)].
- If we now multiply Fig. (h) with a gating function H(u) [Fig. 6.2.2(i)] we can select only the required part of the spectrum [Fig. 6.2.2(k)]. We take the inverse Fourier transform to get back the original signal f(t) [Fig. 6.2.2(j)].
- All this was possible because the sampling period was small (sampling frequency was large). Hence to avoid aliasing, the sampling frequency should be greater than or equal to 2 × Bandwidth (2 × w).
- In other words, $\Delta t \leq \frac{1}{2w}$. This is the Whittaker-Shannon sampling theorem. Only if this condition is met, can we recover the original signal. These 1-D concepts can be easily extended to the 2-D domain. A function f(x, y) is called a band limited function if its Fourier transform F(u, v) is zero outside a bounded region in the frequency plane i.e.

  $F(u, v) = 0 \quad u > u_0, v > v_0$
- Let us consider a band limited signal as shown in Fig. 6.2.2.

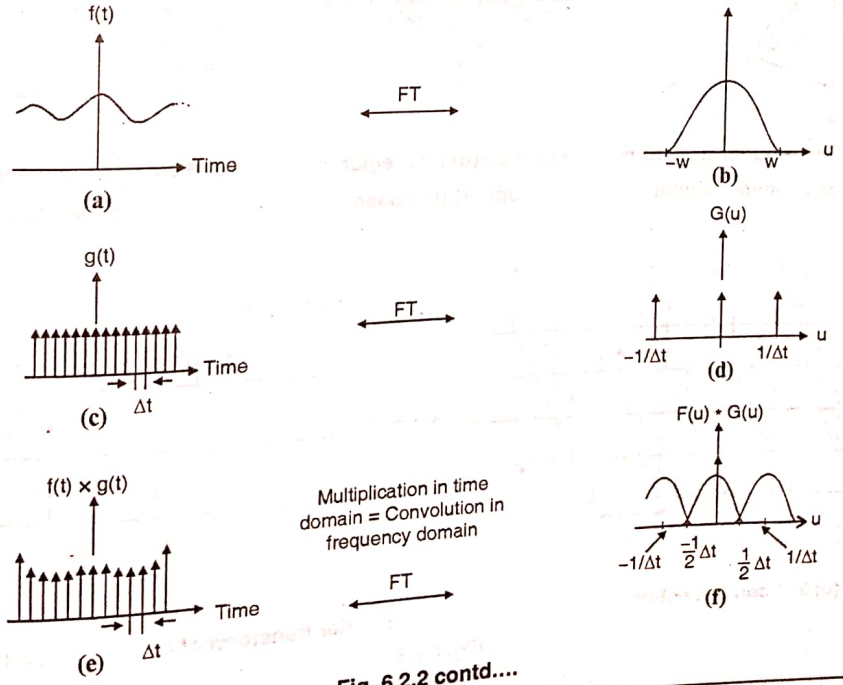


Fig. 6.2.2 contd....

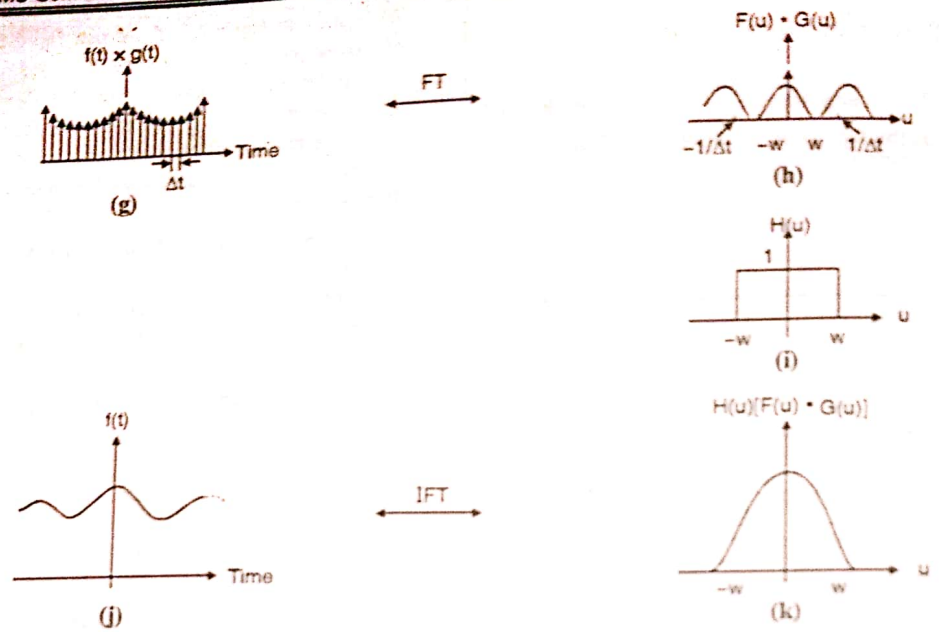


Fig. 6.2.2

– Fig. 6.2.3 (a) and (b) represent the Fourier Transform of a band limited signal and its region of support (the base) respectively.

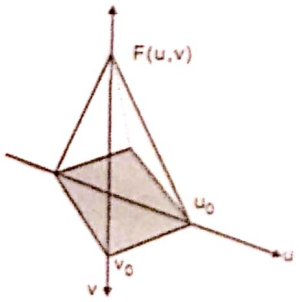


Fig. 6.2.3(a) : Fourier transform of a band limited signal

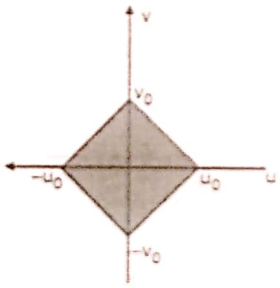


Fig. 6.2.3(b) : Its region of support (the base)

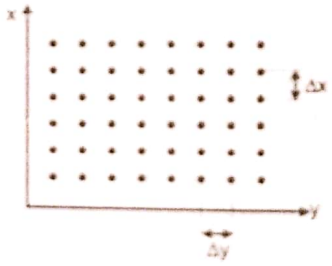
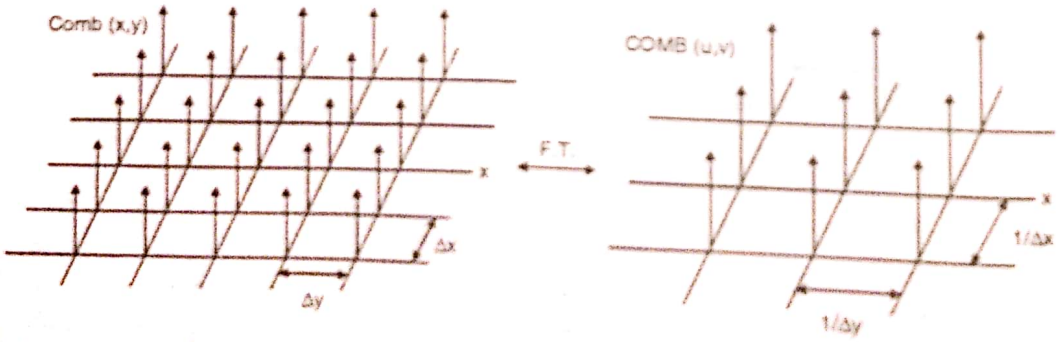


Fig. 6.2.4 : Two dimensional comb function



(a) 2-D comb function

(b) Fourier transform of COMB function

Fig. 6.2.5

- Just as in the case of a 1-D function, consider a 2-D comb function (sampling function) [Refer Fig. 6.2.4]

A 2-dimensional comb function is defined as

$$\text{Comb}(x, y; \Delta x, \Delta y) \triangleq \sum_{m,n=-\infty}^{+\infty} \sum \delta(x - m\Delta x, y - n\Delta y)$$

...(6.2.1)

- The top view of the comb function is given in Fig. 6.2.4. Another way of representing a 2-D function or a sampling function is shown in Fig. 6.2.5(a).
- The Fourier transform of the comb function with spacing of Δx and Δy is another COMB function with spacing $1/\Delta x$ and $1/\Delta y$.

$$\text{Comb}(u, v) = \text{FT}\{\text{comb}(x, y; \Delta x, \Delta y)\}$$
$$= \text{COMB}(u, v; 1/\Delta x, 1/\Delta y)$$

The subsampled image is defined as

$$f_s(x, y) = f(x, y) \times \text{comb}(x, y; \Delta x, \Delta y)$$

$$f_s(x, y) = \sum_{m,n=-\infty}^{+\infty} \sum f(m\Delta x, n\Delta y)\, \delta(x - m\Delta x, y - n\Delta y)$$

...(6.2.2)

- Since multiplication in the time domain is equal to convolution in the frequency domain, we get

$$F_s(u, v) = F(u, v) * \text{COMB}(u, v) \quad ...(6.2.3)$$

- We know from the 1-D case, $F_s(u, v)$ will simply be replicating F(u, v) at all impulses. Hence for a 2-D case, using Fig. 6.2.3(b), we have,

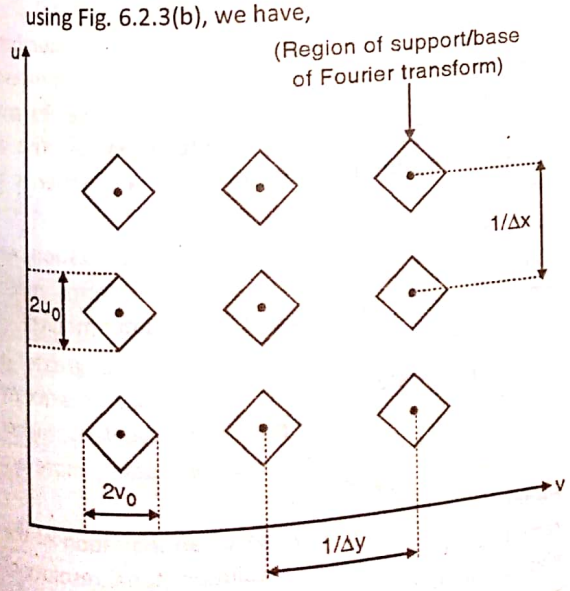


Fig. 6.2.6

- To avoid aliasing, the sampling frequency should be twice the maximum frequency in the signal (image)

$$\Delta x \leq \frac{1}{2u_0}$$

$$\text{and } \Delta y \leq \frac{1}{2v_0} \quad ...(6.2.4)$$

Where, $2u_0$ and $2v_0$ are the bandwidths in the u and v directions respectively. This is known as the Nyquist condition.

$\Delta x = \frac{1}{2u_0}$ and $\Delta y = \frac{1}{2v_0}$ are the lower bounds of the equation and are known as the Nyquist rate.

As seen from the Fig. 6.2.6, there is absolutely no aliasing. If the Nyquist condition is not met, overlapping of frequencies occurs and this is known as aliasing. This overlapping is shown in Fig. 6.2.7.

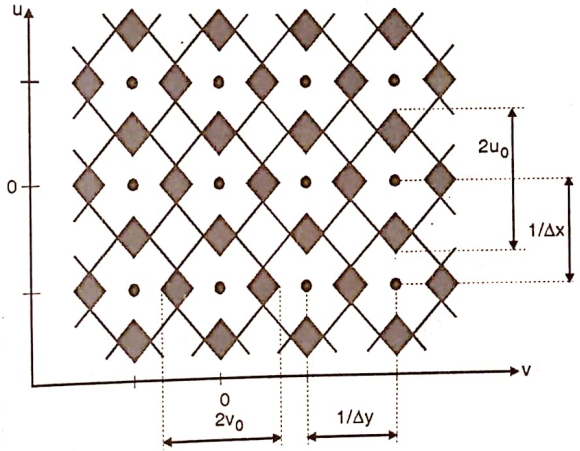


Fig. 6.2.7 : Aliasing due to low sampling frequency

- The shaded portions are where overlapping (aliasing) takes place.
- We observe aliasing when we go to watch movies. Whenever the hero of the movie (could be the villain too) races his car in the climax, have you ever observed the wheels of his car ?
- When the car is slow, the wheels move in the same direction as his car. But what happens when he speeds up his car ? The wheels start moving in the opposite direction. This is because; the camera, which is shooting this scene, has a particular maximum sampling frequency.
- As the car speeds up the Nyquist condition is not met and hence there is aliasing, i.e. high frequencies start looking as low frequencies.

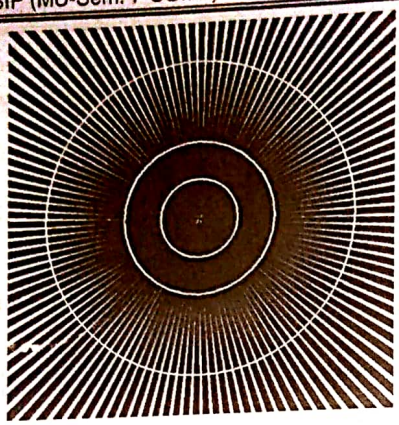


Fig. 6.2.8(a) : Original image

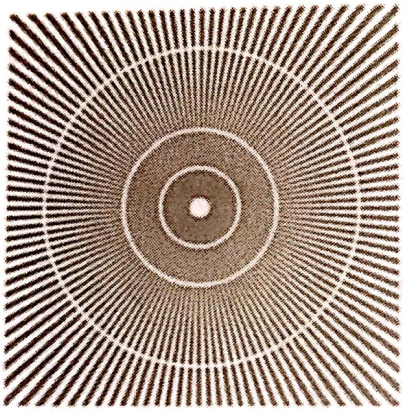


Fig. 6.2.8(b) : Sampled image

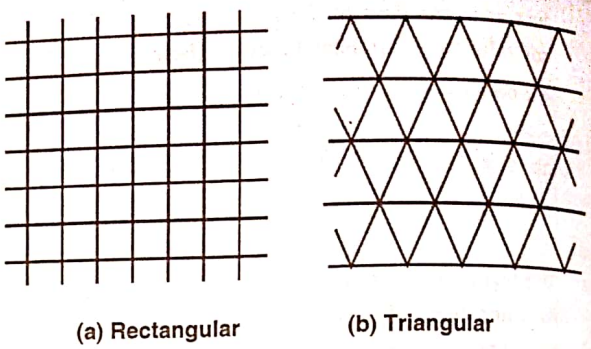
Coming back to the summation Equation (6.2.2),

$$f_s(x,y) = \sum_{m,\,n} \sum_{=-\infty}^{\infty} f(m\,\Delta x, n\,\Delta y)\,\delta(x - m\,\Delta x, y - n\,\Delta y)$$

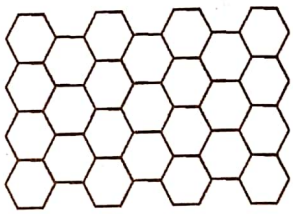
The equation suggests that a sampled 2-D function defined over a domain of size

M $\Delta x \times$ N Δy can be considered as a 2-D array {f(0,0), f(0,1),.....}

- This means that the function is represented as an M × N uniformly spaced sample. Apart from the sampling interval in the x and y directions, another important choice relevant to the image sampling is the spatial arrangement of the sample points called Tessellation.
- We normally consider a comb function, which is a rectangular tessellation of grid. However other types of tessellations of the image plane, namely triangular and hexagonal, may also be used. Throughout this book though, we use rectangular tessellations.



(a) Rectangular (b) Triangular



(c) Hexagonal

Fig. 6.2.9 : Commonly used sampling grids

6.3 Quantization

MU - May 2012, May 2013, Dec. 2013

Q. Write short note on : Quantization.

(May 2012, May 2013, Dec. 2013, 5 Marks)

- This is also a topic that has been widely discussed in the lower semesters. 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 i.e. lower is the image area $\Delta x\,\Delta y$ represented by each sampled point. 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

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & \cdots & f(1,M-1) \\ f(2,0) & f(2,1) & \cdots & f(2,M-1) \\ \vdots & & & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix}_{N \times M}$$

- 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.

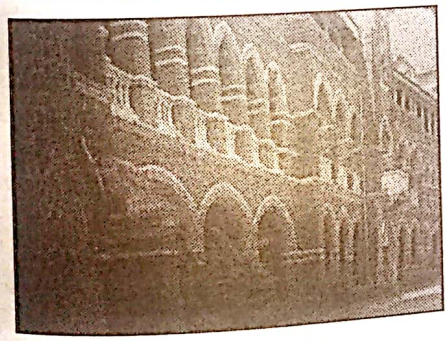
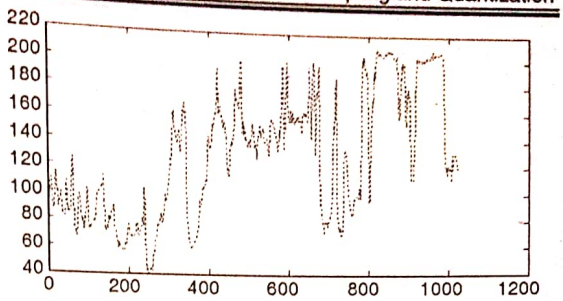


Fig. 6.3.1(a)



(b)

Fig. 6.3.1 : The first row of the image is plotted.

- 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.
- 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 6.3.1 : 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

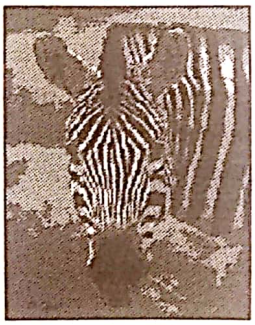
We shall see the effects of reducing quantization levels and reducing spatial resolution separately.

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

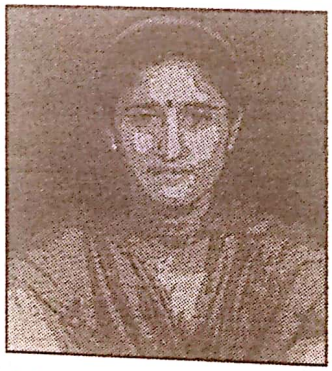
(d) Image using 1-bit

Fig. 6.3.2

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

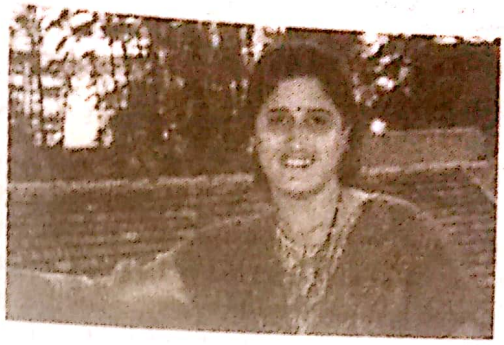
Fig. 6.3.3

Fig. 6.3.3 (c) Down sampled image displayed after zooming to match the size of the original image

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. 6.3.3(a) and Fig. 6.3.3(c) we see that the second image has a "Checker board" pattern due to the reduction of samples.

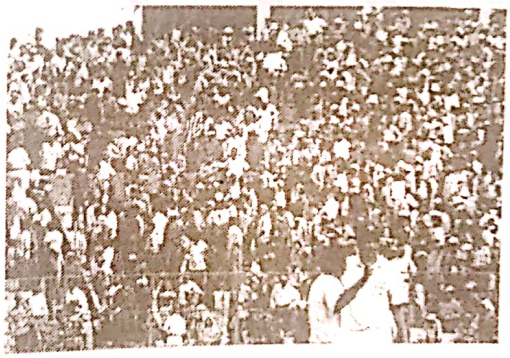
6.4 Isopreference Curves

- 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.

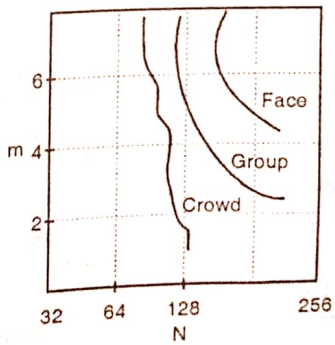


(a)

(b)



(c)



(d)

Fig. 6.4.1

6.5 Non-Uniform Sampling

- We have already seen that as N and m increase, the spatial and the grey level resolution increase. But due to this the storage space required increases enormously.
- To address this problem, we use non-uniform sampling. In this, regions which have a lot of details i.e., were there are abrupt changes in grey levels, a lot of samples are taken while regions where the grey levels reduce very slowly (background) only few samples are used.
- Hence non-uniform sampling can be used to reduce the storage space required without distorting the subjective quality of the image.
- We have all seen non-uniform sampling been used. Whenever a person's identity needs to be concealed on T.V., non-uniform sampling is used.
- Very few samples are taken near the eye regions because of which we see blocks near the person's eye and hence can not make out who that person is.

6.6 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.

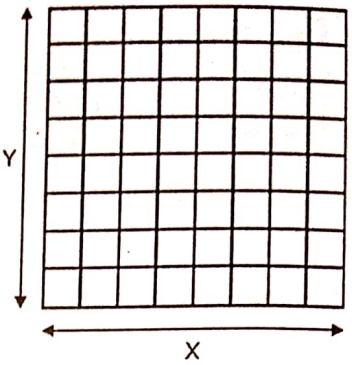


Fig. 6.6.1

- 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.

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

6.6.1 Solved Examples on Physical Resolution

Ex. 6.6.1 : 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. 6.6.2 : 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

$$\text{Aspect ratio is} = 1024/768 = 1.33$$

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. 6.6.3 : How much storage capacity is required to store an image with size of 1024 × 768 and 256 gray levels ?

Soln. :

Storage capacity required = $A \times B \times C$

Here $A \times B$ = Physical size of image = 1024×768

C = Number of bits required to get 256 gray levels, we must have 8 bits ($2^8 = 256$)

Storage capacity required = $1024 \times 768 \times 8 = 6291456$ bits

Ex. 6.6.4 : 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 are

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

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

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

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. 6.6.5 : 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

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

$\therefore$ time taken = 273.06 sec ≈ 4.5 min

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. Spatial resolution and grey level resolution are also explained using MATLAB codes.

Review Questions

Q. 1 Explain the concept of aliasing for 1-dimensional signals.

Q. 2 Explain the concept of aliasing for 2-dimensional signals.

Q. 3 How does one avoid aliasing ?

Q. 4 Explain sampling and quantization.

Q. 5 Explain the effects of reducing sampling and quantization.

Q. 6 Explain isopreference curves.

Q. 7 Explain non-uniform sampling.

□□□

CHAPTER 7

Image Enhancement in Spatial Domain

Syllabus :

Gray Level Transformations, Zero Memory Point Operations, Neighborhood Processing, Spatial Filtering, Smoothing and Sharpening Filters, Median Filter.

7.1 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.

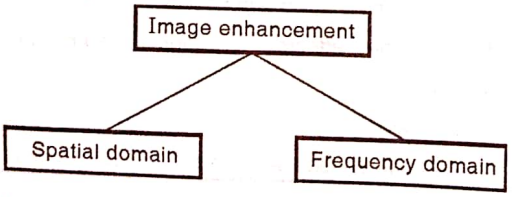


Fig. 7.1.1

7.2 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. 7.2.1.

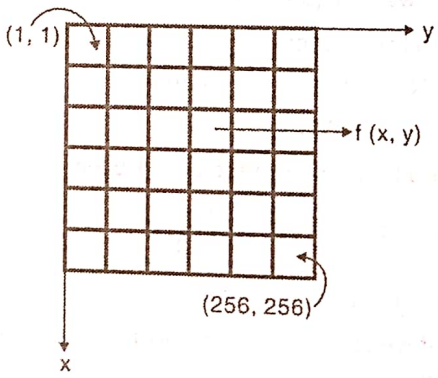


Fig. 7.2.1

Note : We always consider the axis as shown in Fig 7.2.1. 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,

$$g(x, y) = T[f(x, y)] \quad \ldots(7.2.1)$$

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 (7.2.1).
- Spatial domain enhancement can be carried out in two different ways :
  (1) Point processing
  (2) Neighbourhood processing

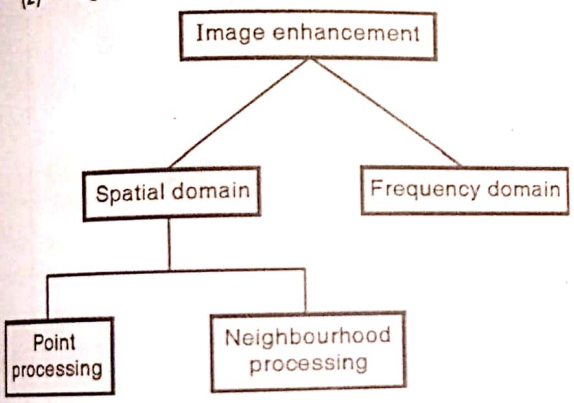


Fig. 7.2.2

7.3 Point Processing / Zero Memory Operation

MU - Dec. 2015, Dec 2016, May 2017, May 2018, Dec 2018

Q. What do you understand by zero memory operation. **(Dec. 2015, 5 Marks)**

Q. Explain any five zero memory operations. **(Dec. 2016, May 2017, 10 Marks)**

Q. Explain Thresholding. **(Dec. 2016, May 2017, 2 Marks)**

Q. Explain zero memory operations. **(May 2018, Dec. 2018, 5 Marks)**

- 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.
- Since this operation does not require past pixel values, we do not need additional memory allocation. Hence point processing is also known as zero memory operation. 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. 7.3.1.
- In the Fig. 7.3.1, 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. 7.3.1 will help us understand the point processing techniques better.

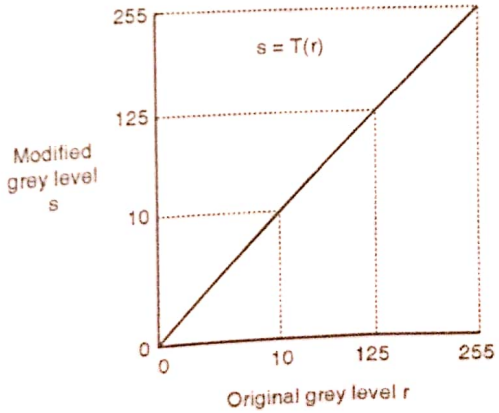


Fig. 7.3.1

(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. 7.3.2 is the digital negative transformation for a 8-bit image.

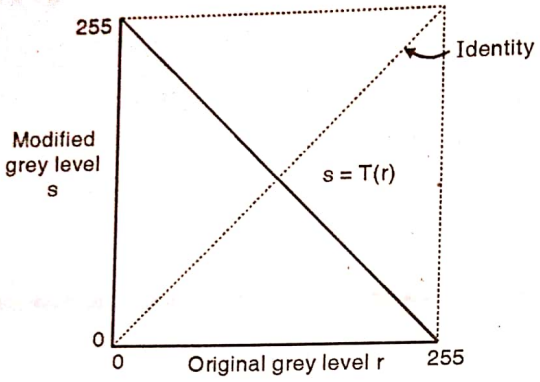


Fig. 7.3.2

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

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

Hence when

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

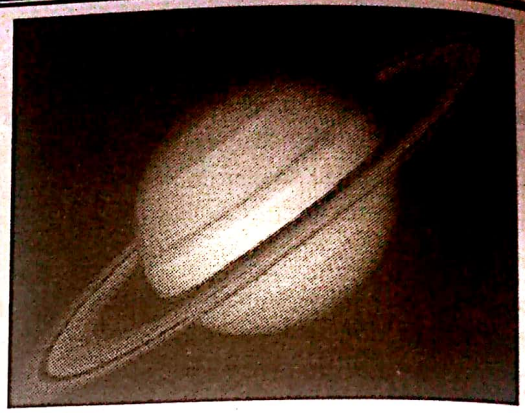
In general,

$$s = (L - 1) - r \qquad \ldots(7.3.1)$$

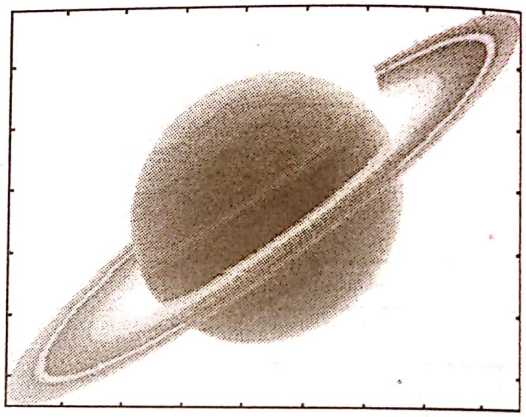
Here L is the number of grey levels. (256 in this case)

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



(b) Digital negative

Fig. 7.3.3

(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. 7.3.4 shows the transformation used to achieve contrast stretching. In the Fig. 7.3.4, the dotted line indicates the identity transformation and the solid line is the contrast stretching transformation.

– As is evident from the Fig. 7.3.4, 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.

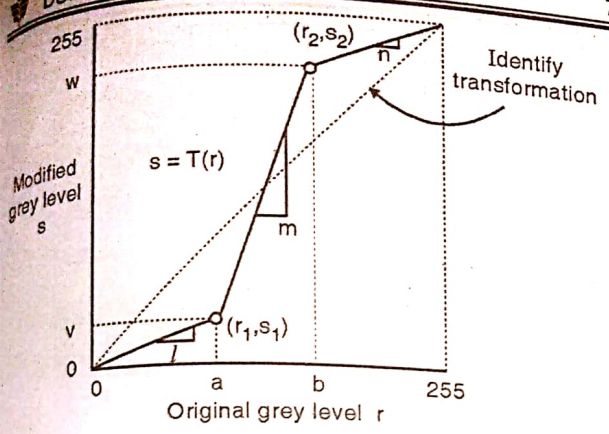


Fig. 7.3.4 : Original grey level r

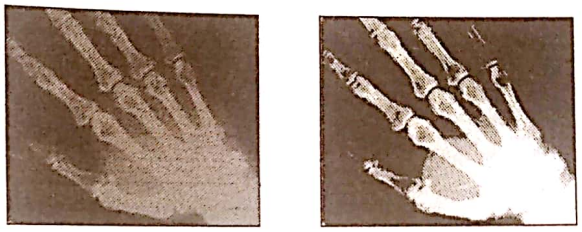
- The formulation of the contrast-stretching algorithm is given as follows :

$$s = \begin{cases} l \cdot r & 0 \le r < a \\ m \cdot (r-a) + v & a \le r < b \\ n \cdot (r-b) + w & b \le r < L-1 \end{cases} \quad \ldots(7.3.2)$$

where *l*, m and n are the slopes. It is clear from the Fig. 7.3.4 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
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. 7.3.5

(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. 7.3.6.
- The formula for achieving thresholding is as follows,

$$\left.\begin{array}{ll} s = 0; & \text{if } r \le a \\ s = L-1; & \text{if } r > a \end{array}\right\} \quad \ldots(7.3.3)$$

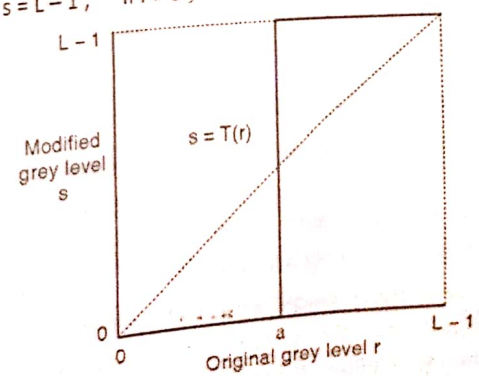


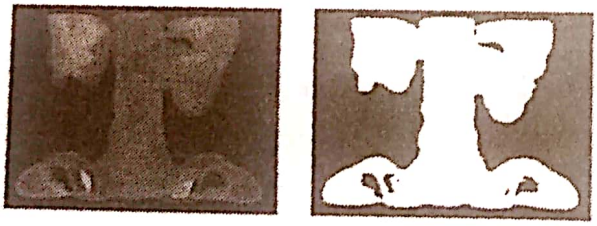
Fig. 7.3.6 : Original grey level r

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)
```

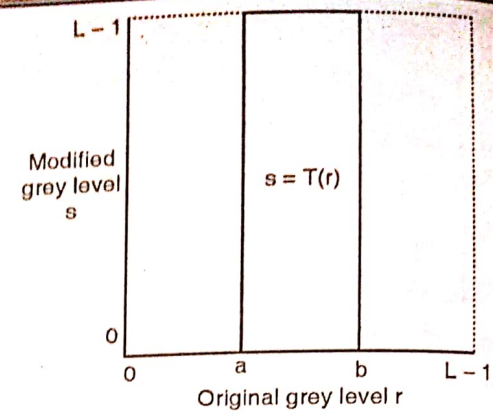


(a) Original image of spine (b) Image obtained using threshold

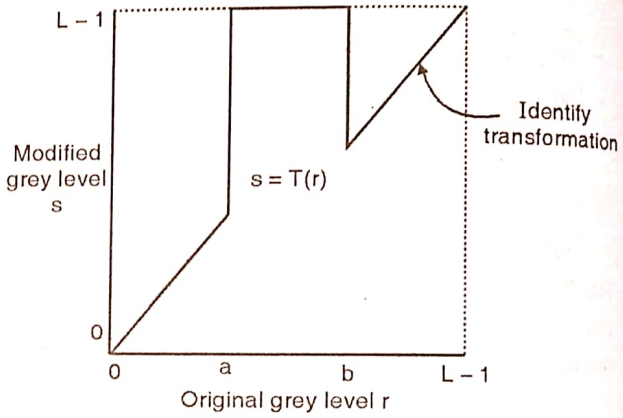
Fig. 7.3.7

(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. 7.3.8(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. 7.3.8

This can be implemented using the formulation

$$\left.\begin{array}{ll} s = L-1; & \text{if } a \le r \le b \\ s = 0; & \text{otherwise} \end{array}\right\} \quad \ldots(7.3.4)$$

- 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. 7.3.8(b).

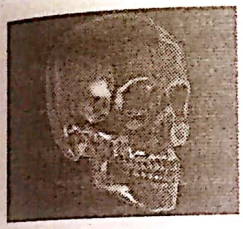
The formulation for this is

$$\left.\begin{array}{ll} s = L-1; & \text{if } a \le r \le b \\ s = r; & \text{otherwise} \end{array}\right\} \quad \ldots(7.3.5)$$

MATLAB program for grey level slicing with and without background is as follows :

```
%% 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))
```



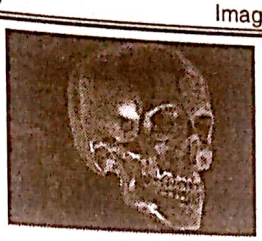
(a) Original image

(b) Gray level slicing without background

Fig. 7.3.9

MATLAB program

```
%% 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. 7.3.10

(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 $256 \times 256 \times 8$ 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.

Solved Example

Ex. 7.3.1 : 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
100	011	010
111	101	010

Binary image

1	0	0
0	1	0
1	1	0

LSB plane

0	1	0
0	1	1
1	0	1

Middle bit plane

0	0	0
1	0	0
1	1	0

MSB plane

MATLAB program

```
%% 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. P. 7.3.1 : 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 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 normalization constant.

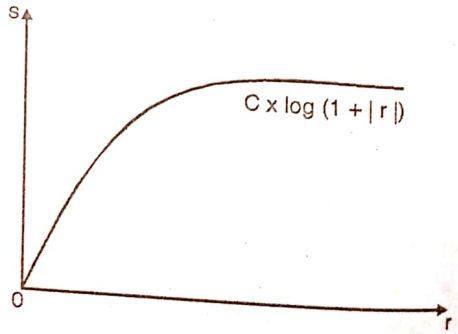
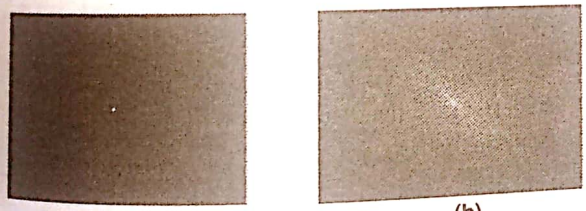


Fig. 7.3.11

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. 7.3.12 : 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 $s = c\, r^{\gamma}$ ...(7.3.6)

- Here c and γ are positive constants. The transformation is shown Fig. 7.3.13 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.

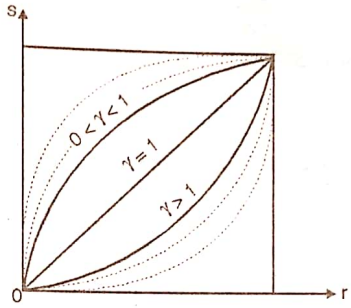


Fig. 7.3.13 : Gamma correction factor

```
%% 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. 7.3.14

7.4 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. 7.4.1. Instead of a 3 × 3 neighbourhood, we could also use a 5 × 5 or a 7 × 7..... neighbourhood.

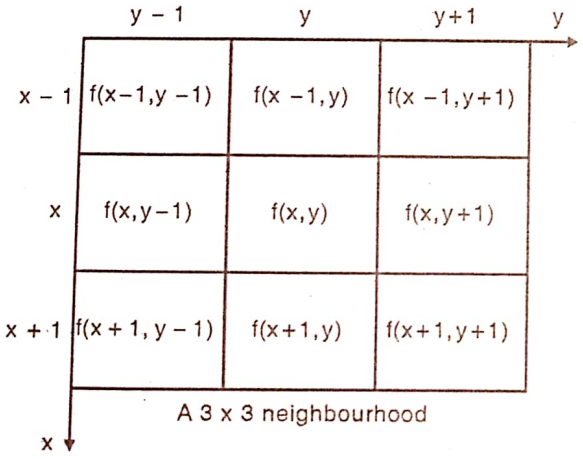


Fig. 7.4.1 : A 3 × 3 neighbourhood

w1	w2	w3
w4	w5	w6
w7	w8	w9

Fig. 7.4.2 : 3 × 3 mask

- There are a lot of things that can be achieved by neighbourhood processing which are not possible with point processing. Fig. 7.4.2 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,

$$g(x, y) = f(x-1, y-1) \times w1 + f(x-1, y) \times w2 + f(x-1, y+1) \times w3 + f(x, y-1) \times w4 + f(x, y) \times w5 + \ldots\ldots\ldots + f(x+1, y+1) \times w9 \quad \ldots(7.4.1)$$

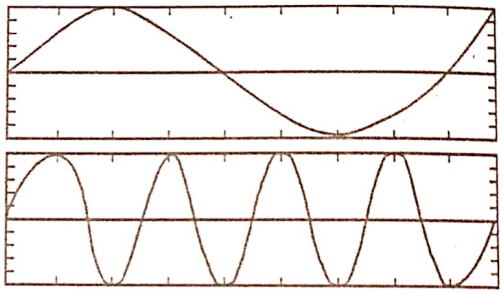


Fig. 7.4.3

- 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. 7.4.3.

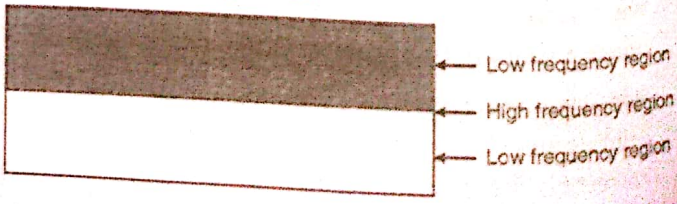


Fig. 7.4.4

- 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.

7.4.1 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.

7.4.2 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,

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

$z \rightarrow$ Grey level

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

$\sigma \rightarrow$ Standard deviation

$\sigma^2 \rightarrow$ Variance

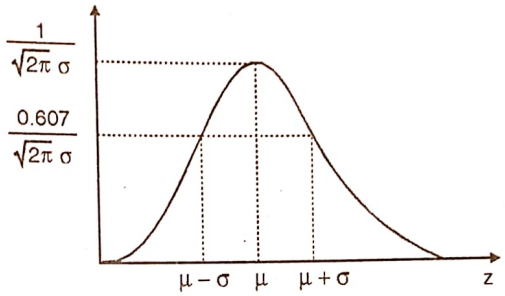


Fig. 7.4.5

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. 7.4.6.

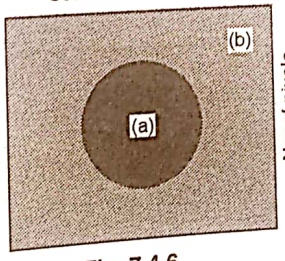


Fig. 7.4.6

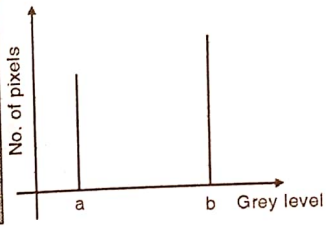


Fig. 7.4.7

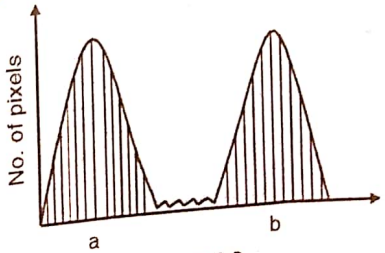


Fig. 7.4.8

There are two constant regions in the image. Hence, the histogram of the image is as shown in Fig. 7.4.7. If in this image, Gaussian noise creeps in, the histogram gets modified as shown in Fig. 7.4.8. Gaussian noise arise in an image due to factors such as circuit noise, sensor noise poor illumination and high temperature.

- 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.

7.4.1 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.

7.4.2 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,

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

$z \rightarrow$ Grey level

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

$\sigma \rightarrow$ Standard deviation

$\sigma^2 \rightarrow$ Variance

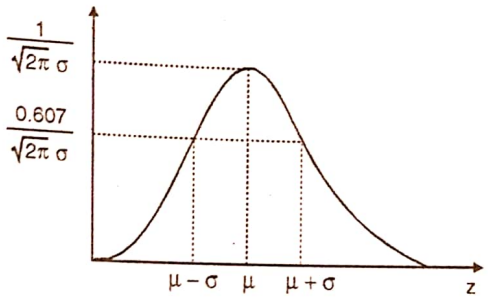


Fig. 7.4.5

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. 7.4.6.

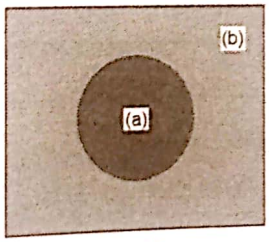


Fig. 7.4.6

Fig. 7.4.7

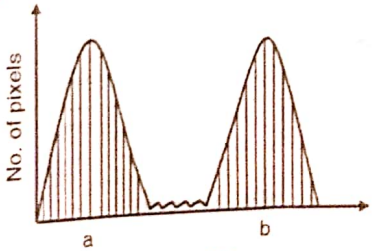


Fig. 7.4.8

There are two constant regions in the image. Hence, the histogram of the image is as shown in Fig. 7.4.7. If in this image, Gaussian noise creeps in, the histogram gets modified as shown in Fig. 7.4.8. Gaussian noise arise in an image due to factors such as circuit noise, sensor noise poor illumination and high temperature.

(2) 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. 7.4.9.

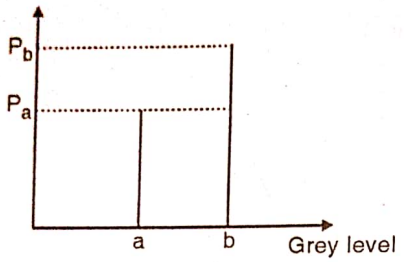


Fig. 7.4.9

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.

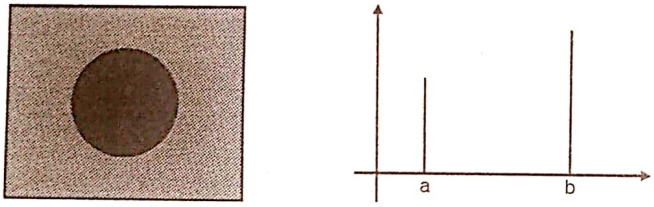


Fig. 7.4.10 (a)

When salt and pepper creeps in, the image looks like

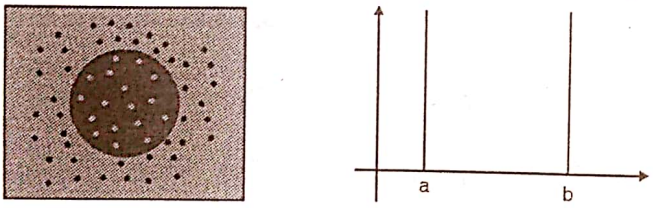


Fig. 7.4.10 (b)

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

7.4.3 Low Pass Averaging Filter

MU - May 2017

Q. Explain lowpass spatial filtering. **(May 2017, 5 Marks)**

- 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. 7.4.11.

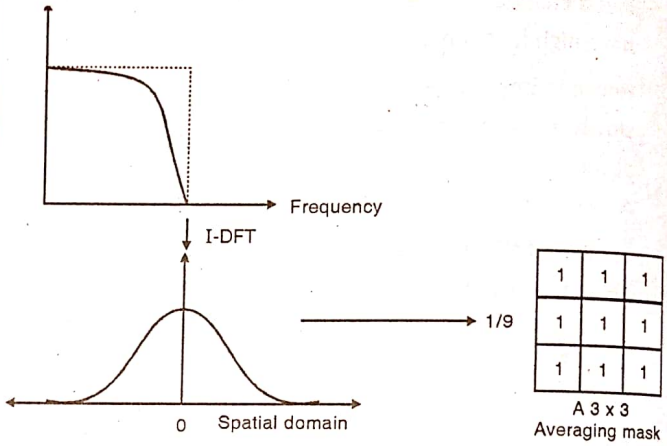


Fig. 7.4.11

- 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. 7.4.12

- 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

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

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

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

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. 7.4.13

Generalized 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(c))
title ('Low Pass Filtered Image using MATLAB');
```

7.4.4 Low Pass Median Filtering

MU - Dec. 2017

Q. Write short notes on : Median Filter. (Dec. 2017, 5 Marks)

- 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.
- 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.

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

Solved Example

Ex. 7.4.1 : 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

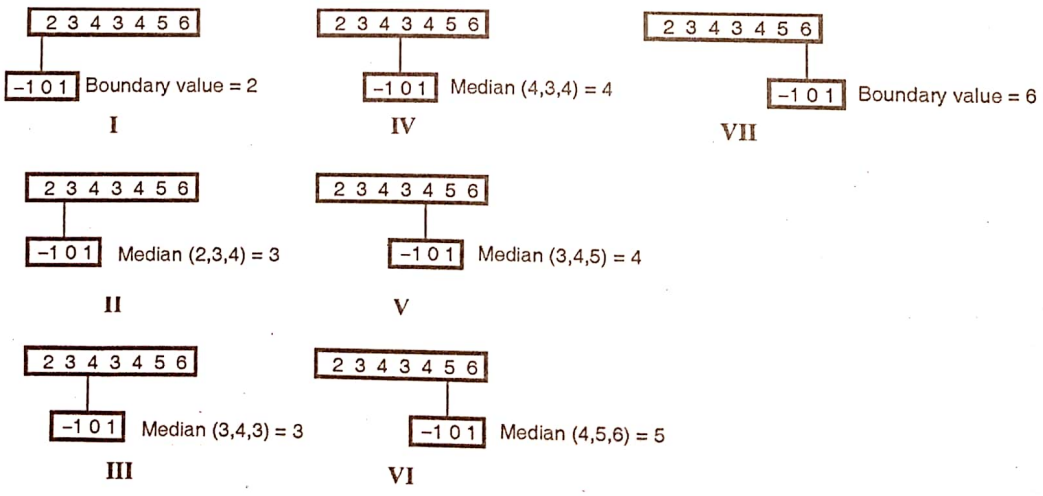


Fig. P. 7.4.1

```
%% 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

(b) Median filtered image and pepper noise

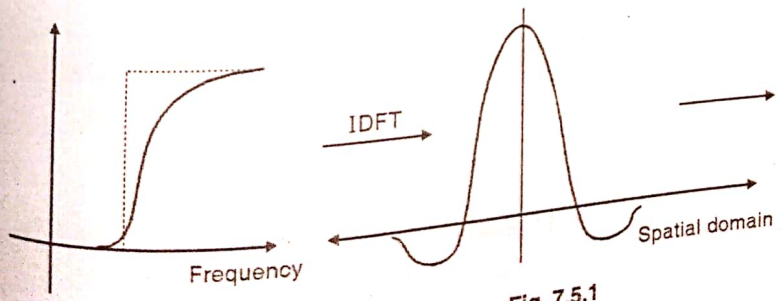
Fig. P. 7.4.1

7.5 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. 7.5.1.
- 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. 7.5.1.
- 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



−1	−1	−1
−1	8	−1
−1	−1	−1

A 3 x 3 high pass mask

Fig. 7.5.1

- 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 are as follows,

$\frac{1}{9}$

−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 as follows

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. 7.5.2

7.6 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 as follows,

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. 7.6.1

The generalized 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')
```

7.7 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.

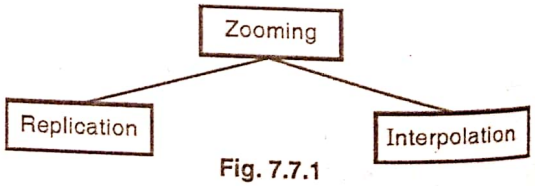


Fig. 7.7.1

7.7.1 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 as follows,

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.

Replication by zooming = Interlacing the rows → Interlacing the columns → Convolve Mask

- 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.

7.7.2 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

Interpolation 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 as follows :

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. 7.7.2

7.8 Solved Examples

Ex. 7.8.1 : 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

MU - May 2014, 10 Marks

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. 7.8.2 : 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

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

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

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. 7.8.3 : For following image find :
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

MU - May 2016, 10 Marks

Soln. :

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. 7.8.3.

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

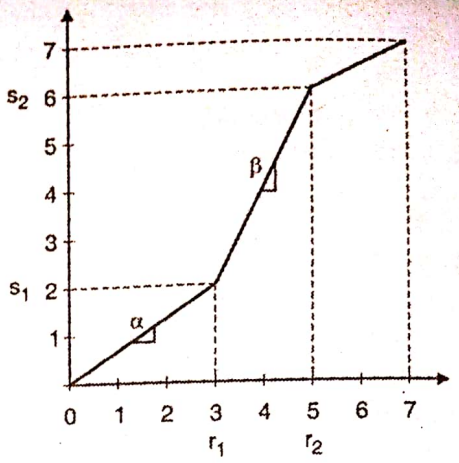


Fig. P. 7.8.3

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. 7.8.4 : 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

$g(x, y) =$

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

(ii) Thresholding with T = 4

The thresholding function is shown in Fig. P. 7.8.4 along with thresholding formula.

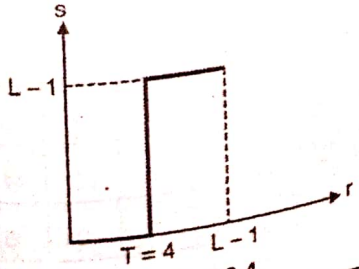


Fig. P. 7.8.4

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

Here

$$L - 1 = 7$$

∴ The thresholded image is as follows,

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 follows,

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. 7.8.4(a).

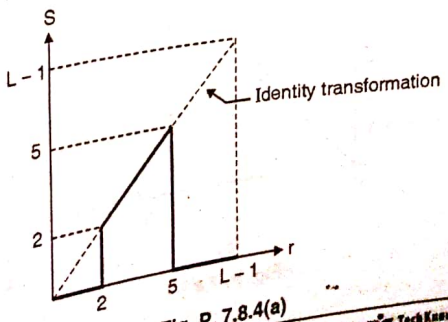


Fig. P. 7.8.4(a)

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

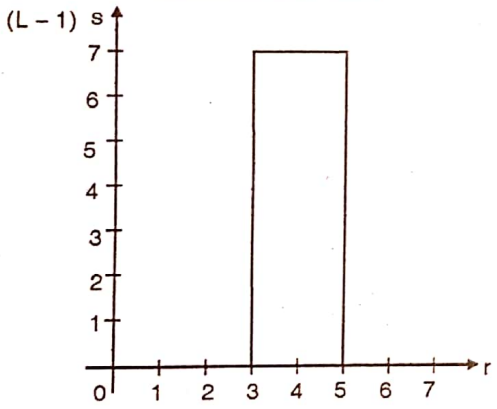
Hence the final clipped image is as follows,

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

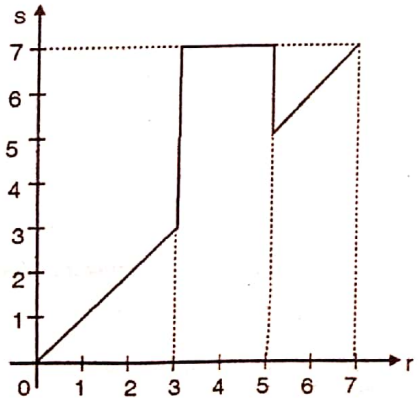
Ex. 7.8.5 : Perform intensity level (grey level) slicing on the 3 BPP image. Let $r_1 = 3$ and $r_2 = 5$. Draw the modified image using with background and without background transformations.

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)



(b)

Fig. P. 7.8.5

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

$s = 0 \quad$ 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

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

$s = r \quad$ 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. 7.8.6 : 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. 7.8.7 : 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 → Original number of grey levels

G → Modified number of grey levels

[.] → Choosing the floor value.

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

∴ 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. 7.8.8 : 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. 7.8.9 : 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.

$f(x, y) =$

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

We move the averaging mask over this image starting from the top left corner as follows,

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

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. 7.8.10 : 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.

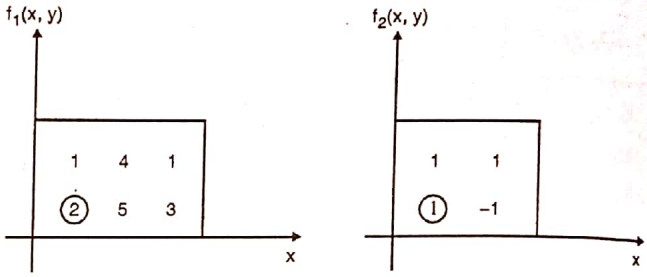


Fig. P. 7.8.10

Soln. :

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

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

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

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

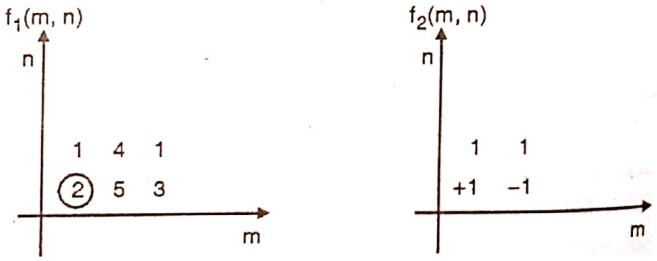


Fig. P. 7.8.10(a)

The basic shift operations are

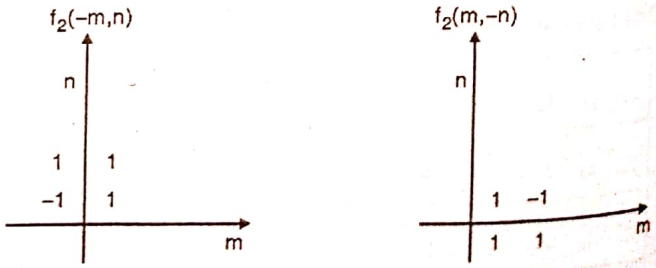
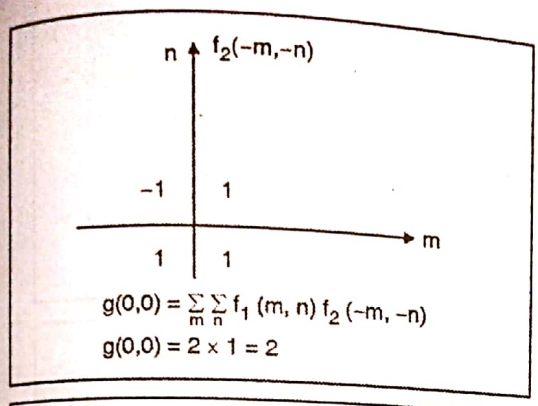


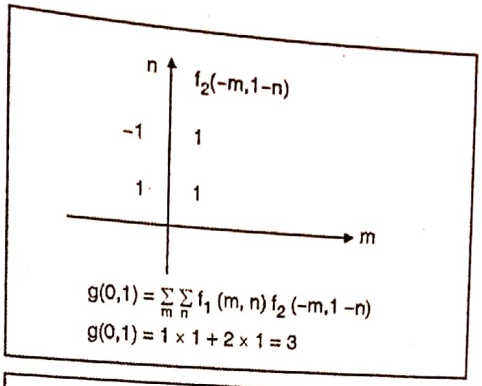
Fig. P. 7.8.10(b)

$f_1(-m, n) \Rightarrow$ fliping about m-axis $\quad f_2(m, -n) \Rightarrow$ fliping about n-axis



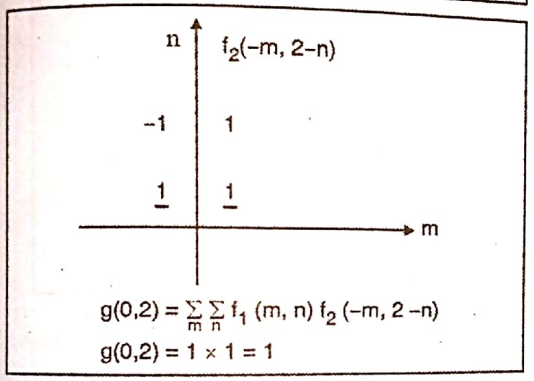
$g(0,0) = \sum_m \sum_n f_1(m, n)\, f_2(-m, -n)$

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



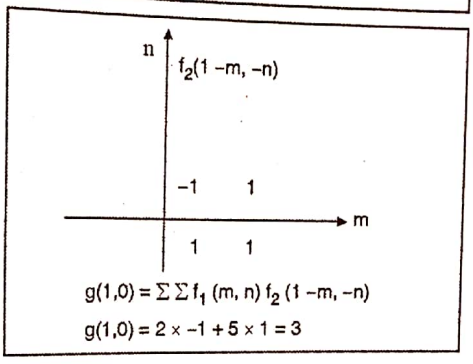
$g(0,1) = \sum_m \sum_n f_1(m, n)\, f_2(-m, 1-n)$

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



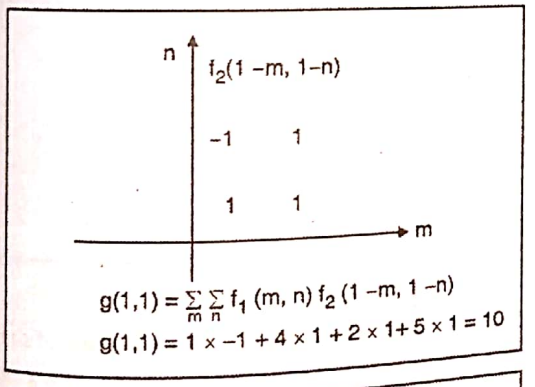
$g(0,2) = \sum_m \sum_n f_1(m, n)\, f_2(-m, 2-n)$

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



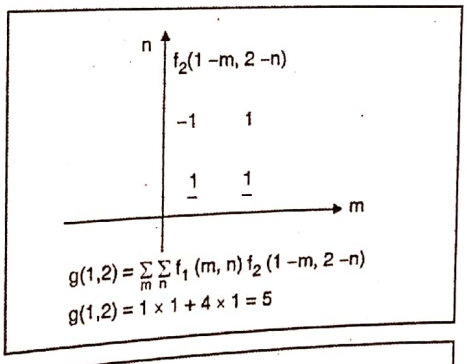
$g(1,0) = \sum \sum f_1(m, n)\, f_2(1-m, -n)$

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



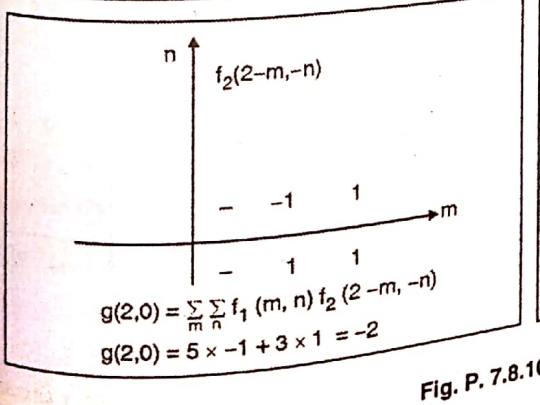
$g(1,1) = \sum_m \sum_n f_1(m, n)\, f_2(1-m, 1-n)$

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



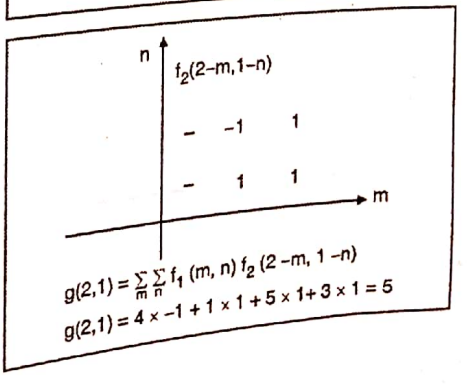
$g(1,2) = \sum_m \sum_n f_1(m, n)\, f_2(1-m, 2-n)$

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



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

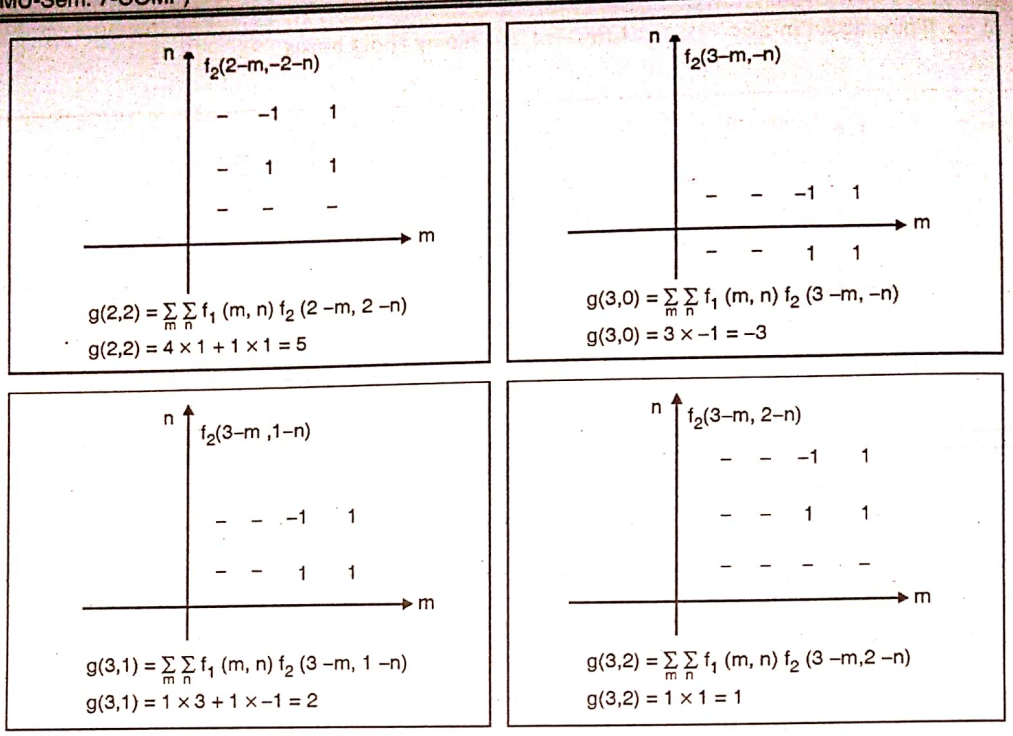
$g(2,0) = 5 \times -1 + 3 \times 1 = -2$



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

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

Fig. P. 7.8.10(c)cont...



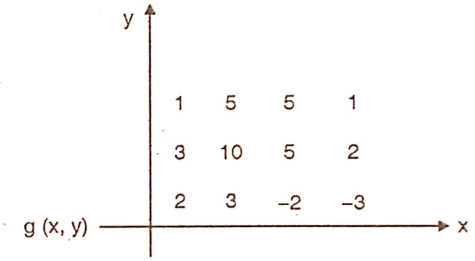


Fig. P. 7.8.10(c)

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. 7.8.11 : Perform discrete convolution on the given image arrays

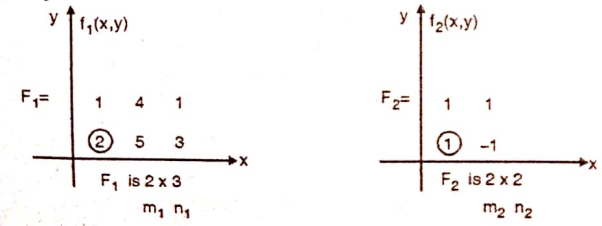


Fig. P. 7.8.11

Soln. :

Zero padding F_1 and F_2 so that both are of size

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

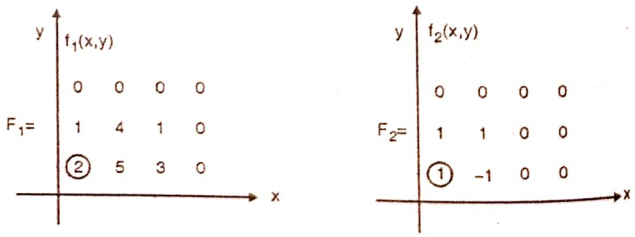


Fig. P. 7.8.11(a)

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

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

$$g_1(x,y) = f(x,y) * h_1(x,y) =$$

23.77	30.66	23.66
30.88	41.22	30.66
24.11	30.88	23.667

(ii) High Pass

High pass mask $h_2(x,y) = \frac{1}{9}$

−1	−1	−1
−1	8	−1
−1	−1	−1

$$g_2(x,y) = f(x,y) * h_2(x,y) =$$

6.22	0.33	8.33
2.11	78.77	−0.667
7.889	1.11	7.33

We now show that High pass = Original – Low pass

Original – Lowpass =

30	31	32
33	120	30
32	32	31

−

23.77	30.66	23.66
30.88	41.22	30.66
24.11	30.88	23.667

=

6.22	0.33	8.33
2.11	78.77	−0.667
7.889	1.11	7.33

This is the same as the High pass result. Therefore, High pass = Original – Low pass

Ex. 7.8.13 : 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.

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

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. 7.8.14 : 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

MU - Dec. 2011, 10 Marks

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. 7.8.14.

Here,

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

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

∴ The final image would be,

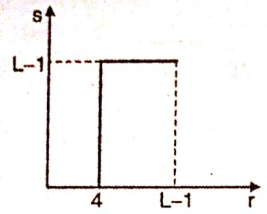


Fig. P. 7.8.14

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. 7.8.14(a).

Here,

$$s = L - 1 = 7 \qquad 2 \le r \le 5$$
$$s = r \qquad \text{otherwise}$$

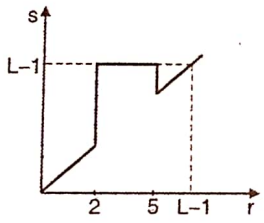


Fig. P. 7.8.14(a)

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 as follows,

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. 7.8.14(b).

Here,

$$s = (L - 1) - r$$
$$s = 7 - r$$

Here the grey levels get inverted.

∴ The final image is as shown in Fig, 7.8.14(b)

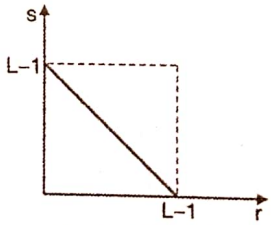
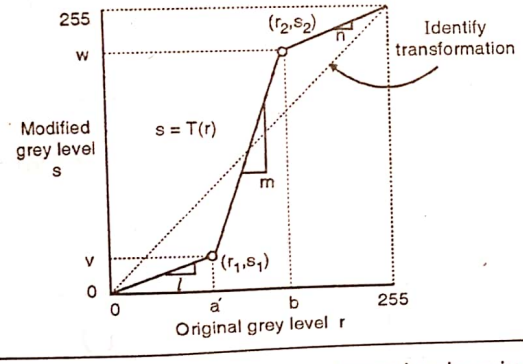
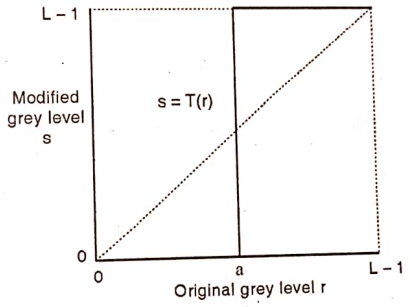


Fig. P. 7.8.14(b)

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

7.9 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 $r > a$
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 Explain spatial domain processing.

Q. 2 Compare and contrast average filtering and median filtering.

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

Q. 4 Develop a procedure for computing the median in $n \times n$ neighbourhood.

Q. 5 What do we mean by Gaussian noise and why is an averaging filter used to eliminate it ?