

# Image Restoration and Compression

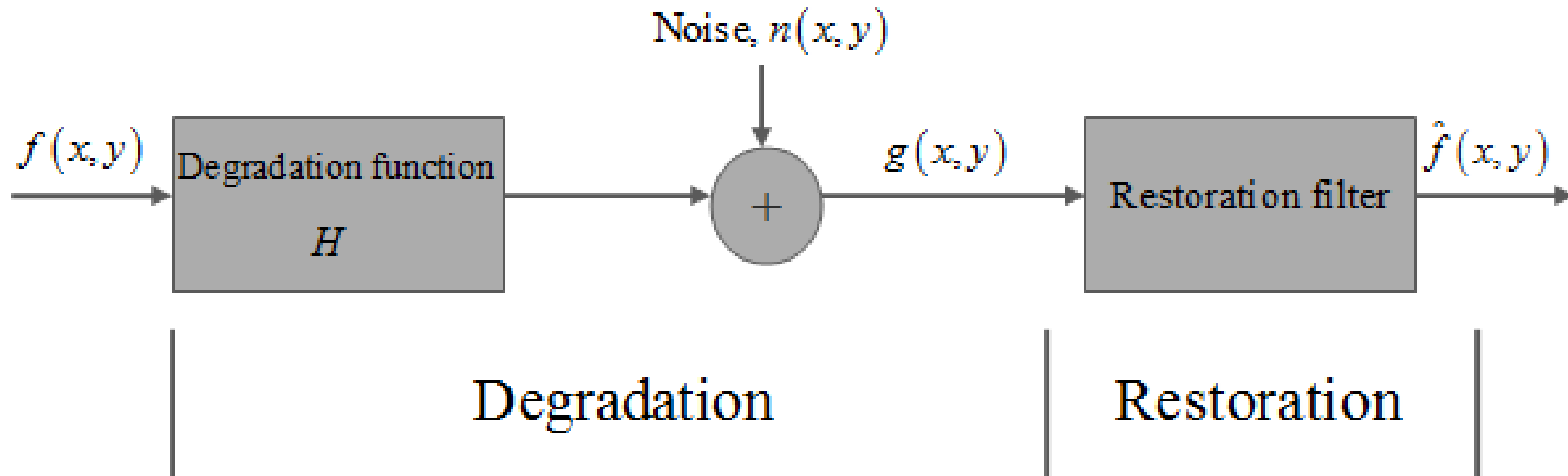
UNIT 4 | Yuba Raj Devkota

NCCS

<b>Unit 4</b>	<b>Image Restoration and Compression</b>	<b>Teaching Hours (8)</b>
Image Restoration	Introduction, Models for Image degradation and restoration process, Noise Models (Gaussian,	2 hrs.
	Rayleigh, Erlang, Exponential, Uniform and Impulse), Estimation of Noise Parameters	
Restoration Filters	Mean Filters: Arithmetic, Geometric, Harmonic and Contraharmonic Mean Filters Order Statistics Filters: Median, Min and Max, Midpoint and Alpha trimmed mean filters Band pass and Band Reject filters: Ideal, Butterworth and Gaussian Band pass and Band Reject filters	2 hrs.
Image Compression	Introduction, Definition of Compression Ratio, Relative Data Redundancy, Average Length of Code Redundancies in Image: Coding Redundancy (Huffman Coding), Interpixel Redundancy (Run Length Coding) and Psychovisual Redundancy (4-bit Improved Gray Scale Coding: IGS Coding Scheme)	2 hrs.
Image compression models:	Lossless and Lossy Predictive Model (Block Diagram and Explanation)	2 hrs.

# What is Image Restoration?

- image restoration is an essential approach used for the retrieval of the uncorrupted, original images from blurred and noisy images because of motion blur, noise, etc. caused by environmental effects and camera misfocus.
- The technique of image restoration involves recovering an image from a damaged state—typically a distorted and noisy image. In image processing, image restoration is a core issue.
- Image restoration is done in both frequency domain and spatial domain.



$$g(x, y) = h(x, y) * f(x, y) + n(x, y) \rightarrow \textcircled{1}$$

$$G(u, v) = H(u, v) F(u, v) + N(u, v) \rightarrow \textcircled{2}$$



True image



Blurred and noisy image



Restoration by Algorithm 1



Restoration by Algorithm 3



True image



Blurred and noisy image



Restoration by Algorithm 1



Restoration by Algorithm 3



True image



Blurred and noisy image



Restoration by Algorithm 1



Restoration by Algorithm 3



True image



Blurred and noisy image

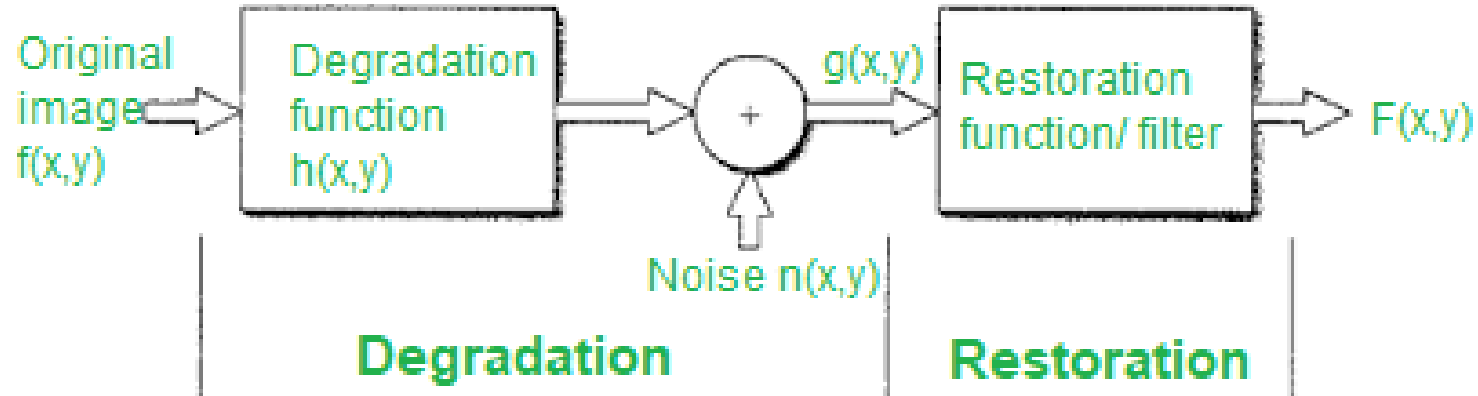


Restoration by Algorithm 1



Restoration by Algorithm 3

# Noise Models in Digital Image Processing (Noise Probability Density Function, PDF)



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

The goal of the restoration function or the restoration filter is to obtain a close replica  $F(x,y)$  of the original image.

## Different Noise Models

1. Gaussian Noise Model
2. Rayleigh Noise Model
3. Erlang (or gamma) Noise Model
4. Exponential Noise Model
5. Uniform Noise Model
6. Impulse Noise Model

### **Gaussian Noise:**

Because of its mathematical simplicity, the Gaussian noise model is often used in practice and even in situations where they are marginally applicable at best. Here,  $m$  is the mean and  $\sigma^2$  is the variance.

Gaussian noise arises in an image due to factors such as electronic circuit noise and sensor noise due to poor illumination or high temperature.

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

# (i) Gaussian Noise: [Normal noise model]

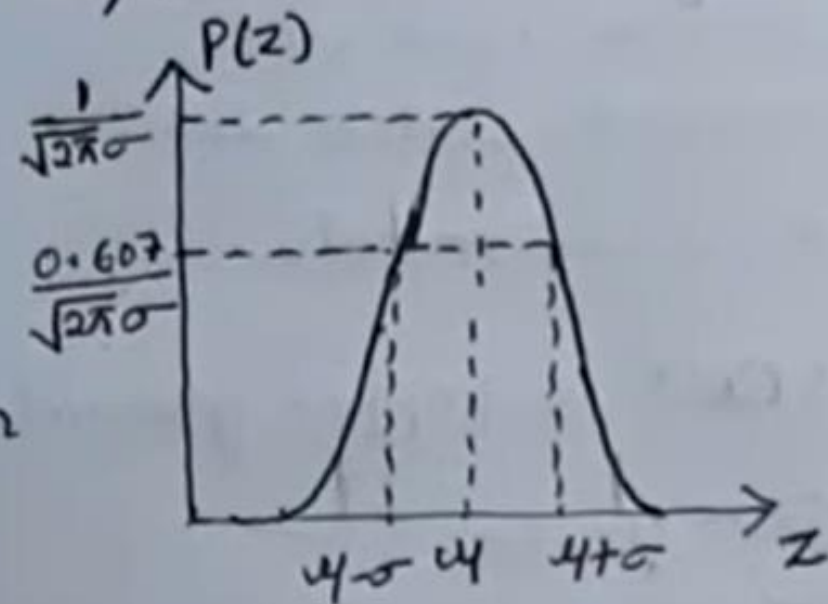
$$\text{PDF, } P(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

$z \rightarrow$  gray level

$\mu \rightarrow$  mean or avg of  $z$

$\sigma \rightarrow$  Standard deviation

$\sigma^2 \rightarrow$  Variance.



70% of values  $\rightarrow [\mu-\sigma, \mu+\sigma]$

95% of values  $\rightarrow [\mu-2\sigma, \mu+2\sigma]$



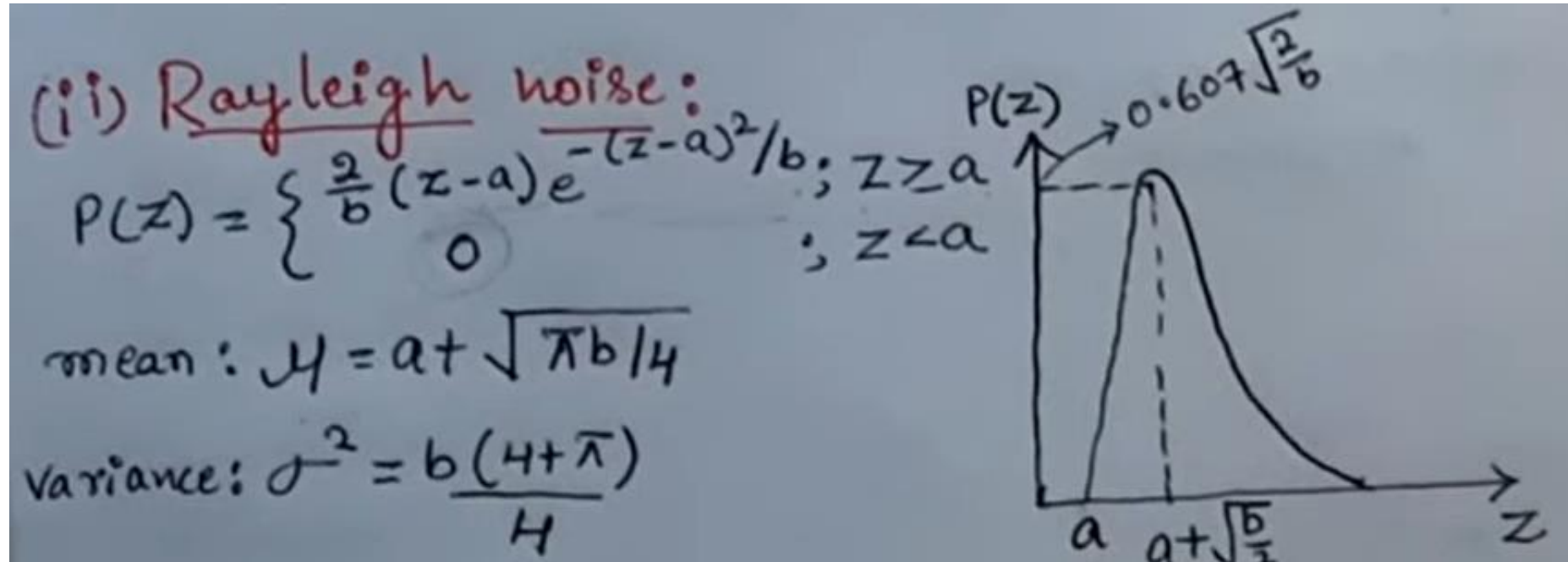
# Rayleigh Noise

$$p(z) = \frac{2}{b}(z - a)e^{\frac{-(z-a)^2}{b}} \text{ for } z \geq a, \text{ and } p(z) = 0 \text{ otherwise.}$$

Here mean  $m$  and variance  $\sigma^2$  are the following:

$$m = a + \sqrt{\pi b/4}$$

$$\sigma^2 = \frac{b(4-\pi)}{4}$$



Rayleigh noise is usually used to characterize noise phenomena in range imaging.



# Erlang (or gamma) Noise

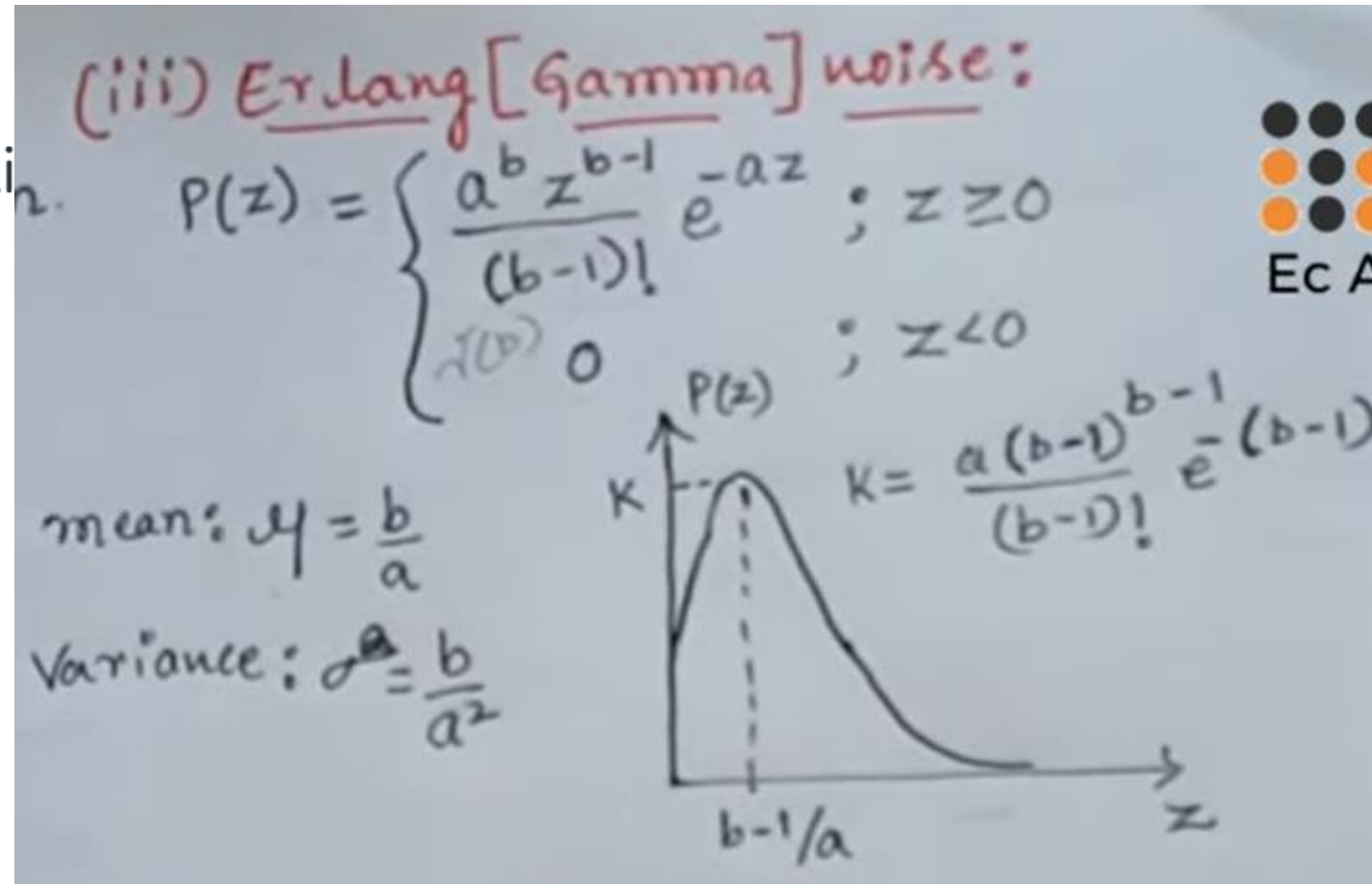
$$p(z) = \frac{a^b z^{b-1}}{(b-1)!} e^{-az} \text{ for } z \geq 0 \text{ and } p(z) = 0 \text{ otherwise.}$$

Here ! indicates factorial. The mean and variance are given below.

$$m = b/a, \sigma^2 = b/a^2$$

Gamma noise density finds appli

Gamma Noise Density finds  
applicable in laser imaging



# Exponential Noise

$p(z) = ae^{-az}$  for  $z \geq 0$  and  $p(z) = 0$  otherwise.

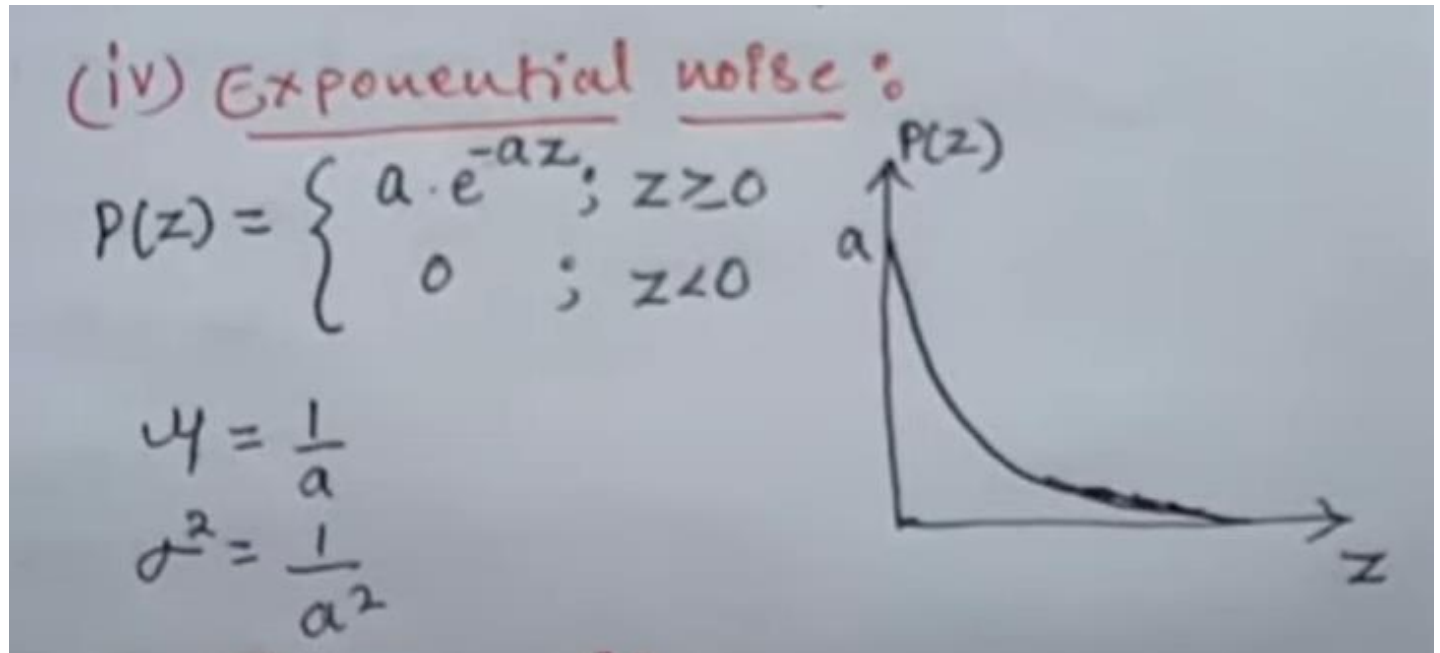
Here  $a > 0$ . The mean and variance of this noise pdf are:

$$m = 1/a$$

$$\sigma^2 = 1/a^2$$

This density function is a special case of  $b = 1$ .

Exponential noise is also commonly present in cases of laser imaging.

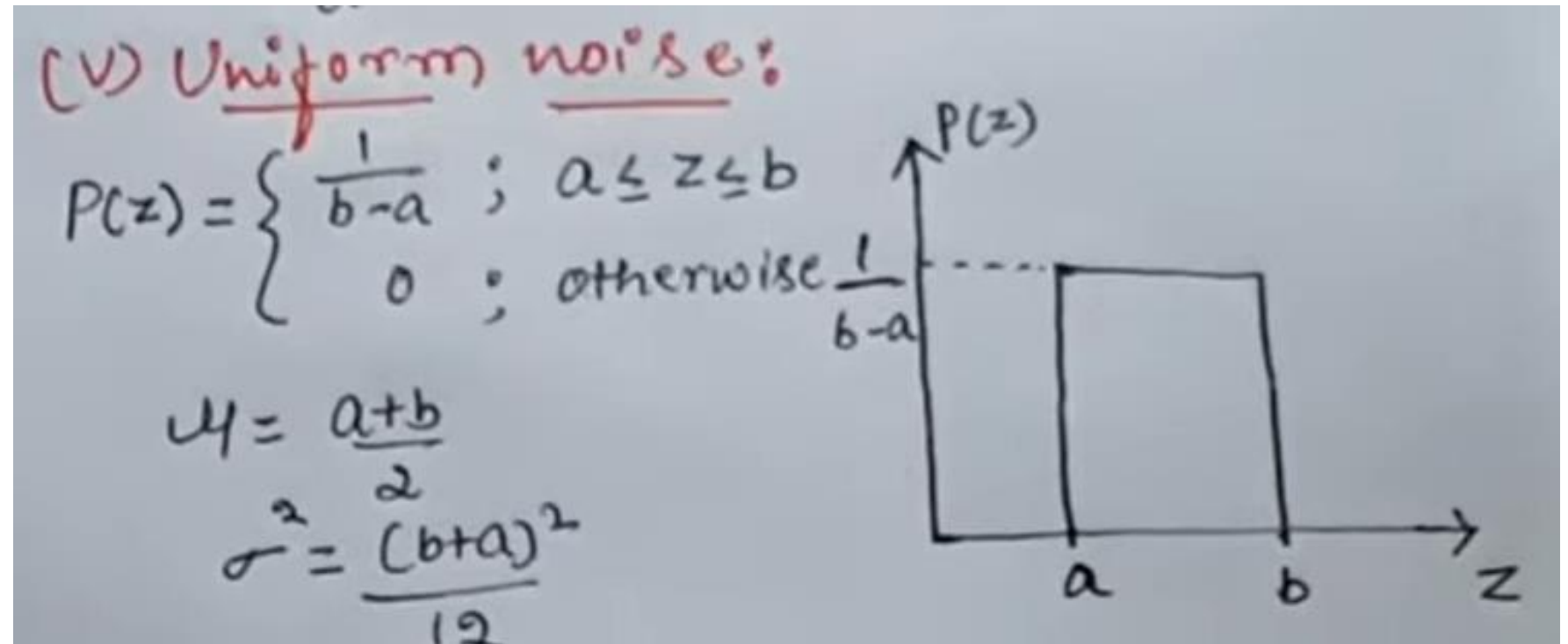


## Uniform Noise

$$p(z) = \frac{1}{b-a} \text{ if } a \leq z \leq b, \text{ and } p(z) = 0 \text{ otherwise.}$$

The mean and variance are given below.

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



Uniform noise is not practically present but is often used in numerical simulations to analyze systems.

# Impulse Noise

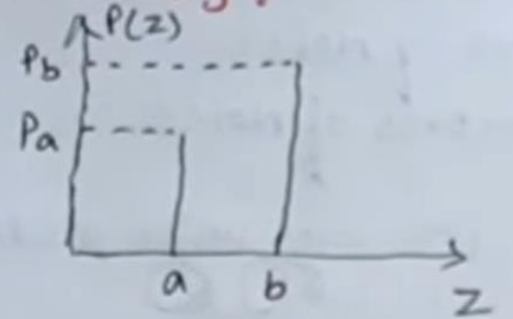
$$p(z) = P_a \text{ for } z = a, p(z) = P_b \text{ for } z = b, p(z) = 0 \text{ otherwise.}$$

If  $b > a$ , intensity  $b$  will appear as a light dot in the image. Conversely, level  $a$  will appear like a black dot in the image. Hence, this presence of white and black dots in the image resembles to salt-and-pepper granules, hence also called salt-and-pepper noise.

When either  $P_a$  or  $P_b$  is zero, it is called unipolar noise. The origin of impulse noise is quick transients such as faulty switching in cameras or other such cases.

(Vi) Salt & Pepper noise [Impulse noise]:

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



→  $b > a \Rightarrow$  gray level  $b \rightarrow$  light dot  
gray level  $a \rightarrow$  dark dot

→  $P_a = 0$  or  $P_b = 0 \rightarrow$  unipolar

→  $P_a \neq P_b \rightarrow$  Salt & Pepper granules.

Shot & Spike noise.

-ve impulses  $\rightarrow$  black (Pepper) point  
+ve impulses  $\rightarrow$  white (salt) point.

8-bit  $\Rightarrow a = 0$  (black)  
 $b = 255$  (white)

# Restoration in the presence of Noise Only (Spatial Filtering)

- The degradation image is given by:

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

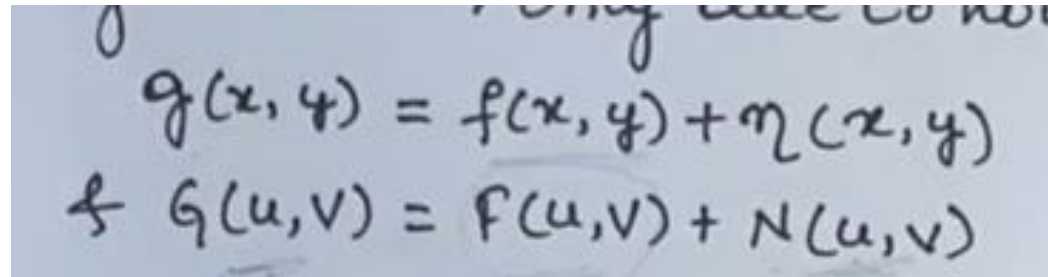
- Since degradation is only due to noise, we omit the part  $h(x,y)$ , so the new equation and its Fourier transform becomes as follows.
- The original image  $F(u,v)$  can be obtained by subtracting noise from Frequency domain image i.e.  $G(u,v) - N(u,v)$ .

## 1. Mean Filters

1. Arithmetic Mean Filters
2. Geometric Mean Filters
3. Harmonic Mean Filters
4. Contra Harmonic Mean Filters

## 2. Order Statistics Filters

1. Median Filters
2. Min & Max Filters
3. Mid Point Filters
4. Alpha Trimmed Mean Filters



Handwritten equations showing the degradation model and its Fourier transform:

$$g(x, y) = f(x, y) + n(x, y)$$
$$\& G(u, v) = F(u, v) + N(u, v)$$

There are two types of spatial filtering methods that restore the image in the presence of noise only:

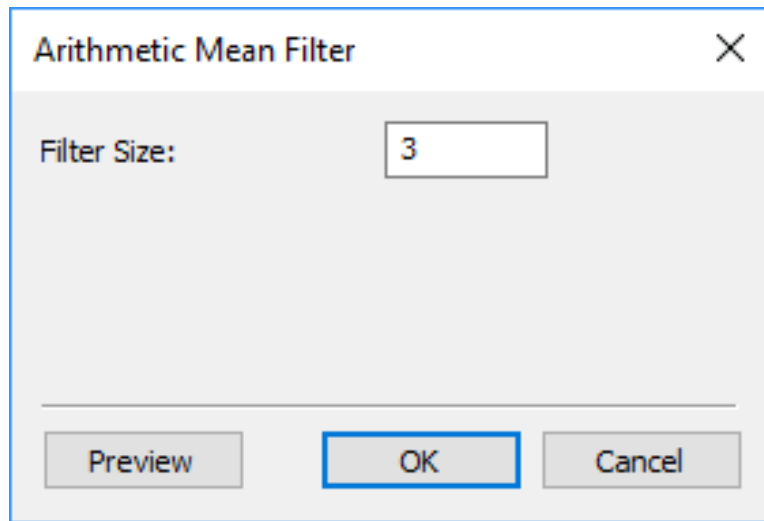
## A. Mean Filters

1. Arithmetic Mean Filters
2. Geometric Mean Filters
3. Harmonic Mean Filters
4. Contra Harmonic Mean Filters

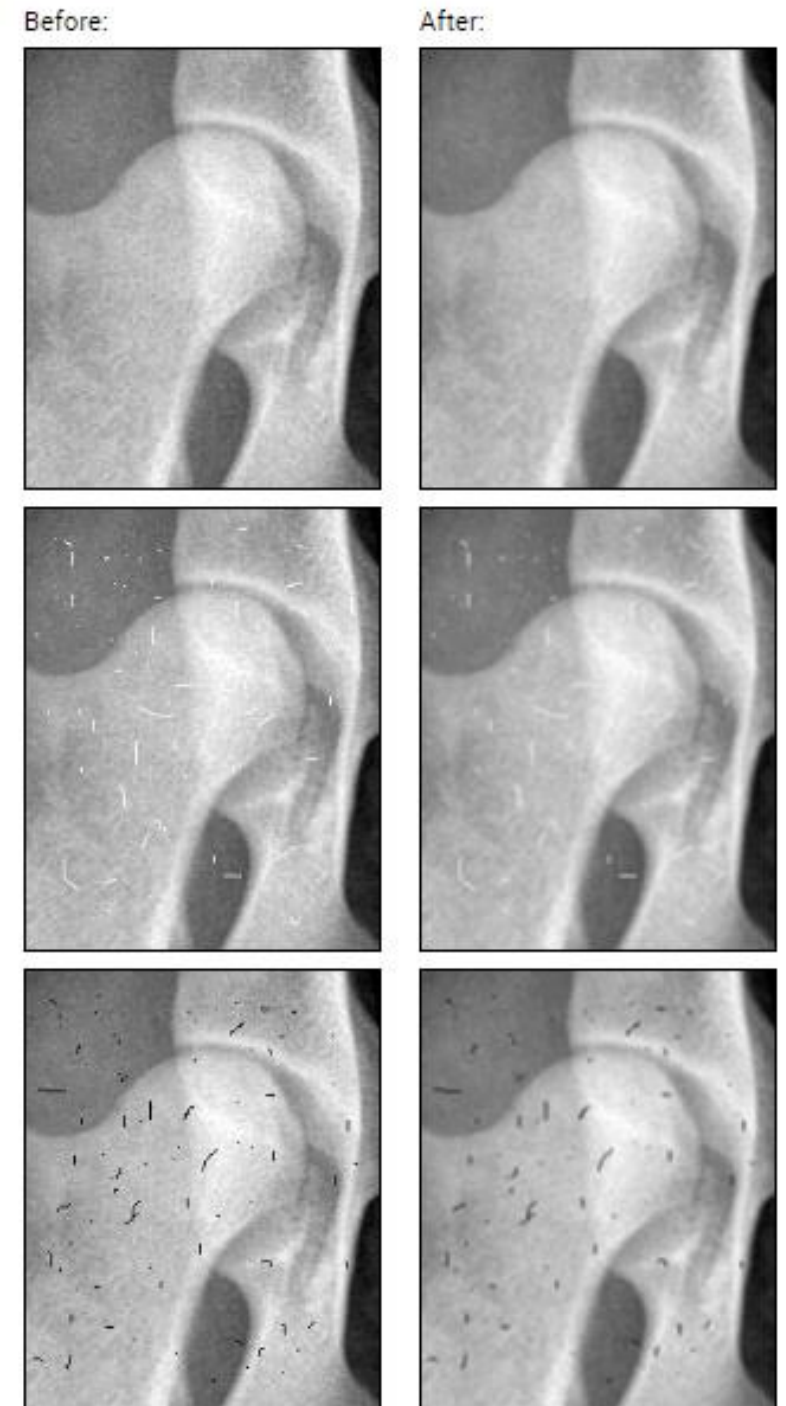
## i) Arithmetic mean filter

- An arithmetic mean filter operation on an image removes short tailed noise such as uniform and Gaussian type noise from the image at the cost of blurring the image.
- The arithmetic mean filter is defined as the average of all pixels within a local region of an image.

$$\bar{x} = \frac{1}{n}(x_1 + \dots + x_n)$$



**Filter Size:** size of the filter mask; a larger filter yields stronger effect.



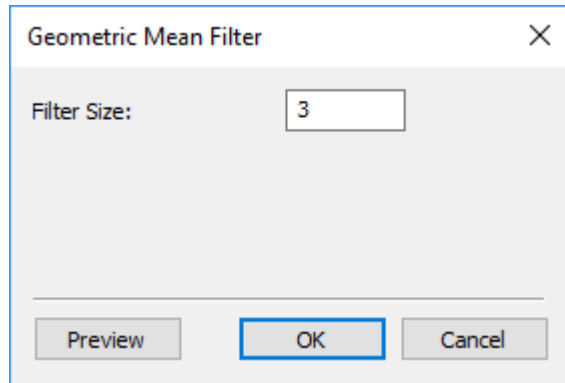


## ii) Geometric mean filter

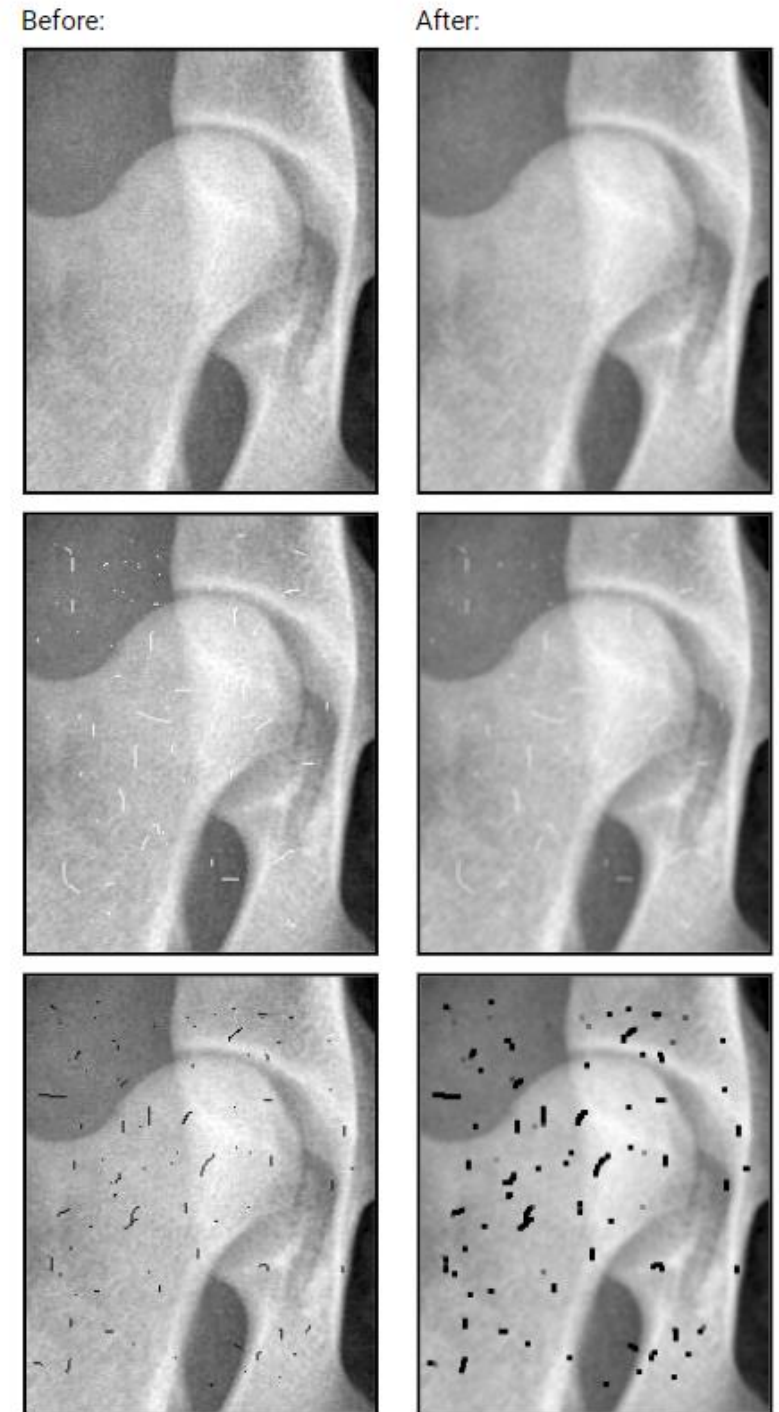
- In the geometric mean method, the color value of each pixel is replaced with the geometric mean of color values of the pixels in a surrounding region.
- A larger region (filter size) yields a stronger filter effect with the drawback of some blurring.

$$G = \sqrt[n]{a_1 \cdot a_2 \cdots a_n}$$

The geometric mean filter is better at removing Gaussian type noise and preserving edge features than the arithmetic mean filter. The geometric mean filter is very susceptible to negative outliers.



**Filter Size:** larger filter yields stronger effect.

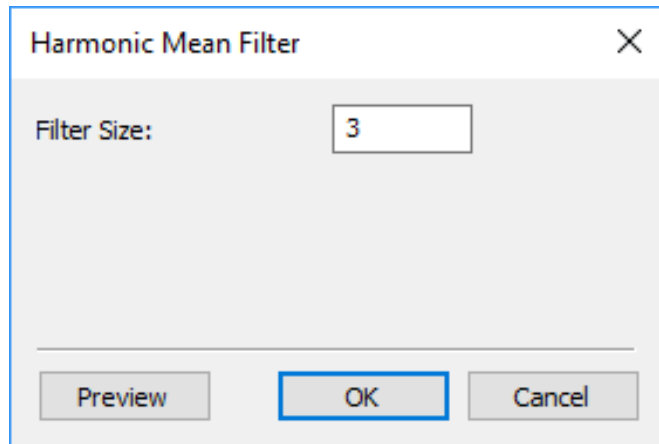


### iii) Harmonic mean filter

In the harmonic mean method, the color value of each pixel is replaced with the harmonic mean of color values of the pixels in a surrounding region.

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \dots + \frac{1}{x_n}}$$

- A larger region (filter size) yields a stronger filter effect with the drawback of some blurring.
- The harmonic mean filter is better at removing Gaussian type noise and preserving edge features than the arithmetic mean filter.
- The harmonic mean filter is very good at removing positive outliers.



Before:



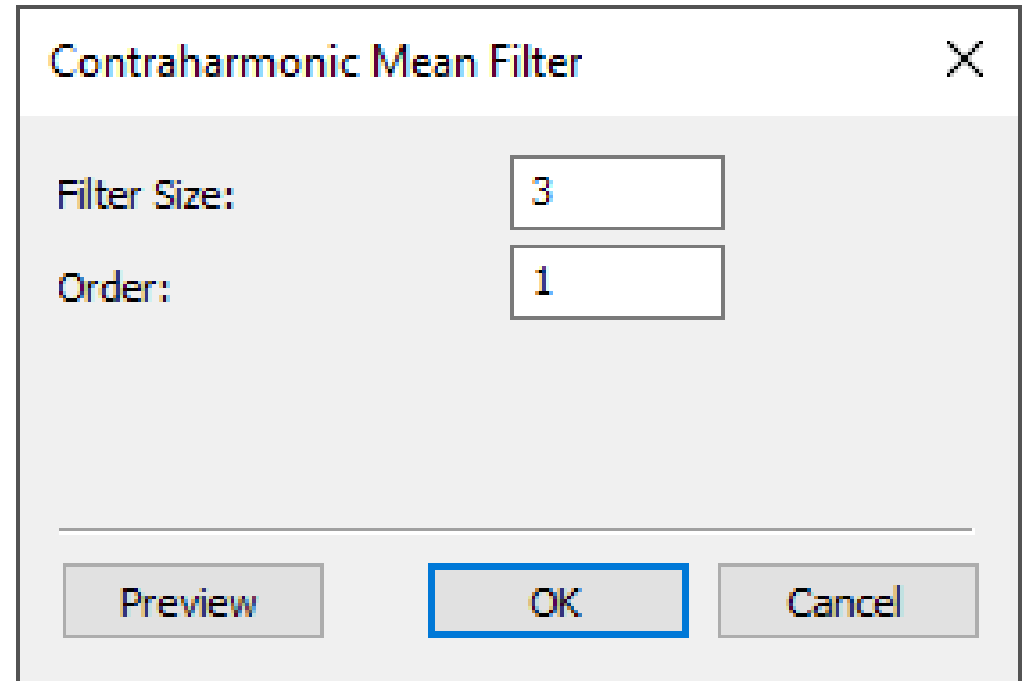
After:



#### iv) Contra-Harmonic mean filter

With a contra harmonic mean filter, the color value of each pixel is replaced with the contra harmonic mean of color values of the pixels in a surrounding region.

$$C_Q = \frac{x_1^{Q+1} + x_2^{Q+1} + \dots + x_n^{Q+1}}{x_1^Q + x_2^Q + \dots + x_n^Q}$$



Contraharmonic Mean Filter

Filter Size: 3

Order: 1

Preview OK Cancel

- A contra harmonic mean filter reduces or virtually eliminates the effects of salt-and-pepper noise. For positive values of  $Q$ , the filter eliminates pepper noise. For negative values of  $Q$  it eliminates salt noise. It cannot do both simultaneously.
- Note that the contra harmonic filter is simply the arithmetic mean filter if  $Q = 0$ , and the harmonic mean filter if  $Q = -1$ .
- A larger region (filter size) yields a stronger filter effect with the drawback of some blurring.

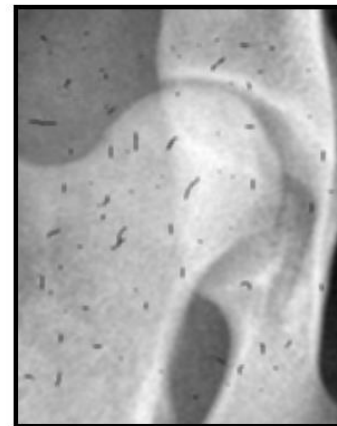
Before:



After (Q=1):



After (Q=-1):



Contraharmonic Mean Filter

Filter Size:

3

Order:

1

Preview

OK

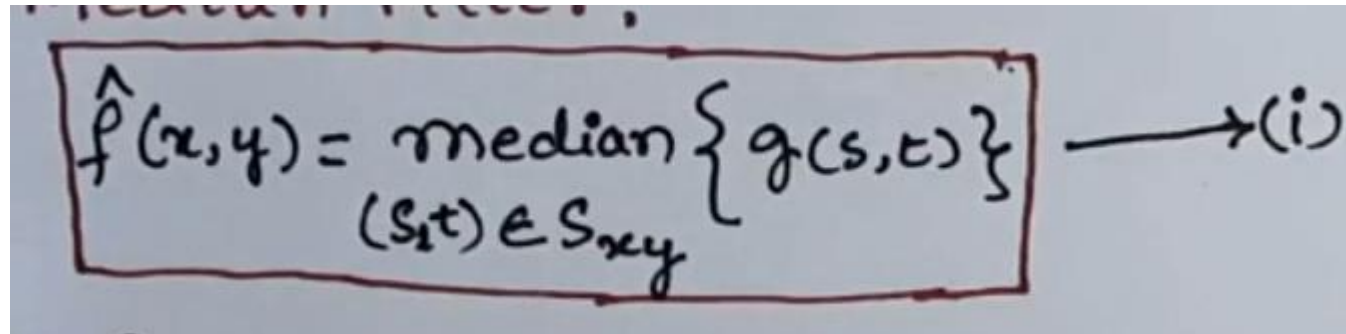
Cancel

## B. Order Statistics Filters

Order-statistics filters are spatial filters whose response is based on ordering (ranking) the pixels contained in the image area compressed by the filter. The response of the filter at any point is determined by the ranking result.

### i) Median filter

- The best-known order-statistics filter is the median filter, which, as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel:
- The original value of the pixel is included in the computation of the median. Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of both bipolar and unipolar impulse noise. In fact, the median filter yields excellent results for images corrupted by this type of noise.



A handwritten equation for the median filter is shown, enclosed in a hand-drawn rectangular box. The equation is  $\hat{f}(x, y) = \text{median} \left\{ g(s, t) \right\}_{(s, t) \in S_{xy}}$ . To the right of the box, there is an arrow pointing to the label  $(i)$ .



(ii) Min & Max Filter:

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

Pepper noise ↓

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

Salt noise ↓

max → Finding the brightest point  
min → Find the darkest point.

(iii) Mid point Filter:

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

→ Computes the mid point b/w  $\max$  &  $\min$  values.

$$\frac{d_1}{2} + \frac{d_2}{2} = \frac{d_1 + d_2}{2}$$

→ Combines order statistic & averaging

→ works best → Randomly distributed noise



(iv) Alpha-trimmed mean Filter:

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

$\frac{d}{2} \rightarrow$  lowest  
 $\frac{d}{2} \rightarrow$  highest

$d \rightarrow 0$  to  $mn$

When  $d=0 \rightarrow$  Reduced to Arith mean Fit.

$d = \frac{(mn-1)}{2} \rightarrow$  median Filter.

noise  $\downarrow$  Other values  $\rightarrow$  useful in multiple type of noise reduction

# Band Pass Filters

**Band pass filters** in image processing allow frequencies within a certain range to pass through while attenuating frequencies outside that range. These filters are useful for isolating specific features in an image that correspond to particular frequency components.

## **Applications:**

- Enhancing specific features in medical imaging.
- Isolating textures in satellite imagery.
- Extracting certain frequencies for artistic effects.

## **Types:**

1. Ideal Band Pass Filter: Passes frequencies within a specific range without attenuation and completely rejects frequencies outside that range.
2. Butterworth Band Pass Filter: Has a smoother transition between the passband and the stopband compared to the ideal band pass filter.
3. Gaussian Band Pass Filter: Provides a smooth transition with a bell-shaped frequency response. Less sharp compared to Butterworth but avoids the ringing artifacts often seen with ideal filters.



Band-pass Filters allow frequencies within a particular range to pass through and attenuate all other frequencies

- ☐ This filter allows the frequencies if they fall in the range  $D_l - D_h$
- ☐ This range is known as Band

**1D Transfer Function is given by:**



$$H(D) = \begin{cases} 1 & \text{for } D_l \leq D \leq D_h \\ 0 & \text{for } D > D_0 \end{cases}$$

*$D_0 \rightarrow$  cut-off freq.*

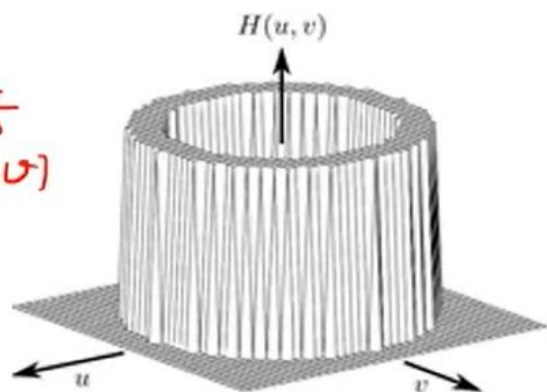
The transfer function for 2D band-pass filter is given as follows:

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

✓  $D(u, v)$

↓  
distance of  
the point  $(u, v)$   
from the  
center

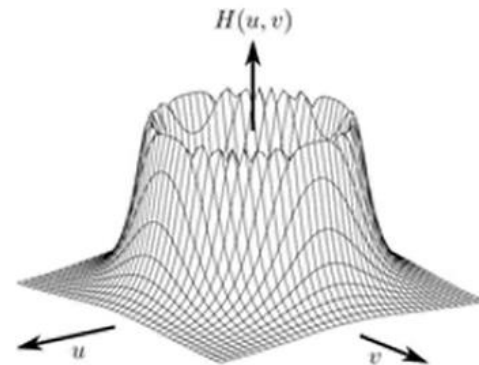
✓  $W \rightarrow$   
width  
of band



Ideal Band-pass Filter

A Butterworth band-pass filter of order n is given as :

$$H_{bp}^{Butterworth}(u, v) = \frac{\left[ \frac{D(u, v) \omega}{D^2(u, v) - D_0^2} \right]^{2n}}{1 + \left[ \frac{D(u, v) \omega}{D^2(u, v) - D_0^2} \right]^{2n}}$$

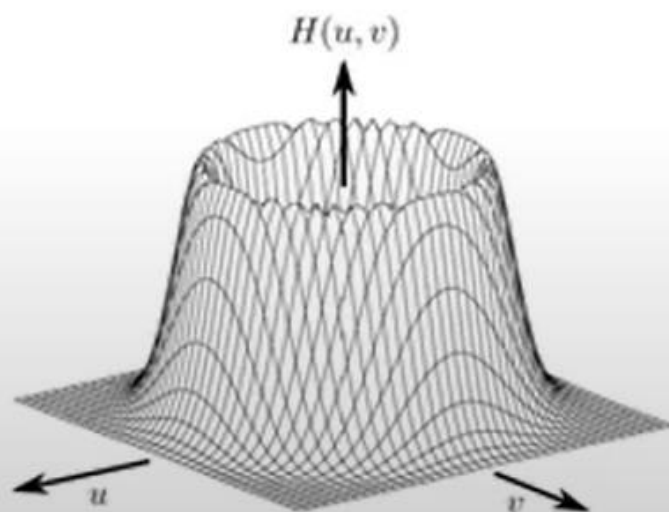


Butterworth Band-pass Filter



A Gaussian band-pass filter is given as follows:

$$H_{bp}^{\text{Gaussian}}(u, v) = e^{-\frac{1}{2} \left[ \frac{D^2(u, v) - D_0^2}{D(u, v) \omega} \right]^2}$$



Gaussian Band-pass Filter

# Band Reject Filters

**Band reject filters** (also known as notch filters) do the opposite of band pass filters: they attenuate frequencies within a specific range and allow others to pass through. These are useful for removing unwanted frequencies such as noise or periodic patterns.

## Applications

- Removing periodic noise in scanned documents.
- Eliminating specific frequency components from an image, such as interference patterns.
- Smoothing images by removing high-frequency noise.

## Types

1. Ideal Band Reject Filter: Completely rejects frequencies within a specific range and passes frequencies outside that range.
2. Butterworth Band Reject Filter: Provides a gradual transition from the stopband to the passband, defined by the filter's order.
3. Gaussian Band Reject Filter: Uses a Gaussian function for smooth transition, similar to the Gaussian band pass filter but inverted.



✓ The objective of Band-reject Filter is to attenuate limited range Frequencies while leaving the other frequencies

Band-reject Filter is the complement of a band pass filter

These filters are very effective in removing periodic noise and ringing effect is normally small

✓  $H_{br}(u,v) = 1 - H_{bp}(u,v)$

This filter rejects the frequencies if they fall in the range  $D_l - D_h$  ✓

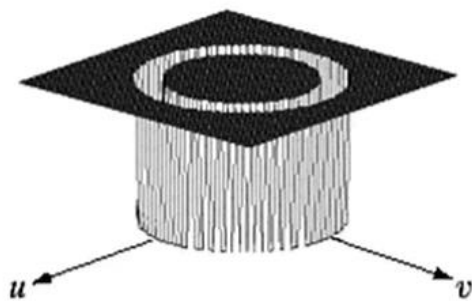
**1D Transfer Function is given by:**

$$H(D) = \begin{cases} 0 & \text{for } D_l \leq D \leq D_h \\ 1 & \text{for } D > D_0 \end{cases}$$



An ideal band reject filter is given as follows:

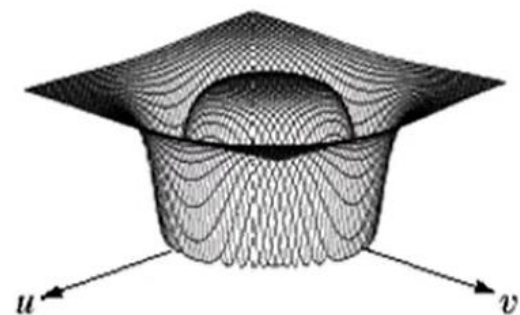
$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) < D_0 - \frac{W}{2} \\ 0 & \text{if } D_0 - \frac{W}{2} \leq D(u, v) \leq D_0 + \frac{W}{2} \\ 1 & \text{if } D(u, v) > D_0 + \frac{W}{2} \end{cases}$$



Ideal Band Reject Filter

A Butterworth band reject filter of order  $n$  is given as :

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



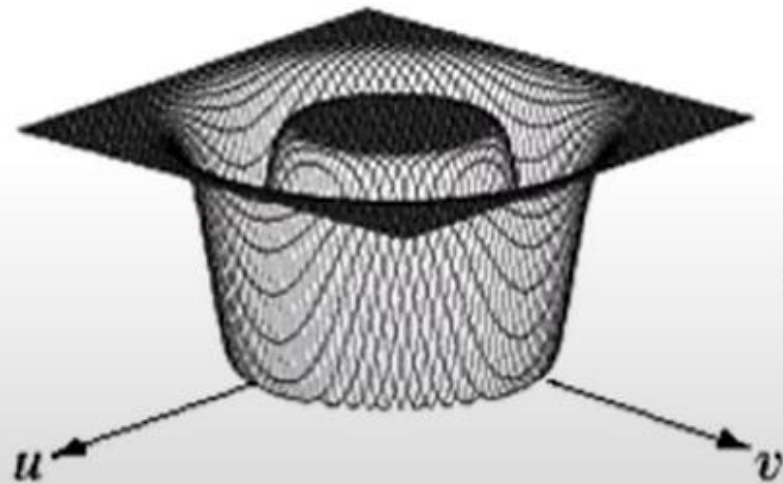
Butterworth Band Reject Filter  
(of order 1)



A Gaussian band reject filter is given as follows:

✓✓

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



Gaussian Band Reject Filter

# Image Compression

- **Image compression** is a process used in image processing to reduce the amount of data required to represent an image.
- The main goal of image compression is to minimize the storage space or transmission bandwidth needed for an image while maintaining an acceptable level of quality.
- Image compression techniques can be broadly classified into two categories: **lossless** and **lossy** compression.



## 1. Lossless Compression:

- No loss of image quality; the original image can be perfectly reconstructed from the compressed data.
- Common techniques include Huffman coding, Run-length encoding (RLE), and Lempel-Ziv-Welch (LZW) coding.
- Used in applications where exact replication is crucial, such as medical imaging and technical drawings.

## 2. Lossy Compression:

- Some loss of image quality; the original image cannot be perfectly reconstructed, but the loss is often imperceptible to human eyes.
- Common techniques include JPEG (Joint Photographic Experts Group), MPEG (Moving Picture Experts Group), and wavelet-based compression.
- Used in applications where a small loss in quality is acceptable, such as web images and multimedia content.

### Applications

- **Storage:** Reducing the amount of storage needed for large image databases.
- **Transmission:** Decreasing the bandwidth required to send images over the internet or other communication channels.
- **Efficient Processing:** Speeding up image processing tasks by working with smaller data sets.

# Key Concepts in Image Compression

## 1. Compression Ratio

The **compression ratio** is a measure of the effectiveness of a compression algorithm. It is defined as the ratio of the original image size to the compressed image size.

$$\text{Compression Ratio} = \frac{\text{Original Image Size}}{\text{Compressed Image Size}}$$

For example, if an original image is 10 MB and the compressed image is 2 MB, the compression ratio is 5:1.

## 2. Average Length of Code

The **average length of code** refers to the average number of bits used to represent each symbol (e.g., pixel value) in the compressed image. This concept is particularly relevant in lossless compression techniques that use variable-length coding schemes, such as Huffman coding.

The average length of code can be calculated using the formula:

$$L_{\text{avg}} = \sum_{i=1}^n P_i \cdot l_i$$

where:

- $P_i$  is the probability of occurrence of the  $i$ -th symbol.
- $l_i$  is the length of the codeword for the  $i$ -th symbol.
- $n$  is the total number of different symbols.

# Coding Redundancy (Huffman Coding)

- Huffman Coding is a technique of compressing data to reduce its size without losing any of the details. It was first developed by David Huffman in the 1950s.
- Huffman Coding is generally useful to compress the data in which there are frequently occurring characters.
- Huffman coding uses a variable-length code for each character in the file. The code lengths are determined by the frequency of each character in the file. The most common characters are assigned the shortest codes, and the less common characters are assigned longer codes.
- This technique can be used to compress any type of file, but it is most commonly used to compress text files.
- Huffman coding is a data compression technique that can be used in a variety of applications. Here are just a few examples:
  1. Image compression:
  2. Audio compression:
  3. Text compression:
  4. Data transmission:



B	C	A	A	D	D	D	C	C	A	C	A	C	A	C
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Each character occupies 8 bits. There are a total of 15 characters in the above string. Thus, a total of  $8 * 15 = 120$  bits are required to send this string.

Using the Huffman Coding technique, we can compress the string to a smaller size.

Huffman coding first creates a tree using the frequencies of the character and then generates code for each character.

Huffman coding is done with the help of the following steps.

1. Calculate the frequency of each character in the string.

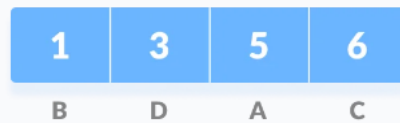
1	6	5	3
B	C	A	D

Frequency of string

Once the data is encoded, it has to be decoded. Decoding is done using the same tree.

Huffman Coding prevents any ambiguity in the decoding process using the concept of **prefix code** ie. a code associated with a character should not be present in the prefix of any other code. The tree created above helps in maintaining the property.

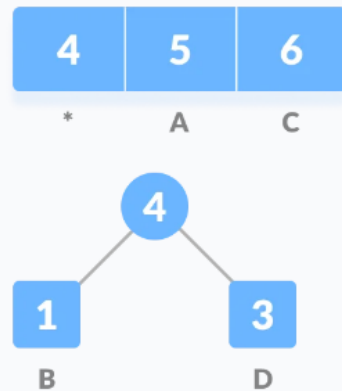
2. Sort the characters in increasing order of the frequency. These are stored in a priority queue  $Q$ .



Characters sorted according to the frequency

3. Make each unique character as a leaf node.

4. Create an empty node  $z$ . Assign the minimum frequency to the left child of  $z$  and assign the second minimum frequency to the right child of  $z$ . Set the value of the  $z$  as the sum of the above two minimum frequencies.

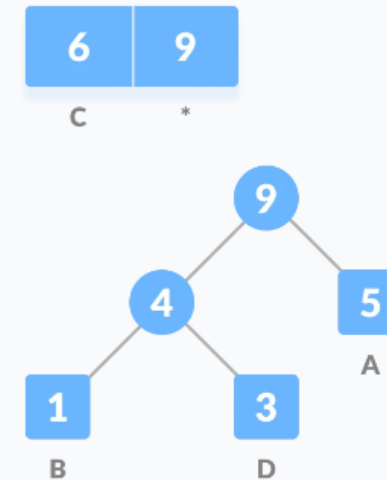


Getting the sum of the least numbers

5. Remove these two minimum frequencies from  $Q$  and add the sum into the list of frequencies (\* denote the internal nodes in the figure above).

6. Insert node  $z$  into the tree.

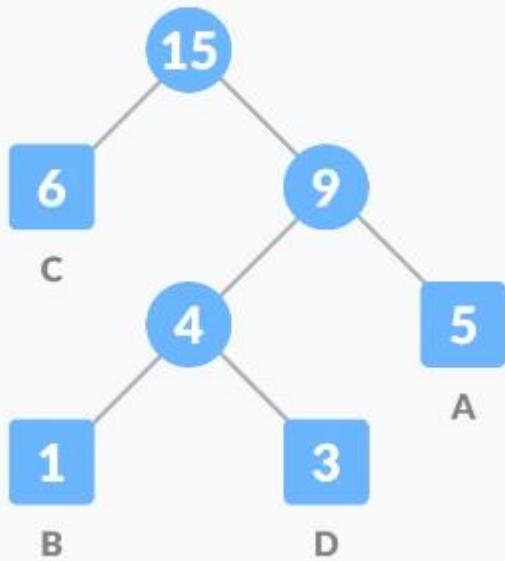
7. Repeat steps 3 to 5 for all the characters.



Repeat steps 3 to 5 for all the characters.

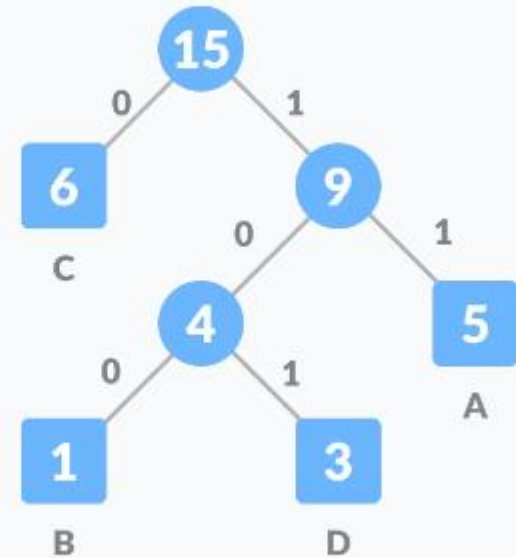
15

\*



Repeat steps 3 to 5 for all the characters.

8. For each non-leaf node, assign 0 to the left edge and 1 to the right edge.



Assign 0 to the left edge and 1 to the right edge

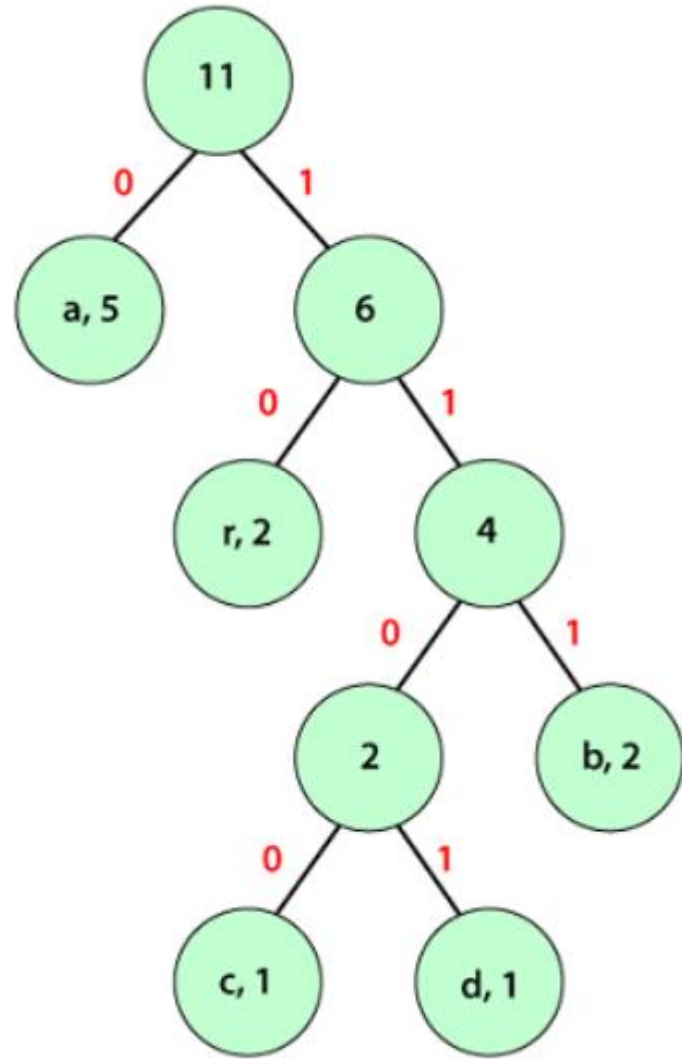
Character	Frequency	Code	Size
A	5	11	$5 * 2 = 10$
B	1	100	$1 * 3 = 3$
C	6	0	$6 * 1 = 6$
D	3	101	$3 * 3 = 9$
$4 * 8 = 32$ bits	15 bits		28 bits

Without encoding, the total size of the string was 120 bits. After encoding the size is reduced to  $32 + 15 + 28 = 75$ .

## Huffman Encoding Example

Suppose, we have to encode string **abracadabra**. Determine the following:

- i. Huffman code for All the characters
- ii. Average code length for the given String
- iii. Length of the encoded string



Character	Frequency	Code	Code Length
A	5	0	1
B	2	111	3
C	1	1100	4
D	1	1101	4
R	2	10	2

Now we can encode the string **(abracadabra)** that we have taken above.

0 111 10 0 1100 0 1101 0 111 10 0



### (ii) Average Code Length for the String

The average code length of the Huffman tree can be determined by using the formula given below:

$$\text{Average Code Length} = \sum (\text{frequency} \times \text{code length}) / \sum (\text{frequency})$$

$$= \{ (5 \times 1) + (2 \times 3) + (1 \times 4) + (1 \times 4) + (2 \times 2) \} / (5+2+1+1+2)$$

$$= \mathbf{2.09090909}$$

### (iii) Length of the Encoded String

The length of the encoded message can be determined by using the following formula:

$$\text{length} = \text{Total number of characters in the text} \times \text{Average code length per character}$$

$$= 11 \times 2.09090909$$

$$= \mathbf{23 \text{ bits}}$$

# Huffman Coding Algorithm

```
create a priority queue Q consisting of each unique character.  
sort then in ascending order of their frequencies.  
for all the unique characters:  
    create a newNode  
    extract minimum value from Q and assign it to leftChild of newNode  
    extract minimum value from Q and assign it to rightChild of newNode  
    calculate the sum of these two minimum values and assign it to the value of newNode  
    insert this newNode into the tree  
return rootNode
```

**Lab 10: Implement Huffman Coding in Java/Python/C++**

# Interpixel Redundancy (Run Length Coding)

✓ Run-length coding (RLC) exploits the repetitive nature of the image

- ✓ ☐ RLC tries to identify the **length** of the **pixel values** and encodes the image in the form of a **run**
- ✓ ☐ Each row of the image is written as a sequence
- ✓ ☐ Then **length** is represented as a **run of black and white pixels**. This is known as Run-length coding
- ✓ ☐ It is very effective way compressing an image
- ✓ ☐ If required, **further compression** can be done using **variable length coding** to code the run lengths themselves

**Example:** Given a sample binary image. Apply Run-length Coding.

Solutions → ① Horizontal RLC

\* Run Length vectors ⇒ (0,5)

\* No. of vectors = 6 (0,3), (1,2)

\* Max. length = 5  
↓  
3 bits in binary

(1,5)  
(1,5)  
(1,5)

\* No. of bits per pixel = 1 (0 or 1)

\* Total no. of pixels =  $6 \times (3+1)$   
 $= 24$  ✓

\* No. of pixels for original image =  $5 \times 5 = 25$  ✓

\* Compression Ratio =  $\frac{25}{24} = 1.042:1$   
=

\* →

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

5×5

Apply Run-length Coding.

② Vertical RLC

\* Run length vectors

✓ (0, 2), (1, 3)

✓ (0, 2), (1, 3)

(0, 2), (1, 3)

(0, 1), (1, 4)

(0, 1), (1, 4)


\* No. of vectors = 10

\* 3 bits

\* 1 bit per pixel

\* Total no. of pixels  $\Rightarrow$   
 $= 10 \times (3 + 1) = 40$  ✓

\* original Image =  $5 \times 5 = 25$

\* compression Ratio =  $\frac{25}{40} = 0.625$  :1  


↓

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

5×5

## Benefits of Run Length Encoding

1. **Compression Efficiency:** Effective for images with large regions of uniform color.
2. **Simplicity:** Easy to implement and understand.
3. **Lossless:** No loss of data during compression.



# Psychovisual Redundancy (IGS Coding)

Lossy Compression

Improved Gray Scale Quantization (IGS)

- Some Data is lost / Some Noise is added
- Irreversible process
- Compression Ratio is High. { 50% Data Compression }

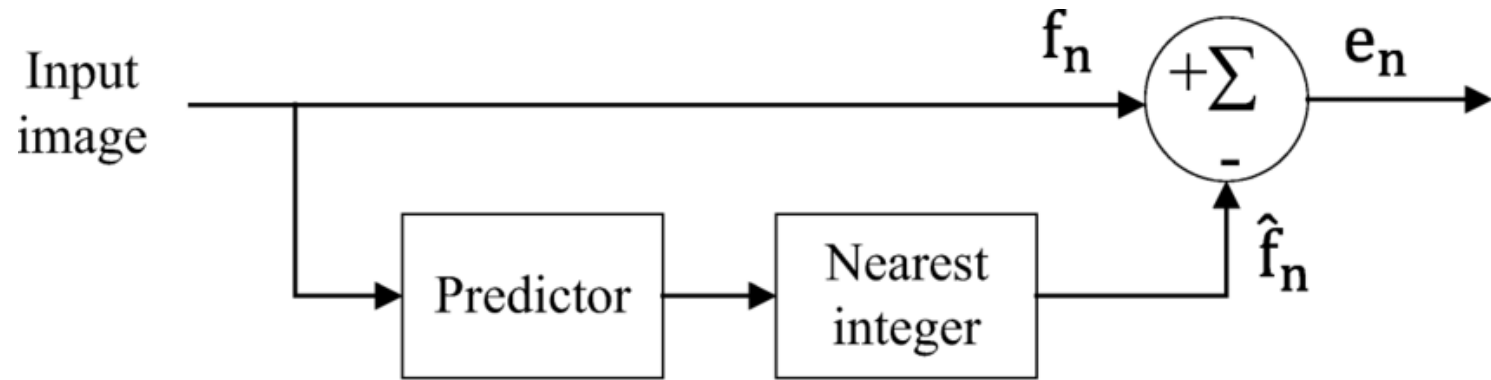


IGTS Code for  $x(n) = \{100, 110, 124, 140, 130, 160, 255, 210\}$

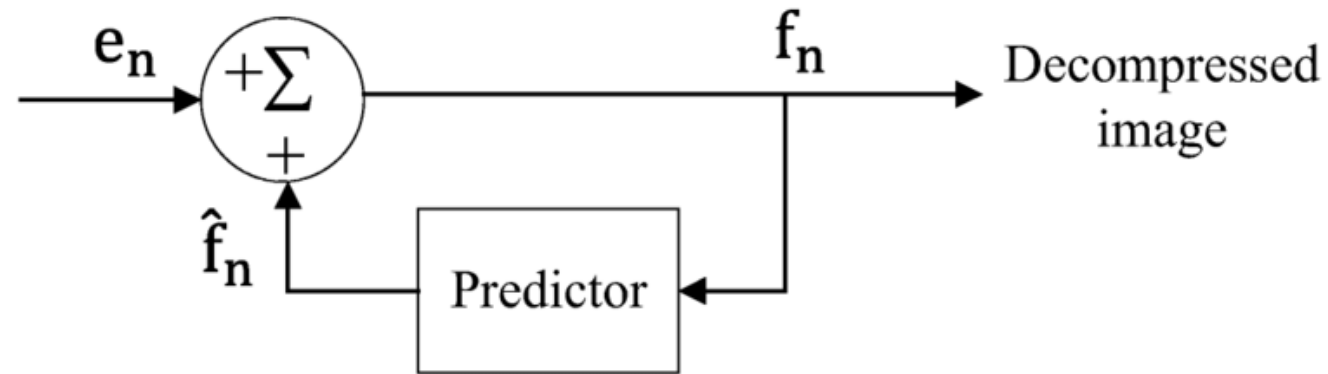
Decimal	Binary Eq		Gray L Eq		IGTS
	HB	LB	HB	LB	
100	1100	0100	1100	<u>0100</u>	1100 (6)
110	0110	<sup>1</sup> 0100	0111	<u>0010</u>	0111 (7)
		1110			
		0010			
124	0111	0010	0111	<u>1110</u>	0111 (7)
		1100			
		1110			
140	1000	<sup>1</sup> 1110	1001	<u>1010</u>	1001 (9)
		1100			
		1010			
130	1000	<sup>1</sup> 0010	1000	<u>1100</u>	1000 (8)
		1100			
		1100			
160	1010	<sup>1</sup> 0000	1010	1100	1010 (10)
		0000			
		1100			

IGS	Decimal Eq $2^n \times \text{IGS}$ $n = 4$	Error. $ \text{Original} - \text{Decimal} $ Decimal after IGS
6	$16 \times 6 = 96$	4
7	$16 \times 7 = 112$	2
7	112	12
9	144	4
8	128	2

# Lossless predictive coding model



(a)



(b)

Lossless predictive coding model: a Encoder; b Decoder



This technique **eliminates** the **inter-pixel dependencies** by predicting new information, which is obtained by taking the **difference** between the actual and predicted value of the pixel

- ☐ The **encoder** takes a **pixel** of the **input image**  $f_n$
- ☐ The **predictor** predicts the anticipated value of that pixel using past inputs
- ☐ Predicted value is rounded to the nearest integer value, which is known as **predicted value**
- ☐ The **error** is the **difference between** the actual and the predicted value
- ☐ This error is sent across the channel. The same predictor is used in the decoder side to predict the value.
- ☐ Reconstructed image is:

# Steps in Lossless Predictive Coding

## 1. Initialization:

- The process begins by selecting a prediction method and initializing the predictor.

## 2. Prediction and Error Calculation:

- For each pixel, predict its value based on previously encoded pixels.
- Calculate the prediction error as the difference between the actual pixel value and the predicted value.

## 3. Entropy Coding:

- Encode the sequence of prediction errors using an appropriate entropy coding method.

## 4. Transmission/Storage:

- The encoded data is then stored or transmitted.

## 5. Decoding:

- The decoder receives the encoded data and decodes the prediction errors.
- Using the same prediction method, the decoder reconstructs the original pixel values by adding the prediction errors to the predicted values.



**Example:** Consider the pixel {23, 34, 39, 47, 55, 63}. Demonstrate the predictive coding.

Value	Predictive Coding
23	23
34	$34 - 23 = 11$
39	$39 - 34 = 5$
47	$47 - 39 = 8$
55	$55 - 47 = 8$
63	$63 - 55 = 8$

\* It saves 6 bits

\* max. value in original sequence  
= 63

↓

requires 6 bits + 1 bit for sign

\* No. of bits required to code original message =  $6 \times 7 = \underline{\underline{42}}$  ✓

\* Predicted coded sequence

→ max. value is 23

↓

(5 bits + 1 bit for sign)

→ Total no. of bits required =  
 $6 \times 6 = \underline{\underline{36}}$  ✓



- ❑ If the **difference** crosses the **threshold limit**, it creates a problem known as **Overloading**
- ❑ Solution of this is to ignore the differences and use the original message for coding

Consider the pixel {23, **64**, 39, 47, 55, 63}

Value	Predictive Coding
23	23 ✓
64	$64 - 23 = 41$
39	$39 - 64 = -25$ ✓
47	$47 - 39 = 8$ ✓
55	$55 - 47 = 8$ ✓
63	$63 - 55 = 8$ ✓

$$* \text{ original sample} = 6 \times (6 + 1) = \underline{\underline{42}}$$

$$\begin{aligned}
 * \text{ After } &\Rightarrow \\
 &= 5 \times (5 + 1) + 1(6 + 1) \\
 &= 5 \times 6 + 7 \\
 &= 30 + 7 = \underline{\underline{37}}
 \end{aligned}$$

## Advantages

- **Perfect Reconstruction:** The original image can be perfectly reconstructed, with no loss of information.
- **Efficient Compression:** Effective for images with high spatial redundancy, leading to lower entropy in prediction errors.

## Disadvantages:

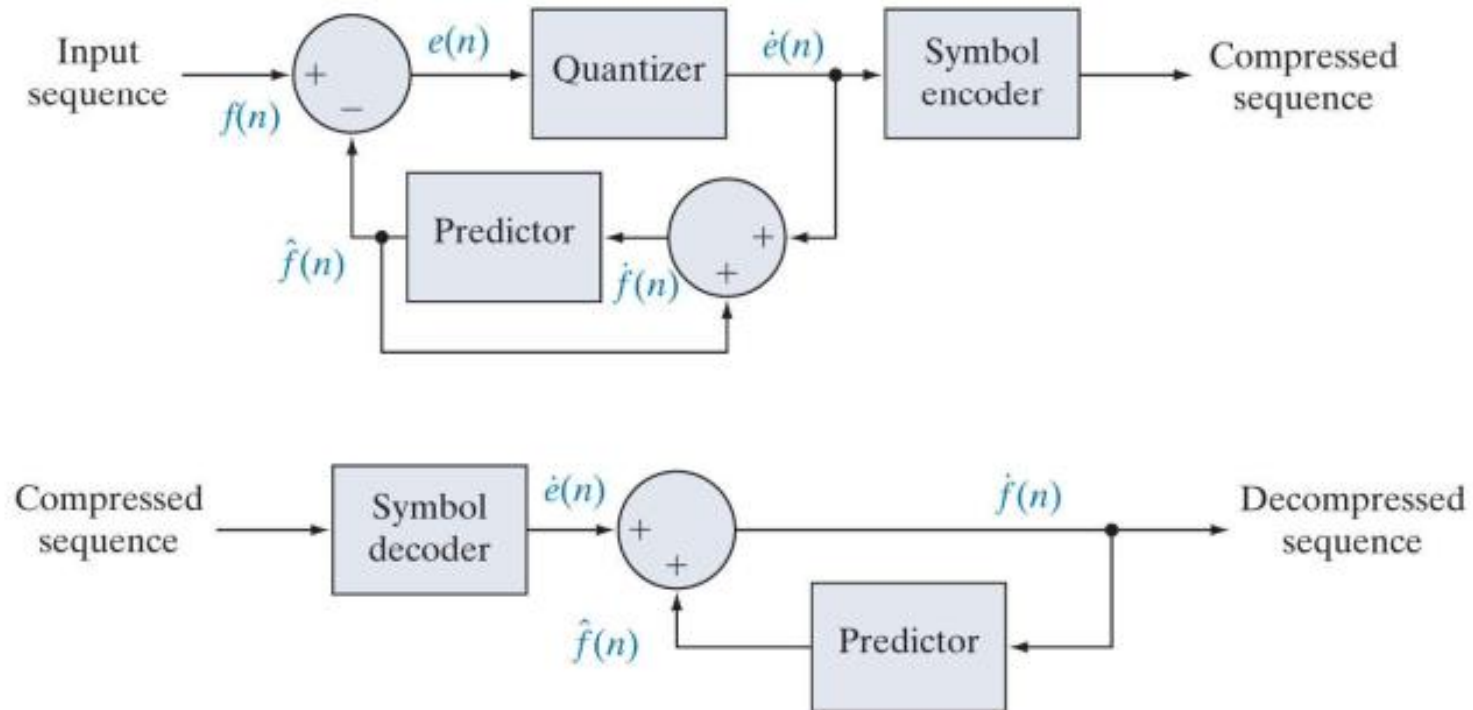
- **Complexity:** The prediction and entropy coding processes can be computationally intensive.
- **Variable Efficiency:** The compression ratio may not be as high as lossy methods for certain types of images (e.g., images with high levels of detail or noise).

## Applications

- Lossless predictive coding is used in applications where preserving the original image data is crucial, such as:
- Medical imaging (e.g., DICOM format)
- Archival of important documents and photographs
- Some scientific and technical imaging applications (e.g., remote sensing)

# Lossy predictive coding model

a  
b



**FIGURE 8.38**

A lossy predictive coding model. (a) encoder, (b) decoder.

- ✓ ☐ Lossy Compression Algorithms incur a **loss of information**
- ✓ ☐ This loss is known as **distortion**
- ✓ ☐ This data loss is **acceptable** if it is tolerable
- ✓ ☐ **Compression ratio** of these algorithms are **very large**

**Lossy predictive coding is an extension  
of the idea of lossless predictive coding**

Lossy predictive coding is a technique used in image processing to compress images by predicting pixel values based on neighboring pixels and then encoding only the difference (or error) between the actual and predicted values. This method exploits the spatial redundancy within images, where adjacent pixels often have similar values.

Consider the pixel {23, **64**, 39, 47, 55, 63}

Value	Predictive Coding
23	23 ✓
64	$64 - 23 = 41$ ✓
39	$39 - 64 = -25$ ✓
47	$47 - 39 = 8$ ✓
55	$55 - 47 = 8$ ✓
63	$63 - 55 = 8$ ✓

overloading  
=

31

$6 \times 6$   
 $= 36$   
bits

5 bits

max. 5 bits are required  
+ 1 sign bit = 6

→ 31 value

→ drastically reduces  
no. of bits

→ loss of information  
increases

## Advantages and Disadvantages

- **Advantages:**

- **Efficient Compression:** By focusing on the differences (which are typically smaller and more predictable), lossy predictive coding can achieve high compression ratios.
- **Simpler Decoder:** The decoding process is straightforward as it only involves reversing the prediction and adding the quantized error.

- **Disadvantages:**

- **Loss of Quality:** Due to quantization, there is an inherent loss of information which can lead to artifacts in the reconstructed image.
- **Error Propagation:** Prediction errors can propagate, potentially magnifying reconstruction errors if not properly managed.

- **Applications**

- Lossy predictive coding is widely used in various image and video compression standards, such as JPEG, MPEG, and H.264/AVC. In these standards, predictive coding helps to reduce the amount of data required to store and transmit images and videos without significantly compromising visual quality.



Construct Huffman code for each gray level. Calculate the compression ratio and the relative data redundancy assuming if 3-bit code is used to code the gray level instead of Huffman code. [7+3 =10]

7. Explain lossy predictive coding in brief.

9. Describe lossless predictive coding model with a suitable block diagram.

8. Explain Contra-harmonic Mean Filters used for image restoration.

3.Explain the contra-harmonic mean filters used in image restoration.

5. What do you mean by Lossy Predictive Coding? Explain it with a suitable block diagram.(6)

5. Construct Huffman code for each gray level given and find the compression ratio and coding efficiency.

Gray Level	0	1	2	4	5	6	7
No. of Pixels	30	35	38	15	10	38	80