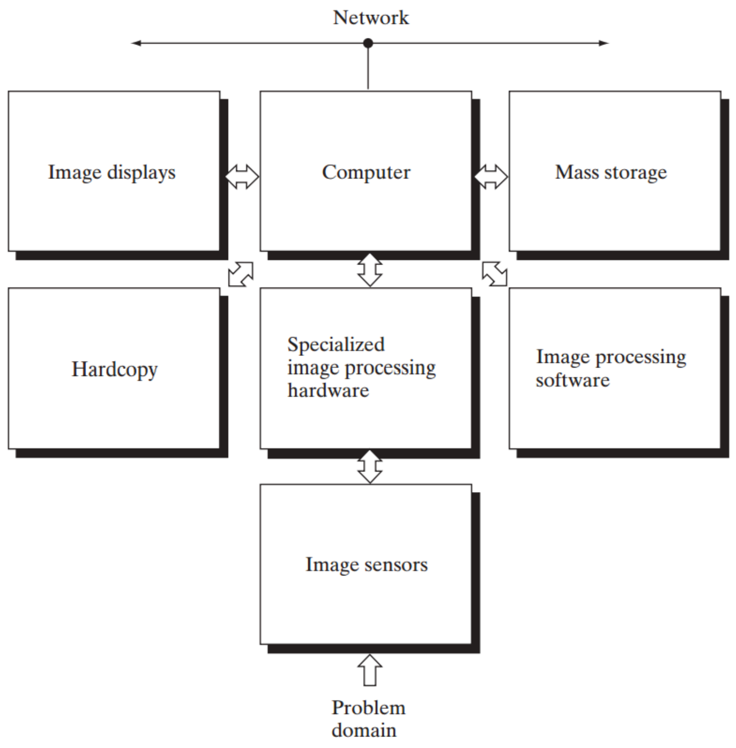
DIP EX

M1Q1) Explain the components of an Image Processing System.

A **Digital Image Processing System** typically consists of several interconnected components that work together to acquire, process, and display images. Each component plays a vital role in transforming raw image data into useful visual or analytical information.



Main Components of an Image Processing System

1. **Image Sensing and Acquisition**
    - Involves capturing the image using sensors.
    - A combination of a physical sensing device and a digitizer (ADC) is used.
    - Example: Camera or scanner capturing an image and converting it into digital format.
2. **Preprocessing**

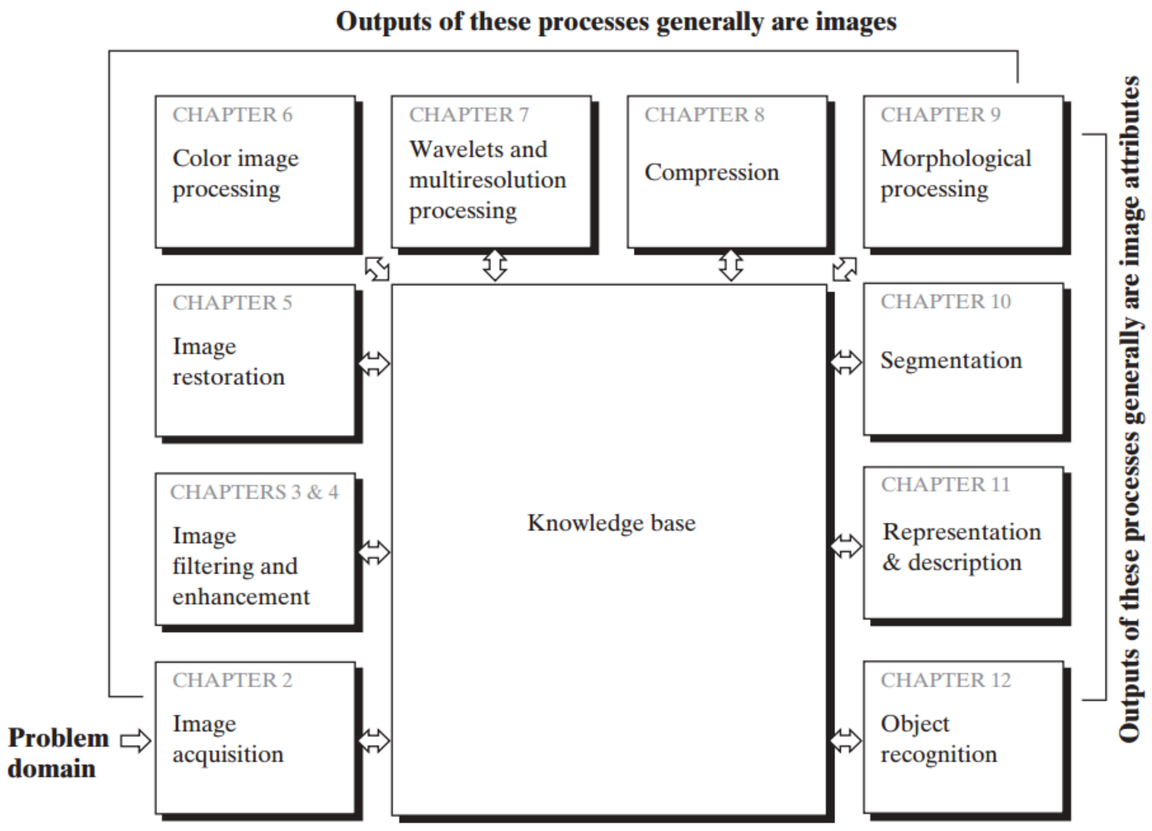
- Enhances the image quality before further processing.
- Common tasks include noise reduction, contrast enhancement, and resizing.
- Often improves the performance of subsequent steps like segmentation.

3. **Segmentation**
   - Divides the image into its constituent parts or objects.
   - A critical step in image analysis, as operations like object recognition depend on it.
   - Example: Separating a tumor region from an MRI scan.
4. **Representation and Description**
   - Once segmented, an object is represented for further processing.
   - Descriptors like boundary, shape, texture, or size are extracted.
   - Converts pixel data into features suitable for recognition.
5. **Recognition (or Classification)**
   - Assigns a label (e.g., "vehicle", "person", "tumor") to an object based on the description.
   - Can use statistical, neural network, or structural methods.
6. **Knowledge Base**
   - Contains domain-specific knowledge to guide processing tasks.
   - For example, expected sizes, shapes, or locations of objects can assist segmentation or recognition.
7. **Image Display and Visualization**
   - Provides tools for displaying the processed image to human observers.
   - Includes monitors, printers, and graphical user interfaces.
8. **Image Storage**
   - Digital images are often stored for future retrieval or analysis.
   - Efficient compression and file formats (like JPEG, PNG) are important considerations.

9. **Communication**
    - Allows image transmission across systems via networks.
    - Essential for remote sensing, telemedicine, etc.

M1Q2) Explain the fundamental steps in Digital Image Processing.

Digital image processing involves a sequence of operations applied to digital images to enhance them or extract useful information. The **fundamental steps** provide a general framework that most digital image processing tasks follow.



Outputs of these processes generally are images
CHAPTER 6 Color image processing
CHAPTER 7 Wavelets and multiresolution processing
CHAPTER 8 Compression
CHAPTER 9 Morphological processing
CHAPTER 5 Image restoration
CHAPTER 10 Segmentation
CHAPTERS 3 & 4 Image filtering and enhancement
Knowledge base
CHAPTER 11 Representation & description
CHAPTER 2 Image acquisition
CHAPTER 12 Object recognition
Problem domain
Outputs of these processes generally are image attributes

1. Image Acquisition

- **First step**: Capturing the image using sensors (e.g., camera, scanner).
- Often includes preprocessing such as:
    - Digitization (analog-to-digital conversion)
    - Calibration or noise removal

2. Image Preprocessing

- **Purpose**: Improve image quality for further processing.
- Common operations:
  - Noise reduction
  - Contrast enhancement
  - Image resizing or cropping
- Typically deals with **gray-level transformations** and **filtering**.

3. Image Segmentation

- **Objective**: Partition an image into meaningful regions.
- Isolates objects or areas of interest (e.g., detecting tumors, identifying roads).
- Techniques include:
  - Thresholding
  - Edge detection
  - Region growing

4. Image Representation

- Converts segmented data into a form suitable for analysis.
- Can be:
  - **Boundary-based** (e.g., contours)
  - **Region-based** (e.g., pixel sets)

5. Image Description (Feature Extraction)

- Quantitative attributes are derived from representations.
- Examples:
  - Shape (e.g., circularity, elongation)
  - Texture (e.g., smoothness, coarseness)
  - Moments and statistical measures

6. Image Recognition

- Assigns a label to an object (e.g., car, tree, tumor).

- Based on extracted features.
- Techniques:
  - Pattern recognition
  - Neural networks
  - Machine learning

7. Image Interpretation

- **High-level processing** that assigns meaning to recognized objects.
- Used in applications like:
  - Scene understanding
  - Decision making in AI systems

Supporting Element: Knowledge Base

- Contains **prior information** used in:
  - Segmentation
  - Recognition
  - Interpretation
- Example: Shape/size of known organs for medical image processing

---

M1Q3) Explain formation of image in human eye with example.

The **human eye** functions like a **biological camera**. Image formation in the human eye is crucial to understanding how visual information is perceived and how digital systems attempt to replicate this perception.

1. Structure of the Human Eye

Key components involved in image formation:

- **Cornea**: Transparent outer surface that **refracts incoming light**.
- **Aqueous Humor**: Fluid-filled space behind the cornea.
- **Iris**: Colored part of the eye that controls **pupil size**.

- **Pupil**: Central opening in the iris, controls **amount of light** entering the eye.
- **Lens**: Flexible, focuses light onto the retina by changing shape (**accommodation**).
- **Vitreous Humor**: Gel-like substance maintaining eye shape.
- **Retina**: Light-sensitive layer where image is formed.
- **Fovea**: Region on retina with **highest visual acuity** (densely packed cones).
- **Optic Nerve**: Transmits visual signals to the brain.

2. Image Formation Process

1. **Light enters through the cornea and pupil**.
2. The **lens adjusts shape** to focus the light rays.
3. **Light is focused onto the retina**, forming an **inverted and real image**.
4. **Photoreceptors (rods and cones)** on the retina convert light to electrical signals.
5. Signals are transmitted to the **brain via the optic nerve**.
6. The **brain processes the signals** and interprets them as upright images.

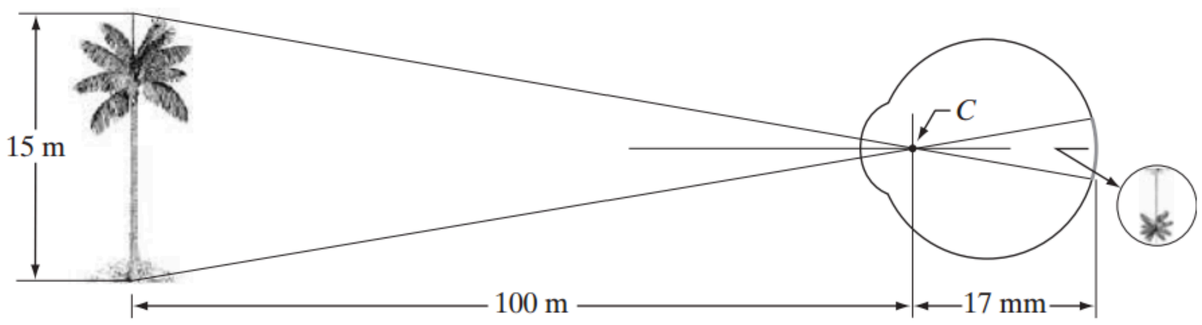
3. Rods and Cones

- **Rods**: Sensitive to low light, no color vision.
- **Cones**: Sensitive to color and fine details.
- The **fovea** contains only cones → responsible for sharp central vision.

Example of Image Formation

If a person looks at a **tree 15 meters high** from a distance of **100 meters**, the eye's lens focuses the light onto the retina. Using geometric optics, the height of the image formed on the retina can be calculated using the ratio of distances:

$\frac{\text{Image Height}}{\text{Retinal Distance (17 mm)}} = \frac{15\text{ m}}{100\text{ m}}$ $\text{Image Height} = \frac{15\text{ m} \times 17\text{ mm}}{100\text{ m}} = 2.55\text{ mm}$



Comparison to a Camera (not required but if asked in exam)

Component	Human Eye	Camera
Aperture	Pupil	Lens aperture
Light Sensor	Retina	CMOS/CCD sensor
Lens Adjustment	Ciliary muscles	Autofocus mechanism
Image Formation	Inverted, real	Inverted, real

M1Q4) Explain various image sensing and acquisition methods.

Image sensing and acquisition refer to the process of capturing a scene and converting it into a digital image. This step forms the **first stage** in any digital image processing system.

1. Components of an Image Acquisition System

- **Physical device**: Captures incoming energy (light, X-rays, etc.).
- **Sensor**: Converts physical energy to electrical signals.
- **Digitizer (ADC)**: Converts analog signal to digital form (pixels).

📌 Together, the combination of sensor and digitizer is often called an image sensor.

2. Methods of Image Sensing and Acquisition

a) Single Sensor Acquisition

- A single imaging sensor with a mechanical setup to scan the scene.
- Common in flatbed scanners or digitizing tablets.

- Uses mirrors, lenses, and movement to capture full image line-by-line.

Example: A **flatbed scanner** moving over a document.

b) Sensor Strips

- A 1D array of sensors (sensor strip) scans across the image plane.
- Typically used in airborne or satellite imaging systems.
- Captures a single row of data at a time as the platform moves forward.

Example: **Remote sensing satellites** scanning Earth strip by strip.

c) Sensor Arrays (2D Sensors)

- 2D array of sensors (e.g., CCD or CMOS arrays).
- Captures entire image in one shot.
- Common in digital cameras, webcams, and smartphones.

Example: **Mobile phone camera sensor**

d) Specialized Sensors

- **Infrared (IR) sensors** for thermal imaging.
- **X-ray sensors** in medical and industrial systems.
- **Ultrasound transducers** in medical imaging.

3. A Simple Image Formation Model

- The image intensity captured can be modeled as:

  $f(x, y) = i(x, y) \cdot r(x, y)$

  where:

  - $i(x, y)$ = illumination component
  - $r(x, y)$ = reflectance component

- This model helps understand how objects reflect or emit energy captured by the sensor.

Summary Table (reference only)

Method	Description	Common Use Case
Single Sensor	Scanning using one sensor & mirror	Flatbed scanners
Sensor Strips	1D array moving across the image	Remote sensing (satellites)
Sensor Arrays	2D matrix capturing entire image	Cameras, medical devices
Specialized Sensors	Based on X-ray, IR, etc.	Medical, industrial, astronomy

M1Q5) With a diagram, explain how image is acquired using sensor strips.

Sensor Strip-Based Image Acquisition

A **sensor strip** consists of a **1D linear array** of sensors arranged in a row. The complete image is acquired by moving the strip or the scene **perpendicularly** to the sensor line — capturing one image line at a time.

Working Principle

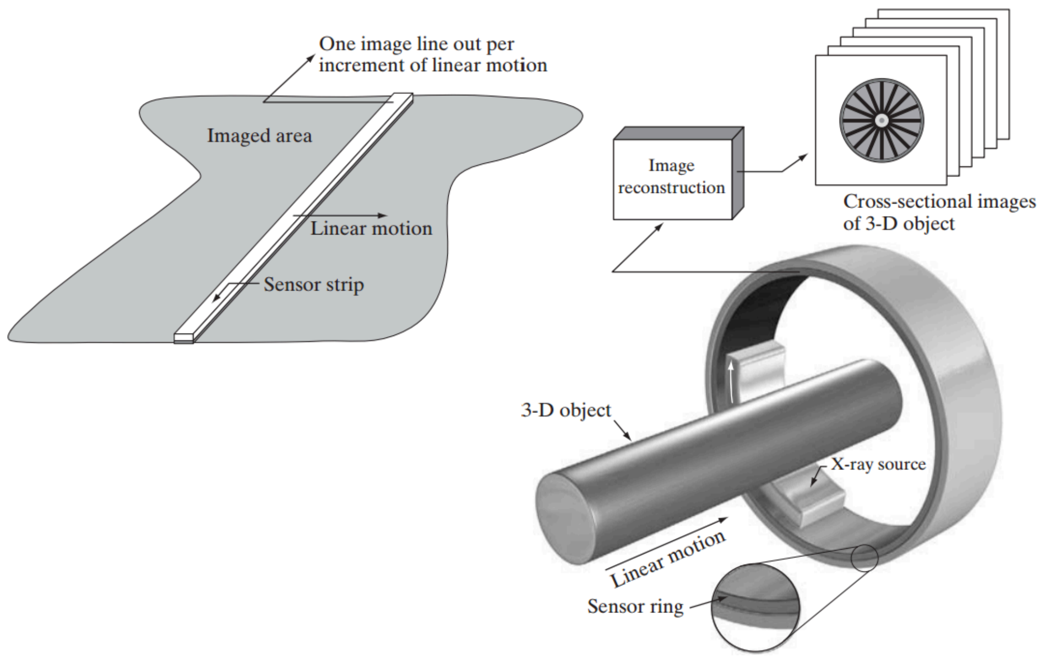
1. A **1D sensor array** captures **one line (row)** of the image at a time.
2. Either the sensor moves across the scene **or** the scene moves past the stationary sensor strip.
3. After each line is captured, the sensor moves slightly and captures the next line.
4. These rows are stacked digitally to form the **2D image**.

Applications

- **Remote sensing satellites** (Earth observation from space).
- **Industrial scanners** (assembly line quality control).
- **Medical scanning devices** (CT or MRI slice acquisition).

Advantages

- High-resolution images.
- Controlled motion and timing yield precise imaging.
- Ideal for continuous or very large scenes.



(a) Image acquisition using a linear sensor strip. (b) Image acquisition using a circular sensor strip.

📌 Alternatively, in satellite systems, the satellite moves forward while the strip scans lines perpendicular to the path.

Example: Satellite Imaging

- A satellite flying over Earth carries a **sensor strip aligned across the flight path**.
- As the satellite moves forward, the strip records each ground line sequentially.
- The collected lines are assembled into a complete image of the terrain.

M1Q6) Explain the process of image sampling and quantization in digital signal processing.

Image **sampling and quantization** are the two key steps that convert a **continuous image** (analog) into a **digital image** for processing.

1. Image Sampling

- **Sampling** refers to the process of **dividing a continuous image** into a **grid of discrete points** (pixels).

- These discrete points represent spatial coordinates (x, y).
- Each sampled location will later be assigned an intensity value through quantization.

Mathematically:

- The continuous image function $f(x, y)$ is sampled at intervals Δx and Δy, creating a grid:

  $f(m\Delta x, n\Delta y) \Rightarrow f(m, n)$

  where mm and nn are integers.

2. Image Quantization

- **Quantization** is the process of **mapping the continuous intensity values** of sampled pixels to a **finite number of gray levels**.
- This turns each sampled value into a digital number (usually binary).

Example:

- If using 8-bit quantization → intensity values range from 0 to 255 (i.e., 2^8 levels).
- A continuous intensity value of 128.734 might be rounded to 129.

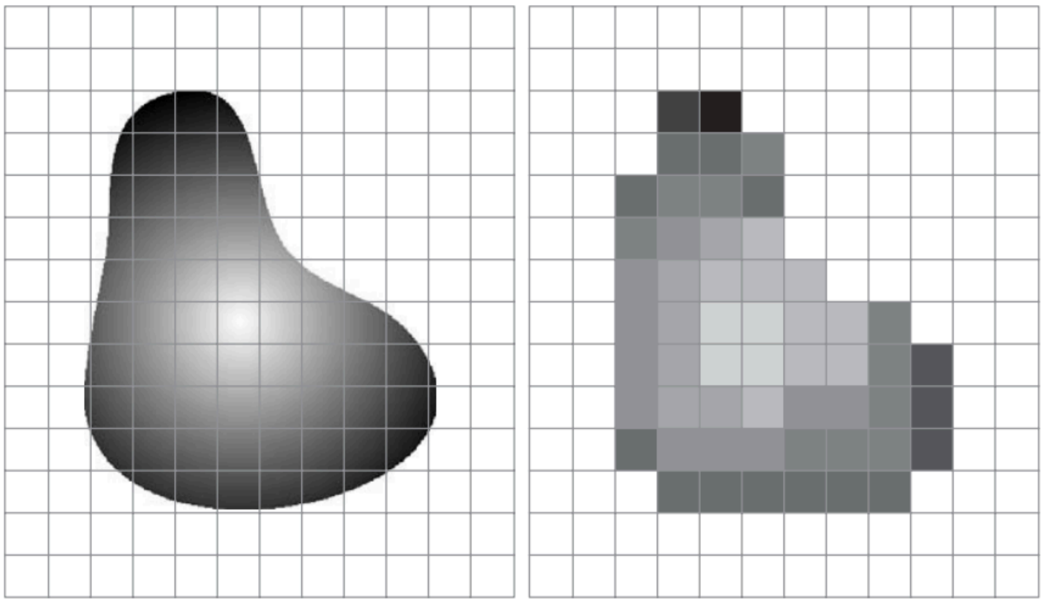
3. Combined Process

- **Sampling** defines the **spatial resolution** (number of pixels).
- **Quantization** defines the **gray-level resolution** (number of bits per pixel).

4. Effect of Sampling and Quantization

- **Too coarse sampling** → **loss of detail**, aliasing.
- **Too coarse quantization** → visible banding and false contours.

📌 Proper choice of sampling rate and quantization levels is essential for high-quality digital imaging.


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

5. Resolution Definitions

- **Spatial Resolution**: How finely an image is sampled (pixels per unit area).
- **Gray-Level (Intensity) Resolution**: Number of possible intensity levels (based on quantization).

Summary Table (reference only)

Concept	Sampling	Quantization
Domain	Spatial (x, y)	Intensity (gray levels)
Purpose	Selects pixel locations	Assigns value to each pixel
Affects	Spatial resolution	Gray-level resolution

M1Q7) Explain some basic relationships between pixels.

1. Neighbors of a Pixel

For a pixel at coordinates (x, y), its neighbors are defined based on **connectivity criteria**:

- **4-neighbors (N_4)**: Horizontal and vertical neighbors

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

- **Diagonal neighbors (N_D)**:

  $(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$
- **8-neighbors (N_8)**: Combination of 4- and diagonal neighbors

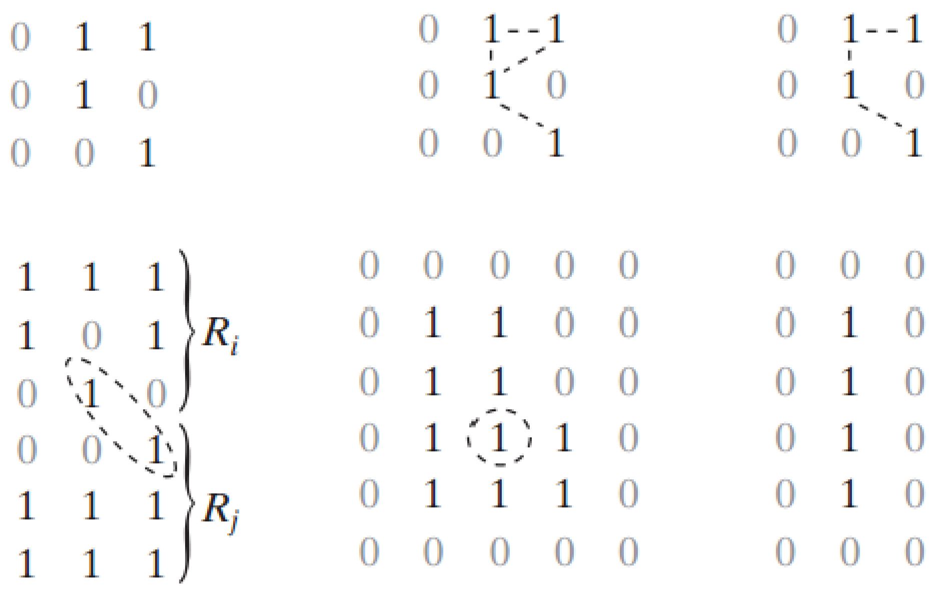
  → Total of 8 neighboring pixels.

2. Adjacency

Defines which pixels are considered **connected** based on neighborhood and gray level criteria:

- **4-adjacency**: Shared edge with similar gray level.
- **8-adjacency**: Shared edge or corner with similar gray level.
- **m-adjacency (mixed)**: Combines both but avoids ambiguity in multiple path connections.

📌 Used in image segmentation and object detection to define object boundaries.



(a) An arrangement of pixels.
(b) Pixels that are 8-adjacent (adjacency is shown by dashed lines; note the ambiguity).
(c) m-adjacency.
(d) Two regions (of 1s) that are adjacent if 8-adjecency is used.
(e) The circled point is part of the boundary of the 1-valued pixels only if 8-adjacency between the region and background is used.
(f) The inner boundary of the 1-valued region does not form a closed path, but its outer boundary does.

3. Connectivity

- A set of pixels is **connected** if there is a path between any two pixels using a type of adjacency (4-, 8-, or m-).
- **Connected components** are maximal sets of connected pixels having the same intensity.

4. Regions and Boundaries

- A **region** is a connected set of pixels with similar properties (e.g., intensity, texture).
- The **boundary** is the set of pixels separating a region from its background.

Example: In a binary image, all foreground pixels (1s) connected together form a region.

5. Distance Measures

Used to compute distances between two pixels $p = (x_1, y_1)$ and $q = (x_2, y_2)$:

- **Euclidean Distance**:

  $$D_E(p, q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- **City-Block Distance** (D4):

  $$D_4(p, q) = |x_2 - x_1| + |y_2 - y_1|$$

- **Chessboard Distance** (D8):

  $$D_8(p, q) = \max(|x_2 - x_1|, |y_2 - y_1|)$$

📌 Used in clustering, path planning, and proximity analysis.

Summary Table (reference only)

Concept	Meaning
4-neighbors	Horizontal and vertical neighbors
8-neighbors	All immediate neighbors (4 + diagonal)
Adjacency	Defines pixel connection rules
Connectivity	Defines object or region based on adjacency
Region	Connected pixels with similar properties
Boundary	Pixels separating region from background

Concept	Meaning
Distance	Measure of closeness between two pixel points

M1Q8) Consider the image segment shown. Let V = {1, 2} and compute the length of the shortest 4-, 8- and m-path between p and q. If particular path does not exist between these two points, explain why?

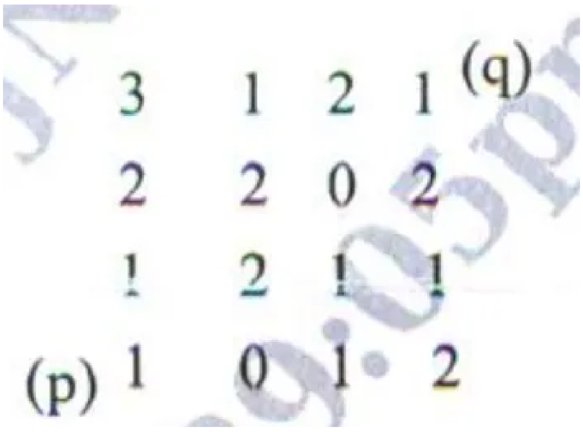


Figure M1Q8

To solve this question, we need to compute the **shortest path** between pixel **p** (located at bottom-left) and pixel **q** (top-right) in the given image using three connectivity types: **4-path**, **8-path**, and **m-path**, using the value set $V = \{1, 2\}$.

Step 1: Matrix Representation with Coordinates

Let's represent the 4×4 matrix with coordinates for clarity:

```
(0,0)  3   (0,1) 1   (0,2) 2   (0,3) 1
(1,0)  2   (1,1) 2   (1,2) 0   (1,3) 2
(2,0)  1   (2,1) 2   (2,2) 1   (2,3) 1
(3,0)  1   (3,1) 0   (3,2) 1   (3,3) 2
```

- Point **p = (3,0)**
- Point **q = (0,3)**
- Valid values (in set V) are: **1** and **2**

Step 2: Identify Valid Pixels (value ∈ V = {1,2})

Let’s mark valid pixels (✓) and invalid (×):

```
(0,0) 3 ×   (0,1) 1 ✓   (0,2) 2 ✓   (0,3) 1 ✓
(1,0) 2 ✓   (1,1) 2 ✓   (1,2) 0 ×   (1,3) 2 ✓
(2,0) 1 ✓   (2,1) 2 ✓   (2,2) 1 ✓   (2,3) 1 ✓
(3,0) 1 ✓   (3,1) 0 ×   (3,2) 1 ✓   (3,3) 2 ✓
```

Step 3: Shortest Path using Different Connectivity

→ 4-Connectivity

- Move only in **N4**: up, down, left, right (no diagonals)
- Valid path:

  $$(3,0) \to (2,0) \to (2,1) \to (1,1) \to (0,1) \to (0,2) \to (0,3)$$
- All values are in VV, and only horizontal/vertical steps are used.
- **Length = 6**

Answer: 4-path length = 6

→ 8-Connectivity

- Allows diagonal moves too.
- Shortest valid 8-path:

  $$(3,0) \to (2,1) \to (1,1) \to (0,2) \to (0,3)$$
- All values in VV, and diagonal steps are allowed.

Answer: 8-path length = 4

→ m-Connectivity

- Mix of 4- and diagonal moves.
- **Diagonal move is allowed only if the 4-connected neighbors of both pixels are not in VV**.
- Let's test shortest candidate 8-path:

  $$(3,0) \to (2,1) \to (1,1) \to (0,2) \to (0,3)$$

Let’s check diagonals:

- (3,0) to (2,1): diagonal
    - (3,1) = 0 → Not in V → okay
    - (2,0) = 1 → in V → ❌ Not allowed under **m-connectivity**

Since diagonal (3,0)-(2,1) is invalid, we fall back to the same path as in 4-connectivity:

$$(3,0) \to (2,0) \to (2,1) \to (1,1) \to (0,1) \to (0,2) \to (0,3)$$

Answer: m-path length = 6

M1Q9) Explain in detail five fields that use digital image processing.

Digital image processing is widely applied across diverse fields where image analysis, enhancement, or recognition is crucial. Below are **five important fields** along with examples from the textbook.

1. Medical Imaging

- Enhances the quality and diagnostic value of medical scans.
- Enables 2D and 3D reconstructions for accurate diagnosis.
- **Applications**:
    - **X-rays**: Detection of fractures, dental issues.
    - **CT scans**: 3D brain/tumor imaging.
    - **MRI**: Soft tissue visualization.
    - **PET scans**: Cancer detection and brain mapping.
- **Example**: head CT scan used for neurosurgical planning.

2. Remote Sensing

- Used by satellites and drones to capture Earth's surface.
- Multispectral and hyperspectral imaging detect environmental changes.
- **Applications**:
    - Forest monitoring
    - Urban planning

    - Crop health analysis
    - Climate change studies
- **Example**: LANDSAT satellite images in different spectral bands over Washington D.C.

3. Industrial Inspection

- Automates quality control and fault detection.
- Reduces human error and increases throughput.
- **Applications**:
    - PCB defect detection
    - Packaging and pill inspection
    - Assembly line robotics
- **Example**: pill inspection using digital image processing.

4. Astronomy

- Enhances telescope-captured images for space exploration.
- Removes noise, detects celestial objects, and enables 3D modeling.
- **Applications**:
    - Galaxy classification
    - Supernova analysis
    - Solar activity mapping
- **Example**: gamma-ray image of the Cygnus Loop supernova remnant.

5. Law Enforcement and Forensics

- Enhances surveillance footage and automates recognition tasks.
- Aids in criminal identification and digital evidence analysis.
- **Applications**:
    - Face and fingerprint recognition
    - License plate reading (ANPR)
    - Surveillance video enhancement

- **Example**: Plate recognition systems using digital cameras and OCR techniques.

M2Q1) Explain the need for image transforms. Discuss two-dimensional orthogonal and unitary transforms with their properties.

Need for Image Transforms

Image transforms are mathematical tools that convert images from one domain (typically spatial) to another (typically frequency or transform domain). The primary reasons for using image transforms in digital image processing include:

1. **Energy Compaction**: Transforms concentrate image information into fewer coefficients (usually low-frequency), making them suitable for compression.
2. **Decorrelation**: Transforms help remove redundancy between pixel values. This is especially important for statistical processing and compression.
3. **Feature Extraction**: Many features become more distinguishable in the transform domain.
4. **Filtering and Enhancement**: Filtering operations like sharpening or smoothing are easier and more efficient in the transform domain.
5. **Restoration and Analysis**: Transforms help in noise reduction and image restoration using frequency domain analysis.

Two-Dimensional Orthogonal and Unitary Transforms

A 2D orthogonal/unitary transform is a generalization of 1D transforms applied to images. For an N×NN \times N image u(m,n)u(m, n), the transform is defined as:

Forward Transform:

$$v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m,n) a_{k,l}(m,n)$$

Inverse Transform:

$$u(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(k,l) a^*_{k,l}(m,n)$$

Where $a_{k,l}(m,n)$ are the orthonormal basis functions.

Separable Transforms: If the basis functions can be written as a product of 1D functions:

$a_{k,l}(m,n) = a_k(m) \cdot b_l(n)$

Then the transform is separable, reducing computation from $O(N^4)$ to $O(N^3)$ operations.

Examples:

- Discrete Fourier Transform (DFT)
- Discrete Cosine Transform (DCT)
- Hadamard Transform
- Haar Transform
- Karhunen-Loève Transform (KLT)

Properties of Unitary Transforms

1. **Energy Conservation**: The total energy (sum of squares) is preserved under transformation.

   $\sum |u(m,n)|^2 = \sum |v(k,l)|^2$
2. **Rotation Interpretation**: A unitary transform can be viewed as a rotation in vector space—just changing the basis while preserving length.
3. **Energy Compaction**: Most of the image energy is compacted into fewer coefficients. Particularly true for KLT and DCT.
4. **Decorrelation**: Transformed coefficients are less correlated than original pixels, aiding in compression and analysis.
5. **Entropy Preservation**: Entropy (a measure of information) is preserved under unitary transforms.

M2Q2) Define the Discrete Cosine Transform (DCT) and derive the 4x4 DCT matrix. State its properties.

1. Definition of DCT

The **Discrete Cosine Transform (DCT)** is a **real-valued orthogonal transform** that converts a finite sequence of data points into a sum of cosine functions oscillating at different frequencies.

For a 1D signal of length NN, the DCT is defined as:

$$(k) = \alpha(k) \sum_{n=0}^{N-1} u(n) \cos\left[\frac{\pi(2n+1)k}{2N}\right], \quad 0 \le k \le N-1$$

Where:

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{if } k = 0 \\ \sqrt{\frac{2}{N}}, & \text{if } k \ge 1 \end{cases}$$

2. 4×4 DCT Matrix Derivation

The DCT matrix C has elements:

$$C_{k,n} = \alpha(k) \cos\left[\frac{\pi(2n+1)k}{2N}\right], \quad k, n = 0, 1, 2, ..., N-1$$

Let's compute for $N = 4$:

Let $\alpha(0) = \frac{1}{2}, \alpha(k) = \frac{1}{\sqrt{2}} for k \neq 0 k \neq 0$

$$C = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2}\cos\left(\frac{\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{3\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{5\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{7\pi}{8}\right) \\ \frac{1}{2}\cos\left(\frac{2\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{6\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{10\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{14\pi}{8}\right) \\ \frac{1}{2}\cos\left(\frac{3\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{9\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{15\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{21\pi}{8}\right) \end{bmatrix}$$

Substituting cosine values and scaling:

$$C \approx \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.653 & 0.271 & -0.271 & -0.653 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.271 & -0.653 & 0.653 & -0.271 \end{bmatrix}$$

This matrix can be normalized to be **orthonormal**.

3. Properties of DCT

1. **Real and Orthogonal**

   $C = C^T = C^{-1}$

   All computations are real (no imaginary components).

2. **Energy Compaction**

   Concentrates most energy in a few low-frequency coefficients — highly efficient for compression (e.g., JPEG).

3. **Fast Transform**

Can be computed in $O(N \log N)$ operations — faster than DFT.

4. **Approximates Karhunen-Loève Transform (KLT)**

   Especially for **highly correlated signals**, DCT is nearly as efficient as KLT.

5. **Separable in 2D**

   2D DCT can be applied by computing DCT along rows and then columns (or vice versa).

6. **Eigenvectors of Tridiagonal Matrix**

   The rows of the DCT matrix are the eigenvectors of a symmetric tridiagonal matrix Q_c

4. Applications

- **Image Compression** (JPEG, MPEG)
- **Image Denoising**
- **Spectral Analysis**
- **Edge Detection**
- **Feature Extraction**

M2Q3) Perform the Discrete Cosine Transform (DCT) for N = 4 and explain its application.

1. DCT Formula for 1D Input of Size N = 4

Given a 1D signal:

$$u(n), \quad n = 0, 1, 2, 3$$

The **DCT coefficients** are computed as:

$$v(k) = \alpha(k) \sum_{n=0}^{3} u(n) \cos\left[\frac{\pi(2n+1)k}{2N}\right], \quad k = 0, 1, 2, 3$$

Where:

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & k = 0 \\ \sqrt{\frac{2}{N}}, & k = 1, 2, 3 \end{cases}$$

For $N = 4$:

- $\alpha(0) = \frac{1}{2}$
- $\alpha(1,2,3) = \frac{1}{\sqrt{2}} \approx 0.7071$

2. Example: Perform DCT for Sample Vector

Let's take an example input vector:

$u = [52,\ 55,\ 61,\ 66]$

Step 1: Compute DCT Coefficients $v(k)$

Using the formula:

$v(k) = \alpha(k) \cdot \sum_{n=0}^{3} u(n) \cdot \cos\left[\frac{\pi(2n+1)k}{8}\right]$

👉 v(0):

$v(0) = \frac{1}{2} \cdot (52 + 55 + 61 + 66) = \frac{1}{2} \cdot 234 = \boxed{117}$

👉 v(1)v(1):

$v(1) = \frac{1}{\sqrt{2}} \cdot \left(52 \cdot \cos\frac{1\pi}{8} + 55 \cdot \cos\frac{3\pi}{8} + 61 \cdot \cos\frac{5\pi}{8} + 66 \cdot \cos\frac{7\pi}{8}\right)$

Using:

- $\cos(\pi/8) \approx 0.9239$
- $\cos(3\pi/8) \approx 0.3827$
- $\cos(5\pi/8) \approx -0.3827$
- $\cos(7\pi/8) \approx -0.9239$

$v(1) \approx 0.7071 \cdot (52 \cdot 0.9239 + 55 \cdot 0.3827 + 61 \cdot (-0.3827) + 66 \cdot (-0.9239))$

$v(1) \approx 0.7071 \cdot (48.043 + 21.048 - 23.34 - 60.976) \approx 0.7071 \cdot (-15.225) \approx \boxed{-10.77}$

Likewise, compute v(2), v(3) using similar steps.

3. Application of DCT

Application Area	Description
Image Compression	DCT concentrates image energy in a few low-frequency components. JPEG uses 8×8 DCT blocks for compression.
Denoising	High-frequency DCT coefficients can be thresholded to remove noise.
Feature Extraction	DCT coefficients capture significant texture and shape information.
Watermarking	Data can be embedded in mid-frequency DCT coefficients for robustness.

Why DCT is Preferred in Practice

- **Real-valued**: Easier to compute and interpret.
- **Energy Compaction**: Much of the image information is in the **first few DCT coefficients**.
- **Computationally Efficient**: Fast DCT algorithms reduce time complexity.

M2Q4) Explain Haar transform with equations. Derive a 4x4 Haar transform matrix and mention its properties.

1. Definition of the Haar Transform

The **Haar transform** is the simplest and oldest example of a **wavelet transform**. It is a **real, orthogonal, and symmetric** transform particularly useful in image processing for detecting **edges**, performing **compression**, and **feature extraction**.

The Haar transform computes:

- **Average (low-pass)** and
- **Difference (high-pass)** information

  from adjacent data elements.

It uses **Haar basis functions** $h_{p,q}(x)$ defined over $[0, 1]$, and the discrete Haar transform is obtained by sampling these functions.

2. Mathematical Definition (1D Case)

Given a signal u(n)u(n) of size $N = 2^n$, the Haar transform coefficients $v(k)$ are computed using:

$v(k) = \sum_{n=0}^{N-1} u(n) \cdot h_k(n)$

Where $h_k(n)$ is the Haar basis function corresponding to index kk.

3. Derivation of the 4×4 Haar Transform Matrix

The 4×4 Haar transform matrix H_4 is:

$$H_4 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Each row represents a Haar basis vector:

- Row 1: DC average (low frequency)
- Row 2: Coarse difference
- Row 3: Fine difference (left pair)
- Row 4: Fine difference (right pair)

4. Properties of the Haar Transform

1. **Real and Orthogonal**

   $H^T = H^{-1}$

   Preserves energy and allows perfect reconstruction.

2. **Fast Computation**

   Computational complexity is $O(N)$ — faster than DFT/DCT.

3. **Sequency Ordered Basis**

   Rows are ordered by the number of zero crossings.

4. **Captures Local Detail**

   Efficient in capturing local changes (edges).

5. **Poor Energy Compaction**

   Compared to DCT or KLT, Haar has lower energy packing efficiency for natural images.

6. **Edge Detection Capability**

   Haar coefficients represent intensity differences — useful in detecting edges.

5. Applications in Image Processing

- **Edge Detection**
- **Image Compression**
- **Feature Extraction**
- **Multi-resolution analysis**

M2Q5) Discuss the properties of the Discrete Fourier Transform (DFT) including translation, periodicity, rotation, and convolution theorem.

1. Translation Property (Circular Shift)

If a signal $u(n)$ is **circularly shifted** by l, the DFT of the shifted signal is:

$$u((n-l) \bmod N) \xrightarrow{\text{DFT}} v(k)e^{-j\frac{2\pi}{N}kl}$$

This means that **shifting** in the spatial domain corresponds to a **phase shift** in the frequency domain.

2. Periodicity Property

The DFT is periodic in both domains with period N:

$$u(n+N) = u(n), \quad v(k+N) = v(k)$$

This implies the spectrum repeats every N samples.

3. Rotation Property (2D DFT)

For a 2D signal, rotation in the **spatial domain** results in an equivalent rotation in the **frequency domain**:

$$f(x,y) \to f(R[x,y]) \quad \Rightarrow \quad F(\omega_1,\omega_2) \to F(R[\omega_1,\omega_2])$$

Where RR is a rotation matrix. The orientation of features in the image is preserved in the frequency spectrum.

4. Convolution Theorem

Circular Convolution (1D and 2D):

The DFT of a circular convolution of two sequences is the **product of their DFTs**:

$$x(n) \circledast h(n) \xrightarrow{\text{DFT}} X(k) \cdot H(k)$$

Linear Convolution (via padding):

To perform linear convolution using DFT:

- Zero-pad both sequences
- Compute DFTs
- Multiply
- Take inverse DFT

2D Case:

$$x_3(m,n) = x_1(m,n) * x_2(m,n) \Rightarrow X_3(k,l) = X_1(k,l) \cdot X_2(k,l)$$

5. Additional DFT Properties

Property	**Explanation**
Linearity	DFT of a sum = sum of the DFTs
Energy Conservation	$\|u(n)\|^2 = \|v(k)\|^2$– Parseval's Theorem
Conjugate Symmetry	DFT of real-valued signals is conjugate symmetric
Separability (2D DFT)	2D DFT = row-wise 1D DFT + column-wise 1D DFT
Basis Images	DFT basis images are complex exponentials $W^{(km+ln)}$
Fast Computation	Fast Fourier Transform (FFT) computes DFT in $O(N \log N)$ operations

M2Q6) State and prove the properties of the two-dimensional DFT.

1. Definition of 2D DFT

Let $f(m,n)$ be a 2D image of size $M \times N$.

The 2D DFT and its inverse are:

Forward DFT:

$$F(u,v) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\cdot e^{-j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

Inverse DFT:

$$f(m,n) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\cdot e^{j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

2. Properties of 2D DFT (With Proofs)

1. Linearity

If:

$$f_1(m,n) \xrightarrow{\text{DFT}} F_1(u,v), \quad f_2(m,n) \xrightarrow{\text{DFT}} F_2(u,v)$$

Then for any scalars a,ba, b:

$$af_1(m,n) + bf_2(m,n) \xrightarrow{\text{DFT}} aF_1(u,v) + bF_2(u,v)$$

✅ *Proof:* Follows directly from linearity of summation and exponential terms.

2. Translation (Shift Property)

If:

$$f(m-m_0, n-n_0) \xrightarrow{\text{DFT}} F(u,v)\cdot e^{-j2\pi\left(\frac{um_0}{M}+\frac{vn_0}{N}\right)}$$

✅ *Proof:* Substituting shift into DFT definition introduces a phase factor.

3. Periodicity

Both $f(m,n)$ and $F(u,v)$ are **periodic** with periods M and N in respective dimensions.

$$f(m+M,n) = f(m,n), \quad F(u+M,v) = F(u,v)$$

✅ *Proof:* DFT is based on periodic complex exponentials.

4. Conjugate Symmetry

If $f(m,n)$ is **real-valued**, then:

$$F(M-u, N-v) = F^*(u,v)$$

✅ *Proof:* Comes from complex conjugation of exponential terms and reality of input.

5. Rotation

If $f(x, y)$ is rotated by θ, the DFT also rotates by θ:

$$f(R[x, y]) \xrightarrow{\text{DFT}} F(R[u, v])$$

Where R is a rotation matrix.

✅ *Proof:* Uses the change of variable in 2D integrals/summations.

6. Scaling

In spatial domain, scaling the image results in **inverse scaling** in frequency domain. However, in DFT, scaling is not straightforward due to fixed sampling and aliasing.

7. Convolution Theorem

If:

$$f_1(m, n) * f_2(m, n) \xrightarrow{\text{DFT}} F_1(u, v) \cdot F_2(u, v)$$

Where $*$ denotes **circular convolution**.

✅ *Proof:* By substituting convolution definition into DFT and rearranging sums.

8. Correlation Theorem

Cross-correlation becomes multiplication of one DFT and the **complex conjugate** of the other:

$$\text{corr}(f_1, f_2) \xrightarrow{\text{DFT}} F_1(u, v) \cdot F_2^*(u, v)$$

9. Parseval's Theorem

$$\sum_{m,n} |f(m, n)|^2 = \frac{1}{MN} \sum_{u,v} |F(u, v)|^2$$

✅ *Proof:* Preserves total energy between domains.

3. Summary Table of 2D DFT Properties

Property	Spatial Domain	Frequency Domain
Linearity	af1+bf2af_1 + bf_2	aF1+bF2aF_1 + bF_2
Translation	f(m−m0,n−n0)f(m - m_0, n - n_0)	F(u,v)⋅e−j2π(⋅)F(u, v) \cdot e^{-j2\pi(\cdot)}
Convolution	f1∗f2f_1 \ast f_2	F1⋅F2F_1 \cdot F_2
Correlation	f1⋆f2f_1 \star f_2	F1⋅F2∗F_1 \cdot F_2^*

Property	Spatial Domain	Frequency Domain
Periodicity	f(m+M,n+N)=f(m,n)f(m+M,n+N) = f(m,n)	F(u+M,v+N)=F(u,v)F(u+M,v+N) = F(u,v)
Conjugate Symmetry	Real ff	F(M−u,N−v)=F∗(u,v)F(M - u, N - v) = F^*(u,v)
Energy Preservation	(\sum	f

M2Q7) Explain the algorithmic steps in computing the Haar transform.

1. What is the Haar Transform?

The **Haar transform** is the simplest example of a **discrete wavelet transform**. It operates by recursively computing **averages and differences** of adjacent pixel values and is useful for **image compression**, **feature extraction**, and **edge detection**.

2. Key Idea

Given an input vector of length $N = 2^n$, the Haar transform recursively performs:

- **Average computation**: $a = \frac{x_1+x_2}{\sqrt{2}}$
- **Difference computation**: $d = \frac{x_1-x_2}{\sqrt{2}}$

This process is repeated on the averages to compute multi-resolution levels of the signal.

3. Algorithmic Steps for 1D Haar Transform

Let the input signal be $f = [f_0, f_1, ..., f_{N-1}]$, with N a power of 2.

Step 1: Initialize

Set current array $f^{(0)} = f$

Step 2: Iterative Averaging and Differencing

For each level $l = 0$ to $\log_2 N - 1$:

1. Let $f^{(l)}$ have size $N_l = \frac{N}{2^l}$

2. For $i = 0$ to $N_l/2 - 1$:
   - Compute **average**:
     $a_i = \frac{f^{(l)}[2i]+f^{(l)}[2i+1]}{\sqrt{2}}$
   - Compute **difference**:
     $d_i = \frac{f^{(l)}[2i]-f^{(l)}[2i+1]}{\sqrt{2}}$
3. Store averages a_i in the first half and differences d_i in the second half of the next level array:
   $f^{(l+1)} = [a_0, a_1, ..., a_{N_l/2-1}, d_0, d_1, ..., d_{N_l/2-1}]$

Repeat until only one average remains.

5. Extension to 2D Haar Transform

For 2D signals (images):

1. Apply **1D Haar transform on all rows**
2. Apply **1D Haar transform on all columns** of the result

This gives the **2D Haar transformed image**, used in compression or multi-resolution analysis.

M2Q8) Compute and illustrate orthogonal transforms for specific image matrix cases.

1. What Are Orthogonal Transforms?

An **orthogonal transform** is a linear transform that preserves energy and is invertible using its transpose:

$$A^{-1} = A^T \quad \Rightarrow \quad A^T A = I$$

Common examples include:

- Discrete Cosine Transform (DCT)
- Discrete Fourier Transform (DFT)
- Haar Transform
- Walsh-Hadamard Transform

2. Simple Image Matrix Example

Let's take a **2×2 image block**:

$$F = \begin{bmatrix} 100 & 150 \\ 200 & 250 \end{bmatrix}$$

We will apply the **2×2 Haar transform**, whose orthogonal matrix is:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

3. Apply 2D Haar Transform

For orthogonal transforms:

$$T = H \cdot F \cdot H^T$$

Step 1: Compute $H \cdot F$

$$H \cdot F = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 100 & 150 \\ 200 & 250 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 300 & 400 \\ -100 & -100 \end{bmatrix}$$

Step 2: Compute $(H \cdot F) \cdot H^T$

Since H is symmetric, $H^T = H$

$$T = \frac{1}{\sqrt{2}} \begin{bmatrix} 300 & 400 \\ -100 & -100 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 700 & -100 \\ -200 & 0 \end{bmatrix}$$

4. Final Transformed Matrix

$$T = \begin{bmatrix} 350 & -50 \\ -100 & 0 \end{bmatrix}$$

5. Interpretation

- **T(0,0) = 350**: Average energy (DC component)
- Other values represent **details**, like edges and variations

6. Properties Verified

Property	Observed In Result
Energy preserved	Total energy before ≈ after
Orthogonality	Inverse is transpose

Property	Observed In Result
Decorrelation	Off-diagonal entries capture edges

7. General Process for Any Orthogonal Transform

Let:

- F : image block
- A : orthogonal transform matrix

Then:

$$T = A \cdot F \cdot A^T \quad \text{(Forward transform)}$$

$$F = A^T \cdot T \cdot A \quad \text{(Inverse transform)}$$

M3Q1) Discuss how periodic noise can be removed by frequency domain filtering.

What is Periodic Noise?

- **Periodic noise** is usually introduced by **electrical interference**, such as power line disturbances.
- It appears as **sinusoidal patterns** or **repetitive artifacts** in the spatial domain.
- In the **frequency domain**, this noise shows up as **distinct spikes or impulses** at specific frequency locations (not at the origin).

Why Use Frequency Domain Filtering?

- Because periodic noise has a well-defined **frequency signature**, it is **easier to isolate and remove** in the frequency domain than in the spatial domain.

Steps to Remove Periodic Noise in Frequency Domain

1. Fourier Transform of the Image

- Apply **2D Discrete Fourier Transform (DFT)** to convert the image to frequency domain.

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

2. Identify Noise Peaks

- Periodic noise appears as **bright symmetric spots away from the center** in the frequency spectrum.
- Usually occurs in **conjugate pairs**.

3. Design a Notch Filter

- Notch filters are used to **block (reject)** the frequencies where periodic noise is located.
- Types:
  - **Ideal Notch Reject Filter**
  - **Butterworth Notch Reject Filter**
  - **Gaussian Notch Reject Filter**

A general notch reject filter removes noise at (u_0, v_0) and its symmetric point $(-u_0, -v_0)$.

4. Multiply with Filter

- Multiply the frequency domain image with the **notch reject filter mask**:
  $G(u, v) = H(u, v) \cdot F(u, v)$

5. Inverse DFT

- Apply **inverse DFT** to get the filtered image in the **spatial domain**:
  $g(x, y) = \mathrm{IDFT}\{G(u, v)\}$

Advantages of Frequency Domain Filtering

- Effectively **targets specific frequencies** without affecting the entire image.
- Works better for **sinusoidal or structured noise** than spatial filtering.

Limitations

- Requires precise **location of noise frequencies**.
- Misidentifying noise spikes may result in loss of image detail.

Summary Table (reference only)

Step	Description
1. Apply DFT	Convert image to frequency domain
2. Identify noise peaks	Detect spikes caused by periodic noise
3. Design notch filters	Create filters to block noisy frequencies
4. Apply filter	Multiply filter with DFT image
5. Inverse DFT	Convert back to spatial domain

M3Q2) Explain the following methods to estimate degradation function used in image restoration: i) Estimation by image observation, ii) Estimation by experiment.

i) Estimation by Image Observation

- This is a **subjective, visual technique**.
- Based on the appearance of the degraded image, we infer the **type and extent of degradation**.
- Most often used when:
  - No prior data is available about the degradation.
  - The degradation is **simple and well-understood** (e.g., motion blur, defocus).

Steps:

1. Visually analyze the degraded image (e.g., blurring, streaking, ringing).
2. Hypothesize the form of degradation (e.g., uniform linear motion).
3. Choose a parametric model for the degradation function $h(x, y)$ (or $H(u, v)$ in frequency domain).
4. Estimate parameters (e.g., length and direction of motion blur) via trial-and-error or intuitive reasoning.
5. Use this estimated function in restoration filters.

Example:

- An image appears smeared in one direction → assume **motion blur**.

- Use motion degradation model:
  $h(x,y) = \frac{1}{L} \quad \text{for a motion length of L pixels}$
- Adjust L until restored image looks acceptable.

Limitation:

- Highly **subjective**.
- Not accurate for complex or unknown degradations.

ii) Estimation by Experiment

- Uses **controlled experiments** to estimate the degradation function objectively.
- Requires knowledge or control over the image formation process.

Steps:

1. Use a known test image (e.g., a delta function or a known pattern).
2. Pass the test image through the **same degradation process** (e.g., same camera, same blur, same conditions).
3. Measure the output image.
4. The resulting degraded image represents the **impulse response** $h(x,y)$, or its DFT gives $H(u,v)$.

Example:

- If degradation is caused by camera defocus:
  - Capture an image of a **point source** (e.g., a small light in dark background).
  - The spread of that point gives the **Point Spread Function (PSF)**, i.e., the degradation function.

Advantages:

- More **objective** and **accurate** than visual observation.
- Can be used to model **complex systems**.

Limitations:

- Requires access to the degradation process.
- Not always feasible in real-world scenarios (e.g., satellite images).

M3Q3) List and explain any three properties of 2D-DFT. (6M)

The **2D Discrete Fourier Transform (2D-DFT)** is fundamental in frequency domain processing of images. It transforms a spatial domain image into its frequency domain representation.

Here are **three key properties** of 2D-DFT (suitable for a 6-mark answer):

1. Linearity

- The DFT is **linear**, which means:

$$\mathcal{F}\{a \cdot f(x,y) + b \cdot g(x,y)\} = a \cdot F(u,v) + b \cdot G(u,v)$$

- Here, $f(x,y)$ and $g(x,y)$ are input images, $F(u,v)$ and $G(u,v)$ are their corresponding DFTs, and a, b are constants.

✔ This helps in analyzing systems made of combined signals or images.

2. Translation (Shifting) Property

- If an image is shifted in spatial domain, its DFT undergoes **phase change**, but **magnitude remains unchanged**:

$$f(x - x_0, y - y_0) \leftrightarrow F(u,v) \cdot e^{-j2\pi\left(\frac{ux_0}{M} + \frac{vy_0}{N}\right)}$$

- Only the **phase spectrum** is affected; the **amplitude spectrum stays the same**.

✔ Used in image registration and alignment.

3. Convolution Theorem

- Convolution in spatial domain is equivalent to multiplication in frequency domain:

$$f(x,y) * h(x,y) \leftrightarrow F(u,v) \cdot H(u,v)$$

- Similarly, multiplication in spatial domain corresponds to convolution in frequency domain.

✔ Widely used in filtering and restoration.

Optional Additional Properties (If needed for elaboration or extra marks):

- **Symmetry**: The DFT of a real image is conjugate symmetric.
- **Periodicity**: The DFT is periodic with period equal to image dimensions.

Summary Table (reference only)

Property	Description	Use Case
Linearity	DFT of a linear combination = combination of DFTs	Image blending, superposition
Translation	Spatial shift changes phase only	Image registration
Convolution	Convolution ↔ Multiplication in frequency domain	Efficient filtering

M3Q4) List and describe the mathematical steps involved in histogram specification.

What is Histogram Specification?

Also known as **Histogram Matching**, histogram specification is a technique used to **transform the histogram of a given image** to closely resemble a **desired histogram**.

- Unlike **histogram equalization**, which aims for a uniform histogram, histogram specification allows **custom control** over the output appearance.

Steps Involved in Histogram Specification

Let:

- r be the gray levels in the **input image**
- s be the gray levels in the **equalized image**
- z be the gray levels in the **specified image**
- $T(r)$: transformation function based on input image
- $G(z)$: transformation function based on specified histogram

Step 1: Compute the Histogram of the Input Image

- Count how many times each gray level r_k occurs.
- Normalize to get the **probability distribution** $p_r(r_k)$.

Step 2: Compute the Cumulative Distribution Function (CDF) for Input

$s = T(r) = \sum_{j=0}^{r} p_r(r_j)$

- This is the transformation used in **histogram equalization**.

Step 3: Compute the Desired (Target) Histogram

- Choose or define a **specified histogram** $p_z(z_k)$.
- This can be:
  - Uniform
  - Gaussian
  - Based on another reference image

Step 4: Compute the CDF of the Specified Histogram

$z = G(z_k) = \sum_{j=0}^{z_k} p_z(z_j)$

Step 5: Match Intensities

- For each intensity r in the input image:
  - Compute $s = T(r)$
  - Find the value z in the specified image such that:

    $G(z) \approx T(r)$
  - Map input pixel of value r to output pixel of value z

Applications

- Enhancing image appearance to match a specific style.
- Normalizing image data for comparison.
- Making lighting consistent across image sets.

M3Q5) Describe the working of a spatial averaging filter (mean filter).

1. What is a Spatial Averaging (Mean) Filter?

A **spatial averaging filter**, also known as a **mean filter**, is a **linear spatial filter** used primarily for **image smoothing** (i.e., reducing noise and minor variations in intensity).

It works by **replacing each pixel value** in an image with the **average value of its neighbors** within a defined window or kernel.

2. Mathematical Expression

For an image $f(x, y)$, the output $g(x, y)$ after applying a mean filter is:

$g(x,y) = \frac{1}{mn} \sum_{s=-a}^{a} \sum_{t=-b}^{b} f(x+s, y+t)$

Where:

- $m \times n$: size of the filter mask (e.g., 3×3, 5×5)
- $a = \frac{m-1}{2}$, $b = \frac{n-1}{2}$
- $f(x+s, y+t)$: pixel values in the neighborhood

3. Filter Mask (Kernel)

Example: **3×3 mean filter kernel**

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

- Each output pixel is the **average of 9 neighboring pixels**.
- The kernel slides across the image and computes the mean in each position.

4. Working Principle (Steps)

1. Select a kernel size (e.g., 3×3).
2. Place the kernel over each pixel, centering it.
3. Multiply overlapping pixels with kernel weights (all equal in this case).
4. Sum the products.

5. Replace the center pixel with the average value.

5. Effect on Image

- Smooths the image by **blurring sharp edges** and **reducing random noise**.
- Removes high-frequency content (details, edges).
- Useful as a **preprocessing step** in edge detection or thresholding.

6. Limitation

- **Blurs edges** and fine details.
- Not effective for preserving important structural content.

Applications

- Removing Gaussian or uniform noise.
- Preprocessing for segmentation.
- General low-pass filtering in spatial domain.

M3Q6) Explain basic intensity transformations and contrast stretching techniques.

1. What Are Intensity Transformations?

Intensity transformations are operations that map one gray level to another using a **transformation function**. They are applied to **enhance image appearance** or to prepare an image for further analysis.

Mathematically:

$s = T(r)$

Where:

- r = input pixel intensity
- s = output pixel intensity
- T = transformation function
- $0 \leq r, s \leq L - 1$, where L is the maximum gray level (typically 256)

2. Types of Basic Intensity Transformations

a) Linear (Identity) Transformation

$s = r$

- Leaves the image unchanged.

b) Negative Transformation

$s = L - 1 - r$

- Produces a photographic **negative** of the image.
- Useful for enhancing white or gray detail in **dark regions**.

c) Logarithmic Transformation

$s = c \cdot \log(1 + r)$

- Expands dark pixel values while compressing bright ones.
- Useful for **enhancing low-intensity regions** (e.g., Fourier spectra, medical images).

d) Power-Law (Gamma) Transformation

$s = c \cdot r^\gamma$

- c and γ are constants.
- Used for **gamma correction** in display systems.
  - $\gamma < 1$: enhances dark regions
  - $\gamma > 1$: compresses bright regions

3. Contrast Stretching

Contrast stretching enhances the contrast of an image by **expanding the range of intensity levels**. Unlike histogram equalization, it's based on simple linear mapping.

a) Piecewise Linear Contrast Stretching

The most common form is **linear stretching** between two intensity limits r_1 and r_2:

$$s = \begin{cases} 0 & r < r_1 \\ \frac{(r - r_1)}{(r_2 - r_1)} \cdot (s_2 - s_1) + s_1 & r_1 \leq r \leq r_2 \\ L - 1 & r > r_2 \end{cases}$$

- r_1, r_2 : input gray level limits
- s_1, s_2 : desired output gray levels

b) Contrast Stretching Example

- If an image has intensities mostly between 100–150:
  - Map 100 → 0 and 150 → 255 to stretch contrast.
- Enhances visibility in images with **poor contrast** due to lighting issues.

M3Q7) Define histogram and explain histogram processing, including histogram equalization. Perform histogram equalization for a 2-bit image.

1. What is a Histogram in Image Processing?

A **histogram** is a graphical representation that shows the **frequency of occurrence of each intensity level** in a digital image.

- For a grayscale image with intensity levels from 0 to $L-1$, the histogram is a plot of:

  $h(r_k) = \text{number of pixels with intensity } r_k$
- r_k : intensity level
- $h(r_k)$: frequency of occurrence of r_k

2. Histogram Processing

Histogram processing includes techniques that **modify the histogram** of an image to enhance its appearance:

- **Histogram Equalization**: Improves contrast by spreading out the intensity values.
- **Histogram Specification**: Matches the histogram to a desired shape.
- **Histogram Stretching**: Linearly expands the intensity range.

3. Histogram Equalization

Goal:

Transform the intensity values so that the **output histogram is uniform**, i.e., all gray levels are equally likely.

Steps:

1. **Compute normalized histogram** (probability of each level):
   $p(r_k) = \frac{h(r_k)}{MN}$
2. **Compute cumulative distribution function (CDF)**:
   $c(r_k) = \sum_{j=0}^{k} p(r_j)$
3. **Map each input intensity to output**:
   $s_k = \text{round}\left[(L-1) \cdot c(r_k)\right]$

Where:

- L = number of intensity levels (e.g., 4 for 2-bit image)
- s_k = output level corresponding to input level r_k

4. Histogram Equalization Example for a 2-Bit Image

Given:

Gray Level r_k	Count $h(r_k)$
0	2
1	3
2	3
3	2

- Total pixels: $MN = 10$
- Number of levels: $L = 4$

Final Mapping Table

Input r_k	Output s_k
0	1
1	2
2	2
3	3

Step 1: Normalized Histogram

$p(r_k) = \frac{h(r_k)}{10} \Rightarrow [0.2,\ 0.3,\ 0.3,\ 0.2]$

Step 2: Cumulative Distribution Function (CDF)

$c(r_k) = [0.2,\ 0.2 + 0.3 = 0.5,\ 0.5 + 0.3 = 0.8,\ 0.8 + 0.2 = 1.0] = [0.2,\ 0.5,\ 0.8,\ 1.0]$

Step 3: Transform Mapping

$$s_k = \text{round}[(L-1) \cdot c(r_k)] = \text{round}[3 \cdot c(r_k)]$$

$$s_k = [\text{round}(0.6),\ \text{round}(1.5),\ \text{round}(2.4),\ \text{round}(3.0)] = [1,\ 2,\ 2,\ 3]$$

M3Q8) Explain bit-plane slicing and intensity slicing techniques, including methods for extracting bit planes from an image.

1. Bit-Plane Slicing

Bit-plane slicing involves separating a digital image into its **individual binary planes** by examining the **contribution of each bit** in the pixel's binary representation.

- For an 8-bit grayscale image, each pixel value is represented as:

  $$f(x,y) = b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$$

  where b_7 is the **most significant bit (MSB)** and b_0 is the **least significant bit (LSB)**.

Purpose:

- To **highlight different levels of detail** in an image.
- **Higher-order bits** (e.g., bit-planes 6 and 7) contain **major visual information**.
- **Lower-order bits** may contain **fine details** or **noise**.

Method to Extract Bit-Plane ii:

Given an 8-bit image $f(x,y)$, to extract the **i-th bit plane**:

$$\text{BitPlane}_i(x,y) = \text{mod}\left(\left\lfloor \frac{f(x,y)}{2^i} \right\rfloor, 2\right)$$

Where:

- $i = 0$ for LSB, up to $i = 7$ for MSB
- `mod(a, 2)` gives 0 or 1 (binary bit)

2. Intensity Slicing

Intensity slicing (also known as gray-level slicing) involves **mapping ranges of intensity values** to particular output values (often colors) to **highlight specific features** in an image.

Techniques:

a) Binary Slicing

- Map a range of intensity values to white (1), others to black (0):

$$s(x,y) = \begin{cases} 1, & \text{if } r_1 \le f(x,y) \le r_2 \\ 0, & \text{otherwise} \end{cases}$$

b) Multilevel Slicing

- Assign **different colors or intensity values** to multiple ranges:
  - 0–50 → dark blue
  - 51–150 → green
  - 151–255 → yellow
- Often used in **pseudocolour image processing**

Applications:

- **Bit-plane slicing**:
  - Image compression
  - Steganography (data hiding)
  - Visual analysis of data significance
- **Intensity slicing**:
  - Medical imaging (tumor highlighting)
  - Satellite imagery (vegetation vs. water)
  - Thermal imaging

M3Q9) Illustrate the concepts of order-statistics filters and propose their applications in image enhancement.

1. What Are Order-Statistics Filters?

Order-statistics filters are **nonlinear spatial filters** that operate on the **rank or order** of pixel values within a neighborhood (mask) rather than their actual intensity values.

- These filters **sort the pixel values** in the neighborhood and then **select one based on rank** (e.g., minimum, maximum, median).
- Commonly used for **noise reduction**, particularly for **impulse noise (salt-and-pepper)**.

2. Types of Order-Statistics Filters

a) Median Filter

- Replaces the center pixel with the **median** of the neighborhood.
- Example:

  Given a 3×3 window:

  $[10, 20, 80;\ 25, \mathbf{250}, 30;\ 40, 50, 60] \Rightarrow$
  $\text{Sorted: } [10, 20, 25, 30, 40, 50, 60, 80, 250]$

  Median = **40**, so center pixel becomes 40.
- **Application**:
  - Removes **salt-and-pepper noise**
  - Preserves **edges better** than linear filters

b) Max Filter

- Replaces the center pixel with the **maximum value** in the neighborhood.
- Enhances **bright regions**, removes **pepper noise (dark spots)**.

c) Min Filter

- Replaces the center pixel with the **minimum value** in the neighborhood.
- Enhances **dark regions**, removes **salt noise (bright spots)**.

d) Midpoint Filter

- Uses the average of **maximum and minimum** values:

  $\text{Midpoint} = \frac{\text{Max}+\text{Min}}{2}$

- Reduces **uniform noise**, especially in low-contrast images.

e) Alpha-Trimmed Mean Filter

- Removes a fixed number of highest and lowest values, then computes the mean of the rest.
- Good **trade-off** between mean and median filters.

3. Applications of Order-Statistics Filters in Image Enhancement

Filter Type	Applications
Median Filter	Removing **salt-and-pepper noise**, edge preservation
Max Filter	Enhancing **bright details**, removing **dark noise**
Min Filter	Enhancing **dark details**, removing **light noise**
Midpoint Filter	Smoothing images with **uniform or Gaussian noise**
Alpha-Trimmed Mean	Used in **noisy images with mixed noise types**

M3Q10) Discuss piecewise linear transformation functions and their roles in image processing.

1. What Are Piecewise Linear Transformation Functions?

A **piecewise linear transformation** is a type of **intensity transformation** where the mapping function is composed of **multiple linear segments**, defined over different ranges of input gray levels.

- They offer more **flexibility** than a single linear or nonlinear function.
- Widely used for **contrast enhancement**, **brightness adjustment**, and **threshold-based processing**.

2. Mathematical Form (General Concept)

The transformation function $s = T(r)$ is defined as:

$$T(r) = \begin{cases} f_1(r), & r_0 \le r < r_1 \\ f_2(r), & r_1 \le r < r_2 \\ \vdots \\ f_n(r), & r_{n-1} \le r \le r_n \end{cases}$$

Each $f_i(r)$ is a **linear function** in its range.

3. Types of Piecewise Linear Transformations

a) Contrast Stretching

- Enhances contrast by expanding the range of gray levels between two thresholds.

$$T(r) = \begin{cases} 0, & r \le r_1 \\ \frac{(r-r_1)}{(r_2-r_1)} \cdot (s_2 - s_1) + s_1, & r_1 < r < r_2 \\ L-1, & r \ge r_2 \end{cases}$$

- **Use case**: Enhancing low-contrast images due to poor lighting.

b) Gray Level Slicing

- Highlights specific intensity ranges by assigning them higher or fixed output values.
- Useful for isolating features in a specific intensity band (e.g., tumors in medical images).

c) Thresholding (Binary Transformation)

- Converts grayscale images to binary (black & white):

$$T(r) = \begin{cases} 0, & r < T \\ L-1, & r \ge T \end{cases}$$

- **Use case**: Image segmentation, document binarization.

Roles in Image Processing

Role	Description
Contrast Enhancement	Expands desired intensity ranges while suppressing others
Feature Emphasis	Highlights specific regions or objects (e.g., in medical or satellite images)
Segmentation Preparation	Used before edge detection or region growing
Brightness Adjustment	Brightens or darkens selected ranges

Advantages

- **Simple to implement**
- **Flexible and customizable**
- Efficient for **real-time processing**

M3Q11) Explain median filtering and why it is preferred for salt-and-pepper noise reduction.

Median filtering is a **nonlinear filtering method** used to clean up images by removing noise. It works by moving a small window (like 3×3 or 5×5 pixels) across the image, and at each step, it replaces the center pixel with the **median** (middle value) of all the pixel values in the window.

How It Works:

1. Choose a small window size (e.g., 3×3).
2. Move the window over the image one pixel at a time.
3. For each window position:
   - Collect all the pixel values inside the window.
   - Sort them from smallest to largest.
   - Find the **median** value (the one in the middle).
   - Replace the center pixel with this median value.

Example:

Suppose we have the following pixel values in a 3×3 window:

$$[12, 200, 14; \quad 13, 255, 15; \quad 11, 12, 13]$$

Sorted values: $[11, 12, 12, 13, 13, 14, 15, 200, 255]$

The **median** (middle value) is **13**, so the center pixel becomes 13.

Why It's Good for Salt-and-Pepper Noise:

- **Salt-and-pepper noise** appears as random black (0) or white (255) dots in an image.
- These noisy pixels are very different from their neighbors.

- The **median filter removes them** because extreme values (0 or 255) are not close to the middle value in a sorted list.
- At the same time, it **preserves edges and details** better than averaging filters, which can blur the image.

M4Q1) Explain the smoothing of images in frequency domain filtering: i) Ideal low pass filter, ii) Butterworth low pass filter, iii) Gaussian low pass filter

i) Ideal Low Pass Filter (ILPF)

Allows all frequencies **within a specified radius** D_0 to pass unchanged and **completely removes** all frequencies outside that radius.

Transfer Function:

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

Where:

- $D(u,v) = \sqrt{(u - M/2)^2 + (v - N/2)^2}$ is the distance from the center of the frequency rectangle.

Effect:

- Sharp cutoff
- Strong smoothing, but causes **ringing artifacts** (Gibbs phenomenon)

ii) Butterworth Low Pass Filter (BLPF)

A smoother alternative to the ideal filter, with a **gradual transition** between passed and blocked frequencies.

Transfer Function:

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

Where:

- n is the **order** of the filter (higher n = sharper cutoff)

- D_0 = cutoff frequency

Effect:

- Reduces ringing compared to ILPF
- Provides a **compromise** between ideal and Gaussian filters

iii) Gaussian Low Pass Filter (GLPF)

Applies a **Gaussian-shaped decay** to high frequencies with **no sharp transitions**.

Transfer Function:

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

Effect:

- **No ringing artifacts**
- Smooth and natural-looking blur
- Best for avoiding sharp transitions and artifacts

M4Q2) With the help of equations, explain how HSI color model is converted to RGB.

1. Overview of HSI to RGB Conversion

- The **HSI model** (Hue, Saturation, Intensity) is useful for color manipulation based on human perception.
- To **display or process** the image on devices (like monitors or cameras), it must be converted to the **RGB model**.
- The conversion is **piecewise**, depending on the hue value H.

2. Definitions

Let the HSI components be:

- H : Hue (in degrees: 0–360°)
- S : Saturation (0–1)

- I : Intensity (0–1)

The RGB outputs:

- $R, G, B \in [0, 1]$

3. Case-wise Equations for HSI → RGB

The conversion depends on the **sector of hue angle**:

Case 1: 0° ≤ H < 120°

Let:

$$B = I \cdot (1 - S)$$

$$R = I \cdot \left[1 + \frac{S \cdot \cos(H)}{\cos(60^\circ - H)}\right]$$

$$G = 3I - (R + B)$$

Case 2: 120° ≤ H < 240°

Let $H' = H - 120^\circ$:

$$R = I \cdot (1 - S)$$

$$G = I \cdot \left[1 + \frac{S \cdot \cos(H')}{\cos(60^\circ - H')}\right]$$

$$B = 3I - (R + G)$$

Case 3: 240° ≤ H ≤ 360°

Let $H'' = H - 240^\circ$:

$$G = I \cdot (1 - S)$$

$$B = I \cdot \left[1 + \frac{S \cdot \cos(H'')}{\cos(60^\circ - H'')}\right]$$

$$R = 3I - (G + B)$$

→ Hue angles must be converted to radians when computing cosines.

4. Summary of Conversion

Hue Range	First Component	Second Component	Third Component
0° to 120°	R	G	B
120° to 240°	G	B	R

Hue Range	First Component	Second Component	Third Component
240° to 360°	B	R	G

5. Scaling the Output

- If working with 8-bit images (0–255), multiply each of R, G, B by 255 after calculation.

M4Q3) Explain the techniques used for pseudocolour image processing.

What is Pseudocolour Image Processing?

- **Pseudocolour image processing** involves assigning artificial colors to a **single-channel grayscale image** to enhance visual interpretation.
- Useful when **fine details or patterns** are not easily distinguishable in grayscale.

Techniques for Pseudocolour Image Processing

There are **two primary techniques**:

1. Intensity Slicing

- The grayscale intensity range (0–255) is divided into **multiple slices or intervals**.
- Each slice is mapped to a **specific color** (R, G, B).

Steps:

1. Define intensity ranges (e.g., 0–63, 64–127, etc.)
2. Assign a color to each range.
3. Replace pixel values in the image with corresponding RGB colors.

Example:

- In a thermal image:
    - 0–50 → Blue
    - 51–150 → Green

    - 151–255 → Red

Application:

- Weather maps, medical scans (like MRI, CT), thermal imaging

2. Gray-Level to Colour Mapping using Look-Up Tables (LUTs)

- Each grayscale value (0 to 255) is **mapped to a specific RGB triplet** using a predefined **Color Look-Up Table (CLUT)**.

Steps:

1. Create or use a predefined LUT (e.g., "Jet", "Hot", "Cool").
2. For each pixel, use its grayscale value as an index into the LUT.
3. Replace the pixel with the RGB value from the LUT.

Example:

- A grayscale value of 120 might map to RGB = (0, 255, 100)

Advantages:

- **Smooth transitions** of colors.
- **Fine control** over appearance.
- Can replicate styles like heatmaps or terrain elevation maps.

Comparison Between the Techniques

Technique	Approach	Output Quality	Flexibility
Intensity Slicing	Assigns colors to intensity intervals	Basic color bands	Medium
LUT-based Mapping	Maps each gray level to an RGB color	Smooth gradients	High (custom LUTs)

M4Q4) Explain RGB colour model in detail.

1. Introduction to RGB Model

The **RGB color model** is the **most widely used model** for image acquisition, display, and processing.

It is a **hardware-oriented, additive model** in which colors are created by **adding different intensities** of the three primary colors:

- **R** = Red
- **G** = Green
- **B** = Blue

2. Additive Nature of RGB

- When **no light** is present: the result is **black** (0,0,0)
- When **maximum intensities** of all three components are added: the result is **white** (1,1,1)
- By varying each component, a **wide range of colors** can be produced.

3. RGB Cube Representation

- The RGB color space can be represented as a **unit cube** in 3D.
- Axes represent **Red**, **Green**, and **Blue** values in the range [0,1] (or [0,255] for 8-bit images).
- Diagonal from (0,0,0) to (1,1,1) = **gray scale axis**
- Corners of the cube represent **pure colors**:
  - Red (1,0,0)
  - Green (0,1,0)
  - Blue (0,0,1)
  - Yellow (1,1,0)
- Cyan (0,1,1)
- Magenta (1,0,1)
- White (1,1,1)
- Black (0,0,0)

4. Mathematical Expression

A color image in the RGB model can be represented as:

$$f(x,y) = [R(x,y), G(x,y), B(x,y)]$$

Where:

- $R(x,y), G(x,y), B(x,y)$ are the red, green, and blue intensity values at pixel (x,y)

5. Characteristics of the RGB Model

Feature	Description
Type	**Additive** color model
Primary Components	Red, Green, Blue
Color Creation	By **adding** light of different intensities
Device Dependency	Used by cameras, monitors, TVs (hardware-based)
Value Range	Typically 0–255 (8-bit) per channel
Total Colors	256^3 = 16,777,216 (for 8-bit RGB)

6. Applications of RGB Model

- Image capture (digital cameras, scanners)
- Display systems (monitors, projectors)
- Graphics design and image editing
- Game development and UI rendering

7. Limitations of RGB Model

- Not intuitive for **color-based image processing** (e.g., segmentation based on hue)
- Device-dependent → same RGB values may appear differently on different displays
- Difficult to adjust **brightness** or **contrast** independently

---

M4Q5) With the help of neat diagram, explain steps involved in frequency domain filtering.

What is Frequency Domain Filtering?

- Frequency domain filtering involves **modifying the Fourier transform** of an image to enhance or suppress specific frequency components.
- In **frequency domain filtering**, the image is processed by modifying its **Fourier Transform**. This method is particularly effective for tasks such as:
    - **Noise removal**

- **Image smoothing**
- **Edge enhancement**

Steps in Frequency Domain Filtering

The process involves the following **five steps**:

1. Input Image

- Start with the original **spatial domain image** $f(x, y)$.

2. Compute Fourier Transform

- Apply **2D Discrete Fourier Transform (DFT)** to convert the image into frequency domain:

$$F(u, v) = \mathcal{F}\{f(x, y)\}$$

- This results in a complex function representing the image's frequency content.

3. Multiply with Filter Transfer Function

- Design a **frequency domain filter** $H(u, v)$ based on the required operation:
  - Low-pass → for **smoothing**
  - High-pass → for **sharpening**
  - Band-reject → for **removing periodic noise**
- Apply the filter:

  $$G(u, v) = H(u, v) \cdot F(u, v)$$

4. Compute Inverse Fourier Transform

- Apply the **Inverse DFT** to convert the result back to spatial domain:

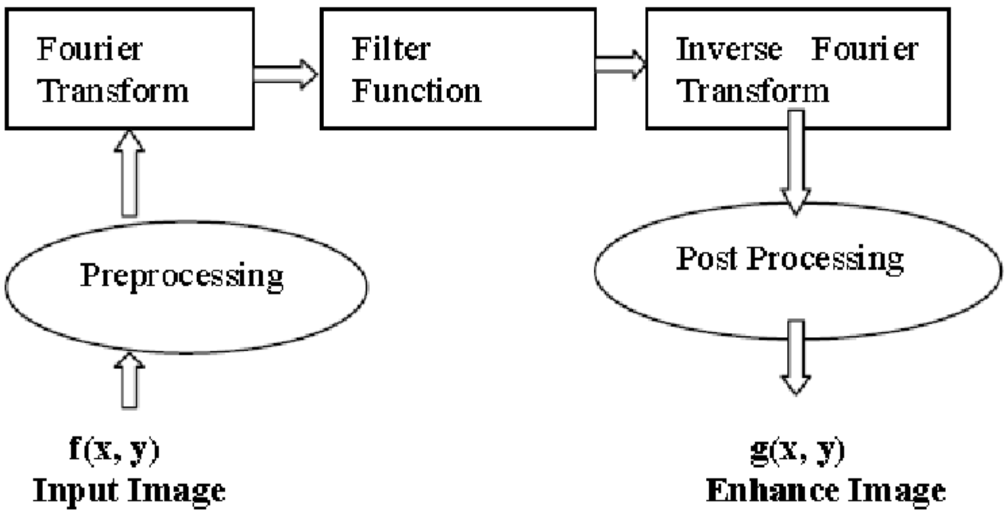
$$g(x, y) = \mathcal{F}^{-1}\{G(u, v)\}$$

- This gives the **filtered image** in the spatial domain.

5. Take Real Part

- Since the result may be complex, **take only the real part** to get the final image:

$$\text{Output Image} = \Re\{g(x, y)\}$$



Applications

- **Image smoothing**: Using low-pass filters to remove noise.
- **Edge enhancement**: Using high-pass filters.
- **Removing periodic noise**: Using notch filters in the frequency domain.

Advantages of Frequency Domain Filtering

- Handles **global frequency components** efficiently.
- Allows **precise control** over different frequency bands.
- Useful for operations that are hard to implement in spatial domain.

M4Q6) Explain sharpening filters for image enhancement in the frequency domain.

1. What Are Sharpening Filters?

Sharpening filters are used to enhance fine details and edges in an image by emphasizing **high-frequency components** in the frequency domain.

- **High frequencies** correspond to **edges, lines, and noise**
- Sharpening aims to **boost** those frequencies while **suppressing low frequencies** (flat regions)

2. Frequency Domain Approach

The steps for sharpening in the frequency domain involve:

1. Transform image using **Fourier Transform**
2. Multiply the transform by a **high-pass filter (HPF)**
3. Perform **Inverse Fourier Transform** to get the enhanced image

3. Common Sharpening Filters

i) Ideal High-Pass Filter (IHPF)

- Removes all low frequencies below a cutoff D_0
- Transfer function:

$$H(u,v) = \begin{cases} 0, & D(u,v) \le D_0 \\ 1, & D(u,v) > D_0 \end{cases}$$

→ Sharpest cutoff, but causes ringing artifacts

ii) Butterworth High-Pass Filter (BHPF)

- Smoother transition compared to ideal filter
- Transfer function:

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

- n : order of the filter
- D_0 : cutoff frequency

→ Less ringing, more control

iii) Gaussian High-Pass Filter (GHPF)

- Smoothest transition
- Transfer function:

$$H(u,v) = 1 - e^{-\frac{D^2(u,v)}{2D_0^2}}$$

→ No ringing artifacts, widely used in applications needing clean sharpening

iv) Laplacian Filter in Frequency Domain

- Directly emphasizes edges using the Laplacian operator in the frequency domain
- Transfer function:

$H(u,v) = -4\pi^2 \left(\frac{u^2}{M^2} + \frac{v^2}{N^2}\right)$

- Used for:

  $g(x,y) = f(x,y) + \nabla^2 f(x,y)$

→ Adds the second derivative to the image to sharpen edges

M4Q7) With a neat diagram explain homomorphic filtering approach for image enhancement. What are the advantages of these filters?

1. What is Homomorphic Filtering?

Homomorphic filtering is a technique used for **simultaneous contrast enhancement and dynamic range compression** in images.

It is based on separating the **illumination** and **reflectance** components of an image, processing them separately in the **frequency domain**, and recombining them.

2. Illumination–Reflectance Model

The image is modeled as:

$f(x,y) = i(x,y) \cdot r(x,y)$

Where:

- $i(x,y)$ = **illumination** component (slowly varying, low-frequency)
- $r(x,y)$ = **reflectance** component (rapidly changing, high-frequency)

3. Homomorphic Filtering Steps

Step 1: Take Logarithm

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

This converts the multiplicative model into **additive**, making separation feasible.

Step 2: Apply Fourier Transform

$Z(u,v) = \mathcal{F}\{\ln[f(x,y)]\}$

Transforms the image to **frequency domain**.

Step 3: Apply High-Pass Filter

$S(u,v) = H(u,v) \cdot Z(u,v)$

Where $H(u,v)$ is a **high-pass filter** (e.g., Butterworth).

This:

- **Suppresses** low-frequency (illumination)
- **Amplifies** high-frequency (reflectance, edges)

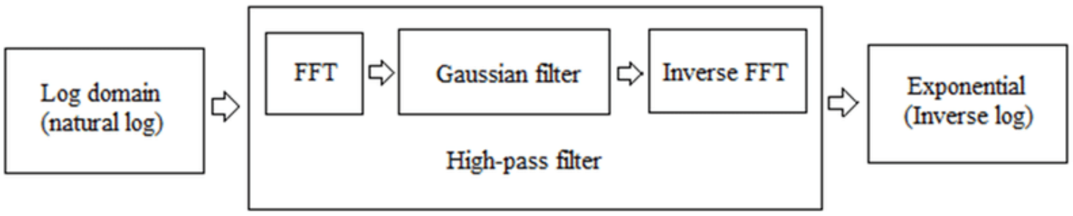
Step 4: Inverse Fourier Transform

$s(x,y) = \mathcal{F}^{-1}\{S(u,v)\}$

Step 5: Exponentiate

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

To revert the log transformation and obtain the **enhanced image**.



4. Advantages of Homomorphic Filtering

Advantage	Explanation
Simultaneous Enhancement	Enhances contrast and compresses dynamic range at the same time
Illumination Correction	Suppresses uneven lighting and shadows
Improves Edge Details	Emphasizes reflectance (edges, textures)

Advantage	Explanation
Flexible Control	Filter design allows control over enhancement level
Frequency Domain Efficiency	Efficient implementation using FFT

M4Q8) Explain the concept of pseudocolour image processing.

1. What is Pseudocolour Image Processing?

Pseudocolour image processing refers to the technique of assigning colors to grayscale images **artificially**, to enhance visual interpretation and analysis.

- It does **not use true color information** from the scene.
- The **intensity values** (gray levels) of the image are mapped to specific **colors**.
- Especially useful when the human eye struggles to distinguish fine details in grayscale.

2. Purpose

- Improves **visual perception** of structures in medical images, satellite data, thermal scans, etc.
- Aids in **data interpretation** where gray level differences are subtle.

3. Main Techniques of Pseudocolour Processing

a) Intensity Slicing

- Assigns different colors to different **ranges of intensity values**.
- The grayscale image is divided into slices (e.g., 0–50, 51–100, ...).
- Each slice is mapped to a specific color (e.g., blue, green, red).

Example:

- A thermal image might display:
    - Low temperatures → blue
    - Medium → green

    - High → red

✔️ Easy to implement, enhances contrast between regions.

b) Gray Level to Colour Mapping

- Uses a **color lookup table (CLUT)** to assign a color to **each individual gray level**.
- Each gray level (0–255) is mapped to an RGB value.

$$\text{Gray level } g \Rightarrow (R, G, B)$$

- Often used in medical imaging and scientific visualization.

✔️ Offers finer control than intensity slicing.

4. Applications

- **Medical imaging**: To highlight tissues or tumors.
- **Remote sensing**: Satellite image bands mapped to RGB.
- **Thermal imaging**: Heat levels mapped to color gradients.
- **Microscopy**: Biological structures enhanced visually.

5. Difference from Truecolour

Feature	Pseudocolour	Truecolour
Based on	Gray levels mapped to colours	Actual RGB components
Source of colour	Artificial assignment	Captured directly from the scene
No. of channels	Single-channel input (grayscale)	Three-channel input (RGB)

M4Q9) With the expression of transfer function explain Laplacian in frequency domain. (6M)

1. What is the Laplacian Operator?

- The **Laplacian** is a **second-order derivative** operator used for **image sharpening**.
- In the **spatial domain**, it highlights **regions of rapid intensity change** (edges).

2. Laplacian in Frequency Domain

The Laplacian in the frequency domain is obtained by taking the **Fourier Transform** of the Laplacian operator.

The **transfer function** H(u,v)H(u, v) of the Laplacian in frequency domain is:

$$H(u,v) = -4\pi^2 \left(\frac{u^2}{M^2} + \frac{v^2}{N^2} \right)$$

Where:

- u, v are frequency coordinates
- M, N are the image dimensions
- The negative sign reflects the edge-enhancing nature of the Laplacian

3. Filtering Operation

Given an image $f(x, y)$, the Laplacian-filtered image in frequency domain is:

$$\mathcal{F}\{\nabla^2 f(x,y)\} = H(u,v) \cdot F(u,v)$$

- $\nabla^2 f(x, y)$ is the Laplacian in spatial domain
- $F(u, v)$ is the DFT of the image
- Multiply $F(u, v)$ with $H(u, v)$ to apply the Laplacian
- Then take the **Inverse DFT** to get the result in spatial domain

4. Application: Image Sharpening

To enhance an image using the Laplacian:

$$g(x,y) = f(x,y) + \nabla^2 f(x,y)$$

In frequency domain:

$$G(u,v) = F(u,v) + H(u,v) \cdot F(u,v) = F(u,v) \cdot [1 + H(u,v)]$$

- This equation sharpens the image by adding high-frequency components amplified by the Laplacian.

5. Advantages

- Enhances **edges and fine details**.
- Can be applied directly in the frequency domain for efficient processing.

Summary Table (reference only)

Term	Meaning
H(u,v)H(u,v)	Transfer function of Laplacian
Use	Enhancing edges and fine details
Formula	$H(u,v) = -4\pi^2 \left(\frac{u^2}{M^2} + \frac{v^2}{N^2}\right)$
Sharpening	Add Laplacian result back to original image

M4Q10) Illustrate and explain the color chromaticity diagram.

1. What Is a Chromaticity Diagram?

The **chromaticity diagram** is a 2D graphical representation of **color perception**, based on the **CIE 1931 XYZ color space**.

It displays **pure chromatic content** (hue and saturation) of colors **independent of brightness (luminance)**.

- Constructed from the **CIE XYZ** model.
- Uses **chromaticity coordinates**:
  $x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}$
- The **(x, y)** values define each color's chromaticity; **Y** controls brightness.

2. Structure of the Chromaticity Diagram

- **Spectral Locus**: Outer curved boundary shows **pure spectral (monochromatic)** colors (wavelengths 380–700 nm).
- **Line of Purples**: Straight line connecting violet (≈380 nm) to red (≈700 nm), containing **nonspectral colors**.
- **White Point**: Represents an equal mix of all wavelengths (e.g., **D65 white** at x ≈ 0.33, y ≈ 0.33).
- **Horseshoe Shape**: All **visible colors** lie inside this curve.
- **Saturation**: Increases as you move away from white point toward boundary.
- **Hue**: Changes as you move along the spectral locus.

3. Chromaticity Coordinates

Given any tristimulus values (X, Y, Z), compute:

$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}$

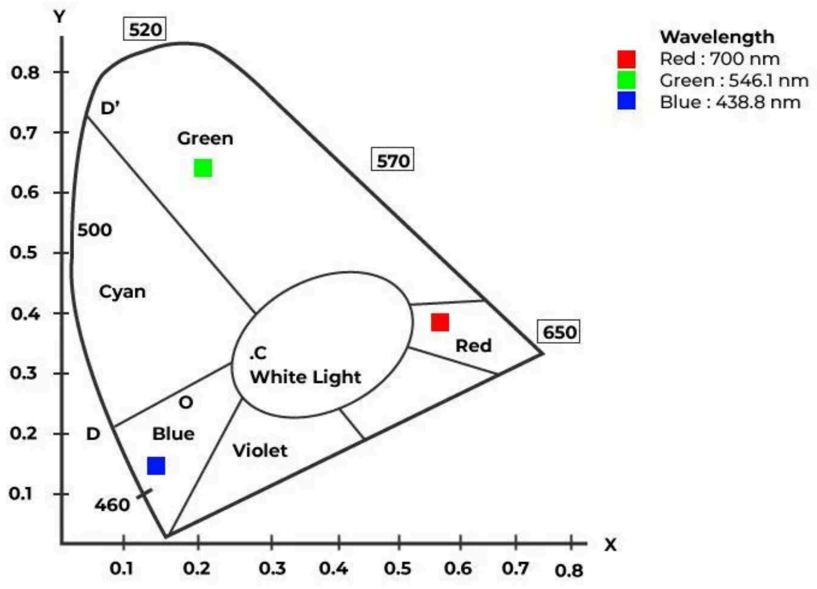
Then:

$z = 1 - x - y$

You only need (x, y) to **locate color chromaticity** on the 2D diagram.

4. Use of the Chromaticity Diagram

Application	Purpose
Color mixing	Shows all combinations of color that can be created
Gamut visualization	Displays range of colors a device (monitor, printer) can reproduce
Color specification	Used in CIE standards for comparing color systems
White balance and lighting	Locate different illuminants (e.g., D65, tungsten)



M5Q1) Explain the Weiner filtering method of restoring images in presence of noise and blur.

The **Wiener filter** is a powerful image restoration technique used to **simultaneously reduce blur and noise** in degraded images. Unlike inverse filtering, it considers both the degradation function and the **statistical characteristics of noise and the original image**.

1. Degradation Model

In the **frequency domain**, a degraded image $G(u, v)$ is modeled as:

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

Where:

- $G(u, v)$: degraded image
- $H(u, v)$: degradation function (blur)
- $F(u, v)$: original image
- $N(u, v)$: additive noise

2. Wiener Filter Formula

The Wiener filter estimates the original image $\hat{F}(u, v)$ as:

$$\hat{F}(u, v) = \left[\frac{H^*(u,v)}{|H(u,v)|^2 + \frac{S_N(u,v)}{S_F(u,v)}} \right] \cdot G(u, v)$$

Where:

- $H^*(u, v)$: complex conjugate of H(u,v)
- $S_N(u, v)$: power spectral density of noise
- $S_F(u, v)$: power spectral density of original image
- The fraction $\frac{S_N}{S_F}$ represents the **noise-to-signal power ratio**

📌 If noise is negligible → Wiener filter approximates inverse filter.

📌 If blur is negligible → Wiener filter behaves like a noise-reducing filter.

3. Interpretation

- The Wiener filter **balances inverse filtering and noise suppression**.
- It **attenuates frequencies** where the signal is weak or noise is strong.
- It is **adaptive**, adjusting based on the image and noise statistics.

4. Practical Case (Simplified Wiener Filter)

If exact power spectra are not known, the ratio $\frac{S_N}{S_F}$ is assumed to be a constant K, and the Wiener filter becomes:

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

Where K is an estimate of the **inverse signal-to-noise ratio**.

5. Advantages of Wiener Filter

- Performs **simultaneous deblurring and denoising**.
- Reduces **amplification of noise**, which is a problem in inverse filtering.
- Works well for **stationary linear degradations**.

6. Limitations

- Requires knowledge or estimation of:
  - Degradation function $H(u, v)$
  - Noise and image power spectra S_N, S_F
- Computationally more intensive than inverse filtering.

7. Example Use Case

- A photograph blurred due to camera shake and corrupted with Gaussian noise can be effectively restored using Wiener filtering

---

M5Q2) With necessary equations and graphs, explain any four probability noise density functions.

Noise in digital images is often described using **probability density functions (PDFs)**. Each PDF defines the **statistical characteristics** of a specific noise type.

Below are four commonly encountered noise models with their **equations**, **graphs**, and **applications**:

1. Gaussian Noise

Equation (PDF):

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

Where:

- z : gray level
- μ : mean (center of the distribution)

Graph:

- Bell-shaped, symmetric curve centered at μ

Characteristics:

- Most common noise model in natural images.

- σ : standard deviation

- Arises due to **electronic circuit noise** or **sensor noise**.

2. Rayleigh Noise

Equation (PDF):

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

Where:

- a : minimum value
- b : scale parameter

Graph:

- Skewed to the right (non-symmetric), rises sharply and then decays.

Characteristics:

- Often used to model **multiplicative noise** (e.g., radar, sonar imaging).

3. Gamma Noise (Erlang Distribution)

Equation (PDF):

$$p(z) = \frac{a^b z^{b-1} e^{-az}}{(b-1)!}, \quad z \geq 0$$

Where:

- $a > 0$: rate parameter
- b : shape parameter (integer)

Graph:

- Positively skewed, like Rayleigh, shape depends on bb

Characteristics:

- Models **positive-only noise**.
- Used in **laser imaging**, **nuclear imaging**, and **ultrasound**.

4. Salt-and-Pepper Noise (Impulse Noise)

Equation (PDF):

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

Where:

- a = minimum gray level (black specks)

Graph:

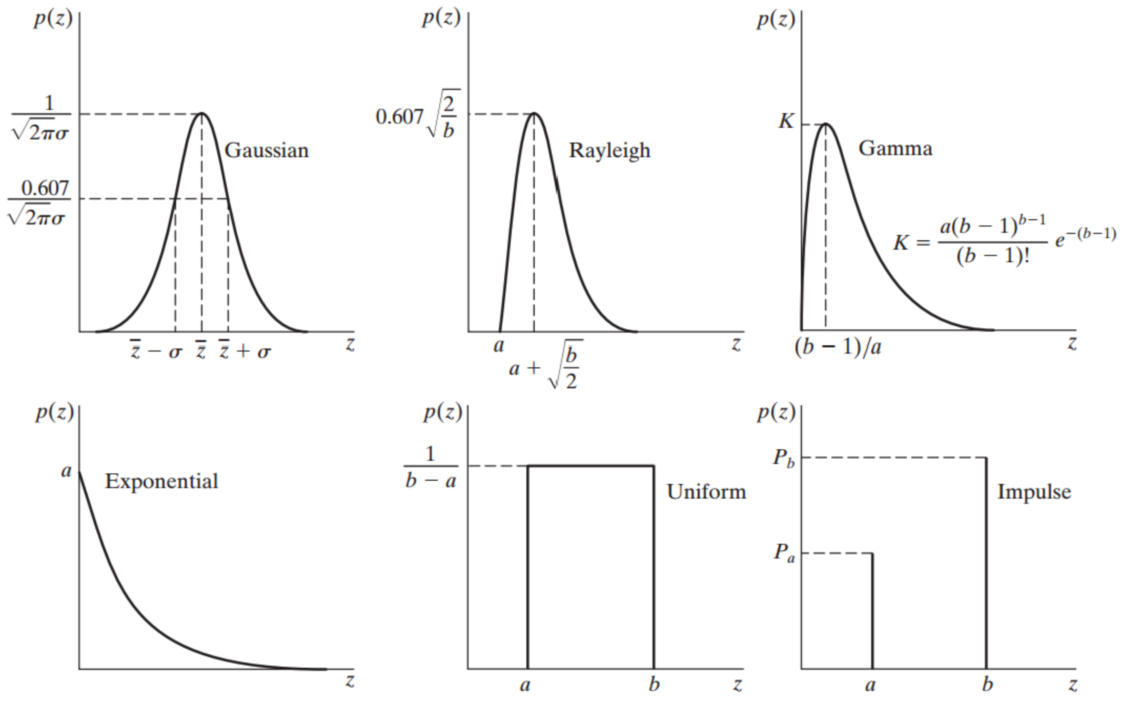
- Two impulses at aa and bb

Characteristics:

- Appears as **random black and white dots** on the image.

- b = maximum gray level (white specks)
- P_a and P_b : probabilities of black and white impulses

- Caused by **bit errors** or **faulty memory locations**.



Comparison Table (ref)

Noise Type	PDF Shape	Common Cause	Appearance
Gaussian	Bell-shaped curve	Sensor/electronic noise	Grainy variation
Rayleigh	Skewed right	Multiplicative noise (e.g., radar)	Smooth gradients with tail
Gamma	Skewed	Laser/nuclear imaging	Bright regions only
Salt-and-Pepper	Two spikes	Transmission/memory errors	Black and white dots

M5Q3) What are adaptive filters? Explain adaptive mean filters and their advantages.

1. What Are Adaptive Filters?

Adaptive filters are spatial filters whose behavior **changes based on local image statistics** like variance or mean within the neighborhood of each pixel.

→ Unlike linear filters (e.g., mean filters), adaptive filters are data-driven and context-sensitive.

2. Adaptive Mean Filter (Also called Local Noise Reduction Filter)

The **adaptive mean filter** is designed to reduce noise while **preserving useful image features**, especially edges and fine details.

Mathematical Formula:

$$\hat{f}(x,y) = g(x,y) - \frac{\sigma_n^2}{\sigma_L^2(x,y)} \cdot [g(x,y) - \mu_L(x,y)]$$

Where:

- $\hat{f}(x,y)$: restored pixel value
- $g(x,y)$: noisy image pixel
- $\mu_L(x,y)$: local mean of neighborhood around (x,y)
- $\sigma_L^2(x,y)$: local variance
- σ_n^2 : estimated noise variance

How It Works:

- If **local variance ≈ noise variance**, the pixel is likely in a **smooth region** → apply more smoothing.
- If **local variance >> noise variance**, the pixel is part of an **edge or detail** → preserve it, apply less smoothing.

3. Advantages of Adaptive Mean Filter

Feature	Benefit
Edge preservation	Reduces smoothing near edges or detailed textures
Noise reduction	Effective against **Gaussian and uniform noise**
Adaptive behavior	Adjusts itself according to local image content
Better than linear mean	Linear mean filters tend to blur all regions equally, losing detail

M5Q4) Explain Inverse filters and their advantages over Wiener filter. (7M)

1. What is an Inverse Filter?

The **inverse filter** is a **frequency domain image restoration technique** that attempts to **recover the original image** from a degraded version by **inverting the degradation function**.

2. Image Degradation Model

In frequency domain, image degradation is modeled as:

$G(u, v) = H(u, v) \cdot F(u, v)$

Where:

- $G(u, v)$: degraded image
- $H(u, v)$: degradation function (e.g., blur)
- $F(u, v)$: original image (to be recovered)

3. Inverse Filtering Formula

To recover the original image, the inverse filter applies:

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

Then apply **Inverse DFT** to obtain $\hat{f}(x, y)$, the restored image.

4. Assumptions

- Assumes **no noise** or **negligible noise** is present.
- Assumes **exact knowledge** of degradation function $H(u, v)$.

5. Advantages of Inverse Filter

Advantage	Description
Simple and direct	Easy to implement mathematically
Effective when noise is minimal	Performs well in noise-free or low-noise conditions
Good for known blur models	Accurate restoration when degradation $H(u, v)$ is known

M5Q5) Explain minimum mean square error filtering method of restoring images (7M)

1. What is MMSE Filtering?

The **Minimum Mean Square Error (MMSE)** filter is a **statistical image restoration method** that seeks to **minimize the average squared error** between the restored image and the original image.

It is essentially a **generalized form of the Wiener filter**, particularly useful when **noise and image signals are random processes** with known statistics.

2. Image Degradation Model

The degraded image $G(u, v)$ in the frequency domain is represented as:

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

Where:

- $F(u, v)$: original image
- $H(u, v)$: degradation function (e.g., blur)
- $N(u, v)$: additive noise
- $G(u, v)$: observed degraded image

3. MMSE Filter Formula

The MMSE estimate $\hat{F}(u, v)$ of the original image is:

$$\hat{F}(u, v) = \frac{H^*(u,v)}{|H(u,v)|^2 + \frac{S_N(u,v)}{S_F(u,v)}} \cdot G(u, v)$$

Where:

- $H^*(u, v)$: complex conjugate of $H(u, v)$
- $S_N(u, v)$: power spectral density (PSD) of noise
- $S_F(u, v)$: PSD of the original image

→ This is identical to the Wiener filter, which is itself an MMSE filter.

4. Interpretation

- **Inverse filtering** tries to undo the effect of $H(u, v)$ directly.
- **MMSE filtering** balances:

- **Deconvolution** of the blur function $H(u,v)$
- **Suppression of noise** via $\frac{S_N}{S_F}$

5. Special Cases

- If **noise is zero** → MMSE filter becomes **inverse filter**:

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

- If $\frac{S_N}{S_F}$ is assumed **constant** K → becomes **simplified Wiener filter**:

  $\hat{F}(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2+K} \cdot G(u,v)$

6. Advantages of MMSE Filter

Advantage	Description
Noise-aware	Effectively suppresses additive noise
Balances blur and noise	Achieves best trade-off via error minimization
Robust to zeros in $H(u,v)$	Avoids division by near-zero values, unlike inverse filter
Statistical accuracy	Based on optimality (minimum error) using second-order statistics

M5Q6) Explain how restoration is done in presence of noise only (6M)

1. Restoration in Presence of Noise Only

When an image is corrupted **only by additive noise**, and there is **no degradation function** (i.e., no blur), the degradation model becomes:

$$g(x,y) = f(x,y) + \eta(x,y)$$

In frequency domain:

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

Where:

- $f(x,y)$: original image
- $\eta(x,y)$: additive noise

- $G(u, v)$: degraded image
- $N(u, v)$: noise in frequency domain
- (u, v) : original image's Fourier transform

2. Objective

To estimate the original image $F(u, v)$ from the noisy observation $G(u, v)$.

3. Restoration Technique – Wiener Filtering

Since there is **no degradation function** $H(u, v)$, it is assumed to be 1:

$H(u, v) = 1$

The **Wiener filter** for noise-only restoration becomes:

$\hat{F}(u, v) = \left[\frac{1}{1+\frac{S_N(u,v)}{S_F(u,v)}}\right] \cdot G(u, v)$

Where:

- $S_N(u, v)$: power spectral density of noise
- $S_F(u, v)$: power spectral density of the original image

→ This formula attenuates frequencies where noise dominates, and preserves those where the signal is strong.

4. Simplified Form (When PSDs Unknown)

If $\frac{S_N}{S_F} = K$ (a constant), then:

$\hat{F}(u, v) = \frac{1}{1+K} \cdot G(u, v)$

This performs **uniform attenuation** of the image in frequency domain.

5. Spatial Domain Alternative – Adaptive Filters

If working in the **spatial domain**, and noise is Gaussian or uniform, filters like:

- **Adaptive Mean Filter**
- **Adaptive Median Filter**

can be used, which adjust smoothing based on **local variance**.

Advantages

- **Effectively suppresses noise** without removing important features.

- **Statistically optimal** when signal and noise characteristics are known.

M5Q7) Explain periodic noise and methods for noise parameter estimation.

1. What is Periodic Noise?

Periodic noise is a type of structured noise that appears in an image due to **electrical interference**, **mechanical vibrations**, or **repeating patterns** during image acquisition or transmission.

Characteristics:

- Appears as **regular ripples, waves, or grids** in the spatial domain.
- In the **frequency domain**, it appears as **distinct spikes** or impulses **away from the center** (i.e., not at the origin).
- It is typically **sinusoidal** in nature, with specific frequencies and orientations.

Mathematical Model:

In spatial domain:

$g(x, y) = f(x, y) + \eta(x, y)$

Where:

- $f(x, y)$: original image
- $\eta(x, y)$: periodic noise (often sinusoidal)
- $g(x, y)$: observed noisy image

2. Methods for Noise Parameter Estimation

Estimating noise parameters helps in designing filters (like notch filters) to remove periodic noise. Below are key methods:

i) Visual Analysis of Frequency Spectrum

Steps:

1. Apply **2D Discrete Fourier Transform (DFT)** to the image.

2. Display the **magnitude spectrum** (usually log-scaled for better visibility).
3. Identify **bright spots or impulses** located symmetrically **away from the center** — these correspond to **periodic noise frequencies**.

Application:

- Locate the coordinates (u_0, v_0) of the noise components for **notch filtering**.

ii) Power Spectral Density (PSD) Analysis

Concept:

- PSD measures how power (energy) is distributed over frequency.
- Used to **quantify noise strength** and frequency location.

$$\text{PSD}(u, v) = |F(u, v)|^2$$

Where $F(u, v)$ is the DFT of the image or the noise component.

Application:

- Helps in selecting filter parameters (e.g., bandwidth of notch filters).

iii) Use of Test Patterns or Calibration Images

Method:

- Capture an image of a **uniform background** (e.g., plain white paper).
- Any non-uniformity observed is likely **noise**, and can be isolated.

Use:

- Analyze this known image to determine:
  - Noise frequency
  - Amplitude
  - Phase

iv) Fourier Transform of Isolated Noise Patch

Steps:

1. Select a region in the image that **contains only noise** (no useful signal).
2. Apply DFT to that region.
3. Estimate noise properties like frequency, orientation, and periodicity.

M5Q8) Explain the model of the image degradation/restoration process with a block diagram.

1. What is Image Degradation and Restoration?

- **Image degradation** refers to any **unwanted modification** (e.g., blur, noise) that reduces image quality.
- **Image restoration** aims to **recover the original image** from the degraded one using mathematical models and filtering techniques.

2. Mathematical Model of Image Degradation

In the **spatial domain**, the degradation model is:

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

Where:

- $g(x, y)$: observed (degraded) image
- $f(x, y)$: original image
- $h(x, y)$: degradation function (e.g., blur kernel or PSF)
- $\eta(x, y)$: additive noise
- $*$: convolution operation

3. Frequency Domain Representation

Using the **Fourier Transform**, the model becomes:

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

Where:

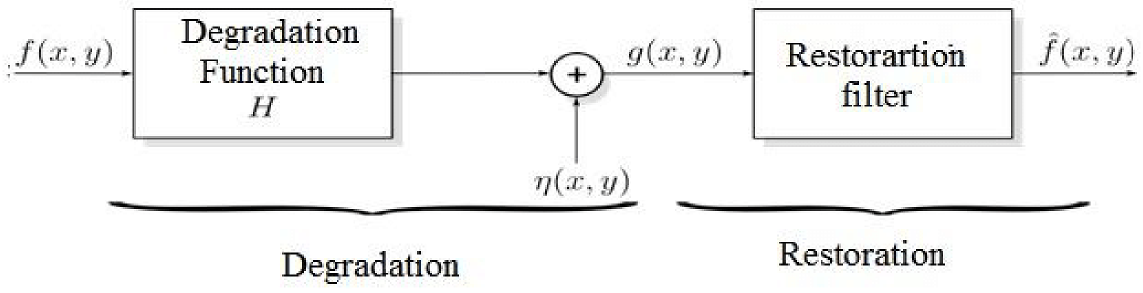
- $G(u, v)$: Fourier transform of the degraded image
- $F(u, v)$: Fourier transform of the original image

- $H(u, v)$: degradation transfer function
- $N(u, v)$: transform of noise

4. Restoration Objective

The goal is to estimate $\hat{f}(x, y)$, an approximation of the original image $f(x, y)$, using the known or estimated degradation function and the observed image.

5. Block Diagram of the Degradation/Restoration Process



6. Restoration Techniques

- **Inverse Filtering**: Assumes no noise; deconvolves blur
- **Wiener Filtering**: Minimizes error in presence of blur and noise
- **Constrained Least Squares**: Uses constraints and regularization
- **Adaptive Filtering**: Varies filter locally based on image statistics

---

M5Q9) Explain the working of the alpha-trimmed mean filter for image restoration.

1. What is the Alpha-Trimmed Mean Filter?

The **alpha-trimmed mean filter** is a **nonlinear spatial filter** used for image restoration in the presence of **mixed noise**, especially **Gaussian noise** and **impulse noise** (salt-and-pepper).

It is a **generalized version** of:

- The **median filter** (when maximum trimming is used)
- The **mean filter** (when no trimming is used)

2. Purpose

- **Reduces random noise** while preserving **edges and details**
- Best suited when the image is affected by **both Gaussian and impulsive noise**

3. Working Principle

Step-by-Step:

1. Select a **neighborhood window** (e.g., 3×3, 5×5).
2. **Flatten** and **sort** the pixel values in ascending order.
3. **Trim** a fixed number of the highest and lowest intensity values.
    - If trimming parameter = d, then **remove** $d/2$ lowest and $d/2$ highest values.
4. Compute the **average (mean)** of the remaining $mn - d$ pixel values.
5. Replace the center pixel with this mean.

Mathematical Formula

Let S be the sorted list of pixel values in an $m \times n$ neighborhood.

Let dd be the total number of values to discard (even number).

$\hat{f}(x, y) = \frac{1}{mn-d} \sum_{i=d/2+1}^{mn-d/2} S_i$

Where:

- $\hat{f}(x, y)$ = restored pixel value
- S_i = the i^{th} sorted pixel in the neighborhood

Example

Neighborhood:

$[10,\ 12,\ 15;\ 250,\ 20,\ 22;\ 25,\ 255,\ 30]$

Sorted:

$[10,\ 12,\ 15,\ 20,\ 22,\ 25,\ 30,\ 250,\ 255]$

If $d = 4$:

- Remove 2 lowest (10, 12) and 2 highest (250, 255)

- Remaining: 15, 20, 22, 25, 30 → Mean = $\frac{112}{5} = 22.4$

Applications

- Restoration of images with **combined Gaussian + impulse noise**
- **Medical imaging**, **industrial inspection**, and **surveillance systems**

M5Q10) Discuss the working of mean filters, including automatic and geometric mean filters.

1. What Are Mean Filters?

Mean filters are **linear spatial filters** used for **image smoothing** and **noise reduction**, particularly for **Gaussian noise**.

They replace each pixel with a **mean value** computed from its neighborhood.

2. Types of Mean Filters

A. Arithmetic Mean Filter (Standard Mean Filter)

Replaces each pixel value with the **average (arithmetic mean)** of the pixel values in its neighborhood.

Formula:

For an $m \times n$ window centered at (x, y) :

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

Where:

- $\hat{f}(x, y)$: output (restored) pixel
- $g(s, t)$: input pixel values in the neighborhood S

Effect on Image:

- Smooths the image
- Reduces **random (Gaussian) noise**
- **Blurs edges** and fine details

B. Geometric Mean Filter

Replaces each pixel with the **geometric mean** of the pixel values in the neighborhood.

Better at preserving edges and details compared to arithmetic mean.

Formula:

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

Where:

- $\hat{f}(x,y)$: output pixel
- $g(s,t)$: pixel values in the $m \times n$ neighborhood
- mn : total number of pixels in the neighborhood

Effect on Image:

- Reduces **Gaussian noise**
- **Preserves image detail** better than arithmetic mean
- Less sensitive to **very high or very low values**

*********************************** EOF ***********************************