Unit-4: Image Restoration and Compression

Introduction

Image Restoration:

The purpose of image restoration is to "compensate for" or "undo" defects which degrade an image. Degradation comes in many forms such as motion blur, noise, and camera miss-focus. In cases like motion blur, it is possible to come up with an very good estimate of the actual blurring function and "undo" the blur to restore the original image. In cases where the image is corrupted by noise, the best we may hope to do is to compensate for the degradation it caused.

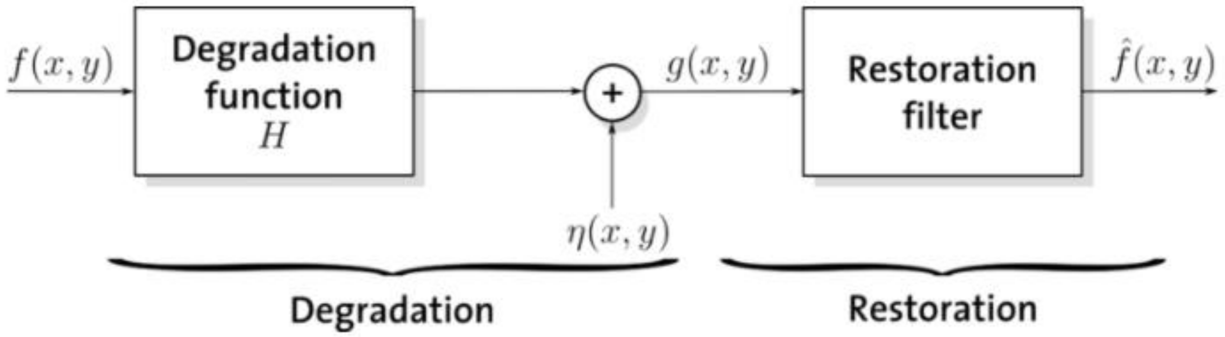
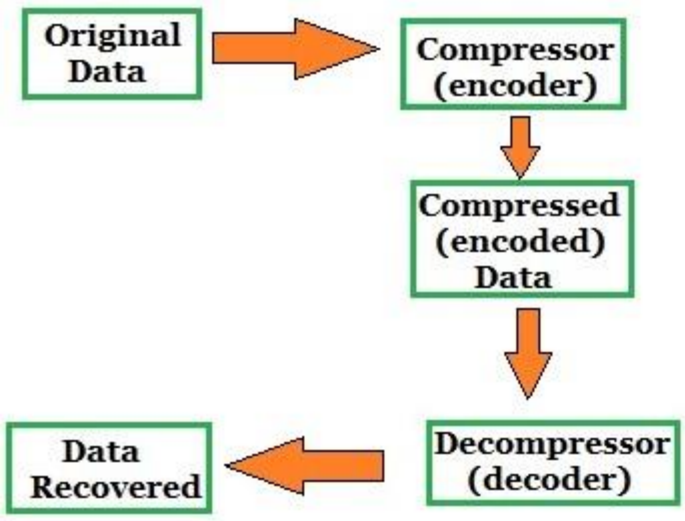


Image Compression:

Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.

There are several different ways in which image files can be compressed. For Internet use, the two most common compressed graphic image formats are the JPEG (*Joint Photographic Experts Group*) format and the GIF (*Graphics Interchange Format*) format. The JPEG method is more often used for photographs, while the GIF method is commonly used for line art and other images in which geometric shapes are relatively simple.



Models for Image Degradation and Restoration

Image restoration is the process of recovering an image that has been degraded by some knowledge of degradation function H and the additive noise term ***n(x,y).*** Thus in restoration, degradation is modeled and its inverse process is applied to recover the original image.

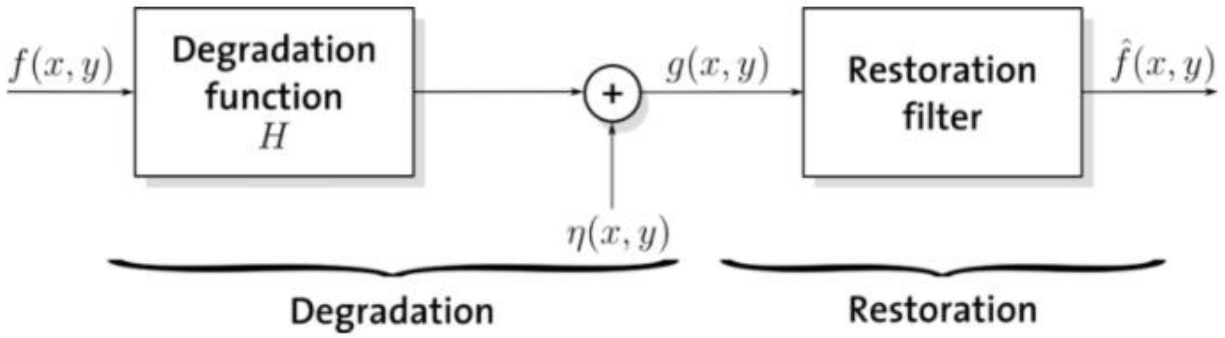


Fig: Image Degradation and Restoration Model

The input image ***f(x,y)*** is processed by degradation function ***H***, that together with an additive noise term ***n(x,y)*** to produce degraded image ***g(x,y).*** the ***g(x,y)*** consist of some knowledge about noise term ***n(x,y)*** and some knowledge about degradation function ***H***.

The objective of image restoration is to obtain an estimate of original image ***f(x,y).*** Here, by some knowledge of ***H*** and ***n(x,y),*** we find the appropriate restoration filters, so that output image ***f'(x,y)*** is as close as original image ***f(x,y).*** Since, it is practically not possible (*or very difficult*) to completely restore the original image.

In Spatial Domain Degraded image is represented by

G(x,y) = h(x,y) * f(x,y) + n(x,y)

Where,

h(x,y) = Degradation function

** = Indicates the convolution operation*

f(x,y) = Original Image

g(x,y) = Degraded image

n(x,y) = additive noise term

In Frequency Domain Degraded image is represented by

G(u,v) = H(u,v) * F(u,v) + N(u,v)

If the restoration filter applied is ***R(u,v),*** then,

F'(u,v) = R(u,v) [G(u,v)]

F'(u,v) = R(u,v) H(u,v) F(u,v) + R(u,v) N(u,v)

F'(u,v) $\approx$ F(u,v)

Where, F'(u,v) is the restored image

Here, restoration filter ***R(u,v)*** is the reverse of degradation function ***H(u,v).***

Noise Models

Spatial noise is described by the statistical behavior of the gray-level values in the noise component of the degraded image. Noise can be modeled as a random variable with a specific probability distribution function (PDF). Important examples of noise models include:

1. Gaussian Noise
2. Rayleigh Noise
3. Impulse (Salt & Pepper) Noise
4. Erlang (Gamma) Noise
5. Uniform Noise
6. Exponential Noise

grow sani gamagam u le eat

Gaussian Noise:

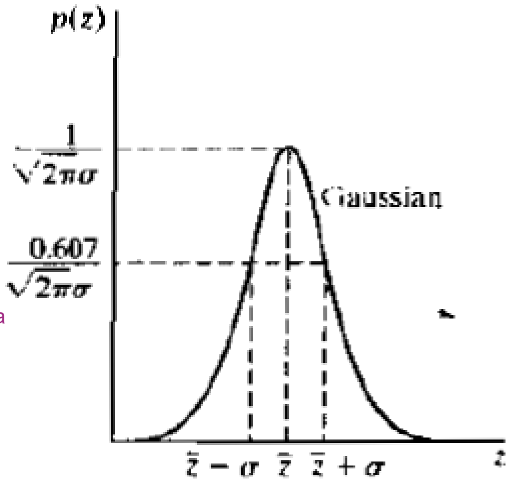
The PDF of Gaussian Noise is

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

z=zbar

z= zbar +sigma

tyo mathi formula ma rakhyo vne niskinxa

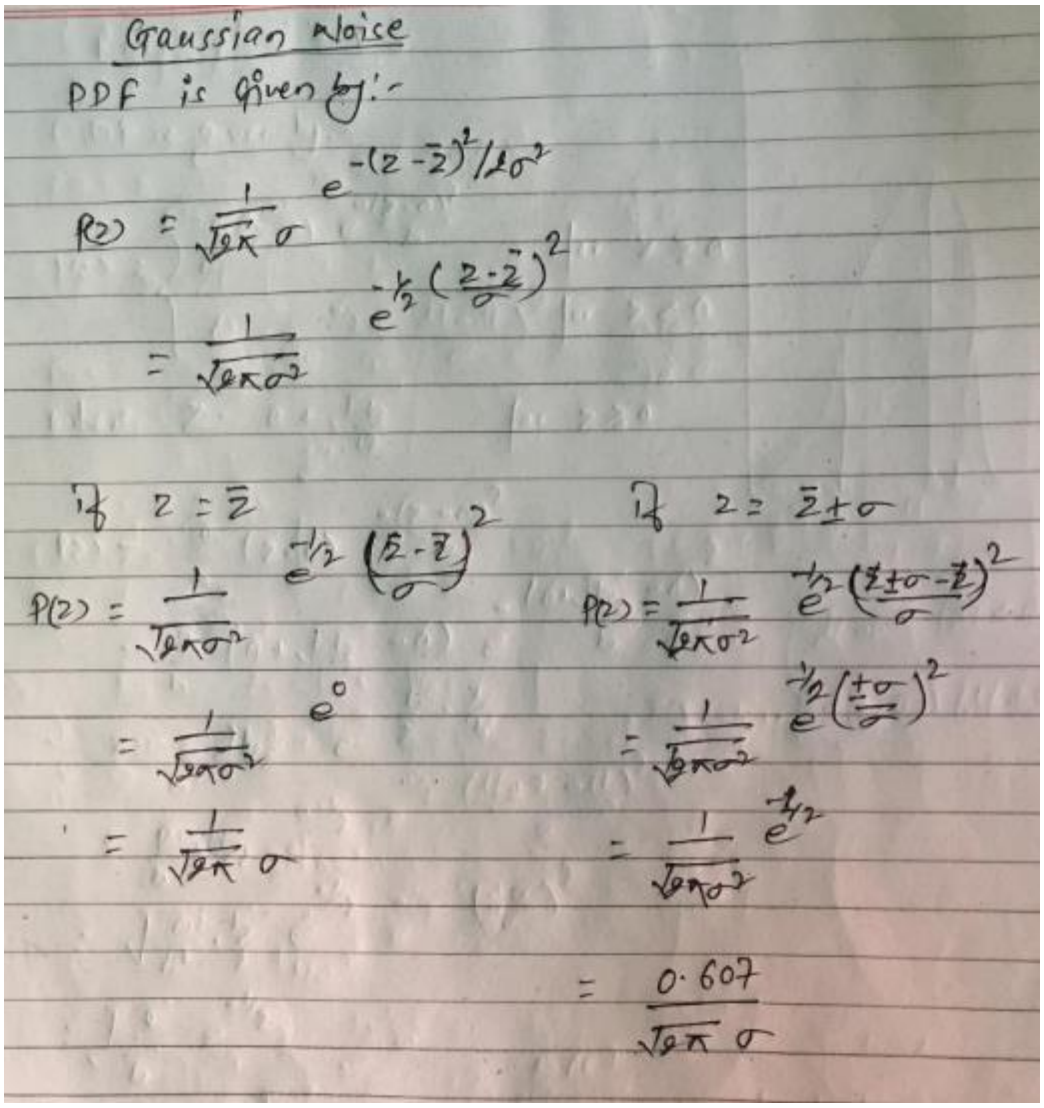


*Where, **z** is the gray value, $\bar{z}$ is the mean and σ is the standard deviation.*

The square of standard deviation σ^2 is called variance.

Approximately 70% of its value lies in range [$\bar{z} - \sigma$, $\bar{z} + \sigma$]

And 95% of its value lies in range [$(\bar{z} - 2\sigma), (\bar{z} + 2\sigma)$]



Gaussian Noise

PDF is given by:-

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

$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{z-\bar{z}}{\sigma}\right)^2}$$

if $z = \bar{z}$

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{\bar{z}-\bar{z}}{\sigma}\right)^2}$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{0}$$

$$= \frac{1}{\sqrt{2\pi}\,\sigma}$$

if $z = \bar{z} \pm \sigma$

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{\bar{z}\pm\sigma-\bar{z}}{\sigma}\right)^2}$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{\pm\sigma}{\sigma}\right)^2}$$

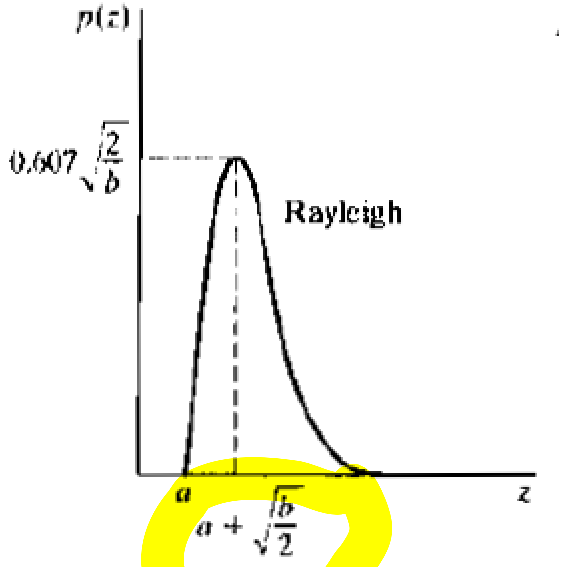
$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-1/2}$$

$$= \frac{0.607}{\sqrt{2\pi}\,\sigma}$$

Rayleigh Noise:

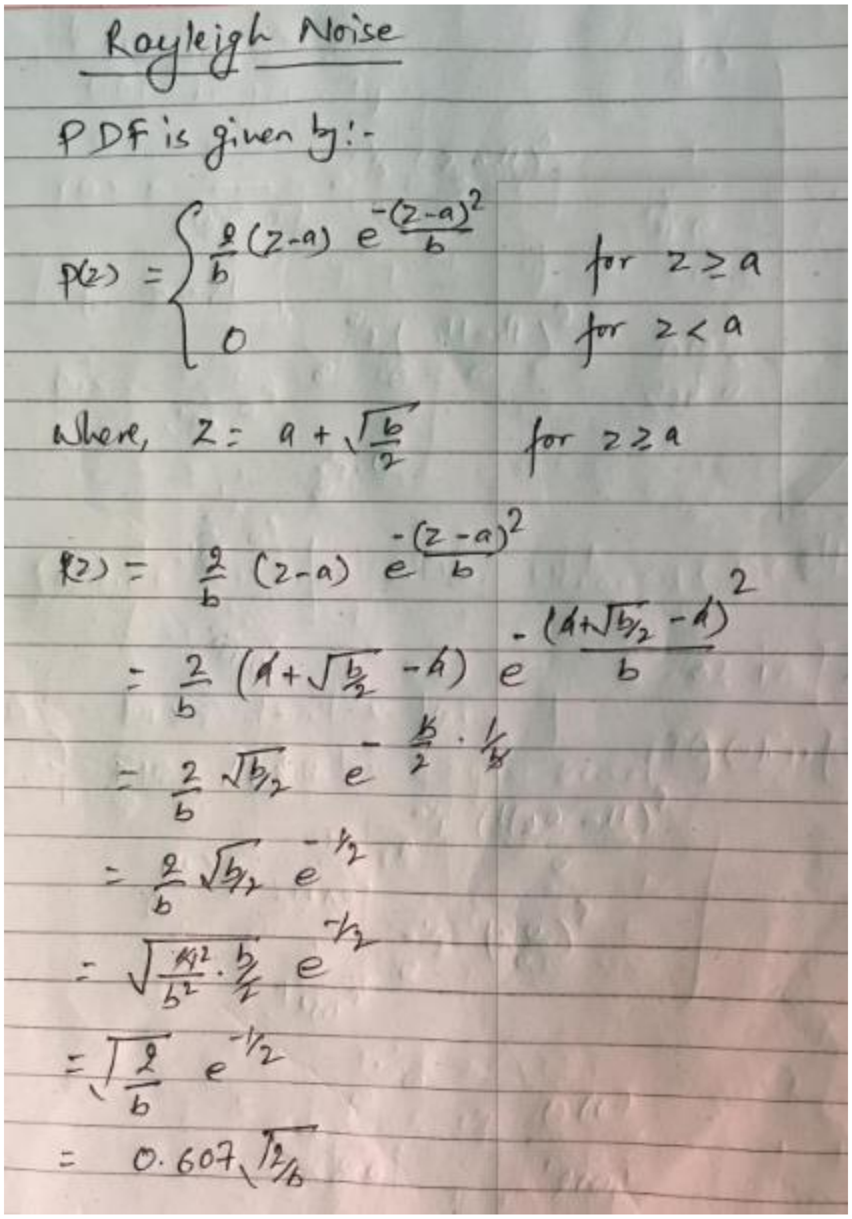
The PDF of Rayleigh Noise is

$$p(z) = \begin{cases} (2/b)/(z-a)e^{-(z-a)^2/b} & for\ z \geq a \\ 0 & for\ z < a \end{cases}$$



The mean and variance of this density is given by

$$\bar{z} = a + \sqrt{\pi b/4} \qquad \sigma^2 = \frac{b(4-\pi)}{4}$$



Rayleigh Noise

PDF is given by:-

$$p(z) = \begin{cases} \frac{2}{b}(z-a)\, e^{\frac{-(z-a)^2}{b}} & \text{for } z \geq a \\ 0 & \text{for } z < a \end{cases}$$

Where, $z = a + \sqrt{\frac{b}{2}}$ for $z \geq a$

$$p(z) = \frac{2}{b}(z-a)\, e^{\frac{-(z-a)^2}{b}}$$

$$= \frac{2}{b}\left(\cancel{a} + \sqrt{\tfrac{b}{2}} - \cancel{a}\right) e^{-\frac{(\cancel{a}+\sqrt{b/2}-\cancel{a})^2}{b}}$$

$$= \frac{2}{b}\sqrt{b/2}\; e^{-\frac{\cancel{b}}{2}\cdot\frac{1}{\cancel{b}}}$$

$$= \frac{2}{b}\sqrt{b/2}\; e^{-1/2}$$

$$= \sqrt{\frac{\cancel{4}^{2}}{b^{\cancel{2}}}\cdot\frac{\cancel{b}}{\cancel{2}}}\; e^{-1/2}$$

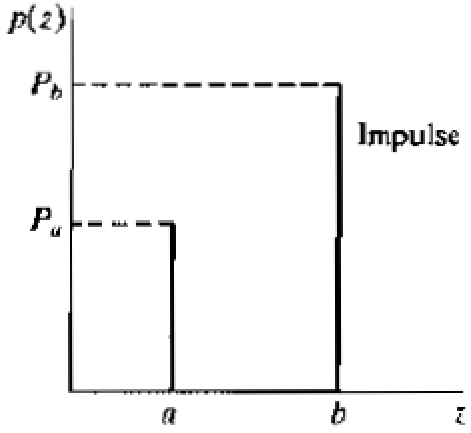
$$= \sqrt{\frac{2}{b}}\; e^{-1/2}$$

$$= 0.607\sqrt{2/b}$$

Impulse (Salt and Pepper) Noise:

The PDF of Impulse Noise is

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

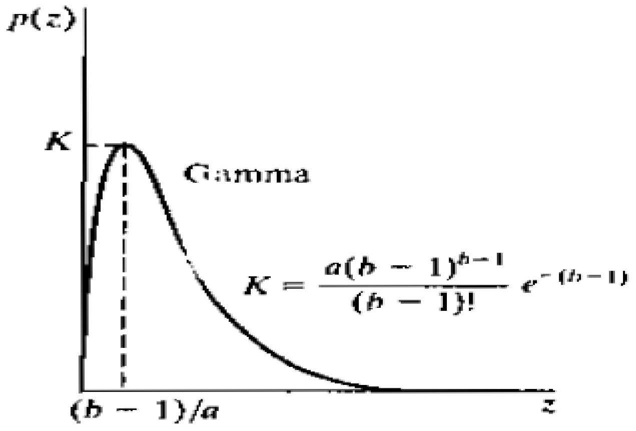


If (***b*** > ***a***), then gray level ‘***b’*** appears as a light dot (*salt*), otherwise the gray level ‘***a’*** appears as a dark dot (*pepper*).

Erlang (Gamma) Noise

The PDF of Erlang Noise is

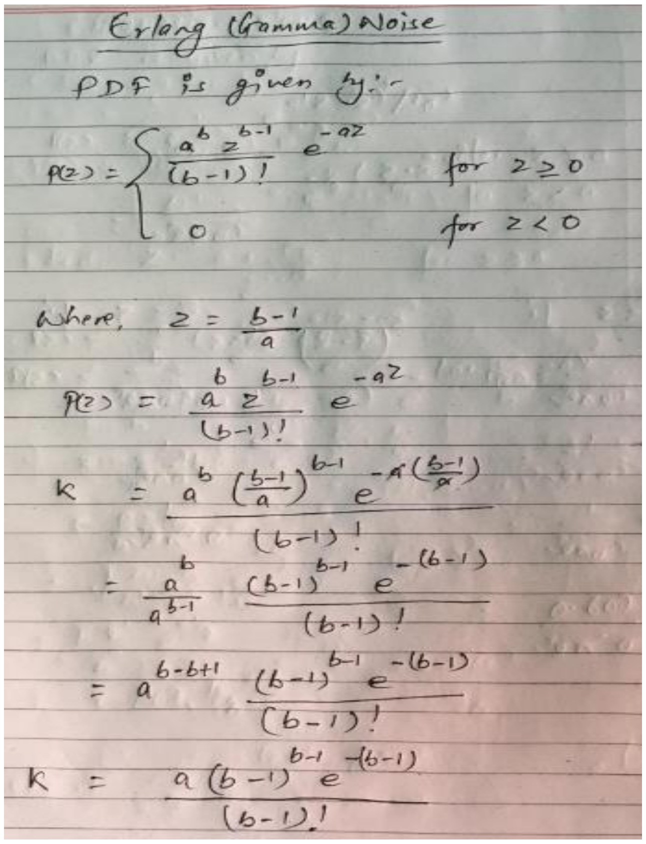
$$p(z) = \begin{cases} \dfrac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$



This graph is correct only when the demonstration is the gamma function.

Where the parameters are (***a*>*0***), ‘***b’*** is positive integer and ***‘!’*** is factorial. The mean and variance of this density is given by

$$\bar{z} = \frac{b}{a} \qquad \sigma^2 = \frac{b}{a^2}$$



Erlang (Gamma) Noise

PDF is given by:-

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$

Where, $z = \frac{b-1}{a}$

$$p(z) = \frac{a^b z^{b-1}}{(b-1)!} e^{-az}$$

$$K = \frac{a^b \left(\frac{b-1}{a}\right)^{b-1} e^{-a\left(\frac{b-1}{a}\right)}}{(b-1)!}$$

$$= \frac{a^b}{a^{b-1}} \frac{(b-1)^{b-1} e^{-(b-1)}}{(b-1)!}$$

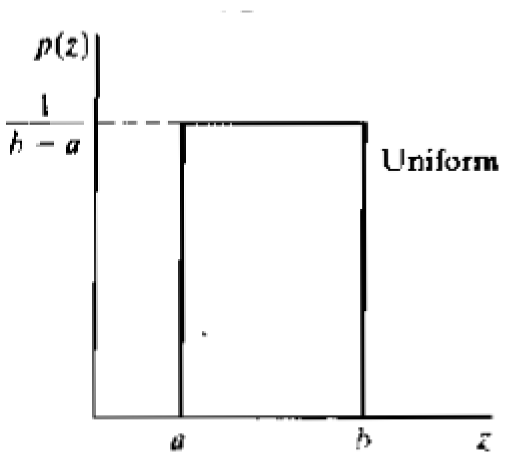
$$= a^{b-b+1} \frac{(b-1)^{b-1} e^{-(b-1)}}{(b-1)!}$$

$$K = \frac{a(b-1)^{b-1} e^{-(b-1)}}{(b-1)!}$$

Uniform Noise:

The PDF of Uniform Noise is

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \le z \le b \\ 0 & \text{otherwise} \end{cases}$$



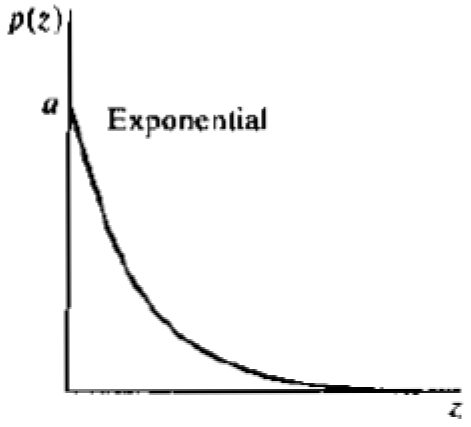
The mean and variance of this density is given by

$$\bar{z} = \frac{a+b}{2} \qquad \sigma^2 = \frac{(b-a)^2}{12}$$

Exponential Noise:

The PDF of Exponential Noise is

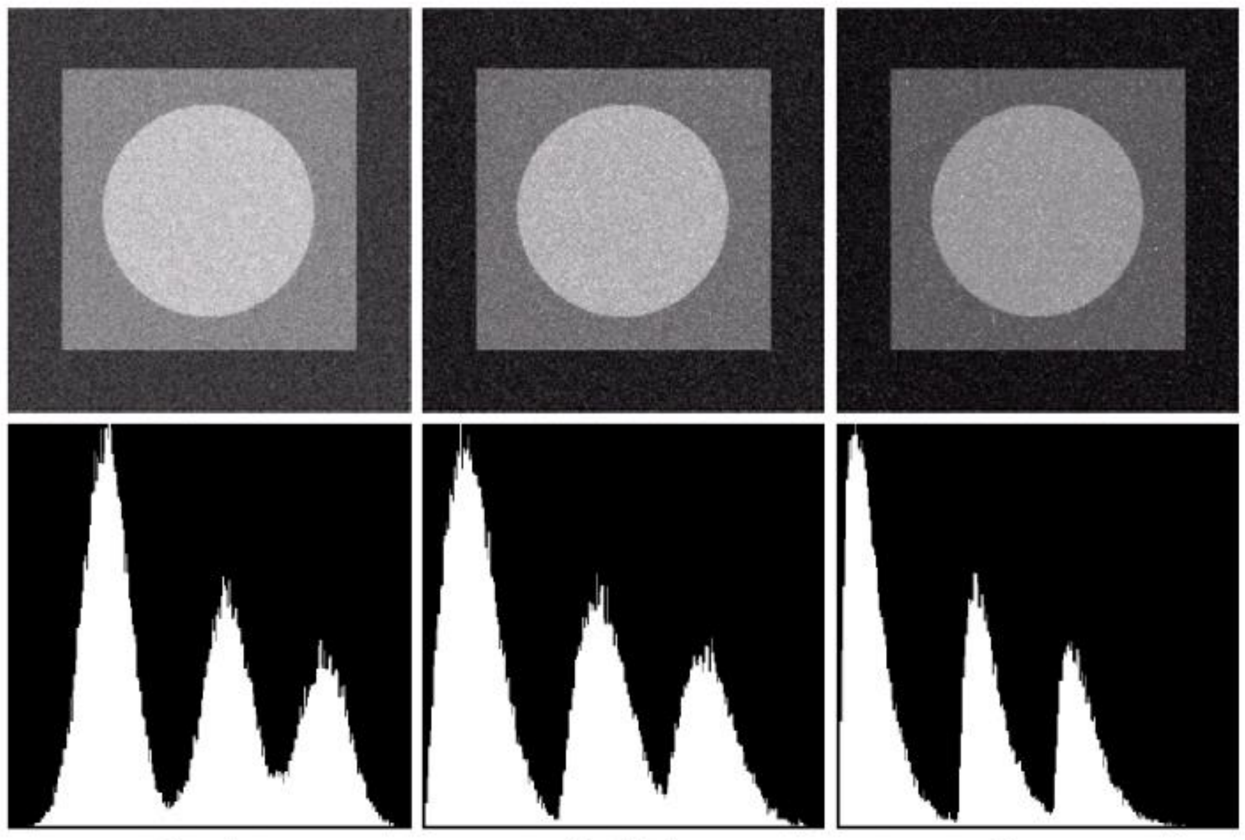
$$p(z) = \begin{cases} ae^{-az} & \text{for } z \ge 0 \\ 0 & \text{for } z < 0 \end{cases}$$

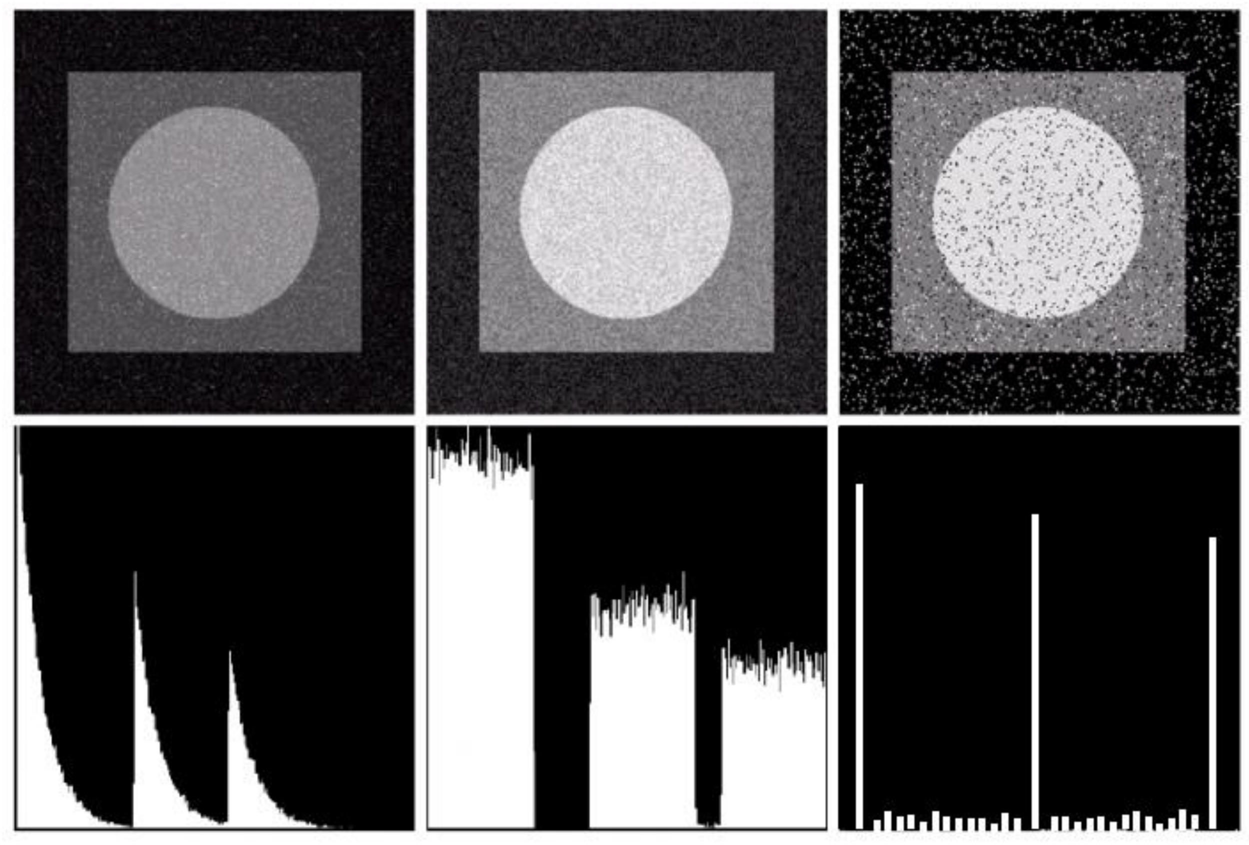


Where, (***a>0***), the mean and variance of this density is given by

$$\bar{z} = \frac{1}{a} \qquad \sigma^2 = \frac{1}{a^2}$$

Following fighures shows the example of above noise models





Estimation of Noise Parameters

Periodic noise in an image arises typically from electrical or electromechanical interference during image acquisition. The periodic noise can be reduced significantly via frequency domain filtering.

The parameters of periodic noise typically are estimated by inspection of the Fourier spectrum (*shape of the signal in frequency domain*) of the image. The periodic noise tends to produce frequency spikes (*projection*) that often can be detected even by visual analysis.

Another approach is to attempt to inter the periodicity of noise components directly from the image but this is possible only in simplistic cases. Automated analysis is possible in situation in which the noise spikes are either exceptionally pronounced or when knowledge is available about the general location of the frequency components of the interference.

The parameter of noise PDF (*probability distribution function*) may be known partially from sensor specifications but it is often necessary to estimate them for a particular imaging arrangement. If the imaging system is available, one simple way to study the characteristics of system noise is to capture a set of image of flat environment. For example, in the case of an optical sensor this is as simple as imaging a solid gray board that is illuminated uniformly. The resulting images are good indicators of system noise.

The simplest use of the data from the image strips is for calculating the mean and variance of intensity levels.

Consider a strip (*sub-image*) denoted by ***S***, and let ***Ps* (*z*$_i$)**, *i = 0,1,2,.....L-1*, denote the probability estimates (*normalized histogram values*) of the intensities of the pixels in ***S***, where, ***L*** is the number of possible intensities in the entire image. The estimate mean and variance of the pixels in ***S*** as flows:

$$\bar{z} = \sum_{i=0}^{L-1} z_i p_S(z_i)$$

$$\sigma^2 = \sum_{i=0}^{L-1} (z_i - \bar{z})^2 p_S(z_i)$$

The shape of the histogram identifies the closest PDF match. If the shape is approximately Gaussian, then the mean and variance are needed because the Gaussian PDF is completely specified by these two parameters. Impulse noise is handled differently because the estimate needed is of the actual probability of occurrence of white and black pixels. To obtain this estimate requires that both black and white pixels be visible, so a mid-gray, relatively constant area is needed in the image in order to be able to compute a histogram.

Restoration Filters

Restoration filters are used to restore the degraded image into original form. There are many restoration filters given below:

Mean Filters:

Arithmetic mean filter:

This is the simplest of the mean filters, S_{xy} represent the set of coordinates in a rectangular sub-image window (*neighborhood*) of size ***m*n***, centered at point ***(x,y).***

The arithmetic mean filter computes the average value of the corrupted image ***g(x,y)*** in the area defined by S_{xy}. The value of the restored image ***f'*** at point ***(x,y)*** is simply the arithmetic mean computed using the pixels in the region defined by S_{xy}. A mean filter smooth local variations in an image, and noise is reduced as a result of blurring.

Arithmetic mean filter can define by an expression

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

Where, ***m*** and ***n*** are the height and width of S_{xy}

Mask of arithmetic mean filter do convolution to image with this mask to filter the image

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Geometric mean filter

A geometric mean filter achieves smoothing comparable to the arithmetic mean filter, but it tends to loss less image detail in the process.

Geometric mean filter can define by an expression

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

Here, each restored pixel is given by the product of the pixels in the sub-image window, raised to the power ***1/mn***.

Example: apply geometric mean filter to given image

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

Solution:

$$(1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9)^{1/9}$$

Result will replace the central pixel of the given image.

Harmonic mean filter

It works well for salt noise, but fails for pepper noise. It also works well for Gaussian noise.

Harmonic mean filter can define by an expression

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

Example: apply harmonic mean filter to given image

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

Solution:

$$\frac{9}{\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \cdots \frac{1}{9}\right)}$$

Result will replace the central pixel of the given image.

Contraharmonic mean filter

It is well suited for reducing or visually eliminating the effect of salt and pepper noise.

Contraharmonic mean filter can define by an expression

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

Where, Q is called the order of the filter

For positive value of ***Q***, i.e. ***Q>0***, the filter eliminates pepper noise. For negative value of ***Q***, i.e. ***Q<0***, the filter eliminates salt noise. It can't do both simultaneously.

The contraharmonic mean filter is reduces to the arithmetic mean filter if ***Q = 0***, and to the harmonic mean filter if ***Q = -1***.

Usages of filter:

- Arithmetic and geometric filters are suited for gaussian and uniform noise
- Harmonic filter is suited for salt and gaussioan noise
- Contraharmonic filter is suited for impulse noise.

Order Statistics Filters:

Spatial filters whose response is based on ordering (*ranking*) the values of the pixels contained in the image area encompassed by the filter is called order statics filters.

Commonly used order statistics filters include

- Median filter
- Max and min filter
- Midpoint filter
- Alpha trimmed mean filter

Median filter

The best known order statistic filter is the median filter, which replaces the value of the central pixel by the median of the intensity levels in the neighborhood pixel.

To find the median value we have to sort the pixels of given image in ascending order first.

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

The value of the pixel at (x,y) is included in the computation of the median.

For certain types of random noise it provides excellent noise reduction capabilities with considerably less blurring than linear smoothing filter of the same size. It is particularly good when salt and pepper noise is present.

Max filter and Min filter

Max filter is good for pepper noise and Min filter is good for salt noise.

Max filter equation is

$$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}}\{g(s,t)\}$$

Min filter equation is

$$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}}\{g(s,t)\}$$

Midpoint filter

It simply computes the midpoint between the maximum and minimum values in the area encompassed by the filter. It is good for Gaussian noise and uniform noise.

$$\hat{f}(x,y)=\frac{1}{2}\left[\max_{(s,t)\in S_{xy}}\{g(s,t)\}+\min_{(s,t)\in S_{xy}}\{g(s,t)\}\right]$$

Alpha-Trimmed Mean Filter

It is useful in situation involving multiple types of noise, such as a combination of salt-and-pepper noise and Gaussian noise.

Suppose, we delete the ***d/2*** lowest and ***d/2*** highest intensity values of ***g(s,t)*** in the neighborhood of S_{xy}. Let g_r ***(s,t)*** represents the remaining ***(mn – d)*** pixels, the filter formed by averaging these remaining pixels is called an Alpha-Trimmed mean filer and represented by expression

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

Where, the value of ***d*** can range from ***0*** to ***mn-1***, when ***d = 0***, the alpha-trimmed filter reduces to the arithmetic mean filter and if ***d = mn-1***, filter becomes median filter.

Type text here

Band Reject Filters:

This filter removes the components within a specific range.

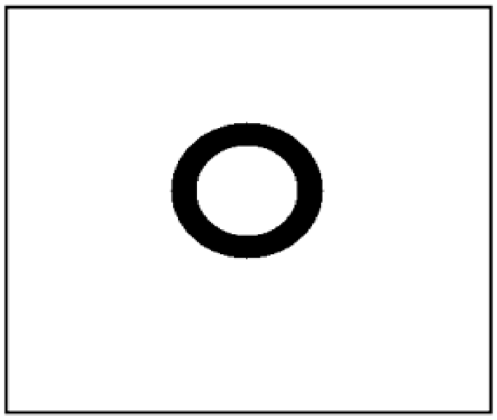
Ideal Band Reject Filter

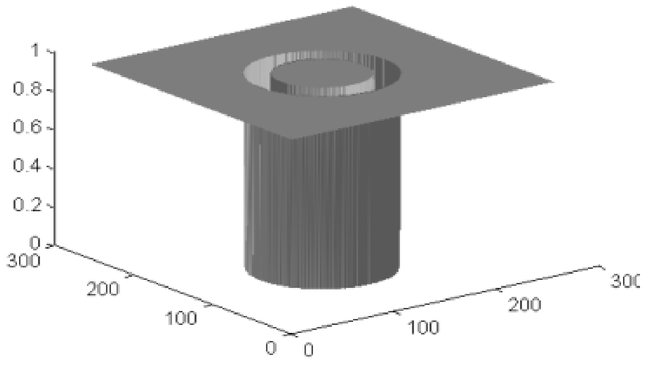
$$H(u,v)=\begin{cases}0 & \text{if } D_0-\frac{W}{2}\le D(u,v)\le D_0+\frac{W}{2}\\ 1 & \text{otherwise}\end{cases}$$

ideal low pass filter sanga milxa teha ni D0 ko value vnda D(u,v) ko value thulo vye 0 hunxa

Where,

- ***D(u,v)*** *is the distance of pixel* ***(u,v)*** *to the frequency center* ***(0,0)*** *and can be calculated by formula:* $D(u,v) = [u^2 + v^2]^{1/2}$
- ***W*** *is the width of the band (range)*
- D_0 *is the radial center*





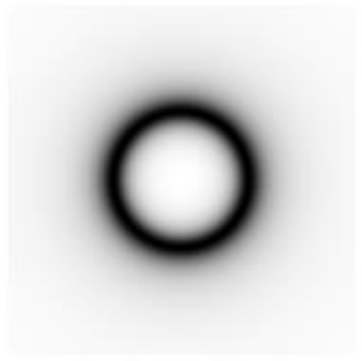
Butterworth Band Reject Filter

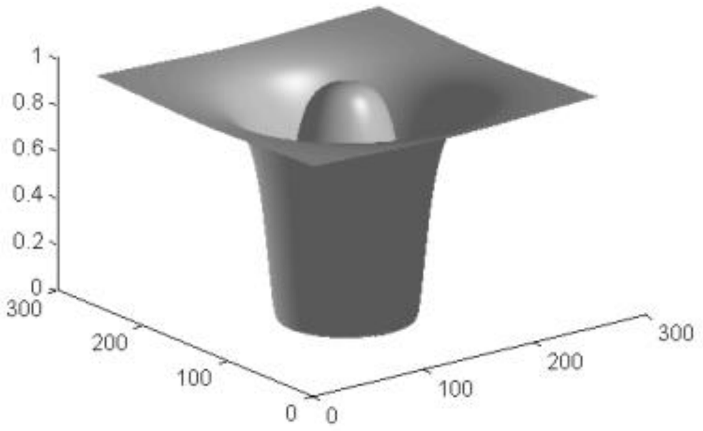
$$H(u,v) = \frac{1}{1 + \left[\frac{WD(u,v)}{D^2(u,v) - D_0^2}\right]^{2n}}$$

D0 lai santi sanga basne didenw
yo band reject filters ma kaile
kei ghatauxa kaile kei add grxa

Where,

- ***D(u,v)*** *is the distance of pixel* ***(u,v)*** *to the frequency center* ***(0,0)*** *and can be calculated by formula:* $D(u,v) = [u^2 + v^2]^{1/2}$
- ***W*** *is the width of the band (range)*
- D_0 *is the radial center*
- ***n*** *is the order of the filter*



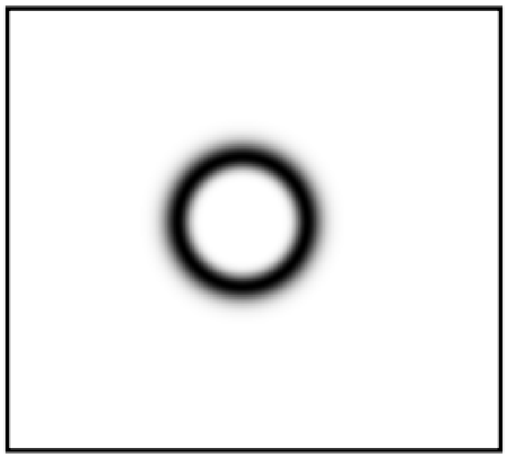


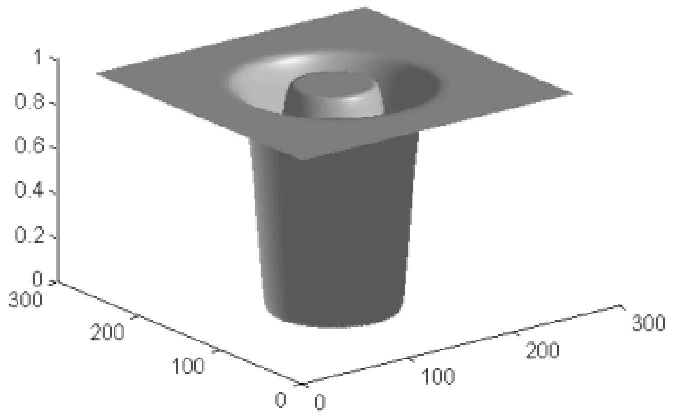
Gaussian Band Reject Filter

$$H(u,v) = 1 - e^{-\left[\frac{D^2(u,v) - D_0^2}{WD(u,v)}\right]^2}$$

Where,

- ***D(u,v)*** *is the distance of pixel* ***(u,v)*** *to the frequency center* ***(0,0)*** *and can be calculated by formula:* $\boldsymbol{D(u,v) = [u^2 + v^2]^{1/2}}$
- ***W*** *is the width of the band (range)*
- $\boldsymbol{D_0}$ *is the radial center*





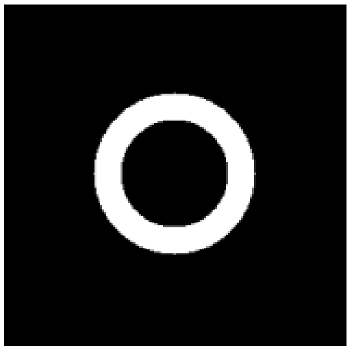
Band Pass Filters:

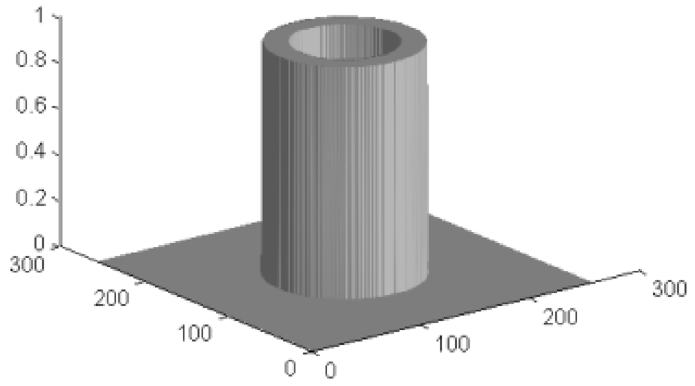
Band pass filter performs the opposite operation of a band reject filter. It let only a portion of the frequency pass. Performing straight band pass filtering on an image is not a common procedure because it generally removes too much image details.

This filter is defined by the expression

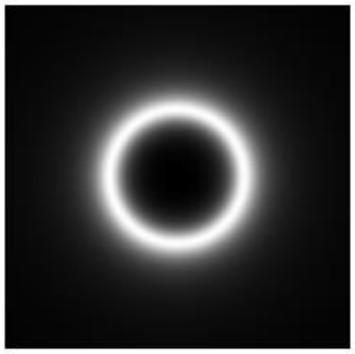
$$H_{BP}(u,v) = 1 - H_{BR}(u,v)$$

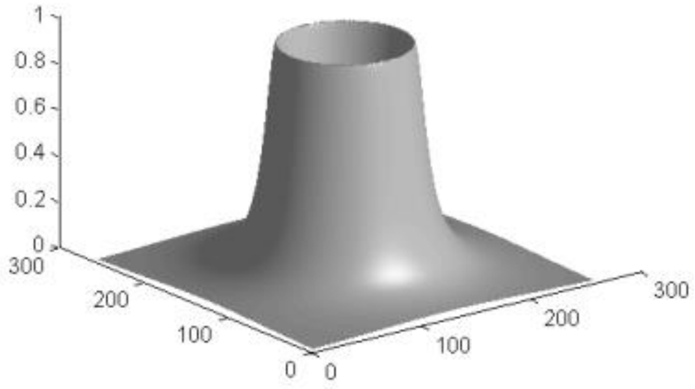
Ideal Band Pass Filter





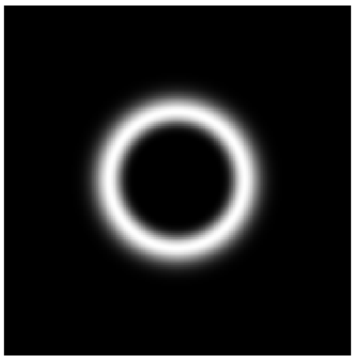
Butterworth Band Pass Filter

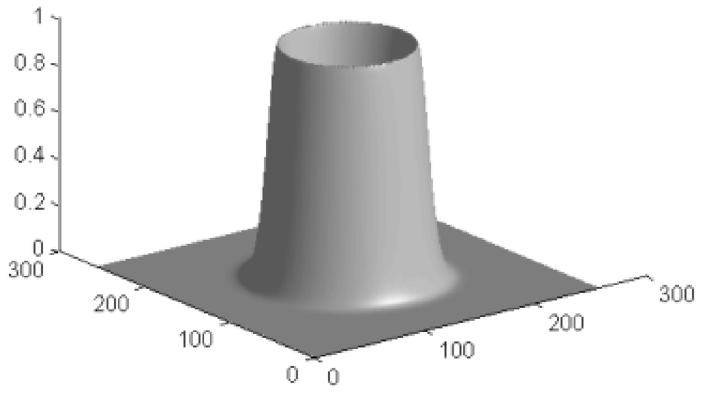




Butterworth band pass filter at order of 1

Gaussian Band Pass Filter





<u>Image Compression:</u>

Introduction overview:

Image compression is the art and science of reducing the amount of data required to represent an image, is one of the most useful and commercially successful technology in the field of digital image processing.

Consider the amount of data required to represent a 2 hour Standard Definition (SD) video using ***(720 x 480 x 24) bit*** pixel arrays. A digital movie or video is a sequence of video frames in which each frame is a full-color still image. For the display of SD video, video player must display the frame sequentially at the rate of near ***30 fps*** (*frame per second*).

$$30\frac{\text{frames}}{\text{sec}} \times (720 \times 480)\frac{\text{pixels}}{\text{frame}} \times 3\frac{\text{bytes}}{\text{pixel}} = 31{,}104{,}000 \text{ bytes/sec}$$

And 2 hour movie consists of

$$31{,}104{,}000\frac{\text{bytes}}{\text{sec}} \times \left(60^2\right)\frac{\text{sec}}{\text{hr}} \times 2 \text{ hrs} \cong 2.24 \times 10^{11} \text{ bytes}$$

Hence, we need ***2.24 x 10^{11} bytes*** (***224 GB***) storage space to hold SD movie of length 2 hour. Which mean ***27*** (*8.5GB dual-layer*) ***DVDs*** (*assuming 12 cm disk*) are needed to store 2 hour movie. To store a 2 hour movie on a single DVD, each frame must be compressed by average factor of ***26.4 (224/8.5)***. The compression must be even higher for High Definition (HD) videos, where image resolution reaches ***(1920 x 1080 x 24)*** bits/image.

Definition of image compression

Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.

There are several different ways in which image files can be compressed. For Internet use, the two most common compressed graphic image formats are the JPEG (*Joint Photographic Experts Group*) format and the GIF (*Graphics Interchange Format*) format. The JPEG method is more often used for photographs, while the GIF method is commonly used for line art and other images in which geometric shapes are relatively simple.

Compression Ratio:

In the image consist of the multiple data with same information which are irrelevant or repeated are ***called redundant data***. We can eliminate or compress the irrelevant data to reduce the file size.

Compression ratio is the ratio of uncompressed data size and compressed data size.

$$\text{Compression Ratio} = \frac{\text{Uncompressed Size}}{\text{Compressed Size}}$$

Thus, a representation that compresses a file's storage size from 10 MB to 2 MB has a compression ratio of 10/2 = 5 *(it is the five times less than original file),* often notated as an explicit ratio, 5:1 ("five" to "one"), or as an implicit ratio, 5/1 (“five” by “One”). This formulation applies equally for compression, where the uncompressed size is that of the original; and for decompression, where the uncompressed size is that of the reproduction.

For Example: The original image is 256×256 pixel, single band (gray scale), 8-bit per pixel. This file is 65,536 bytes (64K). After compression the image file is 6,554 byte *(which is 10 times less than original file)*

The compression ratio = Uncompressed Size : Compressed Size

= 56,536 : 6,554

This can be written as 10:1 This is called a "10 to 1" compression or a "10 times compression", or it can be stated as "compressing the image to 1/10 original size.

Relative Data Redundancy:

Let ***b*** and ***b'*** denotes the number of bits in two representations, uncompressed data and compressed data of the same information, the ***Relative Data Redundancy*** denoted by ***R*** of the representation with ***b*** bits is

R = 1- (1/C)

Where, ***C*** *is commonly called Compression Ratio*

If ***C = 10*** (*sometimes written as 10:1*) larger representation *(original file)* has ***10 bits*** of data for every ***1 bit*** of data in the small representation *(compressed file)*.

Then,

R = 1- (1/10) = 0.9

Which mean 90% of its data is redundant.

<u>Redundancies in Image:</u>

Average Code Length (ACL):

It is an average length of code per pixel of symbol is defined by

$$\text{ACL} = \sum_{i=1}^{n} P(x_i)L(x_i)$$

Where, $P(x_i)$ is probability of pixel or symbol and $L(x_i)$ is length of code for pixel or symbol to represent any character it requires 1 byte (8 bits) but using huffmann coding we can represent with minimum number of varaible length bits

Coding Redundancy (Huffman Coding):

Huffman coding is one of the basic lossless compression methods that have proven useful in image and video compression standards. When applying Huffman encoding technique on an Image, the source symbols can be either pixel intensities of the Image, or the output of an intensity mapping function.

The first step of Huffman coding technique is to reduce the input image to a ordered histogram, where the probability of occurrence of a certain pixel intensity value is as

Probability of Pixel = Number of Pixel / Total Number of Pixel

*Where, **Number of Pixel** is the number of occurrence of a pixel with a certain intensity value and **Total Number of Pixel** is the total number of pixels in the input Image.*

Build a Huffman Tree:

probability ko basis ma lekhdine

1. Combine the two lowest probability leaf nodes into a new node.
2. Replace the two leaf nodes by the new node and sort the nodes according to the new probability values.
3. Continue the steps (a) and (b) until we get a single node with probability value 1.0. We will call this node as **root**

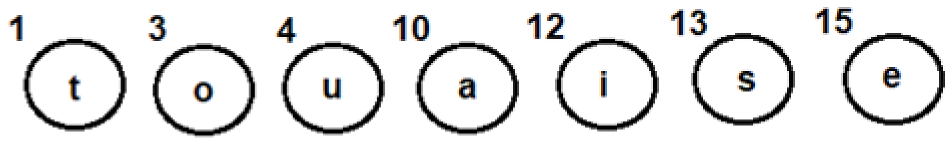
Calculate Huffman Code:

Backtrack from the root, assigning ‘0’ to left child and ‘1’ to the right child in each intermediate node, till we reach the leaf nodes. Then assemble the code 0 and 1 from root to pixel position.

Example: Create Huffman Tree for the following characters along with their frequencies using the Huffman algorithm. Also find the Huffman code for each character.

Characters	**Frequency**	**Probability**
A	10	0.17
E	15	0.26
I	12	0.21
O	3	0.05
U	4	0.07
S	13	0.22
T	1	0.02

Create leaf nodes for all the characters and add them to the min heap.



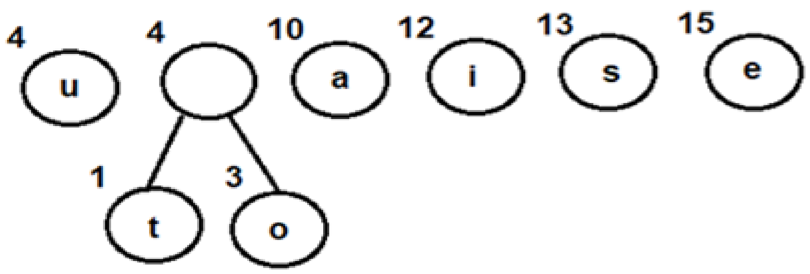
Repeat the following steps till heap has more than one node

Step 1: Extract two nodes, say **x** and **y,** with minimum frequency from the heap

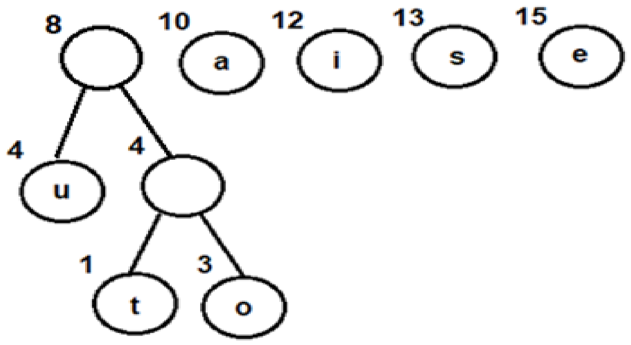
Step 2: Create a new internal node **z** with **x** as its ***left child*** and **y** as its ***right child***. Also frequency (z) = frequency (x) + frequency (y)

Step 3: Add z to min heap

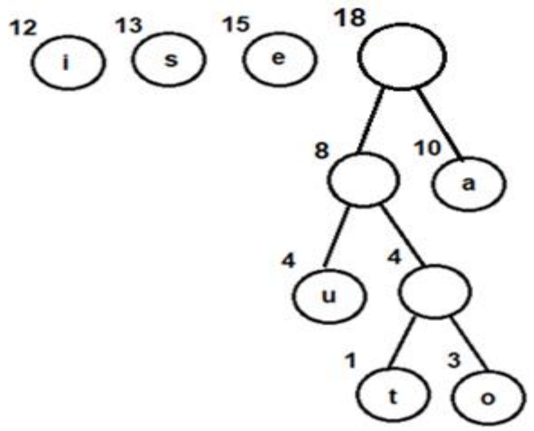
Here in above min heap, node **t** and node **o** have minimum frequency, combine those two and place to min heap.



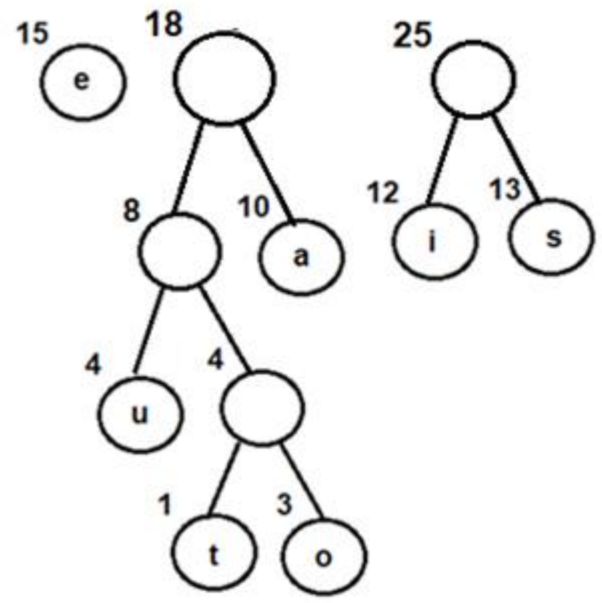
Now, combine node **u** and new tree and add to heap.



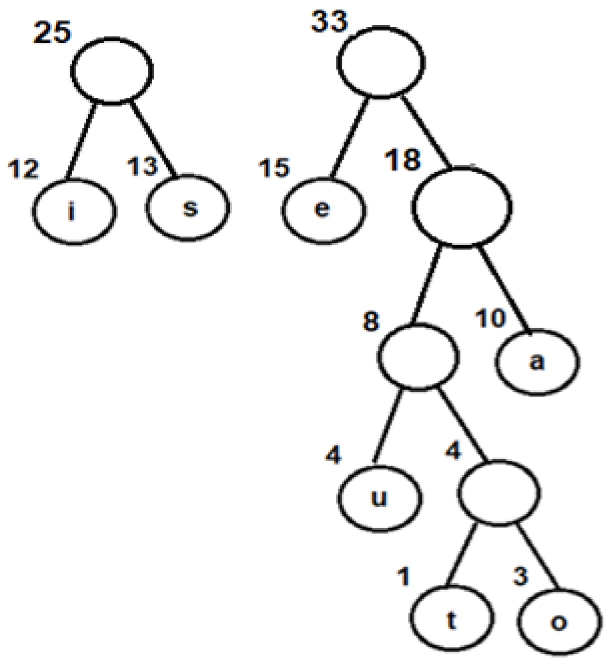
Now, combine node **a** and new tree and add to min heap.



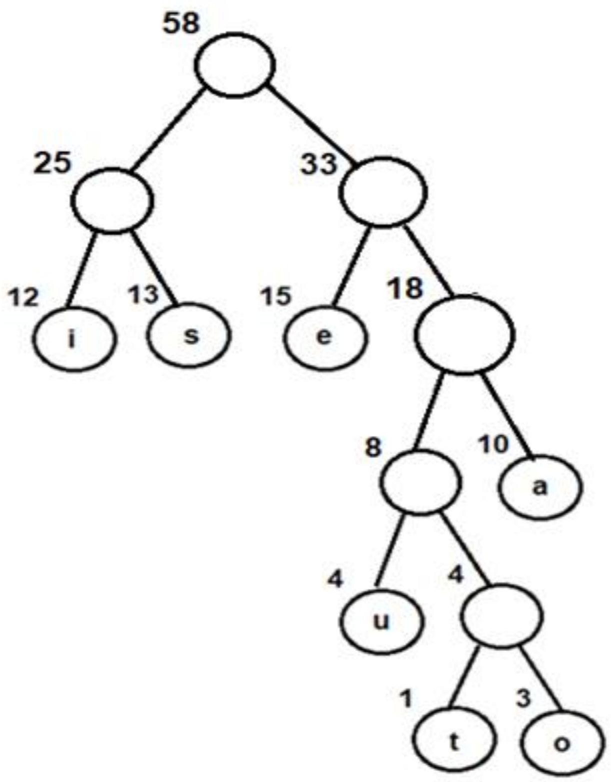
Now, combine node **i** and **s** which are minimum two nodes, and add to min heap



Now, combine **e** with tree having root node 18 and add to min heap



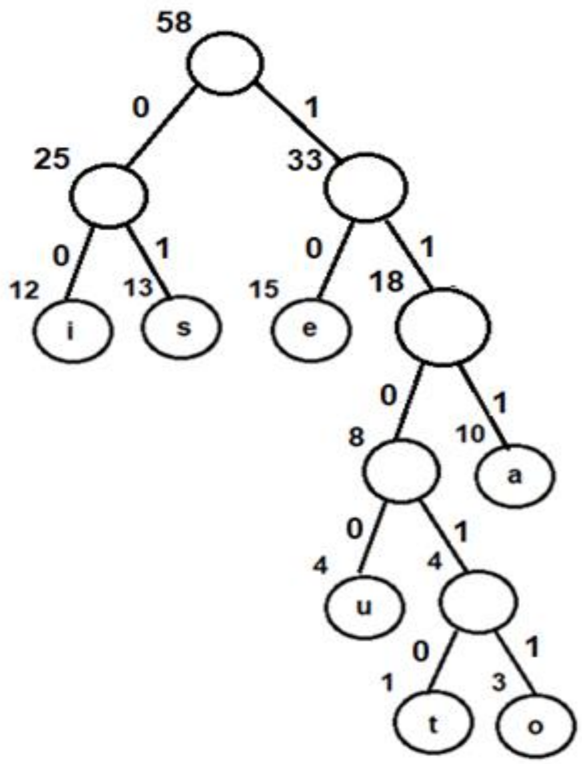
Now, combine remaining two trees



This is the final Huffman tree.

Calculate Huffman Code:
Traverse Huffman tree to get Huffman code for each characters.

Place 0 for left child and 1 for right child in the Huffman tree that we previously calculated.



Characters	Probability	Huffman Codes	Code Length	Average Code Length Sum(Pi x CLi)
A	0.17	111	3	0.51
E	0.26	10	2	0.52
I	0.21	00	2	0.42
O	0.05	11011	5	0.25
U	0.07	1100	4	0.28
S	0.22	01	2	0.44
T	0.02	11010	5	0.1
Average Code Length (ACL)				**2.52**

Inter-pixel Redundancy (Run Length Coding):

It is a simplest data compression technique. It is a form of lossless data compression in which runs of data (*sequences in which the same data value occurs in many consecutive data elements*) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs.

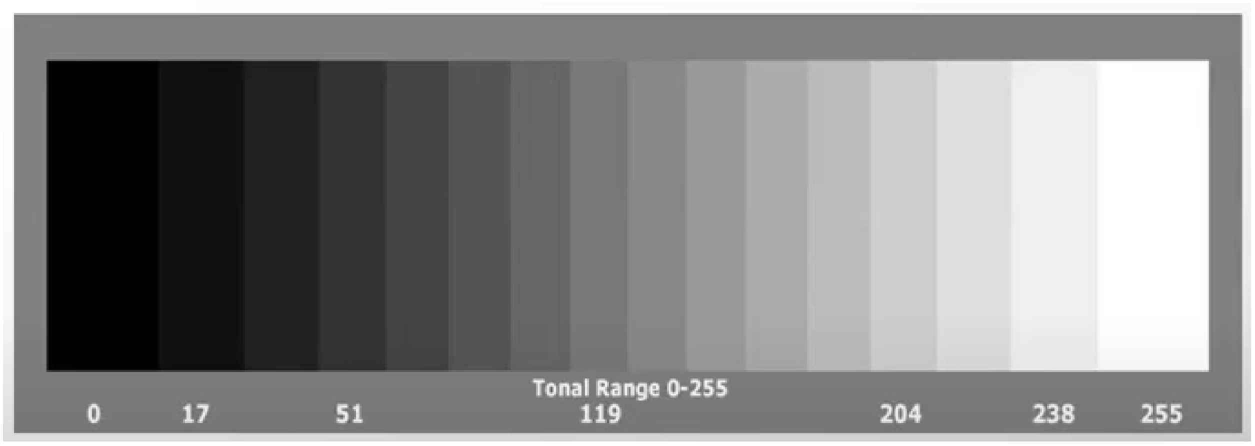
The general idea behind this method is to replace consecutive repeating occurrences of a symbol by one occurrence of the symbol followed by the number of occurrences.

Example 1: Suppose string is AAAAAAA then Run-length encoding is A7 (A is character and 7 is number of times appear that string)

Example 2: If input string is "WWWWAAADEXXXXXX", then the Run-length encoding is W4A3D1E1X6.

Psychovisual Redundancy (4-bit Improved Gray Scale Coding: IGS Coding Scheme):

The tendency of certain kinds of information to be relatively unimportant to the human visual system, The Psychovisual redundancies exist because human perception does not involve quantitative analysis of every pixel or luminance value in the image. During human eye visualization certain information can be eliminated without significantly degrading image quality. For example:



Here, in the above image, there are different levels of shades, human eyes can't differentiate all the shades like gray scale 253, 254, and 255 are like same so, 253 and 254 can illuminate and replace by 255.

Image Compression Models: lossy predictive coding

Compression has two types i.e. Lossy and Lossless technique. A typical image compression system comprises of two main blocks An ***Encoder*** *(Compressor)* which is a software application used to compress an image and ***Decoder*** *(De-compressor)* which is a software application used to de-compress an image. The software application name Codec is performs both encoding and decoding operations.

The image ***f(x,y)*** is fed to the encoder which encodes the image to make it suitable for transmission. The decoder receives this transmitted signal and reconstructs the output image ***f'(x,y).*** If the system is an error free the image ***f'(x,y)*** will be a replica of ***f(x,y).*** Input and output function ***f(x,y,t)*** and ***f'(x,y,t)*** respectively are used for video application where ***t*** specified the time.

@maqus

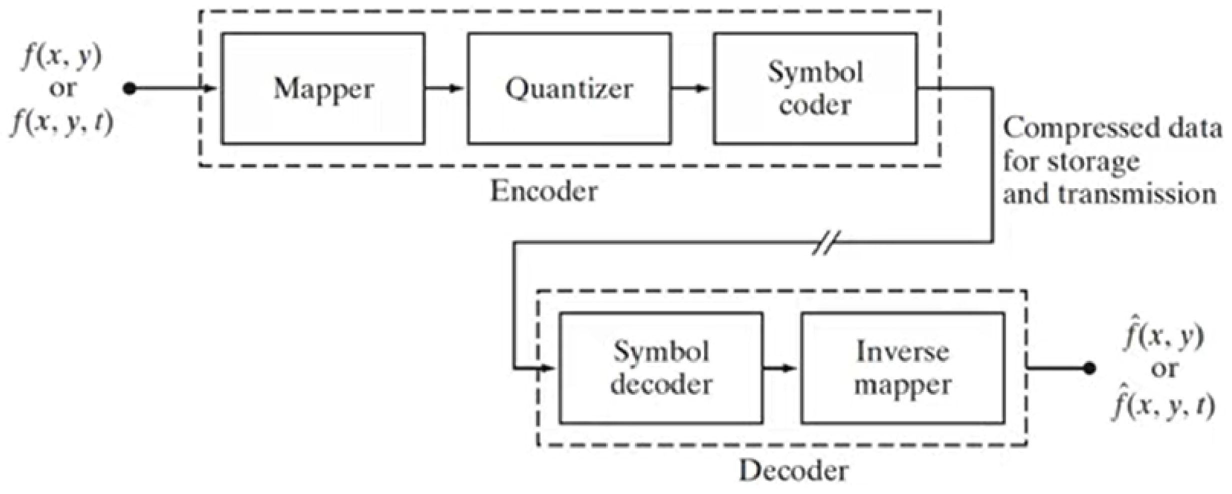


Fig: Block Diagram of Image Compression Model

Encoder (Compressor)

The three basic types of the redundancies in an image are interpixel, coding redundancies and psychovisual redundancies. Run length coding is used to eliminate or reduce inter-pixel redundancies, Huffman encoding is used to eliminate or reduce coding redundancies and psychovisual is used to eliminate interpixel redundancies.

The input image is passed through a ***Mapper***. The mapper reduces the interpixel redundancies. The mapping stage is a lossless technique and hence is a reversible operation. Run-length coding is used to compress the image in this stage.

The output of a mapper is passed to a ***Quantizer*** block. The quantizer block reduces the psychovisual redundancies. It compresses the data by eliminating some redundant information and hence is an irreversible operation. The quantizer block uses JPEG compression which means a lossy compression. Hence in case of lossless compression, the quantizer block is omitted.

The final block of the encoder is ***Symbol Coder***. This block creates a variable length code to represent the output of the quantizer. The shortest code length is used for most frequent quantizer output to reduce coding redundancy. This operation is reversible. The Huffman code is a typical example of the symbol coder.

Decoder (De-compressor)

The ***Symbol Decoder*** and ***Inverse Mapper*** of decoder block perform exactly the reverse operation of the symbol coder and the mapper blocks respectively. It is important to note that the decoder has only two blocks. Since quantization is irreversible hence, an inverse quantizer block does not exist.

End of Unit-4