Digital Image Processing - Semester Exam Notes

1. Introduction [3L]

What is an image?

An image is defined as a two-dimensional function,F(x,y), where x and y are spatial coordinates, and the amplitude of F at any pair of coordinates (x,y) is called the intensity of that image at that point. When x,y, and amplitude values of F are finite, we call it a digital image.
In other words, an image can be defined by a two-dimensional array specifically arranged in rows and columns.
Digital Image is composed of a finite number of elements, each of which elements have a particular value at a particular location.These elements are referred to as *picture elements,image elements,and pixels*.A *Pixel* is most widely used to denote the elements of a Digital Image.

Types of an image

1. BINARY IMAGE- The binary image as its name suggests, contain only two pixel elements i.e 0 & 1,where 0 refers to black and 1 refers to white. This image is also known as Monochrome.

2. BLACK AND WHITE IMAGE- The image which consist of only black and white color is called BLACK AND WHITE IMAGE.

3. **8 bit COLOR FORMAT**- It is the most famous image format.It has 256 different shades of colors in it and commonly known as Grayscale Image. In this format, 0 stands for Black, and 255 stands for white, and 127 stands for gray.

4. 16 bit COLOR FORMAT- It is a color image format. It has 65,536 different colors in it.It is also known as High Color Format. In this format the distribution of color is not as same as Grayscale image.

A **pixel is the smallest unit of a digital image or display and stands for "picture element."** It is a very small, isolated dot that stands for one color and plays the most basic part in digital images.
Pixels when combined help to create the mosaic of colors and shapes contributing towards visual content being displayed on screens such as smartphones, computers TVs, etc. In digital imaging, a grid of pixels can be seen and the combination of thousands or millions of such 'pixel' creates an overall visual representation that users see on their screens.

Background

Digital Image Processing means processing digital image by means of a digital computer. We can also say that it is a use of computer algorithms, in order to get enhanced image either to extract some useful information.

Digital image processing is the use of algorithms and mathematical models to process and analyze digital images. The goal of digital image processing is to enhance the quality of images, extract meaningful information from images, and automate image-based tasks.

The main goal is to improve the quality of an image or to extract meaningful information for further analysis.

Digital Image Representation

📌 Definition: A digital image is a discrete representation of a visual image. It is composed of pixels (picture elements) arranged in a 2D grid, where each pixel holds a value representing the intensity (grayscale) or color at that point.

Mathematically:

f(x,y)f(x, y)f(x,y)

Where:

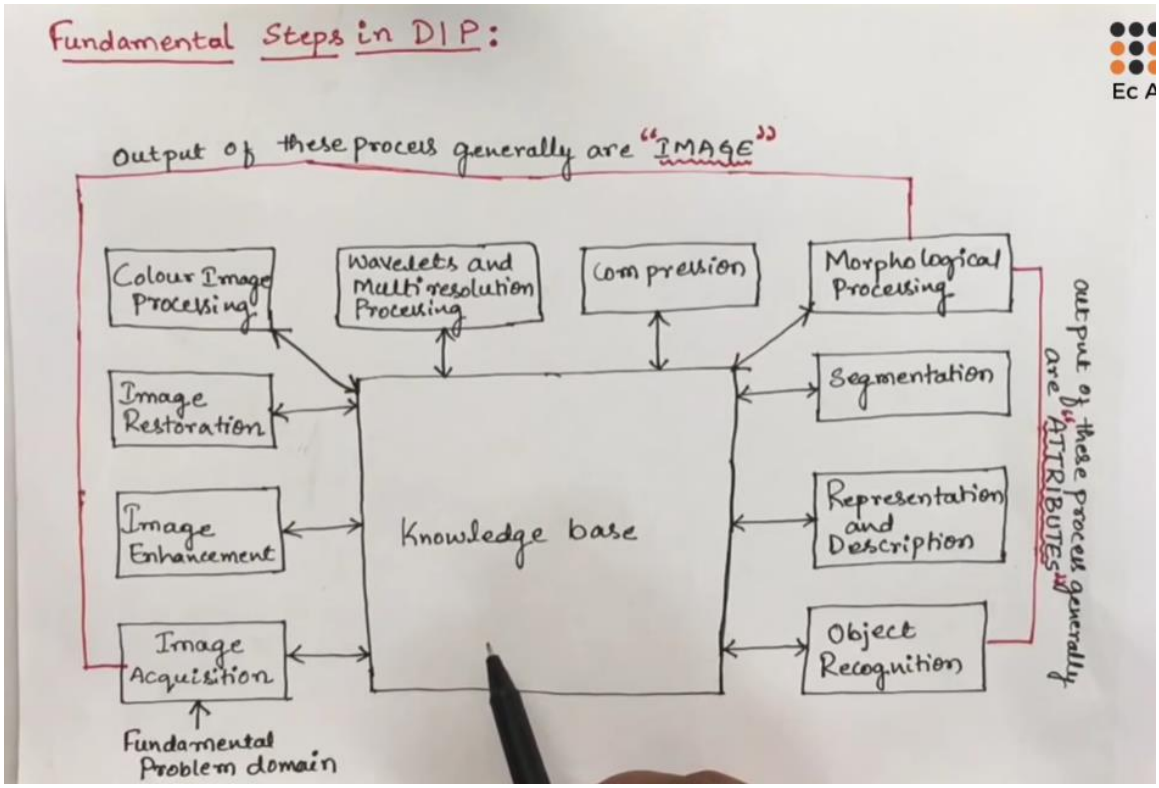
- x,yx, yx,y = spatial coordinates of the image
- f(x,y)f(x, y)f(x,y) = intensity or gray level at the coordinates x,yx, yx,y

---

□ Structure of a Digital Image:

Term	Description
Pixel	Smallest unit of an image (short for "picture element")
Resolution	Number of pixels in an image (e.g., 1920×1080 = Full HD)
Bit depth	Number of bits used to represent the intensity of each pixel
Spatial Resolution	How finely the image is sampled in space (x and y axes)
Gray Level	Intensity of a pixel (0 to 255 in 8-bit images)

Fundamental Steps in Image Processing



1. Image Acquisition

- **Purpose:** The starting point of DIP.
- **Function:** Captures an image using a sensor (e.g., camera, scanner) and converts it into a digital format.
- **Example:** Scanning a document or taking a photo with a camera.
- **Output:** A raw digital image.

---

2. Image Enhancement

- **Purpose:** Improve the visual quality of the image.
- **Function:** Enhances contrast, brightness, sharpness, or removes blur to make important features more visible.
- **Techniques:**
    - Histogram Equalization
    - Contrast Stretching

    - Filtering (smoothing/sharpening)
- **Output:** Enhanced image.

---

3. Image Restoration

- **Purpose:** Recover an original image that has been degraded.
- **Function:** Uses mathematical models to remove known types of distortion or noise.
- **Techniques:**
    - Inverse Filtering
    - Wiener Filtering
- **Output:** Reconstructed or deblurred image.

---

4. Color Image Processing

- **Purpose:** Handle images in color format.
- **Function:** Works in RGB or other color spaces (HSV, CMYK).
- **Applications:**
    - Color enhancement
    - Color segmentation
    - Pseudo-coloring (for grayscale-to-color conversion)
- **Output:** Processed color image.

---

5. Wavelets and Multi-resolution Processing

- **Purpose:** Analyze an image at multiple levels of resolution.
- **Function:** Breaks image down into different scales for detail vs. structure analysis.
- **Techniques:**
    - Wavelet Transform
    - Pyramid Decomposition

- **Use cases:** Compression, denoising, feature extraction.

---

6. Compression

- **Purpose:** Reduce storage and transmission cost.
- **Function:** Reduces the amount of data required to store or transmit an image.
- **Types:**
    - **Lossless:** No loss of information (e.g., PNG, TIFF)
    - **Lossy:** Minor loss acceptable for huge savings (e.g., JPEG)
- **Output:** Compressed image file.

---

7. Morphological Processing

- **Purpose:** Analyze geometric structure and shapes.
- **Function:** Operates on binary or grayscale images using structuring elements.
- **Operations:**
    - Dilation
    - Erosion
    - Opening & Closing
- **Use cases:** Boundary detection, noise removal, skeletonization.

---

8. Segmentation

- **Purpose:** Divide an image into meaningful parts or regions.
- **Function:** Isolates objects of interest from the background.
- **Techniques:**
    - Thresholding
    - Edge-based segmentation
    - Region growing

- **Output:** Segmented image (regions or objects marked).

---

9. Representation and Description

- **Purpose:** Convert segmented regions into information for analysis.
- **Function:** Describes shape, texture, or size of the segmented regions.
- **Types:**
    - **Boundary Representation:** Focus on edges or contours.
    - **Regional Descriptors:** Statistical or topological features (area, perimeter).
- **Output:** Feature vectors or descriptors.

---

10. Object Recognition

- **Purpose:** Assign a label or category to objects.
- **Function:** Uses extracted features to identify known objects.
- **Examples:**
    - Face recognition
    - OCR (Optical Character Recognition)
    - Vehicle or animal detection
- **Output:** Labeled objects or classifications.

□ Knowledge Base (Central Component)

- **Acts as the brain** of the entire pipeline.
- Stores:
    - Processing rules
    - Feedback from output
    - Heuristic or application-specific knowledge
- **Guides each step** dynamically based on the problem or goal.

Elements of Digital Image Processing

 1. Image Acquisition

- Definition: The process of capturing an image and converting it into a digital format.
- Input: Real-world scene/object.
- Devices Used: Digital cameras, scanners, sensors, satellites.
- Steps:
    - Sensing the environment (light, X-ray, etc.)
    - Converting to electrical signals (analog)
    - Digitizing (Analog-to-Digital Conversion - ADC)
- Output: Digital image suitable for processing.

---

 2. Image Storage

- Definition: Saving digital images for future access, manipulation, or transmission.
- Memory Types:
    - Primary storage: RAM (temporary storage during processing)

    - Secondary storage: Hard drives, SSDs, cloud

- File Formats:
    - Lossless: PNG, BMP, TIFF
    - Lossy: JPEG, WebP
- Compression: Often combined with storage to reduce size.

⚙️ 3. Image Processing

- Definition: Applying algorithms to modify, analyze, or interpret images.
- Types:
    - Image Enhancement: Improve visibility (brightness, contrast).
    - Image Restoration: Remove blur or noise.
    - Morphological Processing: Analyze shapes and structures.
    - Filtering: Smoothing or sharpening.
    - Transform-based Processing: e.g., Fourier or Wavelet transforms.

📡 4. Image Communication (Transmission)

- Definition: Sending images across networks or systems.
- Need: For telemedicine, remote sensing, video conferencing, etc.
- Techniques:
    - Compression before transmission
    - Encoding for efficient transfer
    - Protocols: TCP/IP, HTTP, FTP
- Challenges:
    - Bandwidth limitations
    - Lossy transfer
    - Real-time transmission (e.g., video stream)

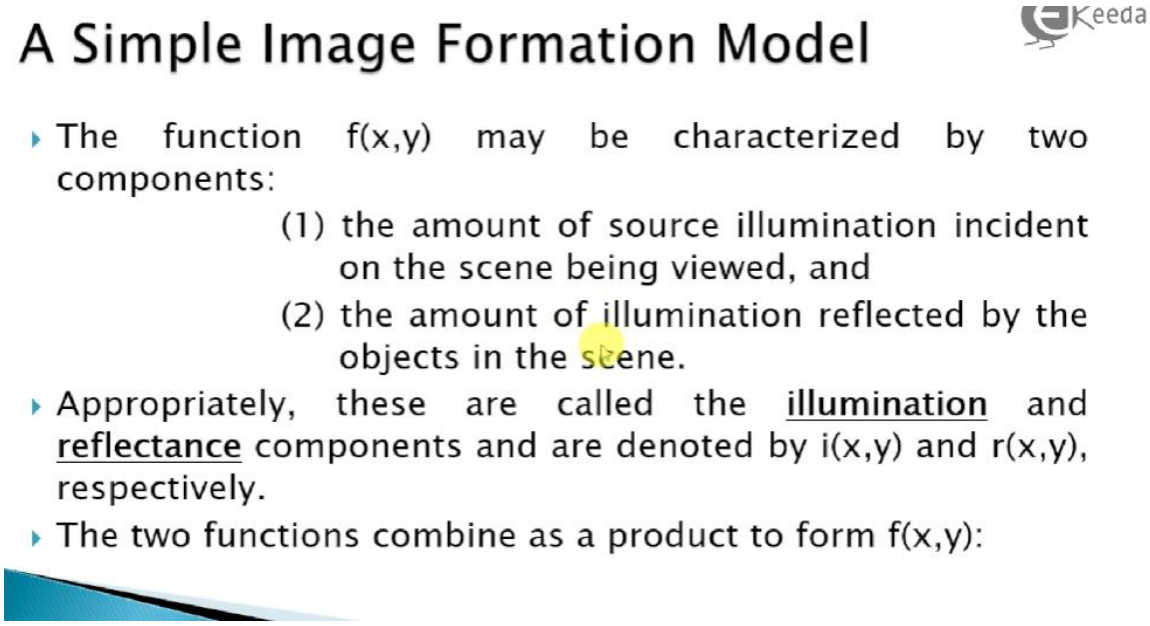
🖥️ 5. Image Display

- Definition: Visualization of processed or raw images.
- Devices:
    - Monitors (CRT, LCD, OLED)

    - Projectors
    - Mobile displays
- Color Models Used:
    - RGB (Red, Green, Blue)
    - Grayscale
- Display Considerations:
    - Resolution
    - Brightness and contrast
    - Frame rate (for dynamic images)

2. Digital Image Formation [4L]

A Simple Image Model



Geometric Model - Basic Transformations

What is a Geometric Model?

The Geometric Model is used in image processing to modify the spatial position of pixels. It transforms the shape or orientation of images by applying mathematical operations to the coordinates of pixels — without changing the intensity (gray value).

---

1. Translation

Definition: Moves an image from one location to another.

Formula:

x' = x + Tx

y' = y + Ty

Where:

- (x, y) = original coordinates
- (x', y') = new coordinates
- Tx, Ty = translation distances along x and y

Example:

If P = (2, 3) and Tx = 4, Ty = 5:

x' = 2 + 4 = 6

y' = 3 + 5 = 8

Result: P' = (6, 8)

---

2. Scaling

Definition: Resizes the image — either enlarges or reduces.

Formula:

x' = Sx * x

y' = Sy * y

Where Sx and Sy are scaling factors.

Example:

If P = (3, 2) and Sx = 2, Sy = 3:

x' = 2 * 3 = 6

y' = 3 * 2 = 6

Result: P' = (6, 6)

---

3. Rotation

Definition: Rotates the image around the origin by angle θ (counter-clockwise).

Formula:

x' = x * cos(θ) - y * sin(θ)

y' = x * sin(θ) + y * cos(θ)

Example:

Rotate point P = (4, 3) by 45 degrees:

- cos(45°) = sin(45°) = 0.7071

x' = 4 * 0.7071 - 3 * 0.7071 = 2.8284 - 2.1213 = 0.7071

y' = 4 * 0.7071 + 3 * 0.7071 = 2.8284 + 2.1213 = 4.9497

Result: P' ≈ (0.7071, 4.9497)

---

4. Shearing

Definition: Slants the image by shifting coordinates.

Formula (X-direction shear):

x' = x + a * y

y' = y

Where "a" is the shear factor.

Example:

If P = (2, 3), and a = 1:

x' = 2 + 1 * 3 = 5

y' = 3

Result: P' = (5, 3)

---

5. Perspective Projection

Definition: Simulates how a 3D object looks on a 2D screen.

Formula:

x' = x / z

y' = y / z

Example:

If 3D point is (6, 9, 3):

x' = 6 / 3 = 2

y' = 9 / 3 = 3

Result: (2, 3)

Homogeneous Coordinates

To apply multiple transformations using matrix multiplication, convert coordinates to homogeneous form:

Homogeneous Point:

[x, y] -> [x, y, 1]

Combined transformation matrix (example):

| x' | | a11 a12 a13 | | x |

| y' | = | a21 a22 a23 | * | y |

| 1 | | 0 0 1 | | 1 |

Summary Table

Transformation	Formula	Description

Transformation	Formula	Description
Translation	x' = x + Tx, y' = y + Ty	Moves image
Scaling	x' = Sx * x, y' = Sy * y	Changes size
Rotation	Uses cos(θ), sin(θ)	Rotates image
Shearing	x' = x + a * y	Slants the image
Perspective Projection	x' = x / z, y' = y / z	Simulates 3D effect

---

Real-Life Applications

- Satellite image registration
- Face alignment in biometrics
- Augmented Reality filters
- Medical imaging (MRI alignment)
- 3D model projection and transformation

Perspective Projection

✅ **Definition:**

Perspective Projection is a geometric transformation that simulates how a 3D object appears to the human eye or a camera on a 2D image plane. It accounts for **depth (z-axis)** — making distant objects appear smaller and closer ones larger.

It is commonly used in:

- 3D rendering and computer graphics
- Robotics vision systems
- Augmented Reality (AR)
- Satellite and drone imaging

□ Mathematical Model:

If a point in 3D space is defined as:

P = (X, Y, Z)

Then its **2D projection** on the image plane (assuming pinhole camera model) is:

x' = f * X / Z

y' = f * Y / Z

Where:

- (x', y') = 2D coordinates on the image plane
- f = focal length of the camera (distance between the pinhole and the image plane)
- Z = depth (distance of the object from the camera)
- X, Y = real-world coordinates of the object

🔂 **Simplified Formula (when f = 1):**

x' = X / Z

y' = Y / Z

This is often used for **normalized perspective projection** in image processing and computer vision.

◆ **Example:**

A 3D point is located at:

P = (6, 9, 3)

Assuming focal length f = 1, the perspective projection gives:

x' = 6 / 3 = 2

y' = 9 / 3 = 3

Result:

Projected Point: (2, 3)

□ Why is Z important?

- If **Z is large** (far from the camera), the projected (x', y') values become **smaller** — making the object appear **smaller**.
- If **Z is small** (close to the camera), the projection appears **larger** — similar to what our eyes see in real life.

🖼️ Visual Intuition:

Imagine you're looking at a railway track:

- The two rails seem to **converge** at a point on the horizon.
- That illusion is caused by **perspective projection** — parallel lines appear to meet due to depth.

🔧 Matrix Representation (Homogeneous Coordinates):

To perform perspective projection using matrix multiplication, we use homogeneous coordinates:

3D Point:

P = [X, Y, Z, 1]^T

Perspective Projection Matrix:

bash

| 1 0 0 0 |

| 0 1 0 0 |

| 0 0 1 0 |

| 0 0 1/f 0 |

Multiplying this matrix and converting back from homogeneous coordinates gives:

x' = X / Z

y' = Y / Z

📌 Applications in DIP:

- 3D reconstruction from 2D images
- Virtual object placement in AR
- Simulating realistic camera behavior in synthetic images
- Aligning multiple views of the same scene (e.g., stereo vision)

.

Sampling and Quantization

□ 1. Sampling

✅ Definition:

Sampling refers to measuring the image at **discrete intervals in space** — selecting points on the continuous image function to represent as pixels.

📌 **Key Concept:**

- It determines the **spatial resolution** of an image.
- Higher sampling = more pixels = finer details
- Lower sampling = fewer pixels = loss of details

□ Example:

Suppose an analog image is defined over a 10×10 cm area. If we sample at 1 cm intervals, we get a 10×10 digital image (100 pixels). If we sample at 0.5 cm intervals, we get a 20×20 image (400 pixels) — thus better resolution.

🖼️ **Visual:**

Sampling turns a **continuous grid** of points into a **discrete matrix of pixels**.

🔘 **2. Quantization**

✅ **Definition:**

Quantization is the process of mapping a range of **continuous intensity values** to **discrete levels**.

📌 **Key Concept:**

- It determines the **grayscale (intensity) resolution** of an image.
- More quantization levels = smoother grayscale image
- Fewer levels = more visible banding or loss of detail

📶 **Example:**

- 2-bit quantization → 4 intensity levels: 0, 1, 2, 3
- 8-bit quantization → 256 intensity levels: 0 to 255 (standard grayscale image)
- 24-bit (color image) → 256 levels each for R, G, B

🔷 **Uniform vs. Non-Uniform**

1. Uniform Sampling/Quantization:

- Equal spacing between sample points or intensity levels
- Easier and faster to implement
- Common in most digital imaging systems

2. Non-Uniform Sampling/Quantization:

- Variable spacing between sample points or intensity levels
- Useful in applications like:
    - Human vision simulation
    - Medical imaging
    - Compression (more bits for critical areas)

□ Important Notes:

- **Oversampling**: Too many samples; may increase file size with little visual benefit.
- **Undersampling**: Too few samples; causes **aliasing** (loss of details or distortion).
- **Aliasing** can be minimized by pre-filtering (blurring) the image before sampling.

📌 **Real-life Applications:**

- Scanners (convert photos into digital format)
- Medical imaging devices (CT, MRI)
- Satellite and aerial imaging
- Digital cameras
- OCR (Optical Character Recognition)

ORGANISER SPECIAL

1.**Explain image sensing and acquisition(using single sensor,sensor strip,sensor array).**

Image Sensing and Acquisition

Image sensing and acquisition are fundamental processes in digital imaging, where a scene is captured and converted into a digital format for processing. This involves using sensors to detect light or electromagnetic radiation and transform it into electrical signals, which are then digitized. Below, I explain the three main approaches to image sensing and acquisition—using a single sensor, sensor strip, and sensor array—with detailed explanations for each.

---

1. Single Sensor

A single sensor is a basic light-sensitive element, such as a photodiode or a single pixel detector, used to capture light intensity at a specific point. To acquire a full image, the sensor must be moved across the scene systematically, either mechanically or optically.

- **Mechanism**:

    - The single sensor captures light intensity at one point at a time.
    - To form a complete image, the sensor scans the scene in a grid-like pattern (row by row or column by column).
    - This scanning can be achieved by moving the sensor physically (e.g., in flatbed scanners) or by using mirrors to direct light from different parts of the scene to the sensor (e.g., in some early imaging devices).
    - The captured intensity values are sampled and quantized to create a digital image.
- **Advantages**:
    - Simple and cost-effective design.
    - High precision in controlled environments, as only one sensor is calibrated.
    - Suitable for applications where cost is a constraint, such as low-cost scanners.
- **Disadvantages**:
    - Slow acquisition process due to sequential scanning.
    - Requires precise mechanical or optical systems for accurate movement.
    - Not suitable for capturing dynamic scenes due to the time taken to scan.
- **Applications**:
    - Flatbed scanners, where a single sensor moves across a document.
    - Early digital imaging systems or specialized scientific instruments.

---

2. Sensor Strip

A sensor strip consists of a linear array of sensors (e.g., a row of photodiodes or CCD elements) that captures a one-dimensional strip of the image at a time. To acquire a two-dimensional image, the strip is moved relative to the scene, typically in a direction perpendicular to the strip's orientation.

- **Mechanism**:
    - The sensor strip captures an entire row (or column) of the scene simultaneously.
    - The strip is moved across the scene (e.g., by a motor in a scanner or by moving the object, as in satellite imaging).
    - Each captured strip is combined to form a complete two-dimensional image.
    - The sensor strip may include filters (e.g., RGB filters) to capture color information.

- **Advantages**:
    - Faster than single-sensor systems, as an entire row is captured at once.
    - More robust than single-sensor systems, as fewer mechanical movements are needed.
    - Suitable for continuous or large-scale imaging tasks.
- **Disadvantages**:
    - Still slower than sensor arrays, as scanning is required.
    - Susceptible to motion artifacts if the object or sensor strip moves irregularly.
    - Limited to applications where linear scanning is feasible.
- **Applications**:
    - Document scanners, where a sensor strip moves across a page.
    - Satellite imaging, where a push-broom sensor captures strips of the Earth's surface as the satellite moves.
    - Industrial inspection systems for conveyor belt objects.

---

3. Sensor Array

A sensor array is a two-dimensional grid of sensors (e.g., a CCD or CMOS sensor array) that captures the entire image simultaneously. Each sensor element (pixel) in the array corresponds to a specific point in the scene, making this the most common method in modern imaging devices.

- **Mechanism**:
    - The sensor array consists of millions of tiny light-sensitive elements (pixels) arranged in a grid.
    - Each pixel captures light intensity and, in color imaging, uses filters (e.g., Bayer filter) to detect specific wavelengths (red, green, blue).
    - The entire scene is captured in a single exposure, with each pixel's signal processed to form a digital image.
    - The signals are read out, amplified, and digitized using an analog-to-digital converter (ADC).
- **Advantages**:
    - Fast acquisition, as the entire image is captured simultaneously.
    - Suitable for dynamic scenes (e.g., video or photography).
    - High resolution and image quality due to the large number of sensors.
    - No mechanical movement required, reducing complexity and wear.

- **Disadvantages**:
    - More expensive due to the complexity of fabricating large sensor arrays.
    - Requires sophisticated electronics for signal processing.
    - Susceptible to noise or defects in individual sensor elements.
- **Applications**:
    - Digital cameras and smartphones (CMOS/CCD arrays).
    - Medical imaging (e.g., X-ray or MRI sensors).
    - Surveillance systems and webcams.

2. **What is CCD Array?**

A **CCD array** (Charge-Coupled Device array) is a type of **sensor array** used in digital imaging to capture light and convert it into electrical signals, forming a digital image. It consists of a two-dimensional grid of light-sensitive elements (pixels) that detect light intensity and, in some cases, color information. CCD arrays are widely used in devices like digital cameras, telescopes, and medical imaging systems due to their high sensitivity and image quality.

Detailed Explanation of CCD Array

1. **Structure**:
    - A CCD array is composed of numerous tiny photosensitive elements (pixels) arranged in a grid (e.g., 1920x1080 pixels for Full HD resolution).
    - Each pixel is a small capacitor that converts incoming photons (light) into an electrical charge via the **photoelectric effect**. The charge is proportional to the light intensity at that point.
    - The array is typically made from silicon, and each pixel is isolated to prevent charge leakage.
2. **Working Principle**:
    - **Light Capture**: When light strikes a pixel, it generates electrons, creating a charge proportional to the light intensity.
    - **Charge Transfer**: The CCD uses a "charge-coupled" mechanism to transfer the accumulated charges across the array in a controlled manner, typically row by row, to a readout register.
    - **Signal Conversion**: The charges are converted into voltage signals, amplified, and digitized using an **analog-to-digital converter (ADC)** to produce a digital image.
    - **Color Imaging**: For color images, a **Bayer filter** (a mosaic of red, green, and blue filters) is placed over the CCD array, allowing each pixel to capture one color component, which is later interpolated to produce a full-color image.

3. **Key Features**:
    - **High Sensitivity**: CCD arrays are highly sensitive to light, making them ideal for low-light conditions, such as in astronomy or night photography.
    - **Low Noise**: They produce images with minimal noise, ensuring high clarity and detail.
    - **Uniformity**: CCD pixels have consistent performance across the array, reducing artifacts.
4. **Advantages**:
    - Superior image quality, especially in low-light or high-precision applications.
    - High dynamic range, capturing both bright and dark areas effectively.
    - Well-suited for scientific applications requiring accurate light measurement (e.g., spectroscopy).
5. **Disadvantages**:
    - **High Power Consumption**: CCD arrays require more power than CMOS arrays, making them less suitable for battery-powered devices.
    - **Expensive**: Manufacturing CCDs is complex and costly compared to CMOS sensors.
    - **Slower Readout**: The sequential charge transfer process can be slower than CMOS, limiting frame rates in video applications.
    - **Blooming**: Bright light can cause charge overflow between pixels, leading to artifacts.
6. **Applications**:
    - **Digital Cameras**: Used in professional and high-end consumer cameras for superior image quality.
    - **Astronomy**: CCD arrays are standard in telescopes for capturing faint starlight (e.g., Hubble Space Telescope).
    - **Medical Imaging**: Used in X-ray imaging, dental imaging, and microscopy due to their precision.
    - **Industrial Imaging**: Employed in machine vision systems for quality control and inspection.

3. **Explain Bilinear Interpolation Method.**

Bilinear interpolation is a method for interpolating values on a 2D grid, commonly used in image processing, computer graphics, and numerical analysis to estimate values at non-grid points. It extends linear interpolation to two dimensions, providing a smooth transition between known data points by considering the four nearest grid points.

Key Properties

- **Smoothness**: Bilinear interpolation produces a continuous and smooth surface, unlike nearest-neighbor interpolation, which can be blocky.
- **Computational Cost**: It is relatively fast, requiring only a few arithmetic operations.
- **Locality**: It only uses the four nearest points, making it local and efficient.
- **Limitations**: It assumes the data varies linearly between grid points, which may not capture complex variations (e.g., sharp edges). Higher-order methods like bicubic interpolation may be used for better accuracy.

Applications:

- Image resizing (scaling up or down)
- Image rotation and geometric transformations
- Texture mapping in computer graphics

Advantages:

- Simple and computationally efficient.
- Produces smoother and more visually appealing results than nearest neighbor.

4. **What is resolution of image? Compute the size of a 640 * 480 image at 240 pixel per inch.**

Image resolution refers to the amount of detail an image holds, typically measured as the number of pixels in the width and height of a digital image (e.g., 640 × 480 pixels). It determines the clarity and quality of the image. Resolution can also be expressed in terms of **pixels per inch (PPI)** when considering the physical size of an image when printed or displayed. Higher PPI values indicate greater detail when viewed at a given physical size.

- **Digital Resolution**: The total number of pixels (e.g., 640 × 480 = 307,200 pixels).
- **Print Resolution**: The density of pixels per unit of physical length (e.g., 240 PPI means 240 pixels per inch).

Computing the Size of a 640 × 480 Image at 240 PPI

To find the physical size of an image with a resolution of 640 × 480 pixels at 240 pixels per inch (PPI), we calculate the dimensions in inches by dividing the pixel dimensions by the PPI.

Step-by-Step Calculation

Step 1: Calculate Width in Inches

Width (pixels) = 640

PPI = 240

Width (inches) = Width (pixels) / PPI

Width (inches) = 640 / 240

Width (inches) ≈ 2.6667

Step 2: Calculate Height in Inches

Height (pixels) = 480

PPI = 240

Height (inches) = Height (pixels) / PPI

Height (inches) = 480 / 240

Height (inches) = 2

Step 3: State the Final Dimensions

The physical size of the image is approximately:

Width: 2.67 inches

Height: 2 inches

Final Answer

The physical size of a 640 × 480 image at 240 PPI is approximately **2.67 inches wide by 2 inches tall**.

5. **What is Digitalization?**

Digitalization is the process of converting analog information, processes, or systems into a digital format to enhance efficiency, accessibility, and functionality. It involves using digital technologies to transform data, operations, or experiences, enabling automation, data analysis, and improved decision-making. Unlike **digitization** (which is specifically converting analog data to digital, e.g., scanning a paper document), digitalization focuses on leveraging digital tools to fundamentally change how processes or businesses operate.

Key Aspects of Digitalization:

- **Data Conversion**: Turning physical or analog data (e.g., paper records, analog signals) into digital formats for storage and processing.
- **Process Transformation**: Redesigning workflows or systems to incorporate digital technologies, such as automating manual tasks or using cloud-based platforms.
- **Enhanced Capabilities**: Enabling real-time data analysis, improved user experiences, and scalability through digital tools like AI, IoT, or software applications.
- **Examples**:

- Businesses adopting digital payment systems (e.g., mobile apps like PayPal).
- Healthcare systems using electronic medical records for better patient data management.
- Factories implementing IoT sensors for real-time monitoring of equipment.

Benefits:

- Increased efficiency and productivity.
- Better data accessibility and analysis.
- Enhanced customer experiences through digital interfaces.
- Cost reduction through automation.

6.**What is 8 bit color image?**

An **8-bit color image** refers to a digital image where each pixel's color is represented using **8 bits** (1 byte) of data. This determines the number of possible colors or shades that can be displayed for each pixel, often used in image processing and computer graphics for efficient storage and display.

Key Details:

- **Bit Depth**: Definition: - Bit depth refers to the number of bits used to represent the color or intensity of each pixel in a digital image. - It determines the number of possible colors or shades a pixel can display. Calculation of Color Possibilities: - For a given bit depth of N bits per channel, the number of possible values is 2^N. - Example: 8-bit depth = 2^8 = 256 possible values per channel.

  Bit Depth in Grayscale Images: - In an 8-bit grayscale image, each pixel uses 8 bits, allowing 256 shades of gray (0 = black, 255 = white).

  Bit Depth in RGB Images:

- **Color Representation**:
  - **Grayscale**: In an 8-bit grayscale image, each pixel represents one of 256 shades of gray, ranging from black (0) to white (255).
  - **Color (RGB)**: Definition: - **An 8-bit RGB color image** is a digital image where each pixel's color is represented by three 8-bit channels: Red (R), Green (G), and Blue (B). - Each channel uses 8 bits, allowing 256 possible intensity values (0 to 255) per channel. - Total bits per pixel = 8 + 8 + 8 = 24 bits, often called "24-bit color."
  - 2. Color Range: - Each channel (R, G, B) has 256 levels (2^8). - Total possible colors = 256 × 256 × 256 = 16,777,216 colors.

- 3. Pixel Representation: - Each pixel is defined by an (R, G, B) triplet. - Examples: - (255, 0, 0): Pure red - (0, 255, 0): Pure green - (0, 0, 255): Pure blue - (255, 255, 255): White - (0, 0, 0): Black
- 4. Difference from 8-bit Indexed Color: - 8-bit RGB uses 24 bits per pixel (8 bits per channel). - 8-bit indexed color uses a single 8-bit value per pixel to reference one of 256 colors in a palette (e.g., used in GIFs).
- 5. Applications: - Used in digital displays, monitors, and TVs for "true color" representation. - Common in image formats like JPEG and PNG for high-quality images. - Used in image processing tasks (e.g., resizing with bilinear interpolation).
- 6. Example Calculation for a 640 × 480 Image: - Pixels = 640 × 480 = 307,200 - Data per pixel = 3 bytes (24 bits) - Total data size (uncompressed) = 307,200 × 3 = 921,600 bytes = 0.9216 MB
- **Indexed Color**: In some 8-bit color images, each pixel stores an 8-bit index pointing to a color in a predefined palette (lookup table) of up to 256 colors. This is common in formats like GIF or PNG-8 to reduce file size.

- **Applications**:
  - Used in formats like GIF for simple graphics or animations due to limited color palettes.
  - Common in early computer displays and video games with restricted hardware.
  - Still used in scenarios where file size or processing power is a concern.
- **Limitations**:
  - Limited to 256 colors (or shades in grayscale), which can lead to color banding in smooth gradients.
  - Less detailed than higher bit-depth images (e.g., 16-bit or 32-bit), which offer more colors or finer gradations.

7. **what is weber ratio?**

1. Definition: - The Weber ratio, based on Weber's law, describes the relationship between a stimulus (e.g., light intensity) and the smallest detectable change in that stimulus, as perceived by the human sensory system. - It is defined as the ratio of the just-noticeable difference (ΔI) to the original stimulus intensity (I): Weber Ratio = ΔI / I - Here, ΔI is the minimum change in intensity required for a human to perceive a difference, and I is the background intensity.

2. Key Concept: - Weber’s law states that the just-noticeable difference (ΔI) is proportional to the original intensity (I), meaning the Weber ratio (ΔI / I) is approximately constant for a given sensory modality (e.g., vision) over a range of intensities. - For vision,

the Weber ratio is typically around 0.01 to 0.1 (1% to 10%), meaning a 1–10% change in light intensity is needed to notice a difference, depending on conditions.

3. Relevance to Digital Image Processing: - **Human Vision Simulation**: The Weber ratio is used in DIP to design image processing algorithms that mimic human perception, such as in non-uniform quantization (noted in the document under "Sampling and Quantization"). Non-uniform quantization allocates more levels to lower intensities where human sensitivity to changes is higher, aligning with Weber's law. - **Image Enhancement**: Techniques like contrast stretching (mentioned in the document) consider the Weber ratio to enhance image features that are perceptually significant to humans, ensuring changes in intensity are noticeable. - **Quantization**: In 8-bit grayscale images (256 levels, as discussed in your 8-bit color question), the Weber ratio helps determine how many intensity levels are needed to avoid perceptible banding, as human perception of intensity differences follows Weber's law.

8. **What is image sampling? Define saturation in digital image**.

1. Image Sampling Definition:
    - Image sampling refers to measuring the image at discrete intervals in space — selecting points on the continuous image function to represent as pixels, as stated in the document.
    - It converts a continuous analog image into a discrete digital image by capturing intensity or color values at specific spatial coordinates.

Key Concepts:

- Determines **spatial resolution** (e.g., number of pixels, such as 640 × 480 from your resolution question).
- Higher sampling rate: More pixels, finer details (e.g., 20×20 image for a 10×10 cm area sampled at 0.5 cm intervals, per document).
- Lower sampling rate: Fewer pixels, potential loss of detail or aliasing (distortion).
- Example: Sampling a 10×10 cm analog image at 1 cm intervals yields a 10×10 digital image (100 pixels); at 0.5 cm, a 20×20 image (400 pixels).

Process:

- Sensors (e.g., CCD array, as in document's "Sensor Array" section) capture light intensity at discrete grid points, forming a 2D pixel array.
- Ties to image acquisition (e.g., single sensor, sensor strip, or array, per your earlier question).

Applications:

- Scanners, digital cameras, medical imaging (CT, MRI), satellite imaging, OCR (document).
- Relates to resolution: A 640 × 480 image sampled at a specific rate yields 2.67 × 2 inches at 240 PPI (your resolution question).

Important Notes:

- Oversampling: Increases file size with minimal visual benefit.
- Undersampling: Causes aliasing, mitigated by pre-filtering (blurring), as noted in document.
- Connects to bit depth (your prior question): Sampling sets pixel count, while bit depth sets intensity levels per pixel.

2. Saturation in Digital Images Definition:

   - Saturation is the intensity or purity of a color in a digital image, indicating how vivid or muted it appears.
   - It is a component of the **HSV (Hue, Saturation, Value)** color model, referenced in the document's "Color Image Processing" section, where high saturation yields vibrant colors and low saturation yields grays.

Key Concepts:

- In HSV:
  - Hue: Color type (e.g., red, blue).
  - Saturation: Color intensity, from 0 (gray) to 1 (pure color).
  - Value: Brightness, from 0 (black) to 1 (bright).
- In 8-bit RGB images (your 8-bit color question), saturation is derived by converting RGB to HSV. E.g., (255, 0, 0) is fully saturated; (128, 128, 128) is desaturated (gray).

Role in Digital Imaging:

- **Color Enhancement**: Adjusting saturation enhances vibrancy or mutes colors (document's color image processing application).
- **Color Segmentation**: High-saturation colors aid object isolation (document's segmentation section).

- **Pseudo-coloring**: Saturation adjustments enhance grayscale-to-color conversions (document).

Example:

- RGB = (255, 0, 0) (pure red) in an 8-bit RGB image has Saturation ≈ 1 (100%) in HSV.
- RGB = (200, 200, 200) (light gray) has Saturation ≈ 0, appearing muted.

Applications:

- Image editing (e.g., increasing saturation for vivid photos).
- Display devices (document's "Image Display" section), where saturation affects RGB/HSV color rendering.
- Ties to 8-bit RGB: Saturation determines color vividness in 16.7 million colors.

9. **What is CMK and CMKY?**

CMY Color Model

Full Form:
Cyan, **M**agenta, **Y**ellow

Definition:
The **CMY color model** is a **subtractive color model** used in image processing and printing. It works by **subtracting light** from white, unlike the RGB model which adds light.

How It Works:

- Cyan absorbs **Red**
- Magenta absorbs **Green**
- Yellow absorbs **Blue**

Conversion from RGB:

If RGB values are normalized between 0 and 1:

C = 1 -R, M = 1 – G, Y = 1 - B

Application:
Used in **image theory**, color printers, and to describe how pigments or dyes combine.

🖨️ CMYK Color Model

Full Form:
Cyan, **M**agenta, **Y**ellow, **K**ey (Black)

Definition:
The **CMYK model** is an extension of the CMY model. It adds a **black (K)** component to produce deeper blacks and improve print quality.

Why Add Black (K)?

- Combining C+M+Y doesn't give a **pure black** — it results in a muddy dark gray.
- Adding a dedicated black ink improves **contrast**, **depth**, and **reduces ink usage**.

3. Mathematical Preliminaries [9L]

Neighbour of Pixels

In digital image processing, the neighbours of a pixel refer to the pixels immediately surrounding a given pixel in an image, typically defined by their spatial arrangement in a grid. The concept is fundamental for tasks like image filtering, segmentation, and feature extraction, as it describes how pixels are connected or related to one another. The two most common types of pixel neighbourhoods are:

1. **4-Neighborhood (4-Connectivity):**
    - A pixel at position (x, y) has four neighbours located horizontally and vertically adjacent to it.
    - These neighbours are at coordinates: (x+1, y), (x-1, y), (x, y+1), and (x, y-1).
    - This is also called the N4(p) neighbourhood, where p is the pixel at (x, y).
    - Example: For a pixel at (2, 2), the 4-neighbors are (3, 2), (1, 2), (2, 3), and (2, 1).
2. **8-Neighborhood (8-Connectivity):**
    - A pixel at position (x, y) has eight neighbours, including those horizontally, vertically, and diagonally adjacent.
    - These neighbours are at coordinates: (x+1, y), (x-1, y), (x, y+1), (x, y-1), (x+1, y+1), (x+1, y-1), (x-1, y+1), and (x-1, y-1).
    - This is called the N8(p) neighbourhood.
    - Example: For a pixel at (2, 2), the 8-neighbors include the 4-neighbors plus (3, 3), (3, 1), (1, 3), and (1, 1).

Additional Notes:

- **Diagonal Neighbours (ND(p)):** The four diagonal neighbours (x+1, y+1), (x+1, y-1), (x-1, y+1), and (x-1, y-1) are sometimes considered separately. The 8-neighborhood is the union of the 4-neighborhood and these diagonal neighbours: $N8(p) = N4(p) \cup ND(p)$.

- Applications: Neighbour definitions are crucial in algorithms like edge detection, morphological operations, and connected component labeling, where the connectivity (4 or 8) determines how pixels are grouped or processed.
- Boundary Considerations: Pixels at the edges or corners of an image have fewer neighbours, requiring special handling in algorithms.

Relations, Equivalence, and Transitive Closure

Relation: A relation on a set of pixels defines how they are related, typically based on spatial proximity (e.g., 4- or 8-neighborhood) or intensity similarity. Formally, it is a subset of the Cartesian product of the pixel set, where $(p, q) \in R$ means pixel p is related to pixel q.

Equivalence: An equivalence relation is a relation that is reflexive ($p \sim p$), symmetric (if $p \sim q$, then $q \sim p$), and transitive (if $p \sim q$ and $q \sim r$, then $p \sim r$). In image processing, it groups pixels into regions, such as connected components, where pixels in the same region are equivalent.

Transitive Closure: The transitive closure of a relation R is the smallest transitive relation containing R. It includes all pairs (p, q) such that there is a path from pixel p to pixel q via the relation R, used to identify all pixels in the same region (e.g., connected component) in image processing.

Distance Measures

Used to determine the proximity between two pixels:

1. Euclidean Distance $d(p, q) = \sqrt{[(x2 - x1)^2 + (y2 - y1)^2]}$

   Straight-line distance between two pixels. Common in edge detection, clustering (e.g. K-means), template matching.

2. Manhattan Distance (City Block Distance)

   $d(p, q) = |x2 - x1| + |y2 - y1|$

   Horizontal and vertical movement only. Faster to compute than Euclidean distance. Used in grid-based processing, morphology.

3. Chebyshev Distance $d(p, q) = \max(|x2 - x1|, |y2 - y1|)$

   Accounts for 8-connected pixels. Useful in morphological operations and flood fill.

Arithmetic and Logic Operations

Operation	Use Case
Addition	Blending, brightness enhancement
Subtraction	Motion detection, image differencing
Multiplication	Contrast adjustment, masking
AND	Apply binary masks to extract regions
OR	Combine masks or segmented regions
XOR	Detect differences between binary images
NOT	Image inversion, creating negative images

Fourier Transformation

The Fourier Transform is a mathematical technique that transforms a function from the spatial or time domain into the frequency domain. It expresses a non-periodic function as an integral of sinusoidal functions, helping to analyze the frequency components present in signals or images.
There are two types of Fourier transform i.e., forward Fourier transform and inverse Fourier transform.

Forward Fourier Transform

The **forward Fourier transform** is a mathematical technique used to transform a **time-domain signal into its frequency-domain representation**. This transformation is fundamental in various fields, including signal processing, image processing, and communications. Forward Fourier Transform is represented by F(k). The symbol for forward Fourier transform is f^*f*^(k) and is defined as:

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} \, dx$$

Inverse Fourier Transform

The inverse Fourier transform is the process of **converting a frequency-domain representation of a signal back into its time-domain form**. This is the reverse process of

the forward Fourier transform. Inverse Fourier Transform is represented by f(x). Symbol for Inverse Fourier transform is f˙*f*(x) and is defined as:

$$f(x) = F_k^{-1}[F(k)](x) = \int_{-\infty}^{\infty} F(k)e^{2\pi ikx}\,dk$$

Properties of 2D Fourier Transform

1. Linearity
2. Translation (Shift Theorem)
3. Scaling
4. Rotation
5. Separability
6. Conjugate symmetry

Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is a mathematical technique used to convert a discrete signal or digital image from the spatial (or time) domain into the frequency domain. It represents the signal as a finite sum of sinusoidal components with discrete frequencies, enabling frequency analysis of digital data.

📌 **DFT Formula**

$$F(u) = \sum_{x=0}^{N-1} f(x) \cdot e^{-j\frac{2\pi}{N}ux}$$

- Where $f(x)$ is the input sequence,
- $F(u)$ is the frequency spectrum,
- N is the total number of samples.

Discrete Cosine and Sine Transforms

Discrete Cosine Transform (DCT):

The DCT expresses a finite sequence of data points as a sum of cosine waves oscillating at different frequencies. It assumes the input data has **even symmetry**, which helps reduce discontinuities at boundaries and minimizes artifacts. The DCT is widely used in practical applications, notably in image and video compression standards such as JPEG and MPEG. It

compacts most of the signal energy into a few low-frequency components, making it highly efficient for compression.

The 1D DCT formula is:

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

where x_n is the input sequence, N is the number of samples, and C_k are the DCT coefficients.

Discrete Sine Transform (DST):

The DST is similar to the DCT but uses sine functions as its basis. It assumes the input data has **odd symmetry**, which is useful in certain boundary condition problems and in solving partial differential equations. The DST transforms the input sequence into a sum of sine waves oscillating at discrete frequencies.

The 1D DST formula is:

$$S_k = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{N}(n+1)(k+1)\right], \quad k = 0, 1, ..., N-1$$

where S_k are the DST coefficients.

Applications:

- **DCT** is extensively used in image and video compression (e.g., JPEG), noise reduction, and signal processing.
- **DST** finds application in solving boundary value problems, filtering, and spectral analysis where odd symmetry is needed.

ORGANISER SPECIAL

1. What is Hough transform? Its Application.

The **Hough Transform** is a feature extraction technique used in image processing and computer vision to detect simple shapes such as lines, circles, or ellipses in an image. It works by transforming points from the image space into a parameter space, where the desired shapes correspond to peaks or accumulations. This makes it robust to noise and partial occlusion.

Applications of Hough Transform

- **Line Detection:** Detect straight lines in images, useful in road lane detection, document analysis, and object recognition.

- **Circle and Ellipse Detection:** Identify circular shapes in images, important in medical imaging (e.g., detecting blood cells), industrial inspection, and robotics.
- **Shape Detection in Noisy Images:** Effective in detecting shapes even when the image contains noise or missing parts.
- **Object Recognition:** Helps recognize specific geometric shapes as part of larger objects.

2. Define Walsh Transform.

The **Walsh Transform**, often interchangeably referred to as the **Walsh-Hadamard Transform (WHT)** or **Hadamard Transform**, is a fundamental non-sinusoidal orthogonal transform used in digital signal and image processing. Unlike the Fourier Transform which decomposes signals into sinusoidal components, the Walsh Transform analyzes signals using a set of rectangular, binary-valued basis functions called **Walsh functions**.

Basis Functions: Walsh Functions : At the core of the Walsh Transform are **Walsh functions**, which are a complete orthogonal set of piecewise-constant waveforms. These functions are characterized by:

- **Binary Values:** They exclusively take on values of +1 or -1.
- **Rectangular/Square Waveform:** Their shape resembles square waves, exhibiting sharp transitions between +1 and -1.
- **Sequency Ordering:** Walsh functions can be ordered based on their "sequency," which is defined as half the number of zero-crossings (sign changes) within a given interval. This concept is analogous to frequency in Fourier analysis. As the sequency index increases, the Walsh function generally exhibits more oscillations or sign changes.
- **Orthogonality:** Any two distinct Walsh functions are orthogonal over their defined interval, meaning their inner product is zero. This property ensures that the transform can uniquely decompose a signal.

Applications : The unique properties of the Walsh Transform make it valuable in various practical applications, including:

- **Digital Signal Processing:** Filtering, compression, and analysis of discrete signals.
- **Image Processing:** Used in image compression (e.g., as an alternative to DCT), feature extraction, and pattern recognition, especially for binary images.

- **Data Compression:** Efficiently representing data with fewer coefficients for storage or transmission.
- **Communications:** Utilized in spread-spectrum communication systems and coding theory due to its orthogonal properties.
- **Medical and Biological Signal Processing:** Analysis of EEG and other physiological signals.

3. What is unitary transform?

A **unitary transform** is a linear transformation in a complex vector space (such as a Hilbert space) that **preserves the inner product (or dot product)** between vectors. In simpler terms, it's a transformation that rotates and/or reflects vectors without changing their length or the angle between any two vectors. This property makes unitary transforms crucial in various fields, especially in quantum mechanics and signal processing.

Unitary transforms are widely applied in:

- **Signal and Image Processing:** For analysis, compression, filtering, noise reduction, and feature extraction. Their energy compaction property (concentrating signal energy into a few coefficients) is highly valuable for compression.
- **Quantum Mechanics:** Unitary operators describe the time evolution of quantum states and are central to quantum mechanics, as they preserve the probability interpretation of wave functions.
- **Linear Algebra:** For diagonalizing Hermitian matrices and understanding symmetries.
- **Statistics:** In principal component analysis (PCA), where an orthogonal transform is used to decorrelate data.

Image Enhancement [8L]

1. Introduction

Image enhancement in digital image processing refers to techniques used to improve an image's visual quality or make it more suitable for specific applications like analysis or display. These methods are generally categorized into spatial domain and frequency domain techniques.

The primary goals of image enhancement are to:

- Improve visual appearance: This includes adjusting contrast, brightness, and sharpness.
- Highlight specific features: Making elements like edges or textures more discernible for better interpretation.
- Reduce noise or artifacts: Cleaning up imperfections in the image.
- Prepare images for further processing: Getting images ready for tasks such as segmentation or recognition.

Main Categories and Techniques

Spatial Domain Techniques

These methods directly manipulate pixel values.

1. Point Processing: Each pixel's value is modified independently.
    - Contrast Stretching: Contrast stretching is an image processing technique used to enhance the contrast of an image by expanding the range of intensity values. It adjusts the pixel intensity values to span a larger range, typically the full range of the display or output device (e.g., 0 to 255 for an 8-bit grayscale image), to make features more distinguishable. This is particularly useful for images with low contrast, where pixel values are clustered in a narrow range.

Applications

- Image Enhancement: Improves visibility of details in low-contrast images (e.g., satellite imagery, medical images like X-rays).
- Photography: Enhances photos taken in poor lighting conditions.
- Computer Vision: Preprocesses images to improve feature detection or segmentation.

Advantages

- Simple and computationally efficient.
- Enhances visibility of details without introducing significant artifacts.
- Works well for grayscale and color images (applied separately to each channel in color images).

Thresholding: Thresholding is an image processing technique used to segment an image by converting it into a binary image or separating it into distinct regions based on pixel intensity values. It involves comparing each pixel's intensity to a threshold value and assigning it to one of two categories (e.g., black or white) based on whether it meets the threshold criteria. Thresholding is widely used in computer vision and image analysis for tasks like object detection, text extraction, and image simplification.

Applications

- **Image Segmentation**: Isolating objects from the background (e.g., detecting shapes or objects in medical imaging).
- **Text Extraction**: Binarizing scanned documents for optical character recognition (OCR).
- **Object Detection**: Simplifying images to detect specific features or regions.
- **Pre-processing**: Preparing images for further analysis in computer vision pipelines.

Brightness Adjustment: Brightness adjustment is an image processing technique that modifies the intensity values of pixels in an image to make it

appear brighter or darker. It involves adding or subtracting a constant value to the pixel intensities, effectively shifting the overall brightness level without altering the contrast or color balance significantly. This technique is commonly used to enhance visibility or correct exposure issues in images.

2. Neighborhood Processing: **Neighborhood Processing** (also known as **Area Processing** or **Spatial Filtering**) is a fundamental category of image enhancement techniques in digital image processing. Unlike point processing methods that modify each pixel independently, neighborhood processing operations determine the new value of a pixel based on the values of its **surrounding pixels** within a defined local region, often called a **neighborhood** or **window**.

- Smoothing (Low-Pass Filtering): **Smoothing (Low-Pass Filtering)** is a fundamental technique in digital image processing, falling under the category of **Neighborhood Processing (Spatial Filtering)**. Its primary purpose is to **reduce noise** and **smooth out sharp intensity variations** in an image, resulting in a softer, often blurred, appearance.

Common Smoothing Filters

Here are the most widely used smoothing (low-pass) filters:

1. Mean (Average) Filter

- **Description:** This is the simplest low-pass filter. It replaces each pixel's value with the **simple average** of all pixel values within its defined rectangular neighborhood (e.g., a 3x3 window).
- **Pros:** Easy to implement, effective at reducing certain types of random (e.g., Gaussian) noise.
- **Cons:** Can significantly blur edges and fine details, as it treats all pixels in the neighborhood equally.

2. Gaussian Filter

- **Description:** This is a more sophisticated low-pass filter that uses a **weighted average**. Pixels closer to the center of the neighborhood are given **more weight** in the averaging process, while pixels further away receive less weight. The weights are determined by a **Gaussian function** (a bell-shaped curve)

- **Pros:** Provides a very smooth and natural-looking blur. It's generally more effective at noise reduction than the mean filter while **preserving edges better** because it gives more importance to the central pixel and its immediate vicinity. The degree of blur can be controlled by the standard deviation (σ) of the Gaussian function.

- **Cons:** Still introduces some blurring of details.

3. Median Filter

- **Description:** Unlike mean and Gaussian filters (which are linear filters), the median filter is a **non-linear spatial filter**. It replaces each pixel's value with the **median** value of all pixel values within its neighborhood. The median is the middle value in a sorted list of pixel intensities.

Pros:

- **Highly effective at removing "salt-and-pepper noise"** (random bright or dark pixels) and other impulse noises without significantly blurring edges. This is because outlier noise values have little impact on the median.
- **Preserves edges much better** than linear smoothing filters.

Cons: Can sometimes round off corners or introduce artifacts, and it's computationally more intensive than mean or Gaussian filters for large kernels because it requires sorting.

Applications of Smoothing (Low-Pass Filtering)

- **Noise Reduction:** This is the primary application, making images cleaner and more visually appealing.
- **Pre-processing for other tasks:** Smoothing is often a necessary first step before operations like edge detection (to prevent noise from being detected as edges), segmentation, or object recognition.
- **Data Compression:** Smoothing can help reduce the amount of high-frequency information, which can aid in certain compression algorithms.
- **Artistic Effects:** Creating a soft-focus or blurred effect for aesthetic purposes.

- Sharpening (High-Pass Filtering): Sharpening (High-Pass Filtering) is a crucial image enhancement technique in digital image processing, belonging to the Neighborhood Processing (Spatial Filtering) category. Its main objective is to enhance edges and fine details in an image, making it appear clearer, crisper, and more defined.

Common Sharpening Filters

Here are the most widely used sharpening (high-pass) filters:

1. Laplacian Filter

- **Description:** The Laplacian is a **second-order derivative filter** that highlights regions of rapid intensity change (edges). It measures the "rate of change of the rate of change" of intensity. A positive result indicates a transition from dark to light, and a negative result from light to dark. Areas of uniform intensity will result in a zero or near-zero output.
- **Application:** The direct output of a Laplacian filter is often a grayscale image where edges are highlighted. To get a sharpened image, this output is typically **added back to the original image**: $g(x,y)=f(x,y)-\text{Laplacian}(f(x,y))$ (The subtraction is often used to ensure the edges become brighter in the output. If the central coefficient of the kernel is positive, then we add; if negative, we subtract.)
- **Pros:** Simple, effective at highlighting edges.
- **Cons:** Very sensitive to noise, as noise itself represents high-frequency variations. Can produce double edges.

2. Unsharp Masking(High- boost Filtering)

- **Description:** This is a very popular and effective sharpening technique widely used in photography software. It works by **subtracting a "blurred" version of the image from the original image** to obtain a "detail mask," and then adding a weighted portion of this mask back to the original image.

  Pros: Provides excellent control over the sharpening effect. More robust to noise than direct derivative filters because it first smooths the image. Produces very natural-looking sharpened images.
- **Cons:** Involves multiple steps (blurring, subtraction, addition).

3. Gradient Filters (First-Order Derivatives, Derivative Filtering)

- **Description:** These filters approximate the **first derivative** of the image intensity. They are primarily used for **edge detection** but inherently produce a sharpened appearance because they highlight regions of significant intensity change. Common examples include Sobel, Prewitt, and Roberts operators.
- **Pros:** Provide both edge magnitude and direction information.
- **Cons:** Can be sensitive to noise, although less so than the Laplacian.

Advantages of Sharpening

- **Improved Visual Clarity:** Makes details more discernible and the image appear crisper.
- **Enhanced Features:** Highlights edges and fine structures crucial for analysis (e.g., in medical images or industrial inspection).
- **Better for Further Processing:** Can improve the performance of subsequent image analysis tasks like object recognition or segmentation.

3. **Histogram-Based Methods**: Histogram-Based Methods are a class of image enhancement techniques that operate by analyzing and modifying the histogram of an image. An image histogram is a graphical representation of the distribution of pixel intensities (or color values) within an image. It plots the number of pixels for each intensity level, providing a global overview of the image's tonal distribution. These methods are particularly effective for improving the contrast of images, especially those that appear dull, washed out, or have pixel values concentrated in a narrow range of the available dynamic spectrum. By manipulating the histogram, these techniques aim to redistribute pixel intensities to achieve a desired visual effect.

- Histogram Equalization:
  - Purpose: To enhance the global contrast of an image by making its intensity histogram as uniform as possible. It aims to spread out the most frequent intensity values across the entire available range of pixel intensities, effectively maximizing the image's dynamic range.
  - **Pros:** Automatic, computationally simple, and effective for many low-contrast images. It is a powerful general-purpose contrast enhancer.
    Cons: Can sometimes lead to an unnatural appearance, especially in photographs, by increasing the contrast of background noise or over-enhancing certain regions. It can also cause a significant shift in the mean brightness of the image. It's a **global** operation, meaning it doesn't consider local variations in contrast.
- Histogram Matching (Specification):
  - Purpose: To transform the histogram of an input image to match the shape of a specified (target) histogram. This allows for more controlled contrast enhancement or for normalizing images to have a consistent appearance (e.g., matching the lighting conditions of a reference image). Histogram Equalization is a special

case of histogram matching where the target histogram is a uniform distribution.

- **Pros:** Offers greater flexibility and control over the enhancement process. Useful for standardizing images from different sources or conditions.
- **Cons:** More complex to implement than histogram equalization. Requires a predefined or derived target histogram, which might not always be readily available or easy to specify.

4. **Color Enhancement** : Color Enhancement in digital image processing refers to a set of techniques aimed at improving the visual quality of color images by adjusting their color characteristics. The goal is often to make colors appear more vibrant, balanced, natural, or to correct for color inaccuracies introduced during image capture. Unlike grayscale image enhancement, which focuses solely on intensity, color enhancement involves manipulating the different color channels or components that define a color. This often necessitates working in different color spaces because some color spaces separate color information (chrominance) from brightness information (luminance), allowing for more intuitive and effective adjustments.

Frequency Domain Techniques

- **Frequency Domain Filters** are used for smoothing and sharpening of image by removal of high or low frequency components. Sometimes it is possible of removal of very high and very low frequency. Frequency domain filters are different from spatial domain filters as it basically focuses on the frequency of the images. It is basically done for two basic operation i.e., Smoothing and Sharpening. These are of 3 types:

- **Low-Pass Filtering**: Low pass filter removes the high frequency components that means it keeps low frequency components. It is used for smoothing the image. It is used to smoothen the image by attenuating high frequency components and preserving low frequency components. High-Pass Filtering: Amplifies high-frequency components to enhance edges and fine details, making the image appear sharper.
- **High pass filter:** High pass filter removes the low frequency components that means it keeps high frequency components. It is used for sharpening the image. It is used to sharpen the image by attenuating low frequency components and preserving high frequency components.

- **Band pass filter:** Band pass filter removes the very low frequency and very high frequency components that means it keeps the moderate range band of frequencies. Band pass filtering is used to enhance edges while reducing the noise at the same time.

Homomorphic Filtering :

Homomorphic filtering is a technique used in image and signal processing to enhance images by handling multiplicative noise or illumination variations. It transforms a multiplicative relationship between signal components (e.g., illumination and reflectance in images) into an additive one by working in the logarithmic domain, allowing linear filtering techniques to be applied.

✅ **Applications of Homomorphic Filtering**

- **Image Enhancement**: Improves visibility in images with uneven illumination (e.g., medical imaging, satellite imagery).
- **Shadow Removal**: Reduces the effect of shadows while preserving details.
- **Speech Processing**: Separates slowly varying vocal tract characteristics from excitation signals.

✅ **Advantages**

- Effectively handles multiplicative noise or illumination variations.
- Enhances details without overly amplifying noise.

✅ **Limitations**

- Requires careful tuning of filter parameters .
- Computationally intensive due to Fourier transforms.
- May introduce artifacts if the filter is not properly designed.

ORGANISER SPECIAL

1. Difference between Image restoration and image enhancement.

Aspect	Image Enhancement	Image Restoration
Definition	Techniques to improve visual quality or utility of an image for human perception or further processing.	Techniques to recover the original image by reversing specific degradations (e.g., blur, noise).
Objective	Enhance perceptual appeal or highlight features (e.g., edges, contrast) without necessarily restoring the original.	Reconstruct the original image as closely as possible to its pre-degraded state.
Nature	Subjective; driven by application or aesthetic goals.	Objective; aims for fidelity to the original image.
Degradation Model	Not required; focuses on manipulating image attributes.	Required or estimated; relies on a model of degradation (e.g., blur function, noise type).
Techniques	- Contrast stretching - Sharpening (e.g., homomorphic filtering, unsharp masking) - Histogram equalization - Color balancing	- Inverse filtering - Wiener filtering - Denoising (e.g., median filter, wavelet-based) - Inpainting
Role of Homomorphic Filtering	Used as a sharpening technique to enhance high-frequency details (edges) and correct uneven illumination.	Can contribute if degradation involves multiplicative illumination, but less precise than dedicated restoration methods.
Example	Sharpening a photo for social media by boosting edges with homomorphic filtering.	Removing motion blur from a photo using deconvolution with a known point spread function.

2. Describe smoothing linear spatial filtering. What is bit plane slicing method?

📌 Smoothing Linear Spatial Filtering

Definition:
Smoothing linear spatial filtering is an image processing technique that applies a linear filter (or kernel) to reduce noise, blur fine details, or smooth an image by averaging pixel intensities in a local neighborhood. It operates directly in the **spatial domain** (i.e., on pixel values) rather than the frequency domain.

🔷 Key Characteristics:

- **Purpose:**
  Reduces high-frequency components (e.g., noise, sharp edges) to produce a smoother image. It is often used as a **preprocessing step** for enhancement or analysis.
- **Mechanism:**
  Convolves the image with a **kernel** (a small matrix), where the output pixel value is a **weighted sum of neighboring pixels**. The kernel defines the weights, which are typically positive and **normalized** (sum to 1).

✅ Applications of Smoothing Linear Spatial Filtering

- **Noise reduction** in medical or satellite images.
- **Preprocessing** for edge detection or segmentation.
- **Smoothing before enhancement** techniques like homomorphic filtering to avoid amplifying noise.

✅ Advantages

- **Simple and computationally efficient** to implement.
- **Effective for reducing random (additive) noise.**

✅ Limitations

- **Blurs edges and fine details**, which may be undesirable for tasks like sharpening.
- **Less effective for non-additive noise** compared to nonlinear filters like the **median filter**

📌 Bit Plane Slicing

Definition:
Bit plane slicing is a technique in image processing that decomposes a grayscale image into its binary bit planes, where each plane represents the contribution of a specific bit position to the pixel intensity. It's used to analyze or enhance specific intensity levels in an image.

🔷 **Key Characteristics:**

- **Image Representation:**
  In an 8-bit grayscale image, each pixel has an intensity value ranging from **0 to 255**, represented using **8 bits**. Bit plane slicing extracts each of these bits (from the **Least Significant Bit (LSB)** to the **Most Significant Bit (MSB)**) into separate binary images.

- **Example:**
  A pixel with intensity **157** has a binary representation of **10011101**. This is split into 8 bit planes:

  - Bit 0 (LSB): 1
  - Bit 1: 0
  - Bit 2: 1
  - Bit 3: 1
  - Bit 4: 1
  - Bit 5: 0
  - Bit 6: 0
  - Bit 7 (MSB): 1

- **Interpretation:**
  Each bit plane is a **binary image** where a pixel is **1 if the corresponding bit is 1**, and **0 otherwise**.

Applications:

- **Image Compression**: Analyze which bit planes (e.g., lower-order bits) can be discarded with minimal visual impact.
- **Image Enhancement**: Manipulate specific bit planes to emphasize certain intensity levels or details.
- **Steganography**: Hide data in lower-order bit planes (e.g., LSB) where changes are less perceptible.

- **Feature Extraction**: Identify patterns or textures in specific bit planes for analysis.

3. What is Quad Tree ?

Definition:
A **Quad Tree** is a tree data structure used in image processing and computer graphics to recursively divide a two-dimensional space (such as an image) into four quadrants or regions. Each node in the quad tree represents a square area of the image and can be subdivided into four smaller nodes if that area is not homogeneous.

🔷 **Key Features:**

- **Recursive subdivision:** The image is divided into four equal parts repeatedly until a criterion (e.g., uniformity of pixel values) is met.
- **Efficient representation:** Helps in compressing images by representing large uniform areas with fewer nodes.
- **Used for:** Image compression, region segmentation, and efficient spatial indexing.

🔷 **Example Use:**

- In a binary image, if a region is entirely black or white, it is represented as a leaf node.
- If mixed, it is subdivided into four child nodes, and the process continues recursively.

4. **What is salt and pepper noise? What is Gaussian noise?**

Salt and Pepper Noise

Definition: Salt and pepper noise is a type of impulse noise where random pixels in an image are set to extreme intensity values, appearing as white ("salt") or black ("pepper") dots. It results in a speckled appearance.

Characteristics:

- **Cause**: Occurs due to errors in image acquisition or transmission, such as faulty sensors, bit errors, or dead/stuck pixels in a camera.
- **Appearance**: Randomly scattered pixels with intensities at the maximum (e.g., 255 in an 8-bit grayscale image, appearing white) or minimum (e.g., 0, appearing black).
- **Model**: A certain percentage of pixels are corrupted, with equal or unequal probability of being "salt" (high intensity) or "pepper" (low intensity).
- **Effect**: Creates sharp, localized distortions that disrupt image details but affect only a small fraction of pixels.
- **Example**: In a grayscale image, a pixel with original intensity 128 might randomly become 0 (black) or 255 (white).
- **Applications:** Common in medical imaging (e.g., X-rays), satellite imagery, or low-quality camera captures.

Gaussian Noise

Definition: Gaussian noise is a type of statistical noise where pixel intensity values are perturbed by random variations following a Gaussian (normal) distribution. It affects all pixels in the image, not just a few.

Characteristics:

- Noise values are distributed around zero with a bell-shaped curve.
- Affects all pixels with varying intensity values.
- Appears as grainy texture, generally less perceptible than salt and pepper noise.
- **Applications:** Common in low-light photography, medical imaging (e.g., MRI), or any scenario with sensor noise.

5. Difference between spatial Domain filtering and frequency domain filtering.

Aspect	Spatial Domain Filtering	Frequency Domain Filtering
Definition	Directly processes the image pixels in the spatial domain.	Processes the image by modifying its frequency components.
Operation	Uses convolution with spatial filters (kernels).	Uses Fourier transform to convert image to frequency domain, applies filters, then inverse transform.
Example Filters	Smoothing (mean), sharpening (Laplacian), edge detection.	Low-pass, high-pass, band-pass filters applied in frequency domain.
Complexity	Computationally simpler and intuitive.	More complex due to transforms but useful for global filtering.
Effectiveness	Effective for local operations and small neighborhoods.	Effective for global features and frequency-based operations.
Noise Handling	Directly suppresses noise by averaging or median filters.	Can selectively filter frequencies to remove periodic noise.
Implementation	Operates on pixel intensities directly.	Requires Fourier transform and inverse transform steps.
Suitability	Good for real-time and simple tasks.	Suitable for tasks requiring frequency analysis and filtering.

Image Restoration

Image restoration is the process of improving or recovering the quality of a degraded or damaged image to make it more visually clear or usable. It involves techniques to remove noise, blur, artifacts, or other distortions while preserving or reconstructing the original details. This is commonly used in fields like photography, medical imaging, satellite imaging, and archival restoration.

Common Techniques in Image Restoration:

1. **Deblurring**:
    - Addresses blur caused by motion, defocus, or atmospheric turbulence.
    - Methods: Inverse filtering, Wiener filtering, blind deconvolution, or deep learning-based approaches (e.g., convolutional neural networks).
2. **Denoising**:
    - Removes noise (e.g., Gaussian, salt-and-pepper) while preserving edges and details.
    - Methods: Median filtering, Gaussian smoothing, wavelet-based denoising, or neural network models like DnCNN.
3. **Inpainting**:
    - Reconstructs missing or corrupted parts of an image (e.g., scratches, tears, or occlusions).
    - Methods: Diffusion-based inpainting, patch-based methods, or generative AI models like GANs (Generative Adversarial Networks).
4. **Super-Resolution**:
    - Enhances resolution to recover fine details from low-resolution images.
    - Methods: Bicubic interpolation, sparse coding, or deep learning models like SRGAN or ESRGAN.
5. **Color Restoration**:
    - Corrects faded colors or color imbalances in old photos or films.
    - Methods: Histogram equalization, color transfer, or AI-based colorization models.
6. **Artifact Removal**:
    - Eliminates compression artifacts (e.g., JPEG blocking) or scanning imperfections.
    - Methods: Total variation denoising, artifact suppression networks.

Degradation Model

In image restoration, a **degradation model** describes how an image is corrupted from its original form. It provides a mathematical framework to understand and reverse the degradation process, enabling the recovery of the original or near-original image. The model typically accounts for factors like blur, noise, or other distortions.

Standard Degradation Model

The degradation of an image is often represented as:

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

Where:

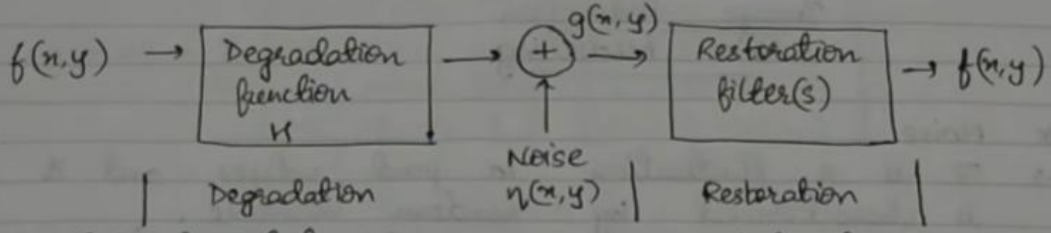
- $g(x,y)$: Degraded (observed) image.
- $f(x,y)$: Original (ideal) image.
- $h(x,y)$: Point Spread Function (PSF) or degradation function (e.g., blur kernel).
- $*$: Convolution operation, modeling effects like motion or defocus blur.
- $n(x,y)$: Additive noise (e.g., Gaussian, Poisson, or salt-and-pepper noise).

Components of the Degradation Model

1. **Blur (Convolution with PSF)**:
    - **Types**: Motion blur (linear or non-linear), defocus blur, atmospheric turbulence.
    - **PSF**: Describes how a single point in the original image spreads in the degraded image. For example, a Gaussian kernel for defocus blur or a line kernel for motion blur.
    - **Example**: Motion blur from camera shake can be modeled as a linear PSF along the motion path.
2. **Noise**:
    - **Types**:
        - **Gaussian Noise**: Random variations with a normal distribution, common in low-light photography.
        - **Salt-and-Pepper Noise**: Random black/white pixels, often from sensor errors.
        - **Poisson Noise**: Shot noise in low-photon-count imaging (e.g., medical or astronomical images).
    - **Impact**: Noise adds random fluctuations that obscure details.
3. **Other Degradations**:
    - **Compression Artifacts**: JPEG/MPEG blocking or ringing effects.
    - **Fading**: Color or intensity loss in old photographs.
    - **Occlusions**: Missing regions due to scratches or obstructions.
    - **Downsampling**: Loss of resolution in low-quality images.

Image Restoration

Image Degradation / Restoration Model



→ The degraded image in the spatial domain is given by:

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

where $h(x,y)$ is the spatial representation of the degradation function.

→ The degraded image in the frequency domain is given by:

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

Noise Models

* Sources of noise

→ Image acquisition, digitization, transmission
→ Image sensors: noise occurs due to environmental conditions and by the quality of the sensing elements.

* White noise

→ In signal processing, white noise is a random signal having equal intensity at different frequencies, giving it a constant power spectral density. The Fourier spectrum of noise is constant.

Types of Noise

1) Gaussian noise
→ Random noise that enters a system can be modelled as a Gaussian or normal distribution.

→ This noise affects both, dark and light areas of image.

→ The PDF of a Gaussian random variable, z, is given by

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

where $z \rightarrow$ gray level
$\mu \rightarrow$ mean of average values of z
$\sigma \rightarrow$ standard deviation
$\sigma^2 \rightarrow$ variance

2) Impulse noise
→ It is also known as shot noise, salt and pepper noise, and binary noise.

→ It occurs mostly because of sensor and memory problem because of which pixels are assigned incorrect maximum values

→ The PDF of (bipolar) impulse noise is given by

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

If either P_a or P_b is zero, impulse noise is called unipolar.

Q1. Why is impulse noise known as salt and pepper noise?

Ans: If neither of the two probabilities P_a or P_b are zero, and especially if they are approximately equal, impulse noise values resemble salt- and -pepper granules randomly distributed over the image. For this reason, bipolar impulse noise is also called salt-and-pepper noise.

3) Poisson noise

→ This type of noise manifests as a random structure or texture in images.

→ It is very common in X-ray images.

→ The PDF of Poisson noise is given by:

$$p(z) = \frac{(nP)^z e^{-nP}}{z!}$$

where $n \rightarrow$ total no. of pixels
$z \rightarrow$ gray level
$P \rightarrow$ ratio of noise pixels to the total no. of pixels

4) Exponential noise

→ This type of noise occurs mostly due to the illumination problems

→ It is observed in laser imaging

→ The PDF of exponential noise is given by:

$$p(z) = \begin{cases} ae^{-az} & \text{for } z \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{where } a > 0.$$

Mean → $1/a$
Variance → $1/a^2$

5) Gamma noise (Erlang)

→ This type of noise also occurs mostly due to the illumination problems.

→ The PDF is given by:

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

where $a > 0$ and b is a positive integer.

Mean → b/a

Variance → b/a^2

a → min. gray value
b → max. gray value

6) Rayleigh noise

→ This type of noise is mostly present in range images.

→ Range images are mostly used in remote sensing applications.

→ The PDF is given by:

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

Mean → $a + \sqrt{\frac{\pi b}{4}}$

Variance → $\frac{b(4-\pi)}{4}$.

7) Uniform noise

→ It is also a very popular noise which occurs in images where different values of noise are equally probable.

→ It occurs because of Quantization noise.

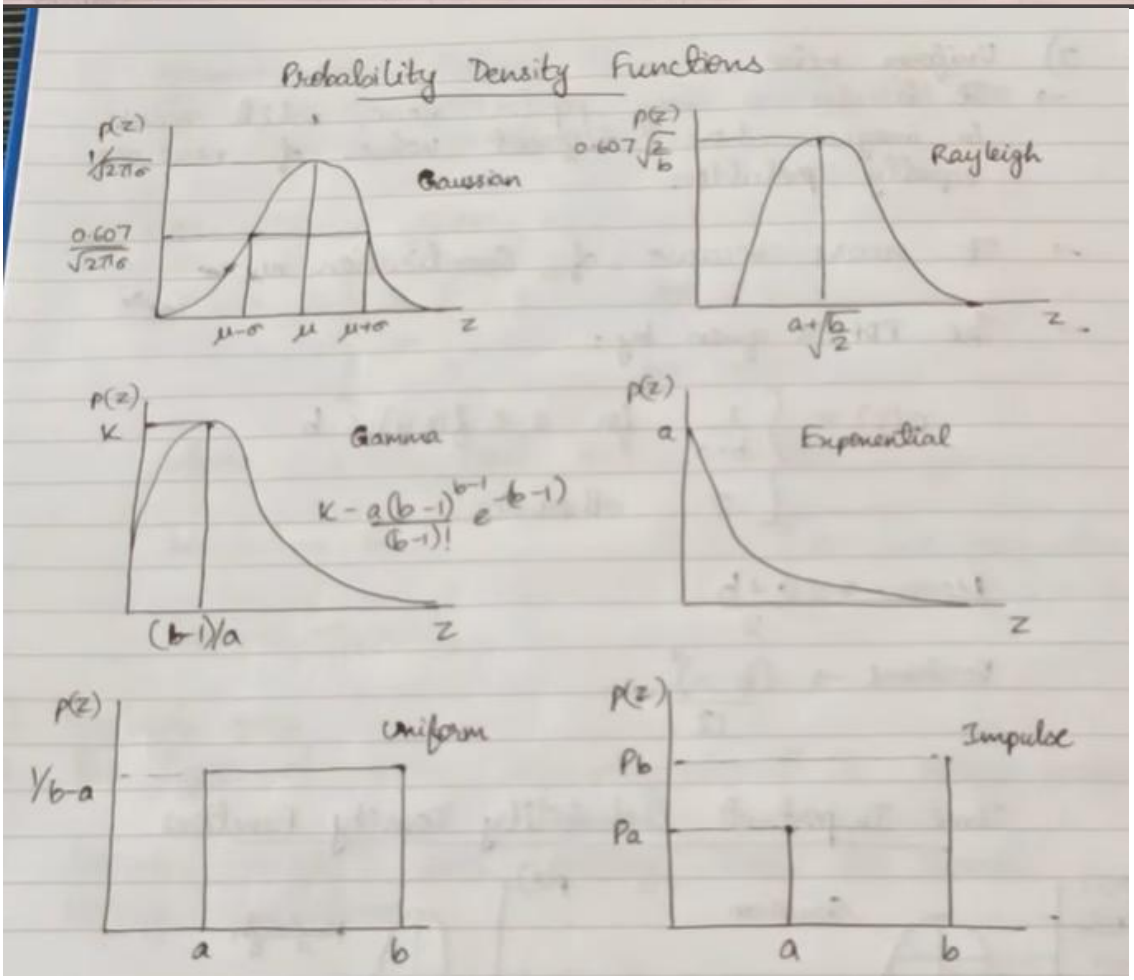
→ The PDF is given by:

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{for } a \le f(x,y) \le b \\ 0 & \text{otherwise} \end{cases}$$

Mean → $\frac{a+b}{2}$

Variance → $\frac{(b-a)^2}{12}$

Probability Density Functions



Noise Modelling

1) Gaussian
   - Electronic circuit noise, sensor noise due to poor illumination and/or high temperature.

2) Rayleigh
   - Range imaging

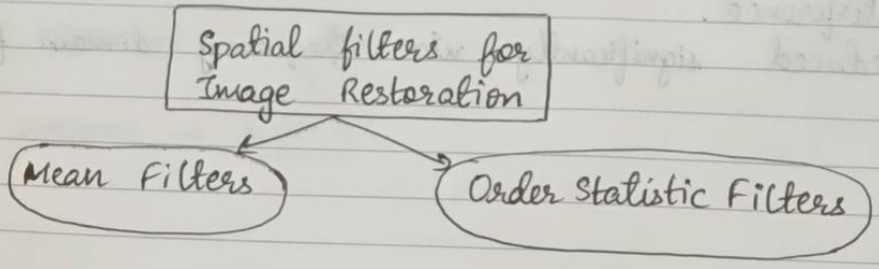
) Exponential and gamma
   - Laser imaging

) Impulse
   - Quick transients, such as faulty switching

Uniform
- Least descriptive
- Basis for numerous random number generators

Image Restoration in presence of Noise Only

Spatial filtering, which is used for image smoothening and sharpening, can also be used to remove noise.



Mean Filters

1) Arithmetic mean filter

→ This filter removes local variations within the image.

→ It is similar to low pass filter.

→ It is useful in removing Gaussian noise and uniform noise.

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

3×3

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

where $g(s,t)$ → mask applied over blurred image.

2) Geometric mean filter

→ This filter eliminates Gaussian noise.

→ It is ineffective for pepper type of noise.

→ This filter achieves smoothing comparable to the arithmetic mean filter, but it tends to lose less image details in the process.

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

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}_{3\times3}$$

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

3) Harmonic mean filter

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

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & (5) & 6 \\ 7 & 8 & 9 \end{bmatrix}_{3\times3}$$

$$f(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}} = \frac{9}{\left(\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}\cdots\frac{1}{9}\right)}$$

4) Contra-harmonic mean filter

→ It is well suited for reducing the effects of salt-and-pepper noise.
→ $Q>0$ for elimination of pepper noise and $Q<0$ for elimination of salt noise.
→ Q is the order of the filter.

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

when $Q=0$, it becomes arithmetic mean filter.
$Q=1$, it becomes harmonic mean filter.

* Usage:

→ Arithmetic and geometric mean filters: suited for Gaussian or uniform noise.
→ Contraharmonic filters: suited for impulse noise.

Order Statistic Filters

1) Median filter.

→ It is an example of non-linear filter.
→ It works by sorting the list and finding the median. The center pixel is then replaced by the median value.
→ It is effective in the presence of both bipolar and unipolar impulse noise.

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

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

2) Maximum filter

→ It selects the largest value in the sorted list.
→ It is used for removing pepper noise

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

3) Minimum filter

→ It selects the minimum value in the sorted list.
→ It is used for removing salt noise.

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

4) Midpoint filter

→ This filter selects the midpoint, which is the average of the minimum and maximum values.
→ It is very effective in removing Gaussian noise and Uniform noise.

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

5) Alpha-trimmed mean filter

→ It is based on the concept of computation of the average of the pixels that falls within the window.

→ It works by deleting the d/2 lowest and the d/2 highest gray-level values.

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

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

[If d = 0 → arithmetic mean filter]

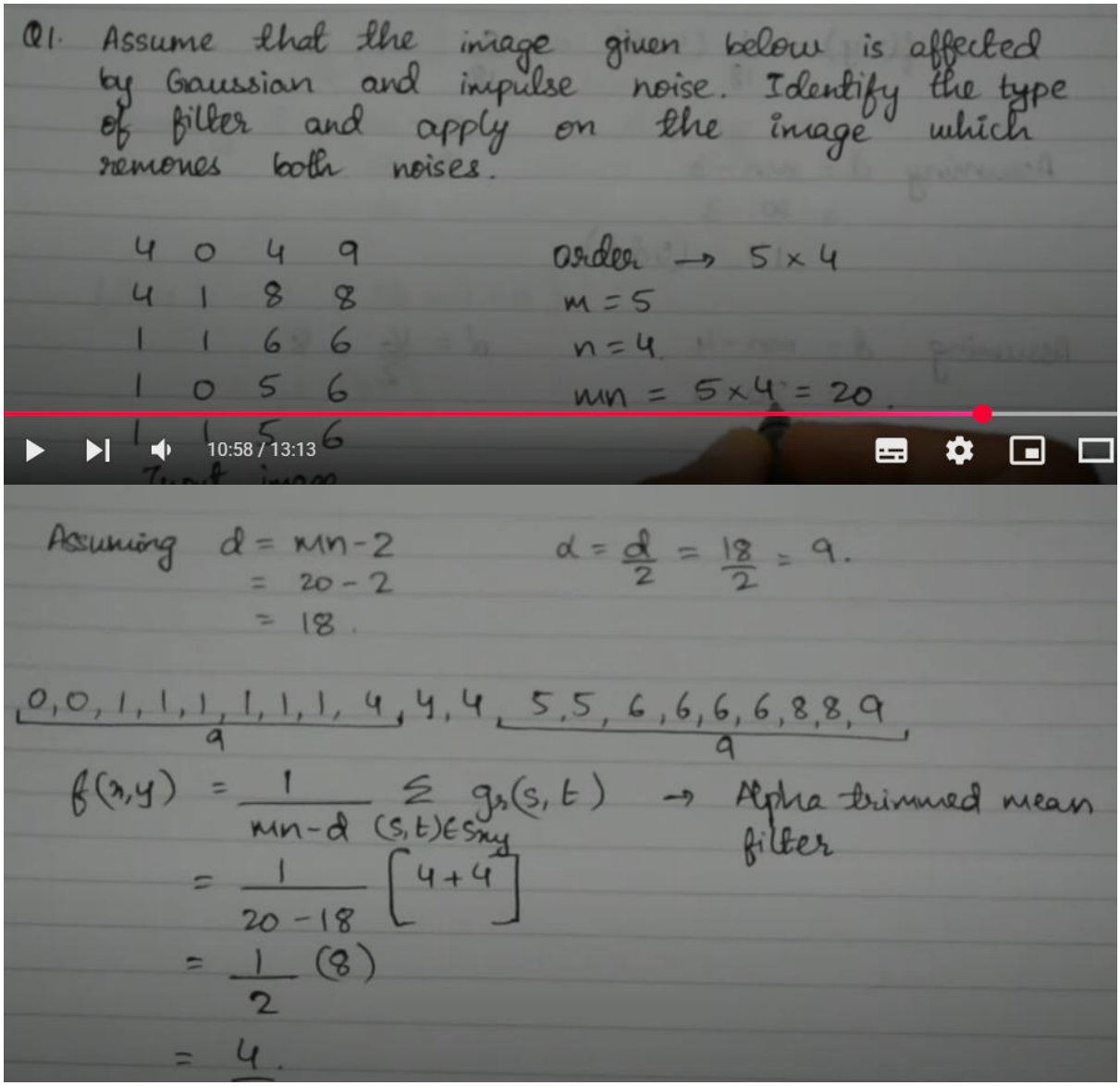
[If d = mn-1 → median filter]

[If d = mn-1 → median filter]

6) Trimmed mean filter

→ It works by removing the max. and min. values and calculating the average of the remaining values.

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



Now we can replace it with centre pixel value in input image.

The center pixel is 1, located at **(2,1)** in the original image.

📌 **Definition:**

Image Restoration is the process of recovering an image that has been **degraded** by known distortions such as **blur**, **noise**, or **geometric distortions**, with the goal of reconstructing an image as **close to the original** as possible.

Unlike **image enhancement**, which is subjective, image restoration is a **mathematical and objective** process based on **models of degradation and recovery**.

🔹 **Image Restoration**

📌 **What is Image Restoration?**

Image restoration is the process of recovering a clear, original image from a **degraded** version that has been blurred, noisy, or distorted. Unlike **image enhancement** (your prior question),

which improves visual appearance subjectively, restoration uses a **mathematical model** to reverse degradation and reconstruct the true image as accurately as possible. Think of it like cleaning a dirty, scratched window to see the scene clearly again—it aims to undo damage like blur, noise, or geometric warping.

📌 **Why Use Image Restoration?**

- To **remove blur** (e.g., from camera shake in photos).
- To **reduce noise** (e.g., speckles in medical images).
- To **correct distortions** (e.g., warped satellite images).
- To **recover details** for analysis (e.g., license plates in surveillance).

Image Degradation Model

The image degradation model is a conceptual framework used in image processing to describe how an observed image is affected by various factors that degrade its quality, such as noise, blur, or distortions. Understanding this model is crucial for designing restoration techniques to recover the original image. Below is a detailed explanation tailored for a 7-mark response, covering the definition, purpose, components, mathematical representation, advantages, limitations, and applications.

1. **Definition (1 mark)**

   The image degradation model describes the process by which an original image $f(x, y)$ is transformed into a degraded image $g(x, y)$ due to factors like blurring, noise, or other distortions. It provides a mathematical representation of the degradation process, typically involving a degradation function and additive noise, to facilitate image restoration.

2. **Purpose (1 mark)**

   The purpose of the degradation model is to characterize the sources of image quality degradation, enabling the development of restoration techniques (e.g., inverse filtering, Wiener filtering) to reconstruct the original image. It helps analyze the impact of degradation factors like blur or noise and guides the design of algorithms to improve image quality.

3. **Components and Mathematical Representation (2 marks)**

   The degradation model typically consists of two main components:

   - **Degradation Function**: Represents systematic distortions, such as blurring caused by motion, defocus, or atmospheric effects. This is often modeled as a convolution with a point spread function (PSF), $h(x, y)$.
   - **Noise**: Represents random variations, such as sensor noise or environmental interference, modeled as an additive term $n(x, y)$.
   - **Mathematical Model**: The degraded image $g(x, y)$ is expressed as:

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

     where:

     - $f(x, y)$: Original image.
     - $h(x, y)$: Degradation function (PSF), typically a blur kernel.
     - $*$: Convolution operation.
     - $n(x, y)$: Additive noise (e.g., Gaussian or salt-and-pepper noise).
     - In the frequency domain, using the Fourier Transform, this becomes:

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

       where G, H, F, N are the Fourier transforms of g, h, f, n, respectively.

   - **Process**: The model assumes that the degradation function $h(x, y)$ blurs the original image, and noise $n(x, y)$ adds random distortions, resulting in the observed degraded image.

Advantages (1 mark)

- **Guides Restoration**: Provides a framework to design inverse or adaptive restoration techniques to recover the original image.
- **Analyzes Degradation**: Helps identify and quantify sources of degradation (e.g., blur type or noise characteristics).
- **Flexible**: Applicable to various degradation types, such as motion blur, defocus blur, or additive noise.

Limitations (1 mark)

- **Model Assumptions**: Assumes a known degradation function and noise characteristics, which may be difficult to estimate in real-world scenarios.
- **Noise Sensitivity**: Restoration based on the model can amplify noise, especially in inverse filtering.
- **Complexity**: Accurate modeling and restoration require significant computational resources, particularly in the frequency domain.

Applications and Color Image Considerations (1 mark)

- **Applications**:
    - **Medical Imaging**: Models degradation in X-rays or MRIs (e.g., blur from motion or noise from sensors) to restore clear images for diagnosis.
    - **Astronomy**: Corrects blur in telescope images caused by atmospheric turbulence.
    - **Photography**: Restores images degraded by camera shake or lens imperfections.
    - **Computer Vision**: Preprocesses degraded images for tasks like object recognition or tracking.
- **Color Images**: For color images (e.g., RGB), the degradation model is applied independently to each color channel, with separate PSFs and noise models for Red, Green, and Blue. Alternatively, in HSV/HSI spaces, degradation is modeled for the intensity channel to focus on brightness-related distortions while preserving color information.

Image Restoration Model

The image restoration model is a framework in image processing designed to recover an original image from its degraded version by reversing the effects of degradation, such as blur, noise, or distortions. It relies on understanding the degradation process to apply appropriate techniques for restoration. Below is a detailed explanation tailored for a 7-mark response, covering the definition, purpose, components, techniques, advantages, limitations, and applications.

1. **Definition (1 mark)**
   The image restoration model describes the process of estimating the original image $f(x, y)$ from a degraded image $g(x, y)$, which has been affected by factors like blur, noise, or other distortions. It typically uses a mathematical model of degradation, such as $g(x, y) = h(x, y) * f(x, y) + n(x, y)$, where $h(x, y)$ is the degradation function (e.g., point spread function, PSF) and $n(x, y)$ is additive noise, to guide the restoration process.
2. **Purpose (1 mark)**
   The primary goal of the restoration model is to reverse the degradation effects to recover a high-quality approximation of the original image. It aims to remove or reduce blur, noise, or other artifacts, improving visual clarity for human interpretation or automated analysis in applications like medical imaging or computer vision.

3. **Components and Techniques (2 marks)**

   The restoration model is based on the degradation model and employs various techniques to recover the original image:

   - **Degradation Model**: Assumes the degraded image is formed as $g(x,y) = h(x,y) * f(x,y) + n(x,y)$, or in the frequency domain: $G(u,v) = H(u,v) \cdot F(u,v) + N(u,v)$. The restoration process estimates $F(u,v)$ or $f(x,y)$.
   - **Restoration Techniques**:
     - **Inverse Filtering**: Directly inverts the degradation function in the frequency domain: $F(u,v) = G(u,v)/H(u,v)$. Effective for known PSFs but sensitive to noise.
     - **Wiener Filtering**: Minimizes the mean square error between the original and restored image, accounting for noise: $F(u,v) = \frac{H^*(u,v) \cdot G(u,v)}{|H(u,v)|^2 + \frac{S_n(u,v)}{S_f(u,v)}}$, where S_n and S_f are power spectra of noise and the original image.
     - **Regularized Least Squares (e.g., Tikhonov Regularization)**: Adds constraints to stabilize restoration against noise, solving for $f(x,y)$ by minimizing a cost function.
     - **Blind Deconvolution**: Estimates both the original image and the PSF when $h(x,y)$ is unknown, often using iterative methods.
   - **Process**: Apply the chosen technique (often in the frequency domain using FFT), adjust for noise, and transform back to the spatial domain to obtain the restored image.

Advantages (1 mark)

- **Improved Image Quality**: Effectively reduces blur and noise, enhancing details for better visualization or analysis.
- **Model-Based**: Leverages the degradation model for systematic restoration, improving accuracy when the PSF is known.
- **Versatile**: Applicable to various degradation types, such as motion blur, defocus, or additive noise.

Limitations (1 mark)

- **Noise Sensitivity**: Techniques like inverse filtering amplify noise, leading to artifacts in the restored image.
- **PSF Dependency**: Requires accurate knowledge of the degradation function $h(x,y)$; unknown or complex PSFs make restoration challenging.
- **Computational Complexity**: Frequency-domain methods and iterative techniques are computationally intensive.

Applications and Color Image Considerations (1 mark)

- **Applications**:
  - **Medical Imaging**: Restores X-rays, MRIs, or CT scans degraded by motion blur or noise for accurate diagnosis.

- **Astronomy**: Corrects telescope images blurred by atmospheric turbulence to reveal celestial details.
- **Photography**: Recovers sharpness in images affected by camera shake or lens imperfections.
- **Computer Vision**: Preprocesses degraded images for tasks like object recognition or tracking.

- **Color Images**: For color images (e.g., RGB), restoration is applied independently to each color channel (Red, Green, Blue) using the same or channel-specific degradation models. In HSV/HSI spaces, restoration focuses on the intensity channel to avoid color distortions while improving clarity.

Discrete Formulation in Image Restoration

Discrete formulation refers to the mathematical representation of the image degradation and restoration processes in a discrete domain, suitable for digital images composed of pixels. It models the degradation process and facilitates the design of restoration algorithms using discrete mathematics, typically in the form of matrix operations or discrete convolutions. Below is a detailed explanation tailored for a 7-mark response, covering the definition, purpose, formulation, techniques, advantages, limitations, and applications.

1. **Definition (1 mark)**
   Discrete formulation in image restoration represents the degradation of a digital image as a discrete mathematical model, where the image is treated as a grid of pixel values. The degraded image is modeled as a result of a discrete convolution with a degradation function (e.g., point spread function, PSF) plus additive noise, and restoration involves solving a discrete system to estimate the original image.

2. **Purpose (1 mark)**
   The purpose of discrete formulation is to provide a computationally tractable model for image degradation and restoration, enabling the use of numerical methods to recover the original image from its degraded version. It allows the application of linear algebra or discrete signal processing techniques to address blur, noise, or other distortions in digital images.

Discrete Formulation and Techniques (2 marks)

- **Degradation Model**:

In matrix form, this becomes:

$$\mathbf{g} = \mathbf{H}\mathbf{f} + \mathbf{n}$$

where $\mathbf{g}$, $\mathbf{f}$, and $\mathbf{n}$ are vectorized forms of the degraded image, original image, and noise, respectively, and $\mathbf{H}$ is a degradation matrix (often Toeplitz or block-Toeplitz for convolution).

Restoration Techniques:

- **Inverse Filtering**: Solves for $\mathbf{f}$ by inverting $\mathbf{H}$: $\mathbf{f} = \mathbf{H}^{-1}\mathbf{g}$. However, this is ill-posed due to noise amplification and non-invertible $\mathbf{H}$.
- **Wiener Filtering**: Minimizes the mean square error using:

$$\mathbf{f} = \left(\mathbf{H}^T\mathbf{H} + \gamma\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{g}$$

  where γ is a regularization parameter to handle noise.
- **Regularized Least Squares**: Solves $\min_{\mathbf{f}} \|\mathbf{H}\mathbf{f} - \mathbf{g}\|_2^2 + \lambda\|\mathbf{R}\mathbf{f}\|_2^2$, where λ is a regularization parameter and $\mathbf{R}$ enforces smoothness or other constraints.
- **Iterative Methods**: Use algorithms like conjugate gradient or Richardson-Lucy to iteratively estimate $\mathbf{f}$, especially for large images or unknown PSFs.

Advantages (1 mark)

- **Computational Feasibility**: Discrete formulation enables the use of linear algebra and numerical methods, making restoration practical for digital images.
- **Flexibility**: Supports various restoration techniques, from simple inverse filtering to advanced regularized methods.
- **Handles Real-World Data**: Models degradation in discrete pixel grids, aligning with the structure of digital images.

5. **Limitations (1 mark)**

- **Ill-Posed Problem**: The degradation matrix $\mathbf{H}$ is often ill-conditioned or singular, making direct inversion unstable and sensitive to noise.
- **Computational Complexity**: Large images result in high-dimensional matrices, requiring significant computational resources.
- **PSF Knowledge**: Accurate restoration depends on knowing or estimating the degradation function h, which may be challenging in practice.

Applications and Color Image Considerations (1 mark)

- **Applications**:
  - **Medical Imaging**: Restores X-rays, MRIs, or CT scans degraded by blur or noise for better diagnosis.
  - **Astronomy**: Corrects telescope images blurred by atmospheric turbulence.
  - **Photography**: Recovers sharpness in images affected by motion blur or lens imperfections.
  - **Computer Vision**: Preprocesses degraded images for tasks like object detection or recognition.

✅ **Algebraic Approach to Image Restoration**

(For 5–7 mark explanation)

📌 **Definition:**

The **algebraic approach to image restoration** refers to modeling the image degradation and restoration process as a system of **linear equations**, which can be solved **algebraically** to recover the original image from a degraded one.

This method is often used when the degradation process is **well understood** and involves **linear operations** such as blurring.

🧮 Mathematical Model:

1. Matrix Representation of Image Degradation:

The degradation model:

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

can be written in **algebraic (matrix) form** as:

$$\mathbf{g} = \mathbf{H} \cdot \mathbf{f} + \boldsymbol{\eta}$$

Where:

- $\mathbf{f}$: vectorized original image (unknown)
- $\mathbf{g}$: vectorized degraded image (known)
- $\mathbf{H}$: degradation matrix (known)
- $\boldsymbol{\eta}$: noise vector

🔁 Goal of Restoration:

Estimate the original image vector $\hat{\mathbf{f}}$ by **solving** the system of equations:

$$\hat{\mathbf{f}} = \mathbf{H}^{-1} \cdot \mathbf{g}$$

However, this direct inverse only works **if $\mathbf{H}$ is invertible and noise is negligible.**

📌 **Advantages:**

- Can handle complex and customized degradation models
- Well-suited for **numerical methods** and **matrix computation software** (MATLAB, NumPy)

⚠️ **Limitations:**

- Computationally intensive for large images (due to matrix size)
- Inversion of large matrices can be unstable or impossible
- Sensitive to noise (unless regularized)

🔍 **Applications:**

- Restoration of **scientific or astronomical images**
- Situations where exact degradation kernel is known
- High-precision systems requiring accurate image recovery

Unconstrained Restoration

Unconstrained restoration is an image restoration technique that aims to recover an original image from its degraded version without imposing explicit constraints on the solution, such as smoothness or regularization penalties. It typically relies on directly inverting the degradation model, making it simple but sensitive to noise.

📌 **Properties:**

1. **Simple**: Direct solution, no constraints.
2. **Noise Sensitivity**: Amplifies noise, especially where $H(u, v)$ is small.
3. **Ill-Posed**: Small changes in $\mathbf{g}$ cause large errors in $\hat{\mathbf{f}}$.
4. **Fast**: Matrix inversion or division in frequency domain.

1. **Definition (1 mark)**
   Unconstrained restoration is a method to estimate the original image $f(x, y)$ from a degraded image $g(x, y)$, modeled as $g(x, y) = h(x, y) * f(x, y) + n(x, y)$, where $h(x, y)$ is the degradation function (e.g., point spread function, PSF) and $n(x, y)$ is additive noise. It solves for $f(x, y)$ without additional constraints, often using direct inversion techniques like inverse filtering.

Purpose (1 mark) The purpose of unconstrained restoration is to reverse the effects of degradation (e.g., blur, noise) to recover the original image with minimal assumptions. It aims to enhance image quality for visual clarity or analysis in applications like medical imaging or photography, assuming the degradation function is known.

4. **Advantages (1 mark)**

   - **Simplicity**: Direct inversion is straightforward when the degradation function $h(x, y)$ is known, requiring no additional parameters.
   - **Effective for Noiseless Cases**: Works well in ideal scenarios with minimal noise and a well-defined PSF.
   - **Preserves Details**: Can recover fine details if the degradation is accurately modeled and noise is negligible.

5. **Limitations (1 mark)**

   - **Noise Amplification**: Highly sensitive to noise, as small values in $H(u, v)$ or an ill-conditioned $\mathbf{H}$ amplify noise in the restored image.
   - **Ill-Posed Problem**: The inverse problem is often unstable, especially for real-world images with unknown or complex PSFs.
   - **Limited Practicality**: Rarely effective in noisy conditions, requiring constrained or regularized methods for robust restoration.

.

Applications and Color Image Considerations (1 mark)

- **Applications**:
  - **Medical Imaging**: Used in idealized cases (e.g., known PSF with low noise) to restore X-rays or MRIs for better diagnosis.
  - **Astronomy**: Restores telescope images with known atmospheric blur and minimal noise.
  - **Photography**: Recovers sharpness in images with known blur (e.g., lens defocus) in controlled settings.
  - **Computer Vision**: Preprocesses images for feature extraction in low-noise scenarios.
- **Color Images**: For color images (e.g., RGB), unconstrained restoration is applied independently to each color channel (Red, Green, Blue) using channel-specific PSFs. In HSV/HSI spaces, restoration focuses on the intensity channel to avoid color distortions, but noise amplification remains a challenge.

📌 **Definition:**

Unconstrained restoration techniques **only minimize the error** between the degraded image and the estimated restored image. They do **not impose any restrictions** (constraints) on the solution.

📐 **Mathematical Form:**

$$\min_f \|Hf - g\|^2$$

Where:

- f: restored image
- g: degraded image
- H: degradation operator (e.g., blur matrix)

This means: find f that makes Hf as close as possible to g.

✅ **Example Methods:**

- **Inverse Filtering**
- **Pseudo-inverse solution**
- **Unregularized least squares**

⚠️ **Limitations:**

- Highly sensitive to noise
- May produce unstable or physically invalid results

Constrained Restoration

Constrained restoration is an image restoration technique that recovers an original image from its degraded version by incorporating constraints or regularization to stabilize the solution and mitigate issues like noise amplification. Unlike unconstrained restoration, it imposes additional conditions (e.g., smoothness or sparsity) to improve robustness.

Purpose (1 mark) The purpose of constrained restoration is to recover a high-quality approximation of the original image by reversing degradation effects (e.g., blur, noise) while mitigating the instability and noise amplification common in unconstrained methods. It aims to produce robust, visually clear images for applications like medical imaging or computer vision.

Definition (1 mark)

Constrained restoration is a method to estimate the original image $f(x, y)$ from a degraded image $g(x, y)$, modeled as $g(x, y) = h(x, y) * f(x, y) + n(x, y)$, where $h(x, y)$ is the point spread function (PSF) and $n(x, y)$ is noise. It solves the restoration problem by incorporating constraints or regularization terms to control noise and stabilize the solution, often using optimization techniques.

🔸 2. Constrained Restoration

📌 **Definition:**

Constrained restoration techniques **minimize the error while satisfying additional conditions** (constraints) such as smoothness, edge preservation, or energy minimization.

📐 **Mathematical Form:**

$$\min_f \|Hf - g\|^2 + \lambda \cdot R(f)$$

Where:

- $R(f)$: Regularization function (e.g., smoothness, total variation)
- λ: Regularization parameter (controls strength of constraint)

This allows balancing between data fidelity and desired image properties.

✅ **Example Methods:**

- **Tikhonov Regularization**
- **Constrained Least Squares Filtering**
- **Total Variation Minimization**
- **Wiener Filtering** (indirectly uses statistical constraints)

✅ **Advantages:**

- More robust to noise
- Produces more natural and visually pleasing images
- Can enforce physical or perceptual constraints

1. **Advantages (1 mark)**
    - **Noise Robustness**: Regularization mitigates noise amplification, producing clearer images than unconstrained methods.
    - **Stability**: Constraints stabilize the ill-posed inverse problem, especially when the PSF is ill-conditioned.
    - **Flexibility**: Various constraints (e.g., smoothness, sparsity) can be tailored to specific degradation types or applications.
2. **Limitations (1 mark)**
    - **Parameter Tuning**: Requires careful selection of the regularization parameter λ, which can be challenging and application-dependent.
    - **Computational Complexity**: Optimization-based methods are computationally intensive, especially for large images or iterative techniques.
    - **Loss of Detail**: Over-regularization may smooth out fine details or edges, reducing restoration accuracy.

3. **Applications and Color Image Considerations (1 mark)**
    - **Applications**:
        - **Medical Imaging**: Restores X-rays, MRIs, or CT scans degraded by blur or noise for accurate diagnosis.
        - **Astronomy**: Corrects telescope images blurred by atmospheric turbulence, preserving fine details.
        - **Photography**: Recovers sharpness in images affected by motion blur or lens imperfections.
        - **Computer Vision**: Preprocesses degraded images for tasks like object recognition or tracking.
    - **Color Images**: For color images (e.g., RGB), constrained restoration is applied independently to each color channel with channel-specific PSFs and regularization. In HSV/HSI spaces, restoration focuses on the intensity channel to avoid color distortions while enhancing clarity.

Aspect	Unconstrained Restoration	Constrained Restoration
Definition	Minimizes error without any additional conditions or restrictions	Minimizes error while imposing extra constraints (like smoothness or regularity)
Mathematical Form	min⁡f‖Hf−g‖2\min_f | Hf - g |^2minf‖Hf−g‖2	min⁡f‖Hf−g‖2+λR(f)\min_f | Hf - g |^2 + \lambda R(f)minf‖Hf−g‖2+λR(f) (where R(f)R(f)R(f) is the constraint term)
Noise Handling	Poor; sensitive to noise and often unstable	Better; more robust to noise due to constraints
Stability	Less stable; solutions may be unrealistic	More stable and physically meaningful solutions
Complexity	Simpler and faster computations	More complex due to additional constraints and parameters
Output Quality	May contain artifacts or amplified noise	Produces smoother, cleaner, and more natural images
Examples	Inverse filtering, direct least squares	Tikhonov regularization, Wiener filtering, constrained least squares

Mathematical Form	$\min_f \|Hf - g\|^2$	$\min_f \|Hf - g\|^2 + \lambda R(f)$ (where $R(f)$ is the constraint term)

🔹 1. Image Degradation Model

Image degradation occurs when an original image is blurred or corrupted due to physical processes during acquisition, transmission, or storage.

The **general degradation model** is:

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

Where:

- $g(x, y)$: degraded image (observed)
- $f(x, y)$: original image (unknown)
- $h(x, y)$: degradation function or point spread function (PSF)
- $\eta(x, y)$: additive noise
- $*$: convolution operation

🔹 2. Examples of Added Noise

Noise can be introduced by sensors, transmission channels, or digitization.

Common noise types:

- **Gaussian noise:** Continuous, normally distributed noise (most common)
- **Salt-and-pepper noise:** Random occurrences of black and white pixels
- **Poisson noise:** Signal-dependent noise (e.g., photon noise in low light)
- **Speckle noise:** Multiplicative noise seen in radar and ultrasound images

---

🔹 3. Ways to Estimate the Degradation Function h(x,y)

a) Observation or Calibration:

- Capture a known test image (like a delta function or impulse) and observe how it spreads.
- The result gives you h(x,y)

b) Experimental Setup Knowledge:

- Use mathematical models based on known physical conditions (e.g., motion blur, defocus).

c) Blind Estimation (when nothing is known):

- Use statistical or optimization methods to jointly estimate f and h from the degraded image g.

2a) Discuss about Threshold Coding Implementation

✅ **Threshold Coding is a lossy compression technique used to reduce image data size by eliminating insignificant coefficients, especially after a transform like DCT or wavelet.**

🔹 Steps in Threshold Coding Implementation:

1. **Transform the Image:**
   - Apply a transform (e.g., DCT, Wavelet) to concentrate energy into fewer coefficients.
2. **Choose a Threshold (T):**
   - Determine a cutoff value. Coefficients below T are considered unimportant.
3. **Apply Thresholding:**
   - Set all values below threshold to **zero**.
   - Keep only coefficients with values ≥ T.
4. **Encode Non-Zero Values:**
   - Use entropy coding (e.g., Huffman or Run-Length) to compress the remaining data.

Advantages:

- Reduces data size efficiently.
- Keeps visually significant details.

🔹 **Disadvantages:**

- Loss of information → may reduce quality.
- Choice of threshold is critical.

🔷 2b) Comparative Study: Least Square vs Constrained Least Square Restoration

Aspect	Least Square Restoration (LSR)	Constrained Least Square Restoration (CLSR)
Objective	Minimize $\|h * f - g\|^2$	Minimize $\|h * f - g\|^2 + \gamma \|\nabla^2 f\|^2$
Noise Handling	Ignores noise characteristics	Includes noise smoothing via regularization
Smoothness Control	No constraint on solution smoothness	Smoothness enforced via Laplacian operator
Regularization	Not included	Uses regularization parameter γ
Output	May amplify noise	Balances restoration and noise suppression
Complexity	Simpler computation	Slightly more complex due to extra term
When to Use	If noise is negligible	If noise is significant or smooth output is needed

✅ **Conclusion:**

- **LSR** is simpler but sensitive to noise.
- **CLSR** is more robust as it balances accuracy and smoothness.

Constrained Least Square Restoration (CLSR)

🔹 **Definition:**

Constrained Least Square Restoration is a **frequency-domain image restoration technique** that reconstructs a degraded image by minimizing a **cost function**. It balances two goals:

1. Keeping the restored image close to the observed degraded image.
2. Ensuring the restored image is **smooth** (i.e., not noisy).

> **Constrained Least Square (CLS) Restoration** is a specific constrained approach that minimizes the error $\|\mathbf{g} - \mathbf{Hf}\|^2$ while constraining the solution's smoothness using a Laplacian operator. It's a practical method to deblur images while controlling noise.

Constrained least square restoration is a technique in image processing that aims to enhance an image by minimizing the difference between the restored image and the observed image while also enforcing constraints on the restored image, such as smoothness. It's particularly useful when dealing with blurred or noisy images, and it finds a balance between fidelity to the observed data and the desired properties of the restored image.

Here's a more detailed explanation:

- **Minimizing the Difference:**

The core idea is to find a restored image that minimizes the squared difference between the restored image and the observed (degraded) image, thus ensuring fidelity to the observed data.

- **Enforcing Constraints:**

Along with minimizing the difference, constraints are imposed on the restored image to achieve specific desired properties. For instance, a constraint might be to maximize smoothness, meaning the restored image should be relatively free from sharp edges and high-frequency noise.

- **Application:**

Constrained least square restoration is often used for deblurring images, reducing noise, and enhancing image quality in various applications like medical imaging, satellite imagery, and astronomy.

📌 **Properties**:

1. **Noise Control**: Laplacian constraint smooths noise.
2. **Tunable**: λ or gamma balances deblurring and smoothness.
3. **Practical**: Works with known PSFs, common in software.
4. **Computationally Intensive**: Requires matrix operations or FFT.

🔹 Degradation Model:

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

Where:

- $g(x, y)$: observed degraded image
- $f(x, y)$: original image (to estimate)
- $h(x, y)$: degradation function (blur/PSF)
- $\eta(x, y)$: additive noise
- $*$: convolution

🔹 Objective:

Minimize the **cost function**:

$$J(f) = \|h * f - g\|^2 + \gamma \cdot \|\nabla^2 f\|^2$$

- First term: **Data fidelity** — ensures similarity to observed image
- Second term: **Regularization** — enforces smoothness using Laplacian
- γ: regularization parameter (controls the trade-off)

Solution in Frequency Domain:

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

here:

$F(u, v)$: Restored image in frequency domain

$G(u, v)$: Fourier transform of degraded image

$H(u, v)$: Fourier transform of PSF

$P(u, v)$: Fourier transform of Laplacian operator

H^*: Complex conjugate of H

Key Points:

CLSR uses **prior knowledge** about image smoothness (through Laplacian).

Works better than inverse filtering or least squares when noise is present.

The parameter γ is crucial — small γ risks noise, large γ over-smooths.

Advantages (1 mark)

- **Noise Robustness**: Regularization reduces noise amplification compared to unconstrained methods like inverse filtering.
- **Stability**: Stabilizes the ill-posed inverse problem by incorporating constraints, improving restoration quality.
- **Customizable**: The constraint operator $\mathbf{C}$ and parameter λ can be tailored to specific degradation types or image characteristics.

Limitations (1 mark)

- **Parameter Tuning**: Requires careful selection of λ, which can be challenging and affects restoration quality.
- **Loss of Detail**: Over-regularization may smooth out fine details or edges, reducing accuracy.
- **Computational Complexity**: Frequency-domain processing and optimization are computationally intensive, especially for large images.

Applications and Color Image Considerations (1 mark)

- **Applications**:
    - **Medical Imaging**: Restores X-rays, MRIs, or CT scans degraded by blur or noise for accurate diagnosis.
    - **Astronomy**: Corrects telescope images blurred by atmospheric turbulence, enhancing fine details.
    - **Photography**: Recovers sharpness in images affected by motion blur or lens imperfections.
    - **Computer Vision**: Preprocesses degraded images for tasks like object recognition or tracking.
- **Color Images**: For color images (e.g., RGB), constrained least square restoration is applied independently to each color channel with channel-specific PSFs and regularization parameters. In HSV/HSI spaces, restoration focuses on the intensity channel to avoid color distortions while enhancing clarity.

Great! Here's a **simple and easy-to-understand example** of **Homomorphic Filtering** for image restoration — similar in style to what you uploaded:

✅ **Example: Restoring an Image with Uneven Lighting (Shadow)**

🔹 **Problem:**

You have a photo of a **document or face** with a **shadow** on one side — making the details hard to see.

🔹 **Step-by-Step Process:**

1. **Input:**
   - Load a grayscale image with **non-uniform lighting** (e.g., shadow on one side).
2. **Log Transform:**
   - Apply:
   - $$\log \text{image} = \ln(1 + \text{image})$$
   - This separates **illumination** and **reflectance**.
3. **Fourier Transform:**
   - Convert the log image to the **frequency domain**:
     - **Low frequencies** = shadows (illumination)
     - **High frequencies** = text, edges (details)
4. **Filtering:**
   - Apply a **high-pass filter** to:
     - Reduce low-frequency components (shadows)
     - Boost high-frequency components (details)
   - Example filter behavior:
     - Boost high frequencies by 2×
     - Reduce low frequencies by 0.5×
5. **Inverse Fourier Transform:**
   - Convert the image back to spatial domain.
6. **Exponential:**
   - Apply exponentiation:

- $$\text{Restored image} = e^{\text{filtered log image}}$$
- Normalize values.

7. **Output:**
   - The result is an image with **reduced shadow** and **clearer details**.

🔸 **Result:**

📄 **Before:** Text is hidden in the shadow.
🖼️ **After:** Shadow removed, and text appears clear and sharp.

🔷 Restoration by Geometric (Spatial) Transformation

✅ Steps:

1. **Identify Deformation**
   - Recognize the distortion (rotation, translation, scaling, warping).
2. **Define Spatial Mapping Function**
   - Set up a transformation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

3. **Apply Inverse Mapping**
   - For each output pixel (x', y'), compute the corresponding input location (x, y).
4. **Use Interpolation**
   - Since (x, y) might be non-integer:
     - Use **nearest neighbor**, **bilinear**, or **bicubic** interpolation to estimate pixel value.
5. **Reconstruct Output Image**
   - Generate the geometrically corrected image.

🔷 Restoration by Homomorphic Filtering

✅ Steps:

1. **Input Image**
   - Load the degraded image (typically with uneven illumination or shadows).
2. **Log Transformation**
   - Convert image from multiplicative to additive model:

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

3. **Fourier Transform**
   - Apply 2D FFT to move into the frequency domain.
4. **Apply High-Pass Filter**
   - Suppress low-frequency components (illumination)
   - Boost high-frequency components (reflectance/details)
5. **Inverse Fourier Transform**
   - Transform back to spatial domain.
6. **Exponential Transformation**
   - Apply:

$$\text{Output} = \exp(\text{Filtered image})$$

7. **Normalize and Display**
   - Adjust intensity levels to 0–255 range.

Gray level interpolation is a technique used in image processing to estimate the intensity (or color) of a pixel at a **non-integer position** in an image. It's needed when transforming images (e.g., rotating, scaling, or warping, as in your image restoration question) because the new pixel positions may not align with the original integer grid of pixels. Interpolation "guesses" the intensity by blending the values of nearby pixels, ensuring the transformed image looks smooth and natural. Think of it like painting a new picture by mixing colors from nearby spots on the original canvas to fill in gaps.

📌 **Why Use Gray Level Interpolation?**

- To **smooth transformed images** (e.g., avoid jagged edges after rotation).
- To **preserve image quality** during scaling, rotation, or warping.
- To **estimate missing pixel values** in geometric transformations (your restoration question).

- To **support applications** like medical imaging, photography, and computer vision in 2025.

🔸 How Gray Level Interpolation Works

📌 **Process in Simple Words:**

1. **Apply Transformation**: Perform a geometric transformation (e.g., rotation, scaling) that maps original pixel coordinates (x, y) to new coordinates (x', y'), which may be non-integer (e.g., $(1.3, 2.7)$).
2. **Locate Neighbors**: Identify the closest original pixels around the non-integer position (x', y').
3. **Blend Intensities**: Combine the intensities of these neighboring pixels using a weighted average, where weights depend on how close each neighbor is to (x', y').
4. **Assign Value**: Set the interpolated intensity as the new pixel's value in the transformed image.
5. **Repeat**: Apply to all pixels in the new image grid.

Visual Analogy: Imagine you're coloring a new grid but the colors don't line up with the original grid's squares. You mix colors from the nearest squares, giving more weight to closer ones, to paint each new square smoothly.

Properties of Gray Level Interpolation

1. **Smoothness**: Bilinear and bicubic produce smoother results than nearest neighbor, reducing artifacts.
2. **Accuracy vs. Speed**: Nearest neighbor is least accurate but fastest; bicubic is most accurate but slowest.
3. **Local Dependency**: Uses nearby pixels (1, 4, or 16), unlike global methods like DFT (your question).
4. **Preserves Intensity Range**: Output stays within the input range (e.g., 0–255), assuming proper clamping.
5. **Essential for Transformations**: Without interpolation, transformed images look blocky or distorted.
6. **Trade-off**: Higher quality increases computation, impacting real-time use in 2025.
7. **Color Images**: Applied separately to R, G, B channels for RGB images.

✅ Applications (5 Points with Detail)

1. **Medical Imaging**:
    - **Use**: Aligns and interpolates MRI/CT scans after geometric correction (your restoration question).
    - **Example**: A rotated MRI is interpolated with bicubic to ensure smooth tissue boundaries for diagnosis in 2025.
2. **Photography**:

- **Use**: Smooths images after rotation, scaling, or lens distortion correction in apps like Photoshop.
- **Example**: A 2025 smartphone rotates a photo 30°, using bilinear interpolation for a natural look.

3. **Satellite Imaging**:
   - **Use**: Corrects terrain warping in aerial images, interpolating to maintain detail.
   - **Example**: A warped Google Maps image is aligned with bilinear interpolation to show clear roads.
4. **Computer Vision**:
   - **Use**: Ensures smooth images for AI tasks like object detection after alignment or resizing.
   - **Example**: A self-driving car resizes camera images, using bicubic interpolation for accurate lane detection.
5. **Animation/Graphics**:
   - **Use**: Interpolates textures during 3D rendering or image warping in video games.
   - **Example**: A 2025 game rotates a character's texture, using bicubic interpolation for realistic visuals.

✅ **Significance (5 Points with Detail)**

1. **Improves Image Quality**: Prevents blocky or jagged artifacts in transformed images, critical for visual clarity.
2. **Enables Geometric Transformations**: Essential for rotation, scaling, or warping (your restoration question), making images usable.
3. **Supports Real-Time Processing**: Bilinear interpolation is fast enough for 2025 devices like smartphones and drones.
4. **Versatile**: Works for grayscale, color, medical, or consumer images, fitting many domains.
5. **Prepares for Analysis**: Smooth, aligned images improve AI, OCR, or compression (DCT, your question).

Image Segmentation

It is a stage of transition from image processing methods whose inputs and outputs are images, to methods in which the inputs are images but the outputs are attributes extracted from those images.

Segmentation refers to the process of partitioning an image into multiple regions.

Image segmentation is typically used to locate objects and boundaries in images.

Methods of finding Discontinuity and Similarity

* Discontinuity
  - Isolated point
  - Line detection
  - Edge detection

* Similarity
  - Thresholding
  - Region growing
  - Region split
  - Region merge

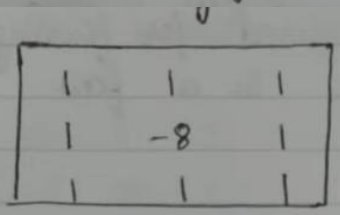
* Discontinuity

1) Detection of isolated points

→ It is done using sharpening filters. We use second order derivative using Laplacian.

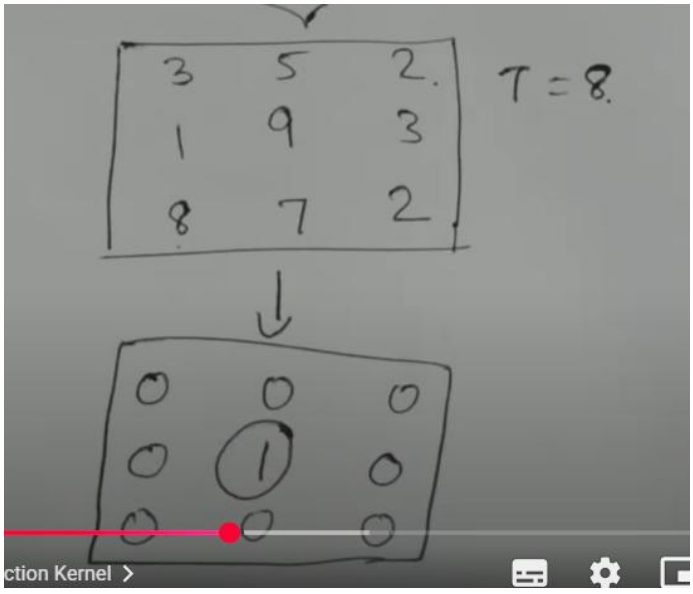
→ A point has been detected at a location (x, y) on which the kernel is centered if the absolute value of the response of the filter at that point exceeds a specified threshold.

Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image.



Kernel

$$g(x,y) = \begin{cases} 1 & \text{if } |Z(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$



2) Line detection

→ We use second order derivatives which result in a stronger filter response and produce thinner lines than first derivatives.

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

2	-1	-1
-1	2	-1
-1	-1	2

+45°

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

-1	-1	2
-1	2	-1
2	-1	-1

-45°

3) Edge detection

→ It is an approach used frequently for segmenting images based on abrupt (local) changes in intensity.

→ First order derivatives such as Roberts-cross, Prewitt and Sobel operators are preferred for thicker lines.

→ Second order derivatives (laplacian) are used for detecting thinner lines

→ Kirsch compass kernels are used for finding the maximum edge strength in a few predetermined directions.

Edge Detection Kernels

1) Roberts cross-gradient operators

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

2) Prewitt operators

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

9:33 / 11:51 • Edge Detection Kernel

3) Sobel operators (have better noise suppression (smoothing) characteristics)

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

4) Prewitt masks for diagonal edges

$$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

5) Sobel masks for diagonal edges

$$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

$+45^\circ$ $\qquad$ -45°

6) Kirsch compass kernels

-3	-3	5
-3	0	5
-3	-3	5

N

-3	5	5
-3	0	5
-3	-3	-3

N.W

5	5	5
-3	0	-3
-3	-3	-3

W

5	5	-3
5	0	-3
-3	-3	-3

S

5	-3	-3
5	0	-3
5	-3	-3

SE

-3	-3	-3
-3	0	-3
5	5	5

E

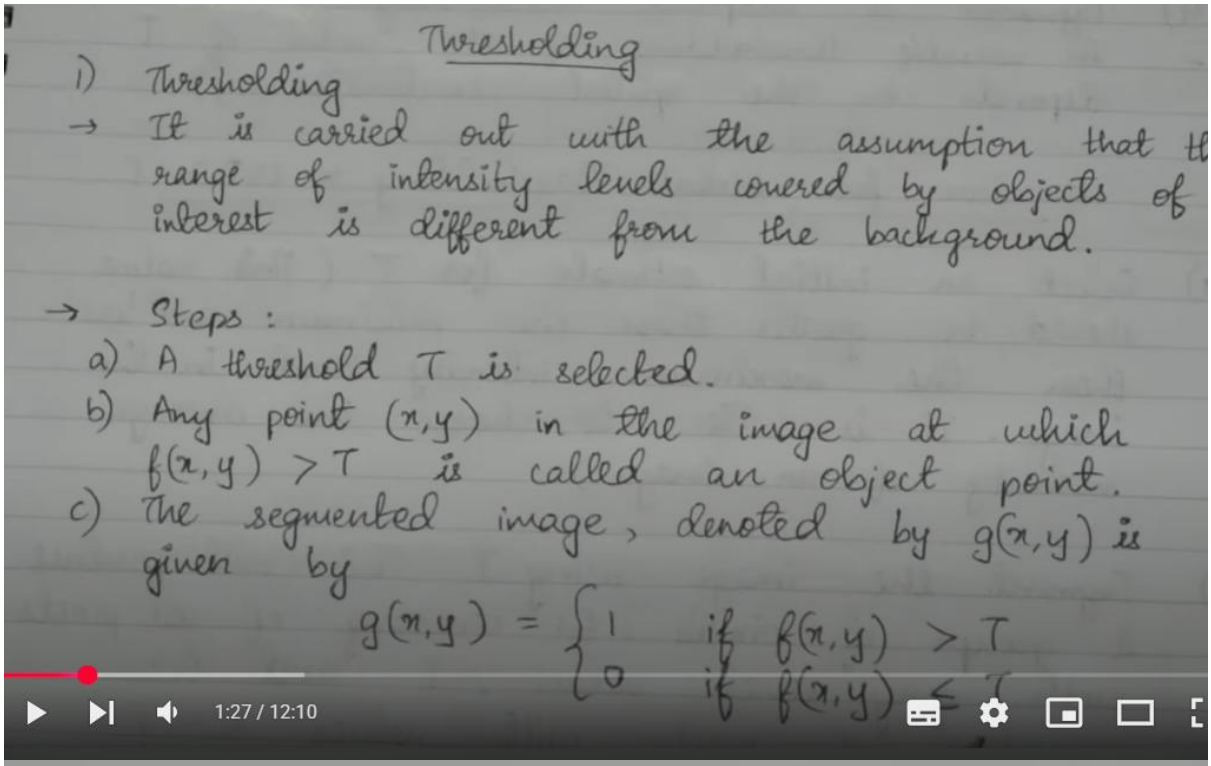
-3	-3	-3
-3	0	5
-3	5	5

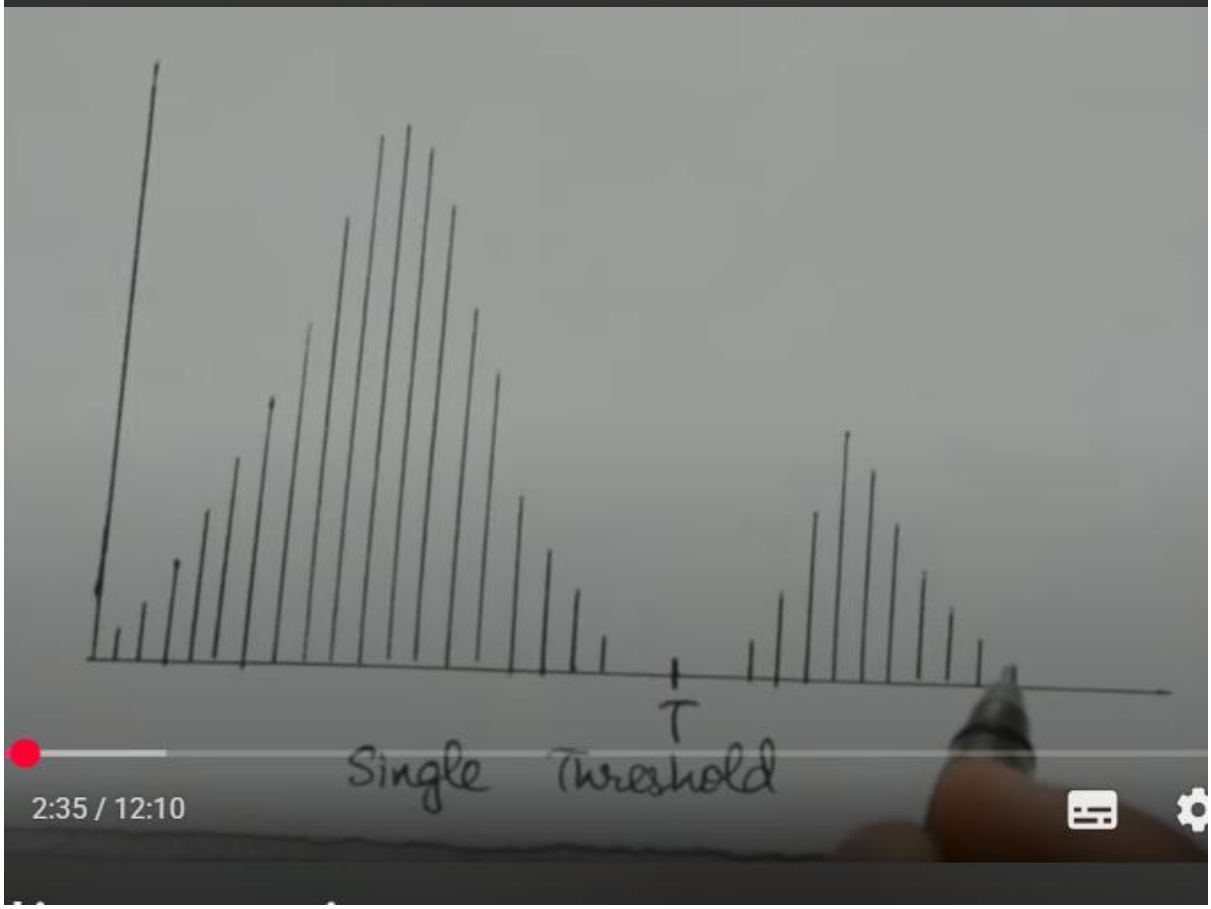
NE

7) Second order kernels (The Laplacian)

0	-1	0
-1	4	-1
0	-1	0

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


Thresholding
1) Thresholding
→ It is carried out with the assumption that th
range of intensity levels covered by objects of
interest is different from the background.
→ Steps :
a) A threshold T is selected.
b) Any point (x,y) in the image at which
f(x,y) > T is called an object point.
c) The segmented image, denoted by g(x,y) is
given by
g(x,y) = { 1 if f(x,y) > T
0 if f(x,y)
1:27 / 12:10


T
Single Threshold
2:35 / 12:10

Types of Thresholding

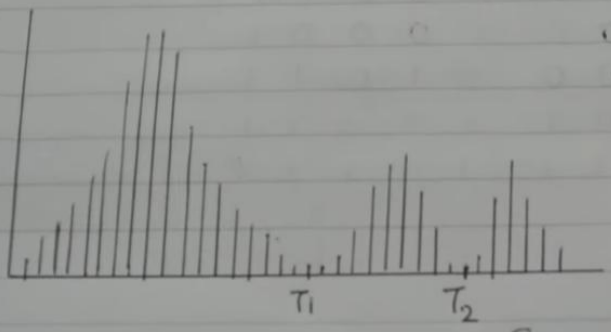
1) Global thresholding
- T is a constant.

2) Variable thresholding
- T changes over an image.

3) Local or regional thresholding
- In variable thresholding, if the value of T at any point (x,y) in an image depends on the properties of a neighborhood of (x,y).

4) Dynamic or adaptive thresholding
- In variable thresholding, if the value of T depends on the spatial coordinates (x,y).

A Histogram with three dominant modes (two types of light objects on a dark background)



$$g(x,y) = \begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \leq T_2 \\ c & \text{if } f(x,y) \leq T_1 \end{cases}$$

Procedure for Global Thresholding to obtain T

1) Select an initial estimate for T. (This value should be greater than the minimum and less than the maximum intensity level in the image. It is better to choose the average intensity of an image.)

2) Segment the image using T. This will produce 2 groups of pixels : G1 consisting of all pixels with gray level values > T and G2 consisting of pixels with values <= T.

3) Compute the average gray level values $\mu 1$ and $\mu 2$ for the pixels in regions G1 and G2.

4) Compute a new threshold value :

$$T = \frac{1}{2}(\mu 1 + \mu 2)$$

5) Repeat step 2 through 4 until the difference in T in successive iterations is smaller than a predefined parameter T_0.

5	3	9
2	1	7
8	4	2

Let $T = \frac{5+3+9+2+1+7+8+4+2}{9}$

$T = \frac{41}{9} = 4.55 \approx 5.$

Segmenting the image using T, we would get

$G1 = \{9, 7, 8\}$ $\quad G2 = \{5, 3, 2, 1, 4, 2\}$

$\mu_1 = \frac{9+7+8}{3}$ $\quad \mu_2 = \frac{5+3+2+1+4+2}{6}$

$= \frac{24}{3} = 8.$ $\quad = \frac{17}{6} = 2 \approx 3.$

$T = \frac{1}{2}(8+3)$

Segmenting the image using T, we would get

$G1 = \{9, 7, 8\}$ $\quad G2 = \{5, 3, 2, 1, 4, 2\}$

$\mu_1 = \frac{9+7+8}{3}$ $\quad \mu_2 = \frac{5+3+2+1+4+2}{6}$

$= \frac{24}{3} = 8.$ $\quad = \frac{17}{6} = 2.83 \approx 3.$

$T = \frac{1}{2}(8+3)$

$= \frac{1}{2}(11) = 5.5 \approx 5.$

Procedure for Adaptive Thresholding

1) Convolve the image with a suitable statistical operator, ie. the mean or median.
2) Subtract the original from the convolved image.
3) Threshold the difference image with C.
4) Invert the threshold image.

Region-based Segmentation

→ Edges and thresholds sometimes do not give good results for segmentation.

→ Region-based segmentation is based on the connectivity of similar pixels in a region.

→ There are two main approaches to region-based segmentation : region-growing and region-splitting.

Procedure for Region-Growing

1) Find all connected components in $S(x,y)$ and reduce each connected component to one pixel. Label all such pixels found as 1. All other pixels in S are labeled 0.

2) Form an image f_Q such that, at each point (x,y), $f_Q(x,y) = 1$ if the input image satisfies a given predicate, Q, at those coordinates, and $f_Q(x,y) = 0$ otherwise.

3) Let g be an image formed by appending to each seed point in S all the 1-valued points in f_Q that are 8-connected to that seed point.

4) Label each connected component in g with a different region label (Ex. integers or letters). This is the segmented image obtained by region growing.

Procedure for Region splitting and Merging

1) If a region R is inhomogeneous (P(R) = False), then R is split into four sub-regions.

2) If 2 adjacent regions Ri, Rj are homogeneous (P(Ri ∪ Rj) = True), then they are merged.

3) The algorithm stops when no further splitting or merging is possible.

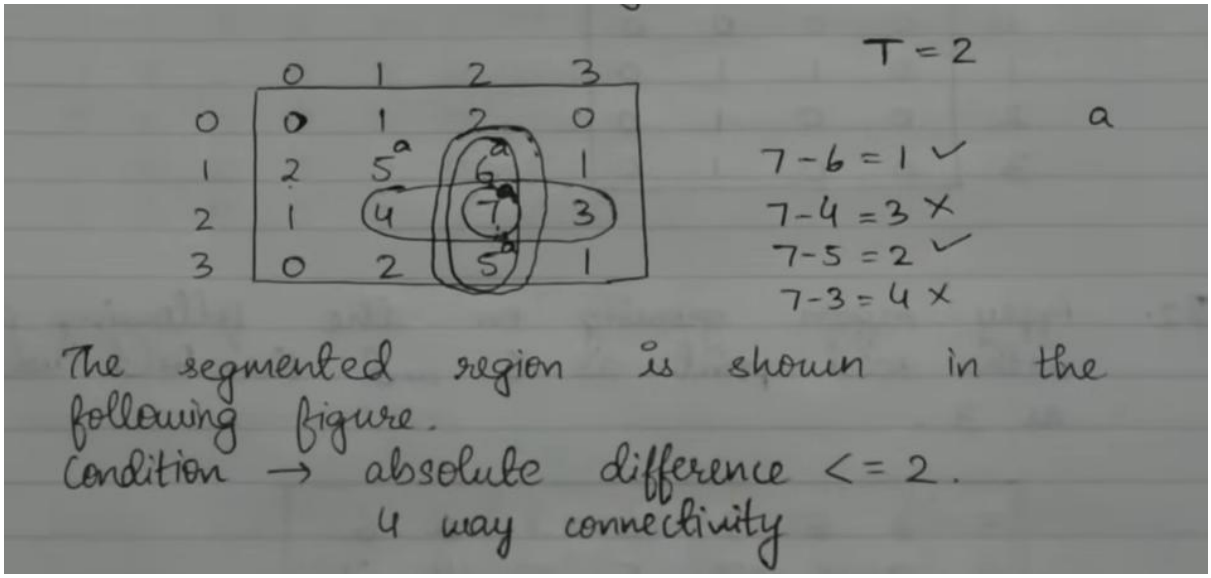
Note: Condition for region growing:
abs(seed value - pixel value) ≤ Threshold

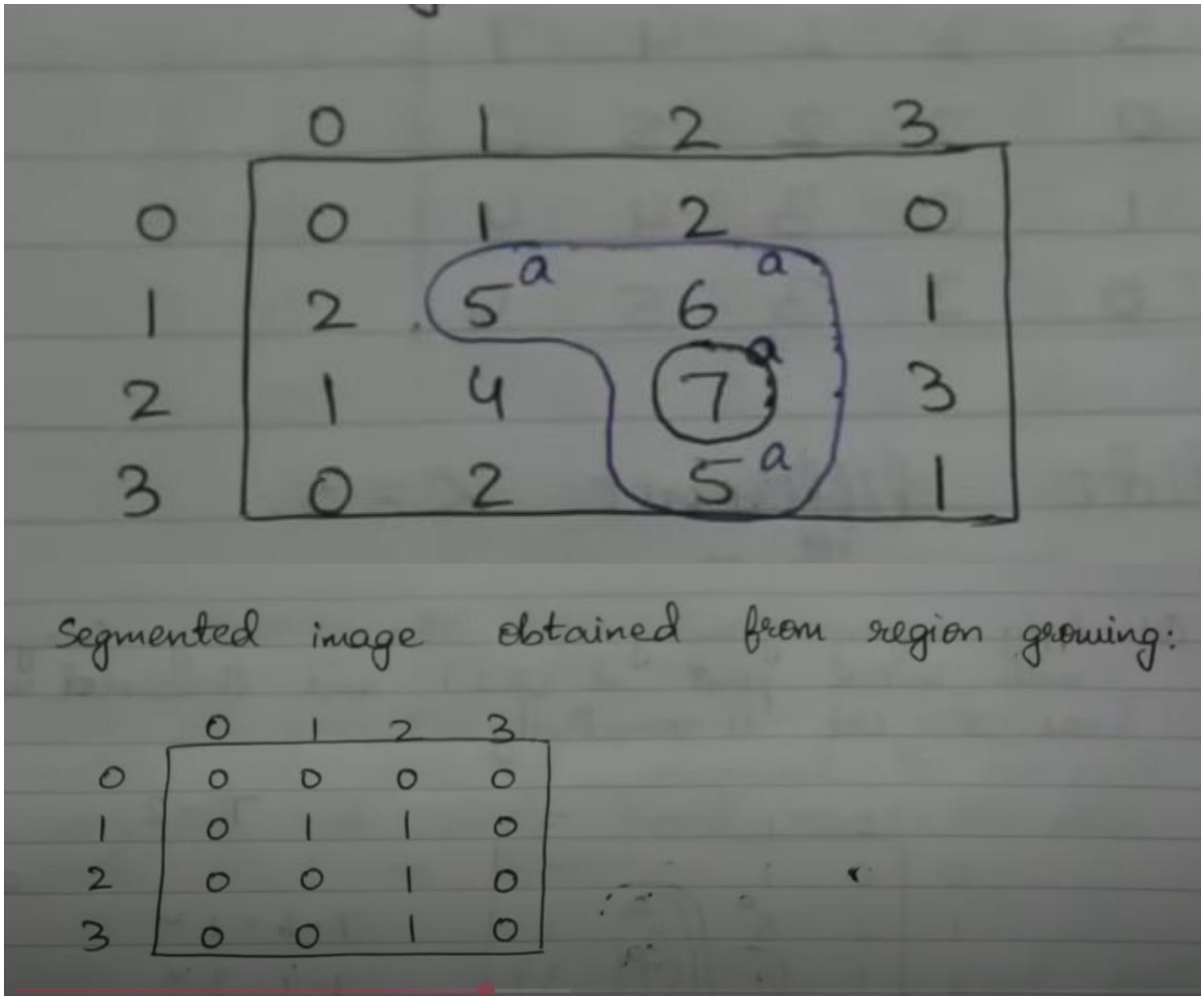
Q1. Apply region growing on the following image with initial point at (2,2) and threshold value as 2. Use 4 connectivity.

T = 2

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

Ans. The segmented region is shown in the following figure.
Condition → absolute difference <= 2.
4 way connectivity





- **Initial seed point**: (2,2) → Value = 7
- **Threshold T = 2**
- **4-connectivity** means: Up, Down, Left, Right neighbors

- **Inclusion condition**:
  $|I(x,y) - I(seed)| \leq 2$

✅ **Step-by-Step Region Growing:**

Start from (2,2) = 7
We'll check its **4-connected neighbors**:

- **(1,2)** = 6 → |6 - 7| = 1 ✅
- **(2,1)** = 4 → |4 - 7| = 3 ❌
- **(2,3)** = 3 → |3 - 7| = 4 ❌
- **(3,2)** = 5 → |5 - 7| = 2 ✅

Now add (1,2) and (3,2) to the region
From here, we check their 4-neighbors **recursively**:

From (1,2) = 6:

- (0,2) = 2 → |2 - 7| = 5 ❌
- (1,1) = 5 → |5 - 7| = 2 ✅
- (1,3) = 1 → |1 - 7| = 6 ❌
- (2,2) already visited ✅

Add (1,1)

From (3,2) = 5:

- (2,2) already visited
- (3,1) = 2 → |2 - 7| = 5 ❌
- (3,3) = 1 → |1 - 7| = 6 ❌

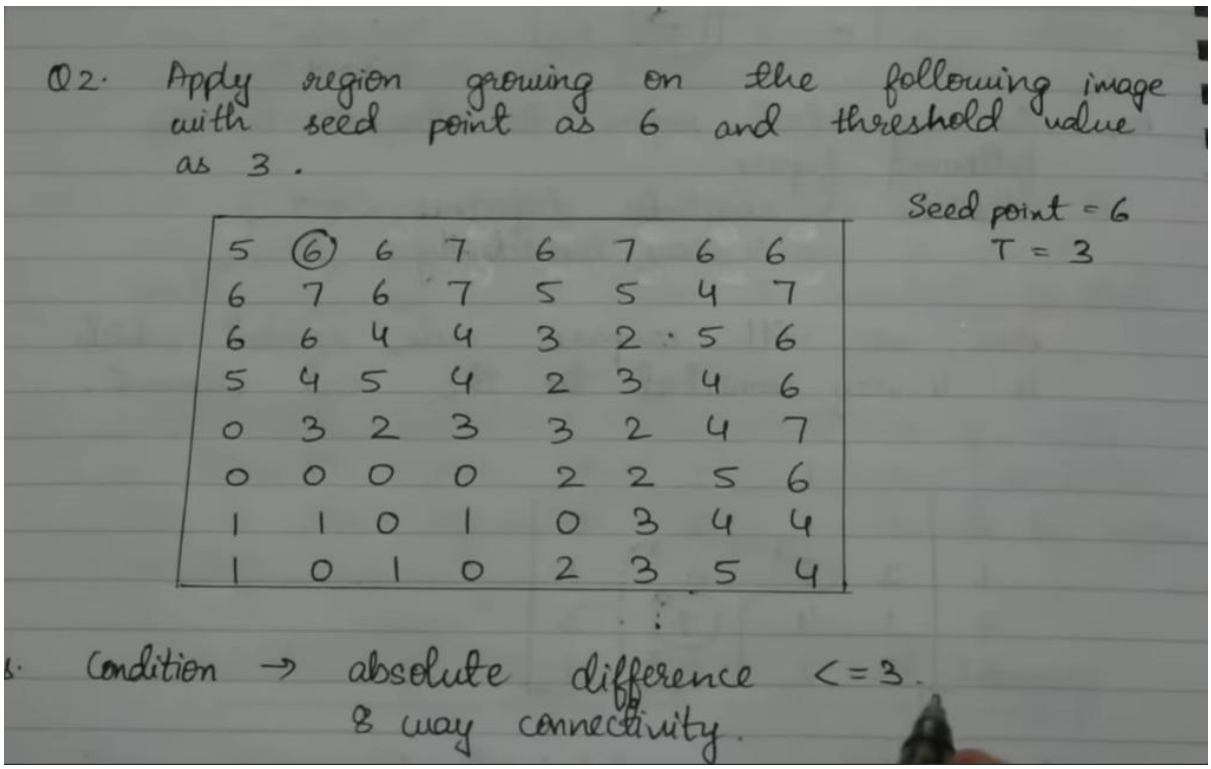
No new points

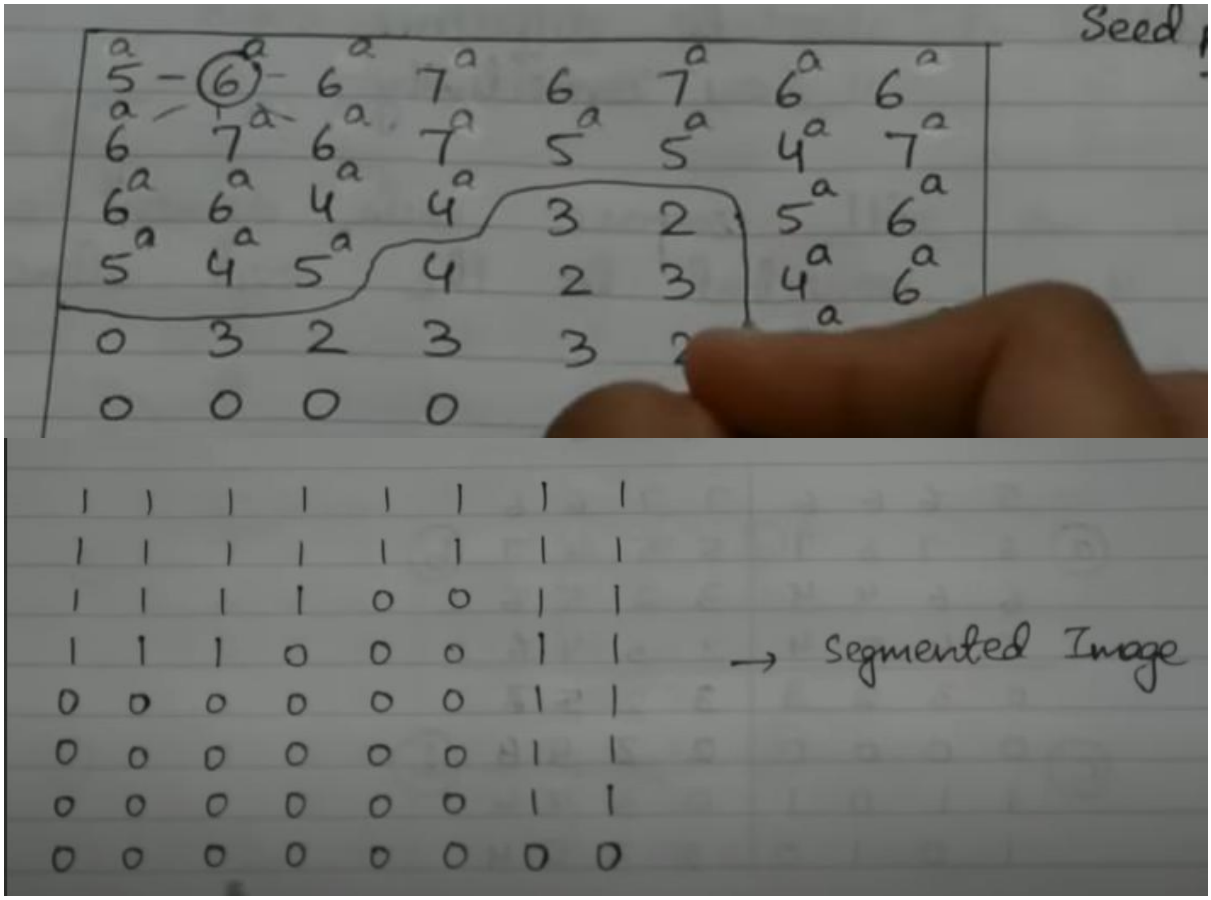
From (1,1) = 5:

- (0,1) = 1 → |1 - 7| = 6 ❌
- (1,0) = 2 → |2 - 7| = 5 ❌
- (1,2) already in region ✅
- (2,1) = 4 → |4 - 7| = 3 ❌

✅ **Final Region Points:**

- **(2,2)** = 7 (seed)
- **(1,2)** = 6
- **(1,1)** = 5
- **(3,2)** = 5





In region growing algorithms, the seed point remains fixed throughout the process of growing the region. The seed point is typically chosen as the starting point for the region growing process, and it does not change as the region expands. Once the seed point is specified, the algorithm iteratively examines neighboring pixels and decides whether to include them in the region based on certain criteria (e.g., similarity to the seed point). However, the seed point itself remains unchanged, serving as a reference point for the region growing process. Changing the seed point during the region growing process could lead to inconsistent segmentation results and may not adhere to the intended segmentation criteria. Therefore, it's common practice to keep the seed point fixed throughout the region growing algorithm.

Q3. Apply splitting and merging on the following image with threshold value equal to 3.

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

T = 3

ns. Condition : absolute difference <= 3.
Max value = 7
Min value = 0
|7 - 0| = 7 which is greater than 3.
Therefore we will split the region into 4 sub regions.

Splitting

	5	6	6	6	7	7	6	6
(a)	6	7	6	7	5	5	4	7
	6	6	4	4	3	2	5	6
	5	4	5	4	2	3	4	6
	0	3	2	3	3	2	5	7
(c)	0	0	0	0	2	2	5	6
	1	1	0	1	0	3	4	4
	1	0	1	0	2	3	5	4

(b) (d)

|7 - 2| = 5

|3 - 0| = 3

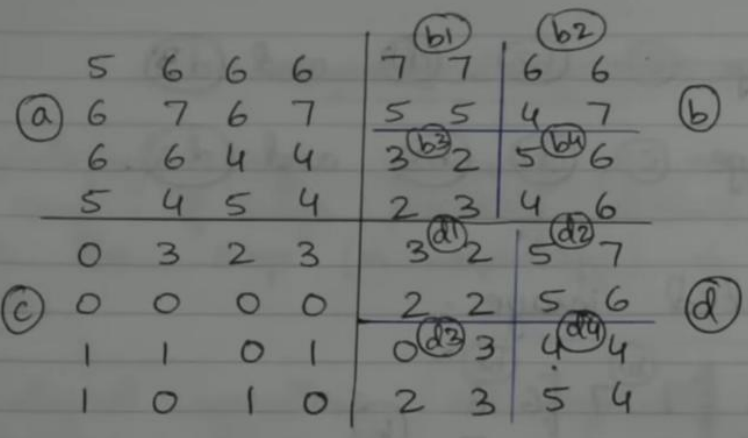
|7 - 0| = 7

In region (a):
Max = 7 Min = 4
|7 - 4| = 3 which is equal to threshold.
Therefore, no need to split.

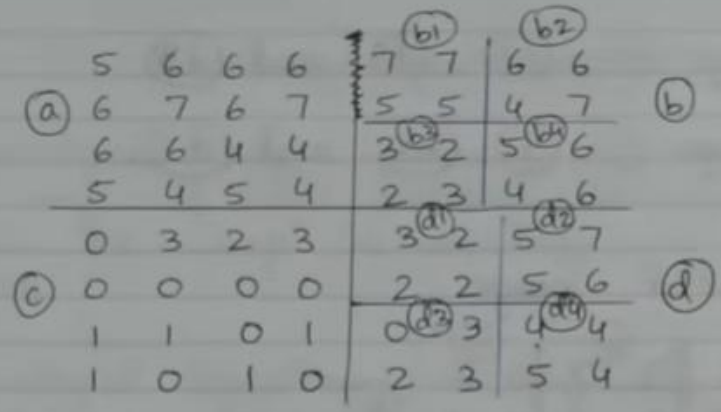
In region (b):
Max = 7 Min = 2
$|7-2| = 5$ which is greater than 3.
Therefore we will split the region (b) into 4 sub-regions.

In region (c):
Max = 3 Min = 0
$|3-0| = 3$.
So no need to split.

In region (d):
Max = 7 Min = 0
$|7-0| = 7$.
So we will split into 4 sub-regions.



Furthermore, we will check all the sub-regions. Since all of them are <= 3, no further splitting is required.



Furthermore, we will check all the sub-regions. Since all of them are <= 3, no further splitting is required.

Merging

Check adjacent regions, if they are falling within the threshold, then merge.

Consider regions (a) and (b1)

Max = 7 Min = 4

|7-4| = 3. ✓ Merge.

Consider regions (a b1) and (b2)

Max = 7 Min = 4

|7-4| = 3. ✓ Merge.

Consider regions (a b1 b2) and (b4)

Max = 7 Min = 4

|7-4| = 3 ✓ Merge

Consider regions (a b1 b2 b4) and d2

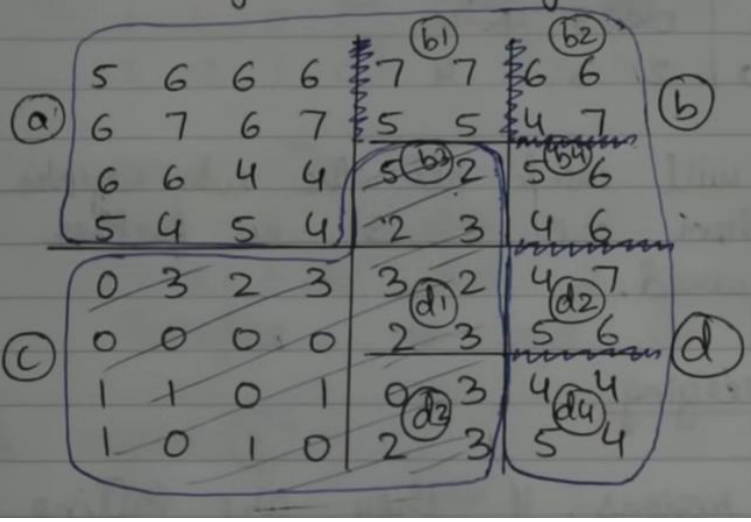
Max = 7 Min = 4

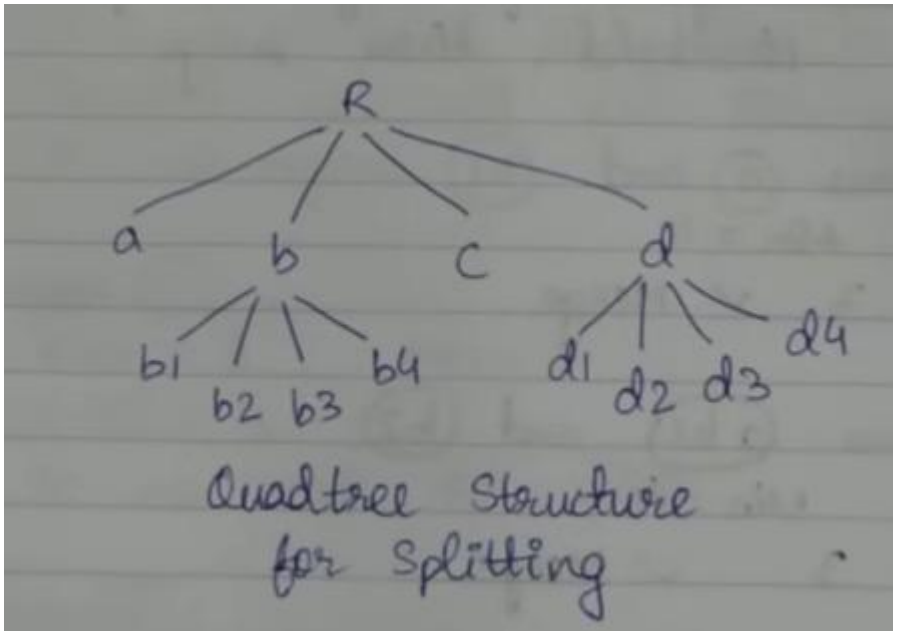
|7-4| = 3 ✓ Merge.

Similarly, merge (a b1 b2 b4 d2) with (d4).

Similarly, merge (c), (d1), (b3) and (d3).

Final segmented image:





Quadtree Structure for Splitting

Morphological Operators

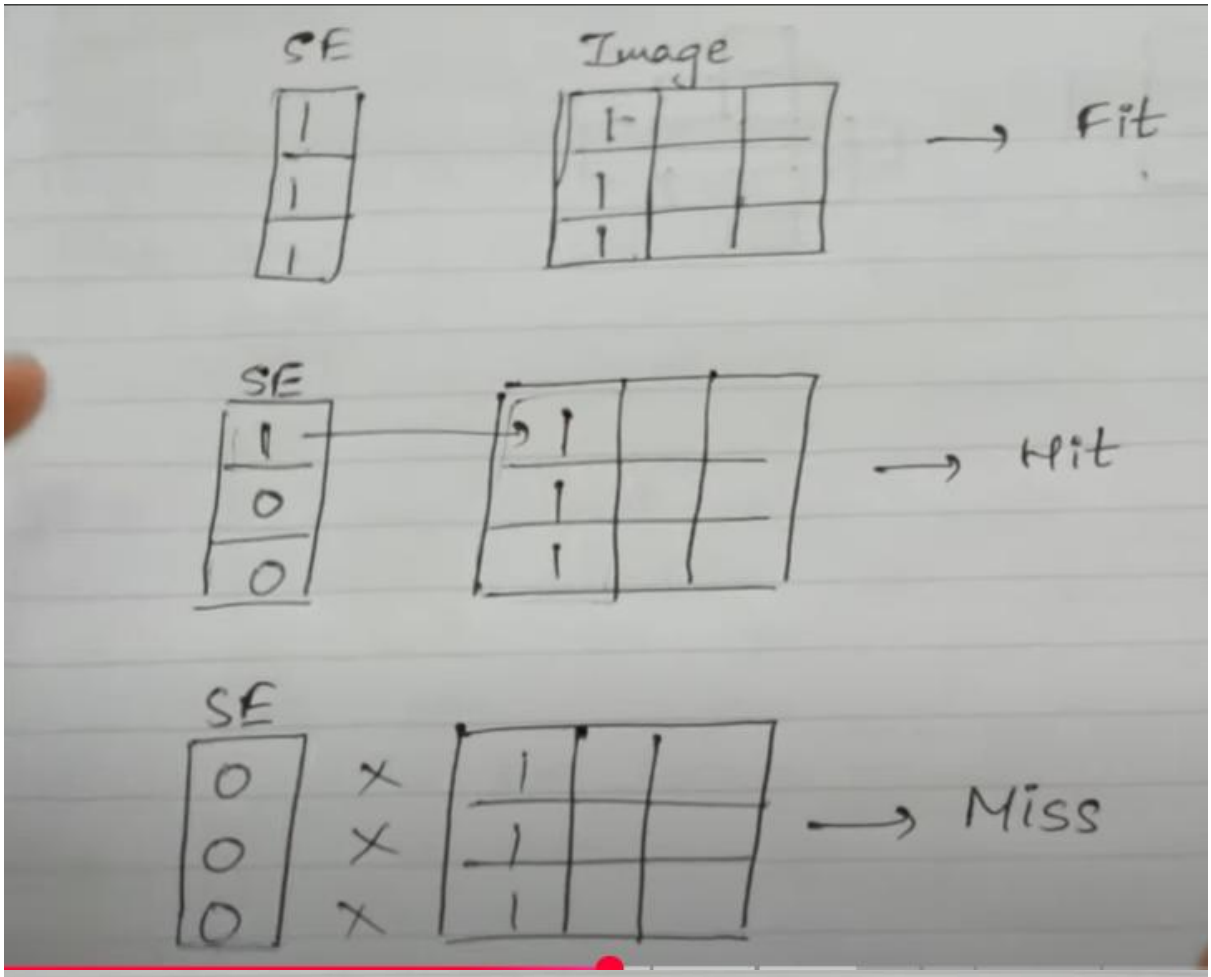
Morphological image processing (or morphology) describes a range of image processing techniques that deal with the shape (or morphology) of features in an image.

Morphological operations are typically applied to remove imperfections introduced during segmentation, and so typically operate on bi-level images.

These operations are performed using the concepts of reflection and translation.

* Structuring Element (SE)

→ It is a small set to probe the image under study.
→ For each structuring element, we should define the origin.
→ The shape and size must be adapted to geometric properties for the objects.
→ We check for three conditions while applying the SE:
  - Fit
  - Hit
  - Miss

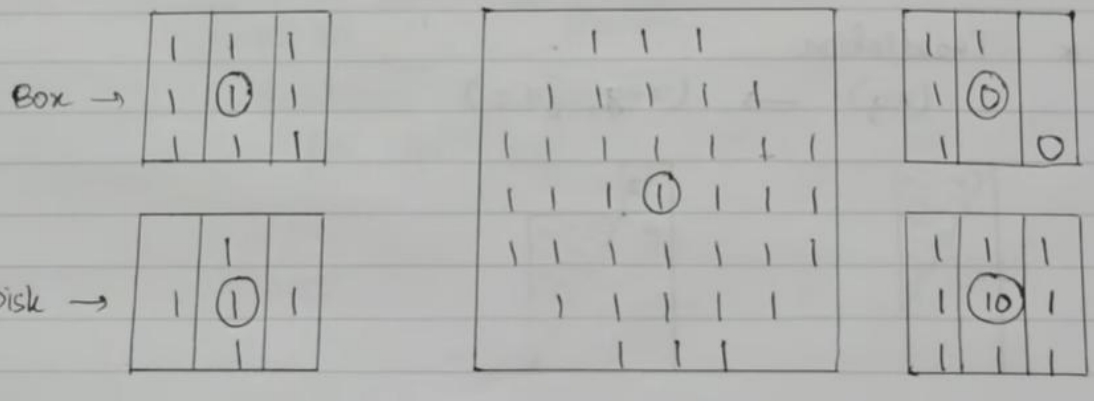


→ SEs can have varying sizes.

→ Usually the element values are 0, 1 and none(!).

→ Empty spots in the structuring elements are don't care's.

→ Examples of SE:



Morphological Operations

→ Basic operations are dual to each other.

1) Dilation and Erosion
→ Dilation enlarges foreground and shrinks background.
→ Erosion shrinks foreground and enlarges background.

* Dilation
→ It is the set of all points in the image where the structuring element "touches or hits" the foreground.
→ Consider each pixel in the input image. If the structuring element matches completely with the pixel values or even if atleast one match is found, write a "1" at the origin of the structuring element.

* Application of dilation
→ Repair breaks (bridging the gaps)
→ Repair intrusions
→ Enlarges the object

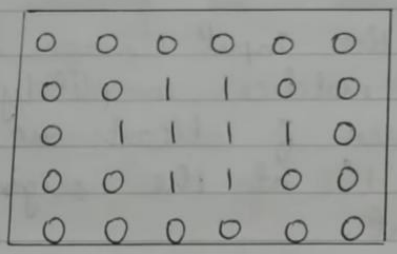
* Dilation is denoted by $A \oplus B$.

* Erosion
→ It is the set of all points in the image where the structuring element "fits into".
→ Consider each foreground pixel in the input image. If the structuring element matches completely with the pixel values, write a "1" at the origin of the structuring element.

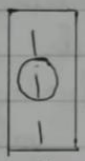
* Application of erosion
→ Split apart joined objects
→ Strip away extrusions
→ Shrink the objects

* Erosion is denoted by $A \ominus B$.

Q1. Compute $A \oplus B$ for the following input images.



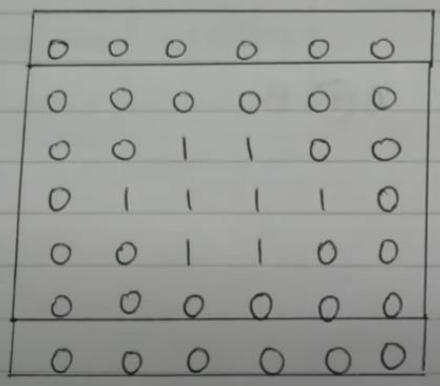
A



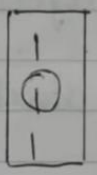
B

Ans. First we will pad the image A at top and bottom.

Ans. First we will pad the image A at top and bottom.



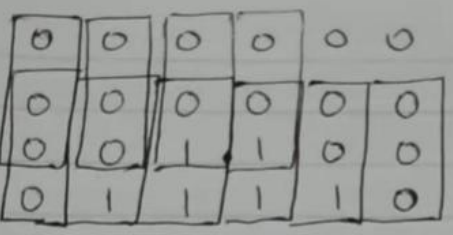
A

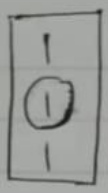


B

Then we will place B over A and check for following conditions for dilation.

Full match = 1.
Some match or atleast one match = 1
No match = 0.



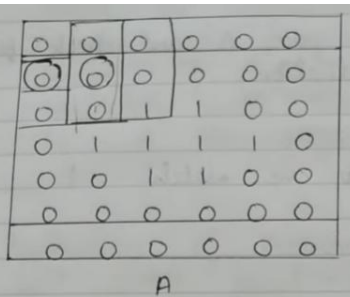


0 0 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0

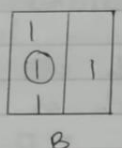
0 0 1 1 0 0
0 1 1 1 1 0

Output image

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



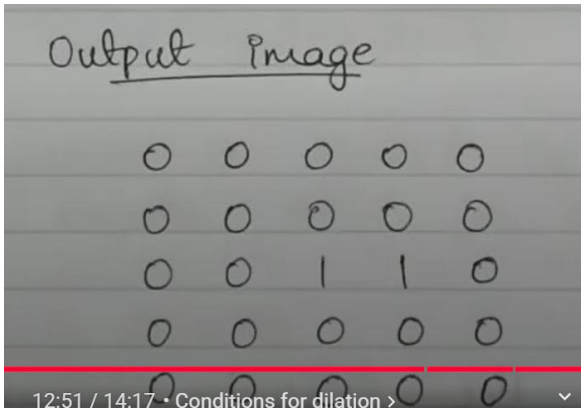
A



B

Output image

0 0 0



📌 **What is Linking and Boundary Detection?**
Boundary detection (also called edge detection) is the process of identifying **edges** or **boundaries** in an image where there are significant changes in intensity, such as the outline of an object. **Linking** is the process of connecting these detected edge points to form continuous boundaries or contours, creating a complete outline of objects. **Local processing** refers to analyzing a small neighborhood around each pixel (e.g., using a 3x3 window) to detect or link edges, making it computationally efficient and focused on local intensity changes. Together, these techniques help segment objects, recognize shapes, or analyze structures in images.

📌 **Why Use Linking and Boundary Detection?**

- To **outline objects** (e.g., detect a tumor in an MRI or a car in a traffic camera).
- To **segment images** (e.g., separate foreground from background).
- To **support computer vision** (e.g., object recognition in self-driving cars).
- To **enhance analysis** (e.g., measure object sizes in industrial inspection).
- 📌 **Focus on Local Processing**:
  Local processing examines a pixel's immediate neighbors (e.g., in a 3x3 or 5x5 window) to detect edges or link them, using techniques like convolution (your prior question) or gradient computation. It's fast and effective for detecting local intensity changes, unlike global methods that analyze the entire image.

✅ 1. **Edge Linking using Local Processing** *(Easy Steps for 7 Marks)*

📌 What is it?

Edge linking connects broken edge pixels into a continuous edge using nearby pixels.

🪜 Steps:

1. **Detect Edges**
   - Use an edge detector (e.g., **Sobel**, **Canny**) to find edges.
2. **Apply Thresholding**
   - Remove weak/noisy edges by keeping only strong edge pixels.
3. **Check Neighboring Pixels (8-Connectivity)**
   - For each edge pixel, check the **8 surrounding pixels**.
   - If a neighbor is also an edge pixel, **link it**.
4. **Link Similar Gradient Directions (optional)**
   - Link pixels only if their **edge direction** is similar.
5. **Use Hysteresis Thresholding (Canny method)**
   - Strong edges are kept.
   - Weak edges are kept **only if connected** to strong edges.

✅ Simple Tip:

If two edge pixels are **close and pointing in the same direction**, connect them.

📌 **Properties**:

1. **Local Connectivity**: Links edges based on nearby pixels, fast and intuitive.
2. **Direction-Based**: Gradient direction ensures meaningful connections.
3. **Gap Sensitivity**: Small gaps may break contours, requiring gap-filling rules.
4. **Robustness**: Works well with strong edges but struggles with noisy or weak edges.

✅ 2. **Boundary Detection using Local Processing** *(Easy Steps for 7 Marks)*

📌 What is it?

Boundary detection means finding the **outline of objects** in an image.

🪜 Steps:

1. **Smooth the Image**
   - Apply blur (e.g., Gaussian filter) to reduce noise.
2. **Detect Edges**
   - Use an edge detector (like **Canny**) to find object outlines.
3. **Link Edge Pixels**
   - Use the above edge linking method to form **continuous edges**.
4. **Trace the Boundary**
   - Follow the connected edge pixels to trace the **shape** of the object.
5. **Label or Mark the Boundary**
   - Store or highlight the object outline for further use (like segmentation or shape analysis).

✅ Simple Tip:

Start at an edge pixel and follow connected pixels until you form a **closed shape**.

📌 **Properties**:

1. **Local Sensitivity**: Detects edges based on nearby pixels, robust to local changes.
2. **Noise Sensitivity**: Noise can produce false edges, requiring preprocessing (e.g., Gaussian low-pass filter, your question).
3. **Directional**: Gradients provide edge orientation, aiding linking.
4. **Threshold-Dependent**: Choice of T affects edge quality (too high misses edges, too low includes noise).

✅ Applications (5 Points with Detail)

1. **Medical Imaging**:
   - **Use**: Detects and links tumor boundaries in MRIs for segmentation.
   - **Example**: Sobel detects tumor edges, and Canny links them into a contour for 2025 AI diagnosis.
2. **Computer Vision**:

- **Use**: Outlines objects (e.g., cars, faces) for recognition or tracking.
- **Example**: A self-driving car uses edge linking to trace lane boundaries in real-time.

3. **Industrial Inspection**:
   - **Use**: Identifies defects (e.g., cracks) by detecting and linking edges.
   - **Example**: A 2025 factory robot detects cracks in metal parts using Sobel and contour tracing.
4. **Document Processing**:
   - **Use**: Detects text or table boundaries in scanned documents.
   - **Example**: Edge detection and linking isolate text boxes in a PDF for OCR.
5. **Satellite Imaging**:
   - **Use**: Outlines roads, rivers, or buildings in aerial images.
   - **Example**: A 2025 satellite image uses Canny to link road edges for mapping.

✅ **Significance (5 Points with Detail)**

1. **Enables Segmentation**: Boundary detection and linking isolate objects, critical for analysis (e.g., medical, vision).
2. **Fast Local Processing**: Uses small neighborhoods, ideal for real-time 2025 applications (e.g., smartphones, drones).
3. **Improves Recognition**: Continuous contours provide shape information for AI and human viewers.
4. **Robust to Local Changes**: Focuses on local gradients, effective in varied lighting or textures.
5. **Foundation for Advanced Tasks**: Feeds into contour analysis, shape recognition, or 3D reconstruction.

🔹 **Thresholding**

📌 **Foundation of Thresholding**

Thresholding is a fundamental **image segmentation** technique that converts a **grayscale image** into a **binary image** by comparing each pixel's intensity to a **threshold value**. Pixels above the threshold are assigned one value (e.g., white, 255), and those below or equal are assigned another (e.g., black, 0). It's used to separate objects (e.g., foreground) from the background based on intensity differences, like sorting items into two groups based on size. Thresholding is a **point processing** method in the **spatial domain**, meaning each pixel is

processed independently, making it simple and fast. It's a cornerstone of image processing, enabling tasks like object detection, text extraction, and medical image analysis.

Why Use Thresholding?

- To **segment objects** (e.g., isolate a tumor in an MRI).
- To **simplify images** for analysis (e.g., binarize text for OCR).
- To **prepare images** for further processing (e.g., edge detection, your prior question).
- To **enhance contrast** in specific regions (complements your enhancement questions).

Visual Analogy: Imagine a grayscale photo as a field with varying shades of gray grass. Thresholding is like drawing a line: all grass darker than a certain shade becomes black, and all lighter grass becomes white, creating a clear map of dark vs. light areas.

🔸 1. Simple Global Thresholding

📌 **Definition**:

Simple global thresholding applies a **single threshold value** T to the entire image to create a binary output. It assumes the image has a clear separation between object and background intensities (e.g., dark objects on a light background). The threshold is often chosen manually or computed from the image's histogram.

📌 **Process in Simple Words**:

1. **Select Threshold**: Pick a fixed threshold T (e.g., 128 for a 0–255 grayscale image) or compute it (e.g., mean intensity, histogram-based).
2. **Compare Pixels**: For each pixel $f(x, y)$:
   - If $f(x, y) > T$, set the output $g(x, y) = 255$ (white).
   - If $f(x, y) \leq T$, set $g(x, y) = 0$ (black).
3. **Apply to All Pixels**: Process the entire image to produce a binary image.

Analogy: Like sorting apples by weight with one cutoff: heavier than 200g goes to one bin (white), lighter to another (black).

📌 **Mathematical Formulation**:

For a grayscale image $f(x, y)$, the output $g(x, y)$ is:

$$g(x,y) = \begin{cases} 255 & \text{if } f(x,y) > T \\ 0 & \text{otherwise} \end{cases}$$

- T: Global threshold, constant across the image.
- **Threshold Selection**:
  - **Manual**: Set T based on visual inspection (e.g., $T = 128$).
  - **Histogram-Based**: Use the image histogram to find a valley between two peaks (e.g., object and background intensities).
  - **Mean/Median**: Set T as the average or median intensity of the image.

📌 **Properties**:

1. **Simplicity**: Easy to implement, requires only one threshold.
2. **Fast**: Point processing, no neighborhood analysis.
3. **Assumes Bimodality**: Works best when the histogram has two distinct peaks (object vs. background).
4. **Limited Flexibility**: Fails if lighting varies or intensities overlap significantly.
5. **Lossy**: Discards grayscale details, producing binary output.

🔸 2. Optimal Thresholding

📌 **Definition**:

Optimal thresholding automatically determines the **best threshold** T by optimizing a criterion, such as minimizing classification error or maximizing separation between object and background classes. It's more sophisticated than simple global thresholding, using statistical methods to find a threshold that best segments the image, especially when the histogram isn't clearly bimodal.

📌 **Process in Simple Words**:

1. **Model Classes**: Assume the image has two classes (e.g., object and background) with different intensity distributions (e.g., Gaussian).
2. **Compute Criterion**: Use a mathematical measure (e.g., between-class variance, error rate) to evaluate possible thresholds.
3. **Optimize**: Find the threshold T that maximizes or minimizes the criterion (e.g., maximizes separation between classes).
4. **Apply Threshold**: Use the optimal T to binarize the image, as in simple global thresholding.

Analogy: Like finding the perfect dividing line between two groups of people based on height, ensuring the groups are as distinct as possible with minimal overlap.

🧠 3. Optimal Thresholding (Otsu's Method)

📌 Definition:

Finds the **best possible threshold** that **minimizes within-class variance** or **maximizes between-class variance** automatically.

🪜 Steps:

1. Compute histogram of the image
2. Try all possible threshold values T
3. For each T, compute:
    - Variance within foreground and background
4. Select T that **minimizes total within-class variance**

✅ Advantages:

- No need to guess a threshold
- Works well for **bimodal histograms** (two peaks: one for object, one for background)

📌 **Properties**:

1. **Automated**: Finds the best T without manual input.
2. **Robust**: Works for complex histograms, unlike simple thresholding.
3. **Statistically Grounded**: Maximizes class separation, minimizing overlap.
4. **Computationally Intensive**: Requires histogram analysis and optimization.
5. **Still Global**: Single T, so it struggles with varying lighting (unlike adaptive thresholding).

✅ **Applications (5 Points with Detail)**

1. **Medical Imaging**:
    - **Use**: Simple global thresholding segments bones in X-rays; optimal thresholding (Otsu) isolates tumors in MRIs.
    - **Example**: A 2025 MRI uses Otsu to binarize a tumor, aiding AI diagnosis.
2. **Document Processing**:
    - **Use**: Binarizes text in scanned documents for OCR.

- **Example**: Simple thresholding with T = 128 converts a scanned book to black text on white, while Otsu optimizes for faded pages.

3. **Computer Vision**:
   - **Use**: Segments objects (e.g., cars, faces) for recognition.
   - **Example**: A self-driving car in 2025 uses Otsu to separate road signs from backgrounds.
4. **Industrial Inspection**:
   - **Use**: Detects defects (e.g., scratches) by thresholding.
   - **Example**: Simple thresholding identifies dark cracks in bright metal parts; Otsu adapts to varying lighting.
5. **Satellite Imaging**:
   - **Use**: Separates land from water in aerial images.
   - **Example**: Otsu thresholds a 2025 satellite image to map rivers accurately.

✅ **Significance (5 Points with Detail)**

1. **Simplifies Analysis**: Converts complex grayscale images to binary, enabling segmentation and recognition.
2. **Fast and Efficient**: Point processing (especially simple thresholding) suits real-time 2025 applications (e.g., smartphones, medical devices).
3. **Versatile**: Works in medical, industrial, and consumer domains, fitting diverse needs.
4. **Robust with Optimal Methods**: Otsu handles complex images, improving reliability over manual thresholds.
5. **Preprocessing Step**: Prepares images for edge detection (your question), contour analysis, or compression (DCT, your question).

✅ Region Oriented Segmentation

📌 What is it?

Region-oriented segmentation methods group **pixels into regions** based on **similarity** (like intensity, texture, or color), rather than just edges or thresholds.

🔹 1. Basic Formulation

✅ Idea:

Divide an image into regions that satisfy a **uniformity criterion**, such as:

- Same gray level
- Same texture
- Similar color

✅ Conditions for a valid segmentation:

Let $R_1, R_2, ..., R_n$ be regions in image I:

1. $\bigcup R_i = I$ (entire image is covered)
2. $R_i \cap R_j = \emptyset$ (no overlapping)
3. Each R_i satisfies a property $Q(R_i) = \text{TRUE}$
4. Neighboring regions R_i and R_j must be **different** (i.e., $Q(R_i \cup R_j) = \text{FALSE}$)

🔹 2. **Region Growing by Pixel Aggregation**

✅ What is it?

Start from a **seed pixel** and **grow** the region by adding neighboring pixels that are **similar** (within a threshold).

🪜 Steps:

1. Choose one or more **seed points**
2. Check **neighboring pixels** (4 or 8-connectivity)
3. Add neighbors that are **similar** in intensity (within a threshold)
4. Repeat until no more pixels can be added

✅ Example Criterion:

If $|f(x, y) - f(seed)| < T$, then add pixel to region.

✅ Use:

- Works well when regions have **distinct intensity values**

🔹 3. **Region Splitting and Merging**

✅ What is it?

A **top-down and bottom-up** approach:

- **Split** large regions that are not uniform
- **Merge** adjacent regions that are similar

🪜 Steps:

1. Start with the **entire image as one region**
2. **Split** the region if it does not satisfy uniformity (e.g., using quadtree splitting)
3. When no more splits are possible, **merge** adjacent regions that are similar
4. Repeat until the final segmentation is done

✅ Quadtree Splitting:

- Divide the region into **4 equal parts**
- Keep splitting each square until each part meets the condition

✅ Region-Based Segmentation: Region Growing & Splitting

🔹 1. Region Growing

📌 Concept:

Region growing is a **bottom-up** segmentation method. It starts with **seed pixels** and adds neighboring pixels that have **similar properties** (like gray level).

🪜 Steps of Region Growing:

1. Select one or more **seed pixels** manually or automatically.
2. For each seed pixel, examine its **4 or 8 neighbors**.
3. If a neighbor's intensity is **similar (within a threshold)** to the seed, **add it** to the region.
4. Repeat for new pixels added, until **no more similar neighbors** are found.

✅ Example:

- Suppose you have a grayscale image.
- Choose a **seed pixel** with intensity 120.
- Set threshold T=10
- Neighbor pixels with intensity between **110 and 130** will be added to the region.
- The region grows outward as long as the condition is satisfied.

🎯 Used for:

- Segmenting tumors in medical images
- Identifying lakes in satellite images
- Segmenting simple objects

🔹 2. Region Splitting

📌 Concept:

Region splitting is a **top-down** approach. It divides the image into **smaller square regions**, and **checks for uniformity** in each. If not uniform, the region is **split into 4 parts** (like a quadtree).

🪜 Steps of Region Splitting:

1. Start with the **entire image** as one region.
2. Check if the region satisfies a **uniformity condition** (e.g., same average intensity).
3. If not uniform, **split** the region into **four equal quadrants**.
4. Repeat the process for each sub-region.
5. Stop when all regions are **uniform** or are small enough.

✅ Example:

- Consider an image with a large background (intensity 200) and a dark square (intensity 50) in one corner.
- The image is first checked for uniformity.
- Since two regions have very different intensities, it gets split into 4.
- The region with the dark square is again not uniform, so it's split again.
- This continues until all regions are uniform or small enough.

🎯 Used for:

- Images with **multiple objects**
- Areas with **different textures or lighting**
- When **seed selection is hard**

🧠 Summary Table

Feature	Region Growing	Region Splitting
Direction	Bottom-up	Top-down
Starts With	Seed pixel(s)	Whole image
Based On	Pixel similarity	Region uniformity
Shape of Regions	Irregular	Always square

Compare and Contrast: Lossless vs. Lossy Compression

Feature	Lossless Compression	Lossy Compression
Definition	Compresses data **without losing any original information**. All data can be perfectly reconstructed.	Compresses data by **removing less important or redundant information**, resulting in some loss of data that cannot be recovered.
Data Recovery	The original data can be **fully and exactly recovered** after decompression.	The original data cannot be perfectly restored; only an approximation is recovered, with some permanent data loss.
Image Quality	Since no data is lost, the **image quality remains unchanged** after compression and decompression.	The image quality may degrade, with artifacts or loss of fine details, depending on compression level.
Compression Ratio	Typically **lower compression ratios**, often between 2:1 and 3:1, because no information can be discarded.	Achieves **higher compression ratios**, sometimes up to 10:1, 50:1, or more, by sacrificing some data quality.
Use Cases	Used when **exact data reproduction is critical**, such as in text files, legal documents, medical images, and technical drawings.	Used where some quality loss is acceptable to save space, such as in photographs, audio files, videos, and streaming media.
Examples	PNG, TIFF, GIF (for images); ZIP (general data); Huffman coding, Run-Length Encoding (RLE)	JPEG (images), MP3 (audio), MPEG (video), HEVC (video codec)

✅ **Popular Lossless Compression Technique: Huffman Coding**

📌 **Purpose:**

Reduce the average number of bits used to represent symbols based on their **frequency**.

🧠 **Basic Idea:**

- **More frequent symbols** get **shorter codes**.
- **Less frequent symbols** get **longer codes**.

🪜 **Steps in Huffman Coding:**

1. **Count Frequencies**
   Calculate the frequency of each symbol (e.g., pixel value).
2. **Build a Priority Queue (Min-Heap)**
   Each symbol is a node with its frequency.
3. **Build Huffman Tree**
   - Pick two lowest-frequency nodes.
   - Merge them into a new node with combined frequency.
   - Repeat until only **one node (root)** is left.
4. **Assign Codes**
   - Traverse the tree:
     - Left branch = 0
     - Right branch = 1
   - This gives each symbol a **unique binary code**.
5. **Encoding**
   Replace each symbol in the original data with its Huffman code.
6. **Decoding**
   Use the same tree to **decode the bitstream** back to original data.

✅ **Example:**

Symbol	Frequency	Huffman Code
A	5	00
B	9	01
C	12	10
D	13	110
E	16	111

Original string: ABCDE
Encoded string: 000110110111

✅ **Advantages:**

- **No data loss**
- Efficient for text, icons, line drawings

🧠 **Conclusion:**

- **Lossless compression** is crucial when **data integrity** is important.
- **Huffman coding** is a widely-used method that offers **efficient encoding** without losing any information.

1. What is Image Restoration?

Image Restoration is the process of recovering an original image that has been degraded by factors like noise, blurring, or distortions during image acquisition or transmission. The goal is to **improve the quality** of the image by **removing or reducing degradations** and reconstructing the true image as closely as possible.

---

2. Differences between Image Deblurring and Image Inpainting

Feature	Image Deblurring	Image Inpainting
Purpose	To remove **blur effects** caused by motion, defocus, or camera shake	To **fill in missing or damaged parts** of an image
Type of Degradation	Blur (loss of sharpness)	Missing or corrupted pixels (holes, scratches)
Techniques Used	Deconvolution, Wiener filtering, blind deblurring	Patch-based methods, exemplar-based, diffusion methods
Result	Sharper, clearer images with restored edges	Reconstructed regions that blend naturally with surroundings
Input Requirement	Blurred but complete image	Image with missing or damaged areas
Example Applications	Restoring old photos blurred by camera shake	Removing scratches, text removal, restoring damaged artworks

---

Summary:

- **Image Restoration** broadly includes all methods to recover degraded images.
- **Image Deblurring** is a restoration task focused on correcting blurs.
- **Image Inpainting** restores **missing or damaged regions** by intelligently filling them.

Image Morphing: Steps and Example

What is Image Morphing?

Morphing is a technique to smoothly transform one image into another by gradually changing shapes and colors over time.

Steps Involved in Image Morphing

1. **Input Images Selection**
   Choose the **source image** (start) and the **target image** (end).
2. **Feature Point Selection**
   Identify corresponding **key feature points** on both images (e.g., eyes, nose, mouth in faces).
3. **Triangulation**
   Divide the images into corresponding **triangles** by connecting the feature points (using Delaunay triangulation).
4. **Warping**
   Gradually **warp** each triangle in the source image to the shape of the corresponding triangle in the target image, for intermediate frames.
5. **Cross-Dissolving (Blending)**
   Blend the pixel colors of the warped source and target images for each intermediate frame, gradually changing pixel values.
6. **Generate Intermediate Frames**
   Create a sequence of images showing smooth transformation from the source to the target.
7. **Display or Save Animation**
   Combine intermediate frames to produce a **video or GIF** showing the morphing effect.

Example

- Morphing a picture of a **cat** into a **dog**:
    - Select key points on cat's eyes, nose, ears.
    - Select corresponding points on the dog's face.
    - Triangulate both images.
    - Warp the cat's triangles gradually to the dog's.
    - Blend colors frame-by-frame.
    - Result: A smooth animation transforming the cat's face into the dog's.

Summary

- Morphing = **Shape + Color transformation** over time.
- Requires accurate **feature correspondence**.
- Used in movies, animation, special effects.

Common Metrics to Assess Segmentation Performance

Image segmentation algorithms are evaluated using various metrics to measure how accurately they partition an image into meaningful regions. Some common metrics are:

---

1. Pixel Accuracy (PA)

- **Definition:** Percentage of correctly classified pixels over the total pixels.
- **Formula:**

$$PA = \frac{\text{Number of correctly classified pixels}}{\text{Total number of pixels}} \times 100\%$$

- **Use:** Simple and intuitive, but may be misleading if classes are imbalanced.

2. Precision and Recall

- **Precision:** Proportion of pixels predicted as belonging to a class that are actually correct.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall:** Proportion of actual pixels of a class that are correctly identified.

$$Recall = \frac{TP}{TP + FN}$$

- Where:
  - TP = True Positives
  - FP = False Positives
  - FN = False Negatives
- **Use:** Important for evaluating specific class performance, especially in object segmentation.

---

3. F1 Score (Dice Coefficient)

- Harmonic mean of precision and recall, balancing both.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

- Values range from 0 (worst) to 1 (best).

4. Jaccard Index (Intersection over Union, IoU)

- Measures overlap between predicted and ground truth regions.

$$IoU = \frac{|Prediction \cap GroundTruth|}{|Prediction \cup GroundTruth|}$$

- Value ranges between 0 and 1, where 1 means perfect overlap.

---

5. Boundary-based Metrics

- Evaluate how well the predicted boundaries match ground truth boundaries.
- Examples: **Hausdorff Distance**, **Boundary F-measure**.

Image Restoration vs. Segmentation vs. Enhancement

Aspect	**Image Restoration**	**Image Segmentation**	**Image Enhancement**
Purpose	Recover original image from degraded/noisy version	Partition image into meaningful regions/objects	Improve visual quality or highlight features
Goal	Remove noise, blur, or distortions	Identify and separate objects or regions	Enhance contrast, brightness, or sharpness
Process Type	Correction and reconstruction	Classification and grouping	Adjustment and modification
Output	Restored image with reduced degradation	Segmented image with labeled regions	Enhanced image with better appearance
Examples	Deblurring, denoising, inpainting	Region growing, thresholding, edge detection	Histogram equalization, smoothing, sharpening
Used In	Medical imaging, satellite images, old photos	Object recognition, image analysis, computer vision	Display improvement, pre-processing for analysis
Focus	Fixing imperfections	Understanding image structure	Making images visually clearer