MACHINE VISION

Contents

MODULE 1

BASICS OF IMAGE PROCESSING

What is Machine Vision?

• Machine vision refers to the ***field of science and technology*** that enables machines to ***interpret and analyze visual data*** from the world.

• By combining ***optics, sensors, image processing***, and ***artificial intelligence***, machine vision systems can perform tasks that involve ***understanding and acting*** on ***visual inputs***, such as identifying objects, detecting defects, measuring dimensions, and making decisions based on visual data

KEY ASPECTS OF MACHINE VISION

1. **Image Acquisition:**
   - ***Capturing images*** using cameras and sensors.
   - Types of cameras: ***Area-scan, line-scan, and 3D cameras***.
   - *Sensors*: ***CCD*** (Charge Coupled Device) and ***CMOS*** (Complementary Metal-Oxide Semiconductor).
2. **Lighting:**
   - Ensures ***uniform illumination*** for capturing ***clear and accurate*** images.
   - *Common lighting techniques*: ***Backlighting***, ***diffuse lighting***, and ***structured lighting***.
3. **Optics**:
   - ***Lenses focus*** the image onto the ***sensor***.
   - *Important properties:* ***focal length, aperture, and depth of field***.
4. **Image Processing and Analysis:**
   - ***Algorithms*** process the captured image to ***extract features*** and ***identify patterns***.
   - Techniques include ***filtering***, ***edge detection, segmentation***, and ***morphological operations***.
5. **Communication and Control:**
   - Machine vision systems communicate with ***other systems*** (e.g., robots) to ***act on processed data***.
   - Industrial protocols like ***Ethernet, Modbus,*** and ***fieldbus*** enable integration.
6. **Software**:
   - Software frameworks and libraries like ***OpenCV, MATLAB***, and proprietary ***tools*** handle image processing tasks

WORKING OF A MACHINE VISION SYSTEM

The process can be broken into several stages:

1. **Image Acquisition:** The camera captures raw visual data.
2. **Preprocessing:** Noise reduction and image enhancement.
3. **Feature Extraction:** Identifying edges, textures, shapes, or regions of interest.
4. **Analysis:** Classification, object detection, pattern matching, or measurement.
5. **Decision Making:** Output is analyzed to trigger actions (e.g., sorting, defect rejection)

KEY TECHNOLOGIES IN MACHINE VISION

1. **2D and 3D Imaging:**
   - ***2D Vision:*** Used for ***flat objects*** or ***surfaces***.
   - ***3D Vision:*** Captures ***depth*** and ***volume*** using stereo vision, laser scanning, or structured light.
2. **Deep Learning**:
   - ***Neural networks*** perform tasks like image classification, object detection, and semantic segmentation.
3. **Edge AI:**
   - ***Processing*** occurs ***locally*** on devices (edge) instead of in the cloud, reducing latency.
4. **Hyperspectral Imaging:**
   - Captures a wide spectrum of light for ***material analysis*** and ***defect detection***.
5. **Thermal Imaging:**
   - ***Infrared cameras*** measure ***heat signatures*** for industrial and safety applications

APPLICATIONS OF MACHINE VISION

Industrial Automation

- **Inspection**: Detecting defects, scratches, or missing parts in manufacturing.
- **Assembly Verification:** Ensures all components are correctly placed.
- **Sorting and Grading:** Classifies products (e.g., food, pharmaceuticals).
- **Robotics Guidance:** Helps robots identify and pick objects in assembly lines.

Healthcare

- **Medical Imaging:** Enhances diagnosis in X-rays, MRIs, and CT scans.
- **Surgical Assistance:** Guides robotic surgery systems.
- **Patient Monitoring:** Tracks vitals and movements in real-time.

Automotive Industry

- **Autonomous Vehicles:** Assists in lane detection, obstacle recognition, and sign identification.
- **Quality Control:** Inspects components like engines, brakes, and paint

Agriculture

- **Precision Farming:** Identifies crop health, pests, and nutrient deficiencies using drones.
- **Sorting:** Automated grading of fruits and vegetables.

Retail and Consumer Applications

- **Facial Recognition:** Used in security and customer personalization.
- **Self-Checkout Systems:** Automatically identifies items for billing.

Security and Surveillance

- **Anomaly Detection:** Monitors unusual activities in public spaces.
- **License Plate Recognition (LPR):** Identifies vehicles for tolling or parking management.

Electronics

- **PCB Inspection:** Detects defects in printed circuit boards.
- **Component Placement:** Ensures precise placement of tiny electronic parts.

Food and Beverage Industry

- **Packaging Inspection:** Verifies labels, seals, and product presence.
- **Foreign Object Detection:** Identifies contaminants in food products.

Aerospace

- **Structural Inspection**: Checks for cracks and defects in aircraft components.
- **Navigation Assistance:** Helps drones and aircraft during takeoff, landing, and in-flight navigation.

ADVANTAGES OF MACHINE VISION

- **Speed**: Performs inspections and measurements faster than humans.
- **Consistency**: Reduces errors caused by human fatigue or subjective judgment.

- **Cost-Effective**: Minimizes labor costs and rejects faulty products early.
- **Enhanced Safety**: Detects hazardous situations in manufacturing environments.

CHALLENGES IN MACHINE VISION

1. **Lighting Conditions:** Varying lighting can affect image quality.
2. **Complexity**: Developing and training robust algorithms for diverse environments.
3. **Cost**: High initial investment for industrial-grade systems.
4. **Scalability**: Adapting to different tasks or industries often requires reconfiguration

FUTURE TRENDS IN MACHINE VISION

1. **AI-Powered Vision:** Increased integration of AI and deep learning for advanced capabilities.
2. **Real-Time Processing**: Faster processing with GPUs and edge computing.
3. **Embedded Vision:** Vision systems embedded into compact devices like drones and smartphones.
4. **Augmented Reality (AR):** Enhances human vision with additional visual data in real time.
5. **Collaborative Robotics (Cobots):** Machine vision-enabled robots working safely alongside humans.

WHAT IS IMAGE PROCESSING

- Image Processing refers to the ***techniques*** used to ***manipulate, enhance, or analyze images*** (still pictures or videos) to ***extract meaningful information*** or improve their quality.
- It involves converting an image into a ***digital format*** and applying various ***algorithms*** to process it.
- Image processing is a crucial step in machine vision and computer vision applications

STAGES OF IMAGE PROCESSING

1. **Image Acquisition:**
   1. Capturing an image using ***cameras***, ***sensors***, or other devices.
   2. Converts visual data into a digital format (matrix of pixel values).
2. **Preprocessing:**
   Improves the ***quality*** of the image or removes unwanted distortions. Techniques include:
   - **Noise Removal:** Filters like median or Gaussian reduce random variations.
   - **Contrast Enhancement**: Adjusts brightness and contrast for better visualization.
   - **Resizing or Cropping:** Adjusts the image size or focus area.
3. **Segmentation**:
   - Divides an image into ***meaningful regions***, such as ***objects*** or ***boundaries***.
   - Methods include ***edge detection, thresholding***, and ***region-based*** segmentation.
4. **Feature Extraction:**
   - Identifies and extracts significant features like ***edges, corners, textures***, or ***shapes***.
   - Used in ***pattern recognition*** or ***object detection***.
5. **Image Analysis:**
   - ***Quantifies*** image content, such as ***size, shape, or intensity*** of objects.
   - Includes tasks like ***counting objects*** or ***measuring dimensions***.
6. **Postprocessing**:
   - Final steps to prepare the processed image for output or decision-making.
   - May involve ***annotation, visualization***, or ***overlay of information***.

TYPES OF IMAGE PROCESSING

1. **Analog Image Processing**: Applied to physical images using traditional techniques like ***magnification*** or ***printing***.

2. **Digital Image Processing:** Applies computational ***algorithms*** to digital images. Examples include: ***Filtering, transformation*** (Fourier or wavelet), and ***morphological*** operations

APPLICATIONS OF IMAGE PROCESSING

Medical Imaging:
- Enhances X-rays, CT scans, and MRIs for diagnosis.
- Helps in tumor detection and segmentation.

Industrial Automation:
- Inspects parts and detects defects in manufacturing.
- Measures dimensions and verifies assembly.

Facial Recognition:
- Identifies faces for authentication or surveillance.

Remote Sensing:
- Processes satellite images for environmental monitoring, agriculture, and urban planning.

Entertainment:
- Enhances images and videos for editing, gaming, and virtual reality.

Robotics:
- Enables robots to recognize objects, navigate, or avoid obstacles.

WHAT IS DIGITAL IMAGE PROCESSING
- Digital Image Processing (DIP) refers to the ***manipulation and analysis*** of digital images using ***computational techniques***.
- It involves ***converting*** an image into a ***numerical form*** that computers can process, applying ***algorithms*** to ***improve, extract, or analyze*** its information.

KEY STEPS IN DIGITAL IMAGE PROCESSING

1. **Image Acquisition:**
   - Captures an image using cameras or scanners.
   - Converts the image into a digital form (a matrix of pixel values).
2. **Preprocessing**:
   Enhances image quality by removing noise, correcting distortions, or improving contrast. Common techniques:
   - **Noise Reduction:** Median, Gaussian, or bilateral filtering.
   - **Histogram Equalization:** Improves contrast in low-light images.
   - **Resampling**: Adjusts image resolution or size.
3. **Image Transformation:**
   Converts the image to a different domain for analysis or enhancement. Example methods
   - **Fourier Transform:** Analyzes ***frequency components*** for ***compression or filtering***.
   - **Wavelet Transform:** Decomposes images into ***multiple resolutions*** for ***texture analysis***.
4. **Image Enhancement:**
   Improves visual quality for better interpretation.
   Techniques:
   - **Sharpening**: Enhances edges or details.
   - **Smoothing**: Reduces noise or pixelation.
5. **Compression**:
   Reduces file size for storage or transmission.
   Types of compression:
   - **Lossless**: Retains original data (e.g., PNG, BMP).
   - **Lossy**: Reduces file size with slight quality loss (e.g., JPEG).
6. **Object Recognition and Analysis**:
   - ***Identifies*** and ***classifies*** objects or patterns in an image.
   - Used in applications like face recognition, license plate detection, etc.
7. **Image Reconstruction**:

- Reconstructs degraded or incomplete images, often used in ***medical imaging*** or ***remote sensing***.

KEY TOOLS AND LIBRARIES FOR DIGITAL IMAGE PROCESSING

- **OpenCV**: Open-source library for image processing and computer vision.
- **MATLAB**: Provides advanced image processing toolboxes.
- **Python Libraries:**
  - **Pillow**: Basic image manipulation.
  - **Scikit-image:** Image analysis.
  - **TensorFlow/Keras:** For deep learning-based image processing.
  - **ImageJ:** Tool for scientific image analysis

WHAT IS AN IMAGE?

- An image is a ***visual representation*** of an object, scene, or concept.
- It is created by ***capturing or generating patterns of light intensity or color***, typically in ***two dimensions***.
- Images can be represented in various forms, such as photographs, drawings, or digital files, and are used to ***convey information, emotions, or artistic expression.***

TYPES OF IMAGES

1. **Analog Image**: A ***continuous-tone*** image, such as a photograph or a painting, created using traditional media. Examples: Film photographs or hand-drawn art.
2. **Digital Image**: A ***numerical representation*** of an image, consisting of a grid of discrete units called ***pixels*** (picture elements). Each pixel has a specific ***intensity or color value***. Examples: Images displayed on screens or stored as files like PNG, JPEG, or BMP

CHARACTERISTICS OF A DIGITAL IMAGE

- **Resolution**:
  - The ***number of pixels*** in an image, typically represented as ***width × height*** (e.g., 1920×1080).
  - Determines the ***level of detail*** in the image.
- **Pixel Values:**
  Represent the ***intensity or color*** of each pixel.
- **Grayscale Images:**
  Pixels have intensity values (e.g., ***0 for black, 255 for white*** in an ***8-bit*** image).
- **Color Images:**
  Pixels are represented in ***color spaces*** like RGB, with values for red, green, and blue channels.
- **Bit Depth:**
  - The number of bits used to ***represent each pixel***.
  - ***Higher*** bit depth allows ***more precise*** representation of colors or shades of gray.
- **Dynamic Range:**
  The range of ***intensity levels*** that an image can display, affecting contrast and detail.

EXAMPLES OF IMAGES

- **Photographs**: Captured by cameras, representing real-world scenes.
- **Drawings**: Created by hand or digitally.
- **Medical Images**: CT scans or X-rays used for diagnostics.
- **Satellite Images**: Representing Earth's surface

IMAGE FORMATION

- Image formation refers to the process by which ***light interacts with optical elements*** such as ***mirrors, lenses, or apertures*** to produce a visual representation (image) of an object.
- This process is ***fundamental*** to the functioning of optical devices like cameras, microscopes, telescopes, and even the human eye.
- In essence, ***light rays*** originating from or reflected by an ***object*** are directed through an ***optical system*** to form an image, either on a ***physical screen*** (real image) or as a ***visual perception*** (virtual image).

APPLICATIONS OF IMAGE FORMATION

1. **Cameras**: Use ***lenses*** to focus light onto a ***photographic film*** or ***digital sensor***, creating an image.
2. **Human Eye:** Works like a ***convex lens*** system to focus light onto the ***retina***, forming a ***real, inverted*** image.
3. **Microscopes:** Magnify small objects by using ***multiple lenses*** to form ***enlarged*** images.
4. **Telescopes:** Focus light from ***distant celestial objects*** to create visible images.
5. **Projectors:** ***Magnify small images*** from a slide or digital screen and project them onto a ***larger screen***

BASIC COMPONENTS OF IMAGE FORMATION

The image formation process involves three critical elements:

1. **Object**
   - **Source of Light:** Can either emit its own light (e.g., a bulb) or reflect light from another source (e.g., a tree under sunlight).
   - **Characteristics:** The object determines the characteristics of the image, including:
     - Brightness
     - Size
     - Orientation
2. **Optical System:** A medium or device that manipulates light to form an image. Examples:
   - **Lenses:** Refract (bend) light to focus it at a specific point.
   - **Mirrors:** Reflect light according to the laws of reflection.
   - **Apertures (e.g., pinhole cameras):** Allow only specific rays to pass through, forming an inverted image.
3. **Image:** The visual representation of the object created by the optical system. Types of Images:
   - **Real Image:** Formed when light rays converge on a screen (e.g., camera sensor, retina).
   - **Virtual Image:** Formed when light rays appear to diverge from a point (e.g., the image seen in a plane mirror).

UNDERSTANDING IMAGE FORMATION

- **Incident Rays:** Rays that originate from an object and strike an optical element.
- **Reflected Rays:** Rays that bounce off a mirror.
- **Refracted Rays:** Rays that bend when passing through a lens or crossing the boundary between two media.

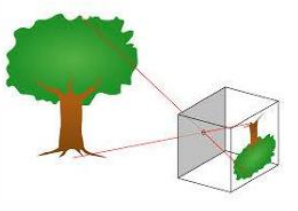
Types of Rays Commonly Used

1. **Parallel Rays:**
   1. Parallel to the principal axis.
   2. Converge (or appear to diverge) at the focal point.
2. **Focal Rays:**
   - Pass through the focal point.
   - Emerge parallel to the principal axis.
3. **Central Rays:**
   - Pass through the optical center without deviation.

VISUAL REPRESENTATION OF IMAGE FORMATION

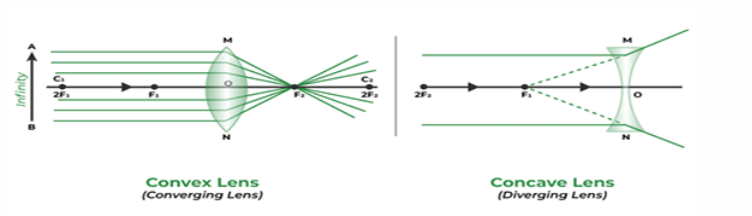
To better understand image formation, let's visualize the basic scenarios:

1. **Pinhole Camera:**
   Light rays pass through a small aperture to form an inverted, scaled down image of the object.



2. **Convex Lens:**

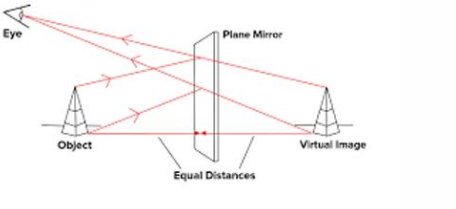
- Focuses parallel rays to a single point (focus) on the opposite side, forming real or virtual images depending on object placement.
- The lens which is thicker at the end than the middle is called the concave lens. It is also called diverging lens as it spreads out the light rays that have been refracted through it. It has the ability to diverge the parallel beam of light.
- A common application of concave lenses is that they are used in optical devices such as binoculars, telescopes, eyeglasses, spy holes in doors, etc.



3. **Concave Lens:**

- Diverges parallel rays outward, creating only virtual images.
- The lens which is thicker at the middle than the end is called a convex lens. It is also called a converging lens as it converges the parallel beam of light into a point.
- A common application of convex lenses is that they are used in optical devices such as microscopes, telescopes, eyeglasses, magnifying glasses, etc.

4. **Plane Mirror:**
   Reflects light rays symmetrically to produce a virtual, upright, and same-sized image.



PIN HOLE CAMERA

What is a Pinhole Camera?

- A pinhole camera is a device with a small aperture (the pinhole) that allows light rays to pass through and form an image on a surface (screen or film) behind the aperture.
- It operates on the principles of rectilinear propagation of light, meaning light travels in straight lines.
- The image formed is inverted and scaled relative to the object.

Mathematical Model of Pinhole Image Formation

- The relationship between the object size, image size, object distance, and image distance is described by similar triangles:

$$\frac{h_i}{h_o} = \frac{d_i}{d_o}$$

- ho: Height of the object.

- hi: Height of the image.
- do: Distance of the object from the pinhole.
- di: Distance of the screen (image plane) from the pinhole.
- Key Observations: • The image height hi can be calculated as:

$$h_i = h_o \cdot \frac{d_i}{d_o}$$

Features of Pinhole Image

- **Sharpness**:
  - A smaller pinhole creates a sharper image but reduces brightness.
  - A larger pinhole increases brightness but causes blurring due to overlapping light rays.
- **Brightness**:
  - Directly proportional to the size of the aperture (pinhole diameter).
- **Distortion-Free:**
  - Unlike lenses or mirrors, a pinhole does not introduce distortion or aberrations.

Problem: **An object 3 meters tall is placed 6 meters away from a pinhole. The screen is located 2 meters behind the pinhole. Find the height of the image and describe its orientation.**

Solution: Using the formula:

$$\frac{h_i}{h_o} = \frac{d_i}{d_o}$$

$$\frac{h_i}{3} = \frac{2}{6}$$

$$h_i = 3 \cdot \frac{2}{6} = 1 \text{ meter}$$

Image height: hi=1 meter
Image orientation: Inverted.

Advantages of the Pinhole Model

1. **Simple Construction**: Requires only a lightproof box, a small aperture, and a screen or photographic film.
2. **No Need for Lenses or Mirrors:** Ideal for studying the fundamentals of optics.
3. **No Optical Distortion:** Produces distortion-free images, unlike lenses, which can introduce chromatic or spherical aberrations.

Limitations

1. **Dim Images:** The small aperture limits the amount of light entering, making the image faint.
2. **Resolution and Sharpness:** Images may blur if the pinhole size is not optimal.
3. **Inverted Image:** The image is always inverted, which may not be practical for some applications.

Applications of Pinhole Cameras

1. **Photography**: Used in pinhole photography to create artistic effects.
2. **Optical Studies**: Demonstrates basic principles of light propagation and image formation.
3. **Astronomy**: Helps in viewing solar eclipses safely by projecting the sun's image onto a surface.
4. **Surveillance**: Can be used as a simple, low-tech surveillance device.

CONVEX LENS IMAGE FORMATION

- **Structure of a Convex Lens**:

- A convex lens is thicker at the center and thinner at the edges.
- It refracts incoming parallel light rays to converge at a single point called the focal point.

Key Terms:

1. **Principal Axis**: A straight line passing through the centers of curvature of the lens surfaces.
2. **Focal Point (F)**: The point where parallel rays converge after refraction.
3. **Focal Length (f)**: The distance between the lens and the focal point.
4. **Optical Center (O)**: The midpoint of the lens where the principal axis passes through.

Principles of Image Formation

- The image formation by a convex lens is governed by the laws of refraction:
  i. A ray parallel to the principal axis passes through the focal point after refraction.
  ii. A ray passing through the focal point becomes parallel to the principal axis after refraction.
  iii. A ray passing through the optical center of the lens does not bend (travels straight).

Types of Images Formed by a Convex Lens

- The properties of the image (real/virtual, upright/inverted, magnified/reduced) are determined by the lens formula:

Lens Formula: 1/f = 1/v – 1/u
Magnification Formula: M = hi/ho = -v/u

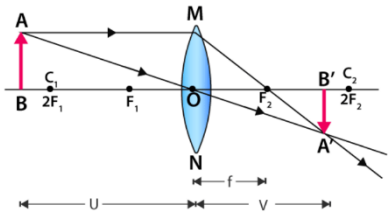
Image Formation for Different Object Positions:

Object Position	Image Position	Image Nature	Image Size
Beyond 2F	Between F and 2F	Real, inverted	Reduced
At 2F	At 2F	Real, inverted	Same size
Between F and 2F	Beyond 2F	Real, inverted	Enlarged
At F	At infinity	Real, inverted	Highly enlarged
Between F and lens	On the same side as the object	Virtual, upright	Enlarged

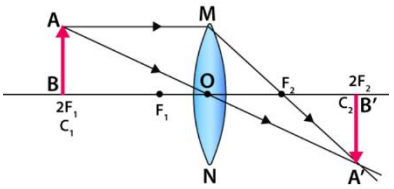
---

Ray Diagrams for Image Formation

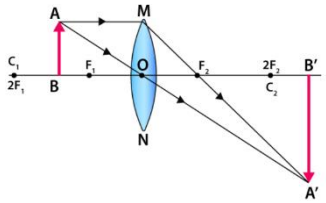
1. **Case 1: Object Beyond 2F**
   - Rays:
     - Parallel ray refracted through F.
     - Ray passing through O goes straight.
   - Image Properties: Position: Between F and 2F, Nature: Real, inverted, Size: Smaller (reduced).



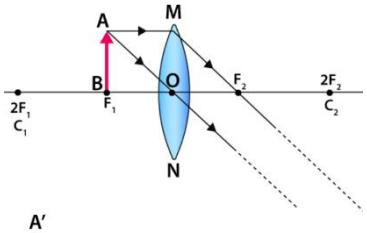
2. **Case 2: Object at 2F**
   - Rays:
     - Parallel ray refracted through F.
     - Ray passing through O goes straight.
   - Image Properties: Position: At 2F, Nature: Real, inverted, Size: Same as the object.



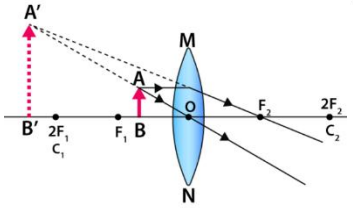
3. **Case 3: Object Between F and 2F**
   - Rays:
     - Parallel ray refracted through F.
     - Ray passing through O goes straight.
   - Image Properties: Position: Beyond 2F, Nature: Real, inverted, Size: Larger (magnified).



4. **Case 4: Object at F**
    - Rays:
        - Parallel ray refracted through F.
        - Ray passing through O goes straight.
    - Image Properties: Position: At infinity, Nature: Real, inverted, Size: Infinitely large.



5. **Case 5: Object Between F and Lens**
    - Rays:
        - Parallel ray appears to diverge from F.
        - Ray passing through O goes straight.
    - Image Properties: Position: Same side as the object, Nature: Virtual, upright, Size: Larger (magnified).



Example

- ***Problem*: An object is placed 15 cm from a convex lens with a focal length of 10 cm. Find the position and nature of the image.**
- **Solution**: Using the lens formula, the image position is 6 cm on the opposite side of the lens, and the nature is real and inverted.

$$\frac{1}{f} = \frac{1}{v} - \frac{1}{u}$$

$$\frac{1}{10} = \frac{1}{v} - \frac{1}{-15}$$

$$\frac{1}{v} = \frac{1}{10} + \frac{1}{15}$$

$$\frac{1}{v} = \frac{3}{30} + \frac{2}{30} = \frac{5}{30}$$

$$v = 6\,\text{cm}$$

Size: Reduced (magnification $M = \frac{-v}{u} = \frac{-6}{-15} = 0.4$).

Applications of Convex Lenses

1. **Cameras**: Focus light to form sharp images on film or sensors.
2. **Magnifying Glass**: Produces a magnified virtual image.
3. **Microscopes and Telescopes**: Forms enlarged images for detailed observation.
4. **Human Eye**: The lens in the eye focuses light on the retina to form images.

CONCAVE LENS IMAGE FORMATION

- **Structure of a Concave Lens**:
    - A concave lens bends incoming light rays outward (diverges them).
    - When parallel rays strike the lens, they appear to diverge from a focal point on the object's side of the lens.

Image Formation for a Concave Lens

- A concave lens always produces the same type of image, irrespective of the object's position.

Image Characteristics:

- **Nature**: Virtual (cannot be projected onto a screen).
- **Orientation**: Upright (same direction as the object).
- **Size**: Diminished (smaller than the object).
- **Position**: Between the lens and the focal point on the same side as the object.

Ray Diagram for a Concave Lens

Steps to Draw a Ray Diagram:

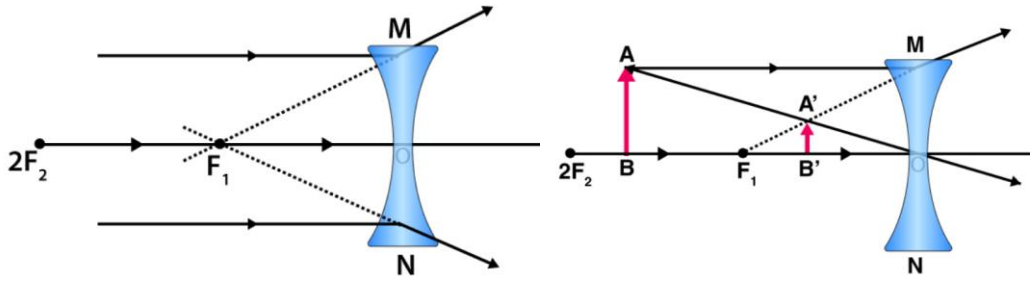
1. **Ray Parallel to the Principal Axis**:
    - A ray parallel to the principal axis diverges after passing through the lens. When extended backward, it appears to come from the focal point F.
2. **Ray Passing Through the Optical Center**:
    - A ray passing through the optical center of the lens travels in a straight line without bending.
3. **Ray Directed Toward the Focal Point**:

- A ray directed toward the focal point emerges parallel to the principal axis after refraction.

Result:

- The rays appear to diverge, and their backward extensions meet at a point. This point is the location of the virtual image.

Object Position	Image Position	Image Nature	Image Size
Beyond 2F	Between F and lens	Virtual, upright	Enlarged
At 2F	Between F and lens	Virtual, upright	Same size
Between F and 2F	On the same side as the object	Virtual, upright	Enlarged
At F	On the same side as the object	Virtual, upright	Highly enlarged
Between F and lens	On the same side as the object	Virtual, upright	Enlarged



Example

- ***Problem*: An object is placed 15 cm from a concave lens with a focal length of -10 cm. Find the position and magnification of the image.**
- **Solution**: Using the lens formula, the image position is 30 cm on the same side

$$\frac{1}{f} = \frac{1}{v} - \frac{1}{u}$$
$$\frac{1}{-10} = \frac{1}{v} - \frac{1}{-15}$$
$$\frac{1}{v} = \frac{1}{-10} + \frac{1}{15}$$
$$\frac{1}{v} = \frac{-3}{30} + \frac{2}{30} = \frac{-1}{30}$$
$$v = -30\,\text{cm}$$

$$M = \frac{v}{u} = \frac{-30}{-15} = 0.2$$

as the object (negative sign indicates the image is virtual). The magnification indicates that the image is virtual, upright, and reduced to 20% of the object's size.

PLANE MIRROR IMAGE FORMATION

- **Principles of Reflection**:
  - The image formation in a plane mirror is governed by the laws of reflection:
    a. The angle of incidence (i) is equal to the angle of reflection (r).
    b. The incident ray, reflected ray, and the normal to the surface lie in the same plane.

Characteristics of Image Formed by a Plane Mirror

- **Nature**: Virtual (the image cannot be projected onto a screen because it appears to exist behind the mirror).
- **Orientation**: Erect (the image is oriented in the same direction as the object).
- **Size**: Same size (the image has the same dimensions as the object).
- **Laterally Inverted**: The left and right sides of the image are reversed.
- **Same Distance from the Mirror**: The image appears at the same distance behind the mirror as the object is in front of it.

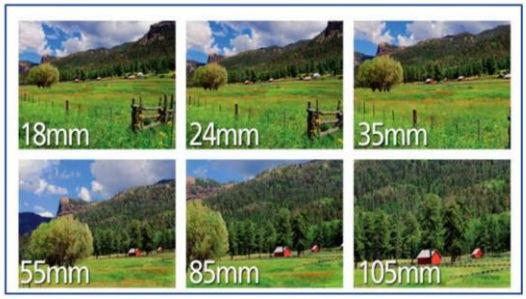
Ray Diagram for Image Formation

- **Problem**: An object is placed 5 cm in front of a plane mirror. Describe the image formed.

- **Solution**:
  - **Nature**: Virtual and upright.
  - **Position**: The image will appear 5 cm behind the mirror (equal to the object's distance in front of it).
  - **Size**: The image will be the same size as the object.
  - **Orientation**: The image will be laterally inverted (left and right swapped).

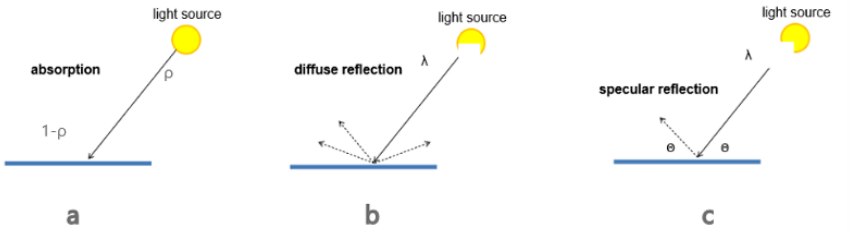
Applications of Plane Mirrors

1. **Household Mirrors**: Used in dressing mirrors and bathroom mirrors for daily use.
2. **Periscopes**: Used in submarines to observe surroundings above the water surface.
3. **Kaleidoscopes**: Plane mirrors reflect light to create beautiful symmetric patterns.
4. **Scientific Instruments**: Used in optical devices like galvanometers for accurate light reflection.
5. **Architectural Decoration**: Creates illusions of larger spaces due to reflections.



Reflection and Scattering

- Images cannot exist without light. Light sources can be a point or an area light source. When the light hits a surface, three major reactions might occur:

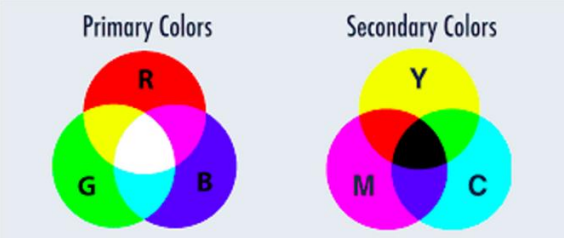


  - Some light is absorbed, depending on the factor called albedo (ρ). A low ρ of the surface means more light will get absorbed.
  - Some light gets reflected diffusively, which is independent of viewing direction, following Lambert's cosine law (the amount of reflected light is proportional to cos(θ)). E.g., cloth, brick.
  - Some light is reflected specularly, which depends on the viewing direction. E.g., mirror.

WHAT IS THE PRIMARY COLOR?

- The primary color is the one which absorbs a primary color and reflects the other two.
- Mixing three primary colors or a secondary color with its opposite primary color produces white light as shown in Fig.
- The primary colors can be added together to produce the secondary colors – magenta, cyan and yellow as you can see in the figure below

- The primary color is the one which absorbs a primary color and reflects the other two



CHARACTERISTICS OF COLOR

There must be some factor or characteristic based on which we can distinguish the color.

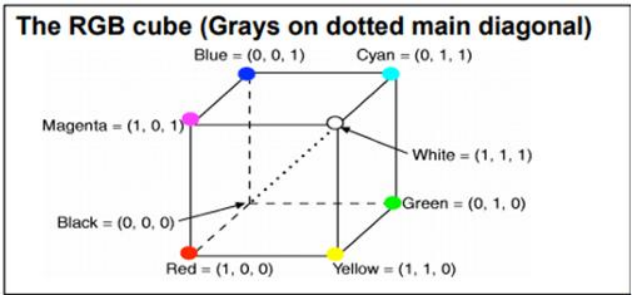
- **Brightness**: This expresses the intensity or amount of the output.
- **Hue**:
  - As we know visible light has energy which is spread over a band of wavelengths.
  - Hue represents the dominant wavelength in a mixture of all light waves.
  - It represents the dominant color as viewed by an observer.
- **Saturation**
  - It shows the amount of white light mixed with the hue.
  - Hue and Saturation, together are called Chromaticity. Therefore, a color may be characterized by its chromaticity and intensity

INTRODUCTION TO COLOR SPACES

- Color spaces are the mathematical representation of a set of colors. There are many color models.
- Some of them are RGB, CMYK, YIQ, HSV, and HLS, etc.
- These color spaces are directly related to saturation and brightness.
- All of these color spaces can be derived using RGB information using devices such as cameras and scanners

RGB COLOR SPACE

- RGBstands for Red, Green, and Blue.
- This color space is widely used in computer graphics. RGB are the main colors from which many colors can be made.
- RGB can be represented in the 3-dimensional form:



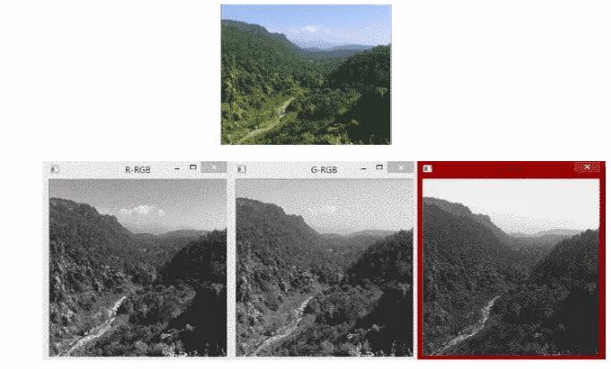
- Below table is 100% RGB color bar contains values for 100% amplitude, 100% saturated, and for video test signal.

	Nominal Range	White	Yellow	Cyan	Green	Magenta	Red	Blue	Black
R	0 to 255	255	255	0	0	255	255	0	0
G	0 to 255	255	255	255	255	0	0	0	0
B	0 to 255	255	0	255	0	255	0	255	0

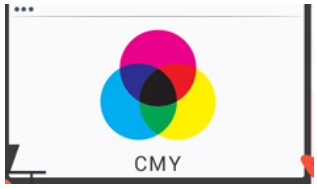
- Colour combination:
  - Green(255) + Red(255) = Yellow

- Green(255) + Blue(255) = Cyan
- Red(255) + Blue(255) = Magenta
- Red(255) + Green(255) + Blue(255) = White

```
import cv2

# Load the image
image = cv2.imread('C:/Users/hp/images/1.jpg')

# Resize the image to 300x300 pixels
image = cv2.resize(image, (300, 300))

# Split the image into its B, G, R components
b = image[:, :, 0:1]  # Blue channel
g = image[:, :, 1:2]  # Green channel
r = image[:, :, 2:3]  # Red channel

# Display the individual color channels
cv2.imshow('B - Blue Channel', b)
cv2.imshow('G - Green Channel', g)
cv2.imshow('R - Red Channel', r)

# Wait for a key press and close the image windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```



CMY MODEL



- This model contains the secondary colors. In this model, any secondary color when passed through white light will not reflect the color from which a combination of colors is made.
- For example- when cyan is illuminated with white light, no red light will be reflected from the surface which means that the cyan subtracts the red light from the reflected white light (which itself is composed of red, green and white light).
- The formula given in equation 1 is for inter-conversion of RGB and CMY models

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

CMY TO RGB CONVERSION

The CMY (Cyan, Magenta, Yellow) color model is a subtractive color model used in color printing. To convert from CMY to RGB (Red, Green, Blue), we can use the following formulas:

CMY to RGB Formula:

- R=255−C
- G=255−M
- B=255−Y

Here:

- C is the cyan component.
- M is the magenta component.
- Y is the yellow component.

RGB values are typically in the range of [0, 255].

RGB TO CMY CONVERSION

The RGB (Red, Green, Blue) color model is an additive color model, commonly used for screens and digital imaging. To convert from RGB to CMY, the following formulas are used:

RGB to CMY Formula:

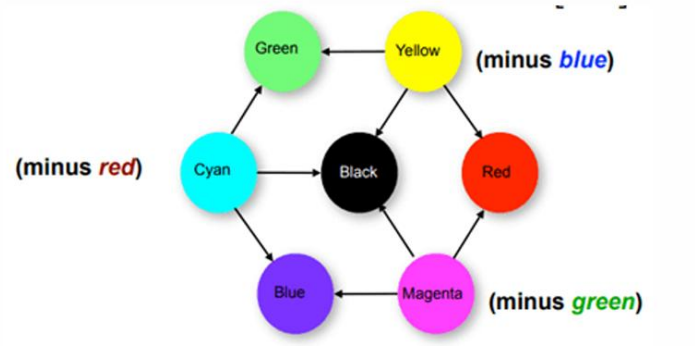
$$C = 1 - \frac{R}{255}$$
$$M = 1 - \frac{G}{255}$$
$$Y = 1 - \frac{B}{255}$$

- R is the red component.
- G is the green component.
- B is the blue component.

CMY values will be in the range of [0, 1] or you can scale them to [0, 255] if needed

CMYK COLOR MODEL

- CMYK colour model is widely used in printers. It stands for Cyan, Magenta, Yellow and Black (key).
- It is a subtractive colour model. 0 represents the primary colour and 1 represents the lightest colour.
- In this model, point (1, 1, 1) represents black, and (0,0,0) represents white.
- It is a subtractive model thus the value is subtracted from 1 to vary from least intense to a most intense colour value



CMYK TO RGB CONVERSION

The CMYK (Cyan, Magenta, Yellow, Key/Black) color model is a subtractive color model often used in color printing. To convert from CMYK to RGB, the following formulas are used:

CMYK to RGB Formula:

- $R=255\times(1-C)\times(1-K)$
- $G=255\times(1-M)\times(1-K)$
- $B=255\times(1-Y)\times(1-K)$

Where:

- C is the cyan component (range 0 to 1).
- M is the magenta component (range 0 to 1).
- Y is the yellow component (range 0 to 1).
- K is the black (key) component (range 0 to 1).
- R, G, B are the resulting red, green, and blue components (range 0 to 255)

RGB to CMYK

1. **Normalize the RGB values:**

$$R' = \frac{R}{255}, \quad G' = \frac{G}{255}, \quad B' = \frac{B}{255}$$

Where R,G,B are the RGB values in the range of 0 to 255, and R′,G′,B′ are the normalized RGB values in the range of 0 to 1.

2. **Calculate the black (K) component:**

$$K = 1 - \max(R', G', B')$$

3. **Calculate the cyan, magenta, and yellow components:**

$$C = \frac{1 - R' - K}{1 - K}$$

$$M = \frac{1 - G' - K}{1 - K}$$

$$Y = \frac{1 - B' - K}{1 - K}$$

4. If K=1 (i.e., the RGB values are the same and equal to 255, resulting in pure black), then:C=0,M=0,Y=0
5. **Convert the CMYK components back to the range [0, 1]:**
   The final CMYK values are in the range [0, 1], representing the intensity of the colors

DIFFERENCE BETWEEN CMY AND CMYK

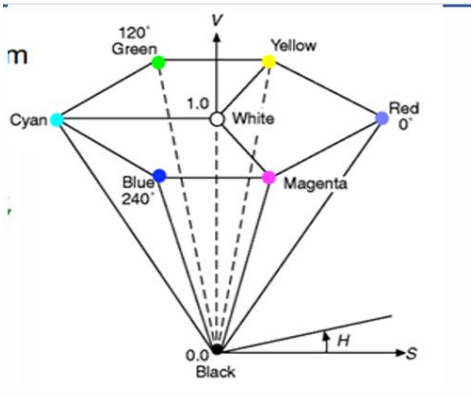
The CMY and CMYK color models are both subtractive color models used in printing and image processing, but they differ in their structure and purpose. Here is a breakdown of the differences:

- **CMY**: Refers to the Cyan (C), Magenta (M), and Yellow (Y) color components. It is a subtractive model where colors are created by subtracting varying percentages of red, green, and blue from white light.
- **CMYK**: Extends the CMY model by adding a Key (K) component, which stands for black ink. The black component is added to improve contrast, reduce ink usage, and produce true black tones.
- **Components**
- **CMY**:
  a. Cyan (C): Subtracts Red
  b. Magenta (M): Subtracts Green
  c. Yellow (Y): Subtracts Blue
- **CMYK**: Includes Cyan (C), Magenta (M), Yellow (Y), and Black (K).
- **Black Representation**
- **CMY**:
  a. Black is achieved by combining 100% of Cyan, Magenta, and Yellow:
  b. C+M+Y=Black
  c. However, this produces a muddy or dark brownish black rather than a pure black.
- **CMYK**: A separate Black (K) channel is added to produce a pure black color, enhancing contrast and detail. This is more efficient and cost-effective in printing.
- **Purpose**
- **CMY**: Primarily used in theoretical color models or digital applications where no actual ink is involved.
- **CMYK**: Used in printing applications like magazines, posters, and packaging, where accurate colors and pure black are essential.
- **Ink Usage**
- **CMY**: Requires a higher quantity of ink to achieve dark tones or black, which can lead to smudging or ink wastage.
- **CMYK**: Reduces ink usage by using a dedicated black (K) ink. It provides better quality prints with sharp black text and darker shades.
- **Applications**
- **CMY**: Used in digital displays, theoretical models, and image processing calculations.
- **CMYK**: Standard model for color printing in physical media like newspapers, books, and brochures

Feature	CMY	CMYK
Black Representation	Combination of C, M, and Y	Separate K channel
Output Quality	Muddy black in dark areas	Clean and pure black output
Ink Usage	High ink consumption	Reduced ink usage
Use Case	Digital color models	Physical color printing

HSV COLOR MODEL

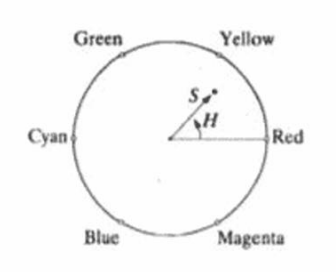
- HSV stands for Hue, Saturation, and Value (brightness).
- It is a hexcone subset of the cylindrical coordinate system.
- The human eye can see 128 different hues, 130 different saturations and number values between 16 (blue) and 23 (yellow).



- Red colour falls between 0 and 60 degrees in the HSV cone.
- Yellow colour falls between 61 and 120 degrees in the HSV cone.
- Green colour falls between 121 and 180 degrees in the HSV cone.
- Cyan colour falls between 181 and 240 degrees in the HSV cone.
- Blue colour falls between 241 and 300 degrees in the HSV cone.
- Magenta colour falls between 301 and 360 degrees in the HSV cone.

HUE

- Hue is a color component that describes a pure color (pure yellow, orange or red)
- Saturation component represents the measure of the degree to which color is mixed with white color.
- 0 degree – Red
- 120 degree – Green
- 240 degree – Blue
- 60 degree – Yellow
- 300 degree – Magenta
- Intensity
  - Range is [0, 1]
  - 0 means white
  - 1 means black



- The formula for conversion of RGB to HSI is quite complicated as compared to other color models.

RGB TO HSV CONVERSION

Steps for RGB to HSV Conversion

1.Normalize RGB values:

$$R' = \frac{R}{255}, \quad G' = \frac{G}{255}, \quad B' = \frac{B}{255}$$

2. Find the maximum and minimum values of the normalized RGB values:

$$C_{\max} = \max(R', G', B'), \quad C_{\min} = \min(R', G', B')$$

$$\Delta = C_{\max} - C_{\min}$$

3.Calculate the Hue (H):

If C=0, then H=0 (undefined, achromatic color).

$$H = \begin{cases} 60^\circ \times \frac{(G'-B')}{C} + 360^\circ \pmod{360}, & \text{if } C_{\max} = R' \\ 60^\circ \times \frac{(B'-R')}{C} + 120^\circ, & \text{if } C_{\max} = G' \\ 60^\circ \times \frac{(R'-G')}{C} + 240^\circ, & \text{if } C_{\max} = B' \end{cases}$$

4. Calculate the Saturation (S): If Cmax=0, then S=0, otherwise:

$$S = \frac{\Delta}{C_{\max}}$$

5. Calculate the Value (V): V=Cmax

HSV TO RGB CONVERSION

Steps for HSV to RGB Conversion

1.Normalize the Hue (H), Saturation (S), and Value (V) to [0, 1]: H∈[0,360], S∈[0,1] , V∈[0,1] .

2.Calculate the chroma (C): C=V×S

3.Calculate X: X=C×(1−|(H/60)mod 2−1|)

4.Calculate the RGB components (R', G', B'):

- If 0≤H<60, then (R′,G′,B′)=(C,X,0)
- If 60≤H<120, then (R′,G′,B′)=(X,C,0)
- If 120≤H<180, then (R′,G′,B′)=(0,C,X)
- If 180≤H<240, then (R′,G′,B′)=(0,X,C)
- If 240≤H<300, then (R′,G′,B′)=(X,0,C)
- If 300≤H<360, then (R′,G′,B′)=(C,0,X)

5.Calculate the RGB values by adjusting with the value of m:

m=V−C

R=(R′+m)×255,G=(G′+m)×255,B=(B′+m)×255

YIQ

• YIQ is the most widely colour model used in Television broadcasting.

• Y stands for luminance part and IQ stands for chrominance part. In the black and white television, only the luminance part (Y) was broadcast.

• The y value is similar to the grayscale part. The colour information is represented by the IQ part.

• There exist a formula to convert RGB into YIQ and vice-versa.

From RGB to YIQ

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \approx \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.5959 & -0.2746 & -0.3213 \\ 0.2115 & -0.5227 & 0.3112 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

From YIQ to RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.619 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

YCbCr COLOR SPACE

- **Definition**:
  - YCbCr is a color space used in digital photography and video that separates brightness from color. It is a transformation of the RGB color space that separates the luminance (brightness) information from the chrominance (color) information.
- **Purpose**:
  - The primary purpose of this separation is to allow efficient compression by reducing the chrominance data (which the human eye is less sensitive to) more than the luminance data.

Components of YCbCr:

1. **Y (Luminance)**:
   - Represents the brightness or intensity of the color.
   - It is the weighted sum of the RGB values and determines the grayscale version of an image.
   - **Formula**: $Y=0.299R+0.587G+0.114B$
   - The Y channel is responsible for most of the perceived image quality and can be used for black-and-white versions of an image.
2. **Cb (Blue-difference Chrominance)**:
   - Represents the difference between the blue component of the image and the luminance channel.
   - It indicates the amount of blue in the image.
   - **Formula**: $Cb=(B-Y)/2$
   - The Cb channel varies between -128 and 127, and its range can be offset by 128 for better representation in digital formats.
3. **Cr (Red-difference Chrominance)**:
   - Represents the difference between the red component of the image and the luminance channel.
   - It indicates the amount of red in the image.
   - **Formula**: $Cr=(R-Y)/2$
   - Like Cb, the Cr channel typically ranges between -128 and 127, and the value can be shifted by 128.

YCBCR TRANSFORMATION

To convert from RGB to YCbCr and vice versa, specific formulas are used.

RGB to YCbCr Conversion:

For the conversion from RGB (Red, Green, Blue) to YCbCr:

- $Y=0.299R+0.587G+0.114B$
- $Cb=-0.169R-0.331G+0.500B$
- $Cr=0.500R-0.419G-0.081B$

YCbCr to RGB Conversion:

To convert back from YCbCr to RGB:

- $R=Y+1.402(Cr-128)$
- $G=Y-0.344136(Cb-128)-0.714136(Cr-128)$
- $B=Y+1.772(Cb-128)$

APPLICATIONS OF YCBCR

1. **Video Compression**:
   - YCbCr is commonly used in video encoding and compression standards (MPEG, H.264, etc.), where chroma subsampling is applied.
2. **Digital Television**:
   - Broadcast standards such as PAL, NTSC, and SECAM use YCbCr as the standard color space for encoding and transmitting video signals.
3. **Image Processing**:
   - YCbCr is used for image processing tasks like noise reduction, edge detection, and other operations that require separation of brightness and color.
4. **JPEG Compression**:
   - JPEG uses YCbCr with chroma subsampling for compressing images, resulting in efficient storage and transmission of digital images.

IMAGE REPRESENTATION

- **Definition**:
  - In digital image processing, image representation refers to how images are stored, processed, and visualized in digital systems.
- **Purpose**:
  - Representation involves capturing and encoding the visual information of an image into a structured, numerical format that computers can process.

- **Types**:
  - **Spatial Domain Representation**:
    - Direct pixel values of the image.
  - **Frequency Domain Representation**:
    - Representation using transformations (e.g., Fourier Transform).

COMPONENTS OF IMAGE REPRESENTATION

1. **Pixel**:
   - The smallest unit of an image.
   - Represents the intensity (brightness) or color at a specific point in the image.
   - Pixels are arranged in a grid (rows and columns).
2. **Resolution**:
   - **Spatial Resolution**: Number of pixels in an image (e.g., 1920x1080).
   - **Gray-Level Resolution**: Number of intensity levels a pixel can represent (e.g., 8-bit images have 256 intensity levels).
3. **Bit Depth**:
   - Number of bits used to represent each pixel.
   - **Common Bit Depths**:
     - 1-bit: Black and white (binary images).
     - 8-bit: Grayscale (256 shades).
     - 24-bit: RGB color images (16.7 million colors).
4. **Image Size**:
   - Determined by the number of pixels and the bit depth.
   - **Storage Requirement**: Size=Width×Height×Bit Depth

TYPES OF IMAGE REPRESENTATION

1. **Binary Images**:
   - Pixels have values 0 (black) or 1 (white).
   - Used in applications like document scanning and object detection.
2. **Grayscale Images**:
   - Pixels represent intensity levels, typically ranging from 0 (black) to 255 (white) for 8-bit images.
   - Used in medical imaging, pattern recognition, etc.
3. **Color Images**:
   - Represented using multiple channels, typically:
     - **RGB (Red, Green, Blue)**: Additive color model.
     - **CMY/CMYK (Cyan, Magenta, Yellow, Black)**: Subtractive color model used in printing.
   - Each channel has its own intensity values.
4. **Indexed Images**:
   - Use a color palette (colormap) where pixel values are indices in the palette.
   - Efficient for storage when the image has limited colors.
5. **Vector Images**:
   - Use mathematical descriptions (e.g., lines, curves) rather than pixels.
   - Common in graphic design and scalable illustrations.

IMAGE REPRESENTATION IN DIFFERENT DOMAINS

- **Spatial Domain**:
  - Image is represented as a matrix of pixel values.
  - **Example**: A 3x3 grayscale image:

```
[[100, 150, 200],
 [50,  75,  125],
 [0,   25,  255]]
```

- **Frequency Domain**:
  - Transform image using techniques like Fourier Transform or Discrete Cosine Transform (DCT).
  - Used in image compression and filtering.

- **Feature-Based Representation**:
  - Images are represented using extracted features such as edges, corners, textures, or objects.
  - **Example**: Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT).

MATHEMATICAL REPRESENTATION OF IMAGES

- An image can be represented as a 2D function: $f(x,y)$ Where:
  - x,y: Spatial coordinates.
  - $f(x,y)$: Intensity or color value at (x,y).
- For color images: $f(x,y)=[R(x,y),G(x,y),B(x,y)]$ Where R,G,B represent intensity values in the red, green, and blue channels.

REPRESENTING IMAGES IN COMPUTER MEMORY

- **Grayscale Image**:
  - Stored as a 2D array or matrix of intensity values.
  - **Example in Python (using NumPy)**:

```
import numpy as np
grayscale_image = np.array([[255, 128, 64],
                            [0, 192, 128],
                            [64, 32, 16]])
```

- **Color Image**:
  - Stored as a 3D array (height, width, channels).
  - **Example**:

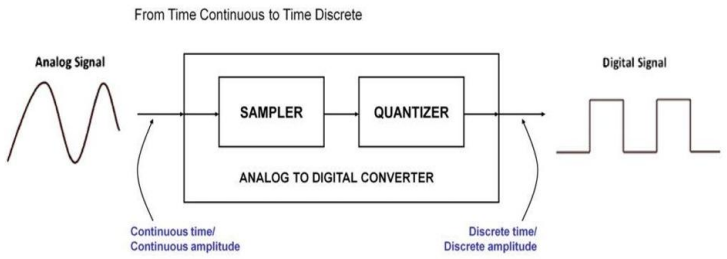
```
color_image = np.array([[[255, 0, 0], [0, 255, 0], [0, 0, 255]],
                        [[128, 128, 0], [0, 128, 128], [128, 0, 128]]])
```

DIGITIZATION

- **Definition**: Digitization in Image Processing is the process of converting a continuous-tone image (analog image) into a digital image represented by discrete numerical values (pixels). It forms the foundation of modern digital image processing systems.
- **Process**: Digitization involves converting an analog image into a digital format by sampling its intensity values at discrete spatial locations.

Steps:

1. **Sampling**: Discretizing the spatial coordinates (x, y) of the image.
2. **Quantization**: Assigning discrete intensity (brightness) levels to the sampled points.



IMPORTANCE OF DIGITIZATION

- **Enables Processing**: Allows images to be manipulated, analyzed, and stored using digital systems.
- **Compression**: Makes it easier to compress images for efficient storage and transmission.
- **Automation**: Facilitates automated tasks like object detection, medical diagnosis, and industrial inspection.
- **Integration**: Digital images can be easily integrated into machine learning, computer vision, and other technologies.

STEPS IN DIGITIZATION

1. **Image Acquisition**:

- Capture the analog image using a device like a camera or scanner.
- Analog signals (light intensity) are recorded over a continuous surface.

2. **Sampling**:
   - Breaks the continuous image into a grid of pixels.
   - Determines spatial resolution, i.e., the number of pixels in the grid (e.g., 1920x1080 for Full HD).
3. **Quantization**:
   - Converts the intensity values of each sampled point into discrete levels.
   - The number of quantization levels determines the image's bit depth (e.g., 8-bit images have 256 intensity levels per channel).

PARAMETERS IN DIGITIZATION

- **Spatial Resolution**:
  - Refers to the number of pixels used to represent the image.
  - Higher resolution results in better detail but requires more storage.
- **Gray-Level Resolution**:
  - Refers to the number of intensity levels per pixel.
  - Higher bit depth provides more accurate representation of intensity variations.

EFFECTS OF DIGITIZATION

- **Low Sampling Rate**:
  - Causes aliasing (loss of detail or misrepresentation of high-frequency components).
- **Low Quantization Levels**:
  - Leads to quantization noise or banding effects in the image.

KEY CHALLENGES

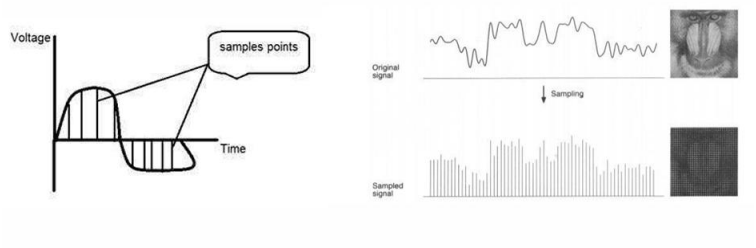
- **Trade-Off Between Quality and Storage**:
  - Higher resolution and bit depth improve quality but increase storage and processing requirements.
- **Aliasing**:
  - Insufficient sampling can distort the image, requiring anti-aliasing filters.
- **Noise**:
  - Introduced during digitization due to sensor imperfections or environmental factors.

TOOLS FOR DIGITIZATION

- **Hardware**: Digital cameras, scanners, and specialized digitizers.
- **Software**: MATLAB, OpenCV, and Python libraries like PIL (Pillow) or NumPy for processing digitized images.

SAMPLING

- **Definition**: Image sampling is the process of converting a continuous image (analog) into a discrete image (digital) by selecting specific points from the continuous image. This involves measuring the image at regular intervals and recording the intensity (brightness) values at those points.



How Image Sampling Works:

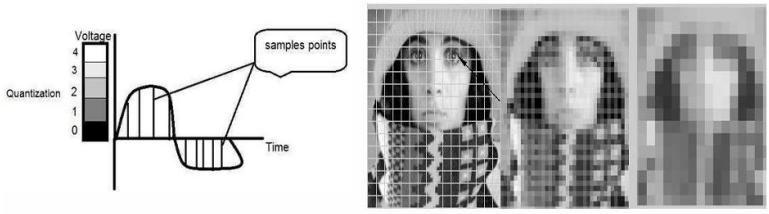
- **Grid Overlay**: A grid is placed over the continuous image, dividing it into small, regular sections.
- **Pixel Selection**: At each intersection of the grid lines, a sample point (pixel) is chosen.

Examples of Sampling:

- **High Sampling Rate**: A digital camera with a high megapixel count captures more details because it samples the image at more points.
- **Low Sampling Rate**: An old VGA camera with a lower resolution captures less detail because it samples the image at fewer points.

QUANTIZATION

- **Definition**: Image quantization is the process of converting the continuous range of pixel values (intensities) into a limited set of discrete values. This step follows sampling and reduces the precision of the sampled values to a manageable level for digital representation.



How Image Quantization Works:

- **Value Range Definition**: The continuous range of pixel values is divided into a finite number of intervals or levels.
- **Mapping Intensities**: Each sampled pixel intensity is mapped to the nearest interval value.
- **Assigning Discrete Values**: The original continuous intensity values are replaced by the discrete values corresponding to the intervals.

Examples of Quantization:

- **High Quantization Levels**: An image with 256 levels (8 bits per pixel) can represent shades of gray more accurately.
- **Low Quantization Levels**: An image with only 4 levels (2 bits per pixel) has much less detail and appears more posterized.

Key Differences Between Image Sampling and Quantization

Feature	Image Sampling	Image Quantization
Definition	Conversion of a continuous image into a discrete set of points by selecting specific pixel positions	Conversion of continuous pixel intensity values into discrete levels
Process Focus	Spatial information (locations of pixels)	Intensity values (brightness or color levels)
Outcome	A grid of pixel values representing spatial resolution	A set of discrete intensity values for each pixel
Resolution Aspect	Affects spatial resolution (detail and clarity of the image)	Affects color/gray level resolution (number of shades or colors)
Determined By	Sampling rate (number of pixels sampled)	Quantization levels (number of intensity levels)
Higher Value Effects	Higher sampling rate captures more spatial detail	More quantization levels represent finer intensity variations
Example	High megapixel count in cameras for detailed images	8-bit color depth for more color variations
Application Impact	Crucial for applications needing high spatial detail like medical imaging	Crucial for applications needing high color fidelity like graphic design
Storage Requirement	Increases with higher sampling rates	Increases with more quantization levels

Typical Values	Measured in pixels per inch (PPI) or dots per inch (DPI)	Measured in bits per pixel (bpp)

ADVANTAGES AND DISADVANTAGES OF IMAGE SAMPLING

Advantages:

1. **Data Reduction**: Converts a continuous signal into a finite set of points, making storage and processing more manageable.
2. **Compatibility**: Sampled images are easily processed by digital systems and algorithms.
3. **Resolution Control**: Allows for control over image resolution by adjusting the sampling rate.

Disadvantages:

1. **Information Loss**: Inevitably loses some information by approximating a continuous signal.
2. **Aliasing**: Can cause distortions and artifacts if the sampling rate is too low.
3. **Computationally Intensive**: High-resolution sampling demands significant computational resources and storage space.

ADVANTAGES AND DISADVANTAGES OF IMAGE QUANTIZATION

Advantages:

1. **Data Compression**: Reduces the amount of data by limiting the number of possible values for each pixel.
2. **Simplified Processing**: Makes image processing operations simpler and faster with fewer distinct values.
3. **Noise Reduction**: Helps reduce the impact of noise by mapping small variations in intensity to the same value.

Disadvantages:

1. **Loss of Detail**: Reduces the range of colors or intensity levels, leading to a loss of fine detail and potential color banding.
2. **Quantization Error**: Introduces differences between the original and quantized values, which can become noticeable.
3. **Reduced Image Quality**: Overly aggressive quantization can significantly degrade image quality, making the image appear blocky or posterized.

MODULE 2

Image Enhancement Techniques

IMAGE ENHANCEMENT TECHNIQUES

- **Definition**: Image enhancement is the process of emphasizing specific details within an image while reducing or removing superfluous elements.
- **Goals**:
  - Removing noise
  - Revealing obscured details
  - Adjusting image levels to highlight particular features
- **Purpose**: Image enhancement methods are techniques used to improve the quality, visibility, or contrast of an image.

Categories of Image Enhancement Techniques

1. **Spatial Domain:**
   - Enhances the image by manipulating individual pixels based on their spatial coordinates at a specific resolution.
2. **Frequency Domain:**
   - Enhances the image by applying a Fourier Transform indirectly to the spatial domain, manipulating pixels in groups.

IMAGE ENHANCEMENT IN SPATIAL DOMAIN

Subcategories:

1. **<u>Point Operations</u> (Intensity Transformations)**:
    - Apply the same transformation to each pixel in a grayscale image based on its original pixel value, independent of its location or neighboring pixels.
2. **<u>Spatial Filters</u> (Mask, Kernel)**:
    - The output value of these operations depends on the values of the function $f(x,y)$ and its neighborhood.

Point Operations

- **Definition**: Point operations are used to change the grayscale range and distribution. The concept is to map every pixel onto a new image with a predefined transformation function.

$$g(x,y)=T(f(x,y))$$

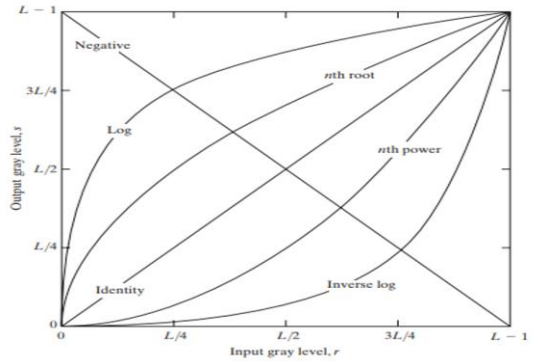
- $g(x,y)$: Output image
- T: Operator of intensity transformation
- $f(x,y)$: Input image

BASIC INTENSITY TRANSFORMATION FUNCTIONS

- The simplest image enhancement method uses a 1 x 1 neighborhood size, which is a point operation. In this case, the output pixel (s) depends only on the input pixel (r):

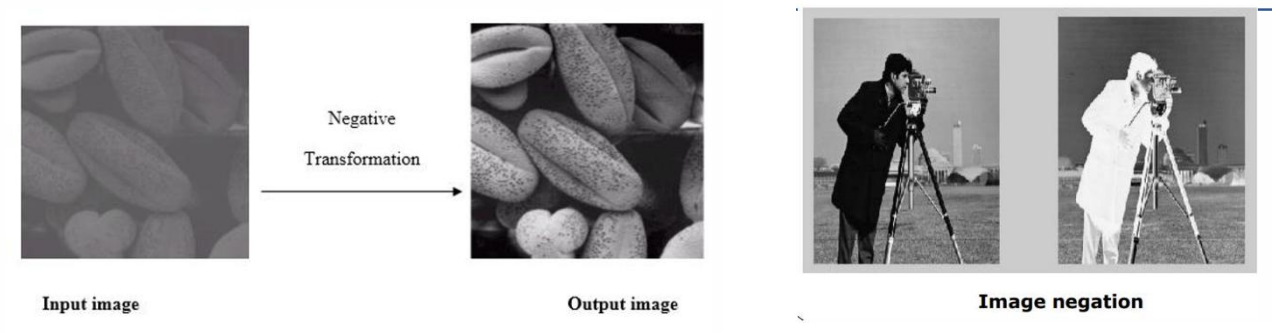
$$s=T(r)$$

- s: Gray level of the output pixel
- r: Gray level of the input pixel
- Different transformation functions work for different scenarios.



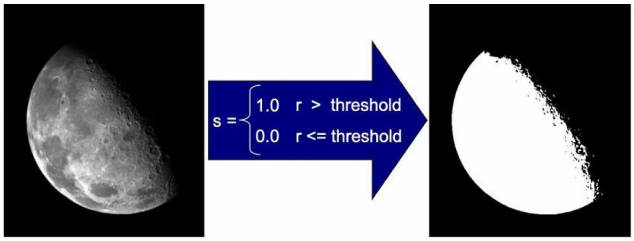
TYPES OF BASIC INTENSITY TRANSFORMATION FUNCTIONS

1. **Linear Transformations**:
    - **Identity Transformation**:
        - The input image is the same as the output image: $s=r$
    - **Negative Transformation**:
        - Defined as: $\boldsymbol{s = L-1-r = 256-1-r = 255-r}$
        - L: Largest gray level in an image.
        - Suitable for enhancing white or gray details in dark areas (e.g., analyzing breast tissue in digital mammograms).

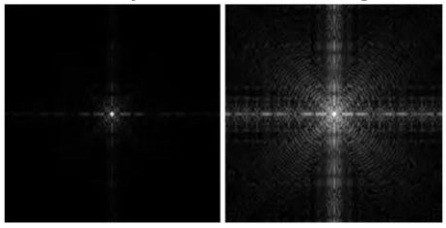


2. **Thresholding**:
    - Useful for segmentation to isolate an object of interest from the background.

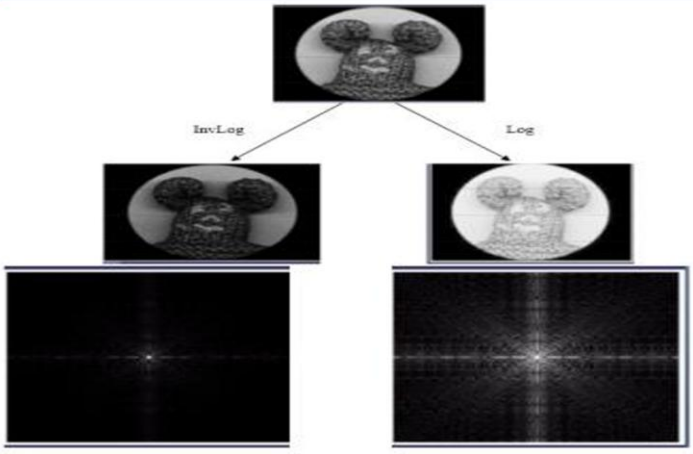
- If thresholding is too low, the image contains higher intensity values more.
- If thresholding is too high, the image contains lower intensity values more.



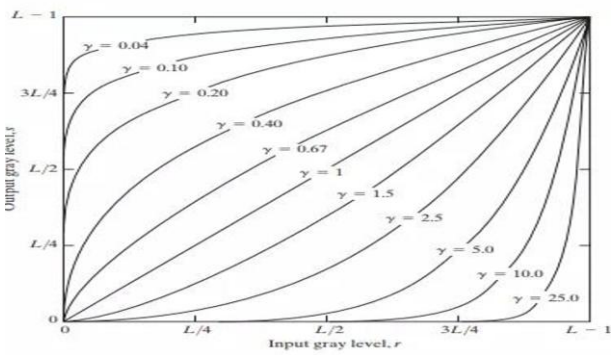
3. **Logarithmic Transformations**:
   - **General Log Transform**:
     - The equation is: $s=c \cdot \log(1+r)$
     - c is a constant; to map from [0,255] to [0,255], **c=255/log(256)**
     - The log transformation works best for dark images, mapping low-intensity values to higher intensity values.



   - **Inverse Log Transform**:
     - Opposite of log transform
     - Expands values of light-level pixels while compressing darker-level values.
     - **s = power(10, r * c)-1**
     - *Note*:
       •s,r: denote the gray level of the input pixel and the output pixel.
       •'c' is a constant; to map from [0,255] to [0,255], **c =LOG(256)/25**

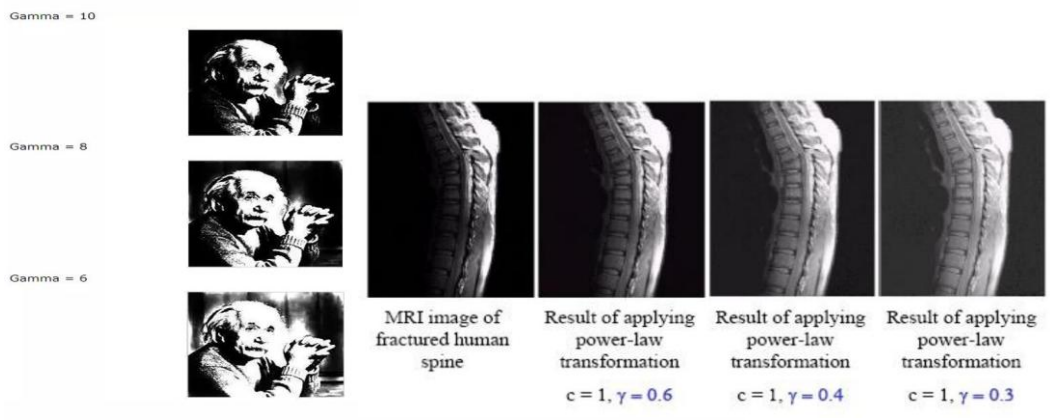


4. **Power-Law (Gamma) Transformation**:
   - A.k.a exponential transformation or power transformation,
   - The gamma transform is a commonly used grayscale non-linear transformation: $s=c \cdot \text{power}(r,\gamma)$
     - s: Gray level of the output pixel
     - r: Gray level of the input pixel
     - c: Constant
     - γ: Gamma coefficient
   - Plots of the equation [formula] for various values of γ (c = 1 in all cases)



Gamma Transformation Characteristics

- Common practice: Convert intensity level from [0, 255] to [0, 1], perform gamma conversion, then restore to original range.
- Comparing to log transformation, gamma transformation can generate a family of possible transformation curves by varying the gamma value.
- Different γ values generate a family of transformation curves:
  - **γ>1:** Enhances contrast of light gray areas.
  - **γ<1:** Enhances contrast of dark gray areas.
  - **γ=1:** Linear transformation; original image is unchanged.



PIECEWISE TRANSFORMATION

- Piecewise transformation is a technique used to modify the intensity values of an image by applying different transformations to specific intensity ranges.
- It is a form of intensity mapping that improves the visual quality of an image or emphasizes certain features.

Key Concept:

- The input intensity range is divided into several sub-ranges.
- Each sub-range is transformed using a different mathematical function or scaling factor.
- The output image is formed by mapping the input pixel values to the new intensity values based on the specified transformations.

Mathematical Representation:

Let $f(i,j)$ represent the intensity of a pixel at (i,j) in the input image, and $g(i,j)$ be the corresponding intensity in the output image. The piecewise transformation is defined as:

$$g(i,j) = \begin{cases} T_1(f(i,j)), & f(i,j) \in [a_1, b_1] \\ T_2(f(i,j)), & f(i,j) \in [a_2, b_2] \\ \vdots & \\ T_n(f(i,j)), & f(i,j) \in [a_n, b_n] \end{cases}$$

Here:

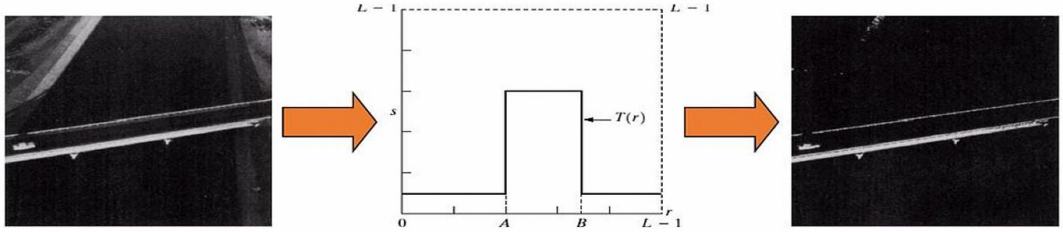
- $[a_k, b_k]$: Intensity range for the k-th transformation.
- T_k: The transformation function applied to the k-th range.

Types of piecewise transformations:

1. Grayscale Threshold Transform (Gray-level Slicing)
2. Contrast Stretching
3. Bit-plane Slicing

PIECEWISE TRANSFORMATION: GRAYSCALE THRESHOLD TRANSFORM (GRAY-LEVEL SLICING)

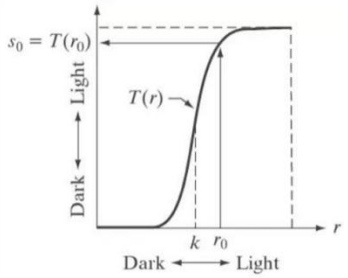
• Grayscale Threshold Transform converts a grayscale image into a black and white binary image.
• The user specifies a value that acts as a dividing line.
• If the gray value of a pixel is smaller than the dividing, the intensity of the pixel is set to 0, otherwise it's set to 255.
• The value of the dividing line is called the threshold. The grayscale threshold transform is often referred to as thresholding, or binarization.



PIECEWISE TRANSFORMATION: CONTRAST-STRETCHING TRANSFORMATION

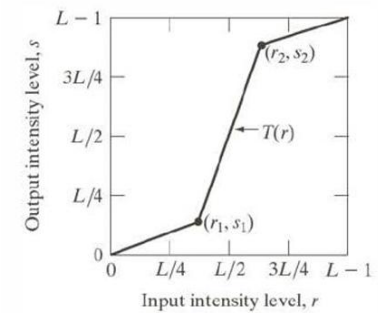
- The goal of the contrast-stretching transformation is to enhance the contrast between different parts of an image, that is, enhances the gray contrast for areas of interest, and suppresses the gray contrast for areas that are not of interest.
- Low contrast images result from the following
  a. Poor illumination
  b. lack of dynamic range in the imaging sensor
  c. Wrong settings of the lens aperture during acquisition
- It is a process that expands the range of intensity levels in an image so that it spans full intensity range of the recording medium or display device

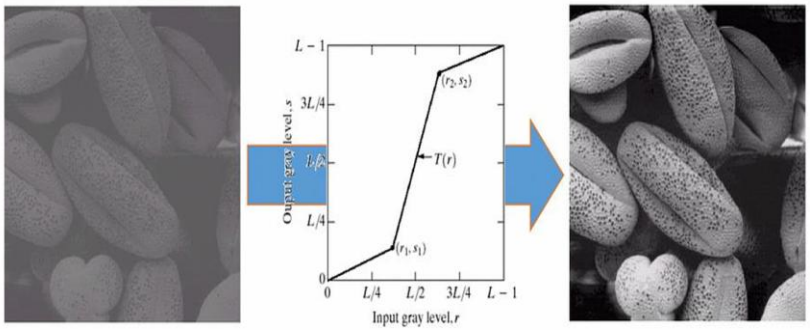
1. **Power-Law Contrast-Stretching Transformation**
   - If T(r) has the form as shown in the figure, the effect of applying the transformation to every pixel to generate the corresponding pixels produce higher contrast than the original image, by:
   - Darkening the levels below k in the original image
   - Brightening the levels above k in the original image



2. **Linear Contrast-Stretching Transformation**

- Points (r1, s1) and (r2, s2) control the shape of the transformation.
- The selection of control points depends upon the types of image and varies from one image to another image.
- If r1 = s1 and r2 = s2 then the transformation is linear and this doesn't affect the image. In other case we can calculate the intensity of output pixel, provided intensity of input pixel is x, as follows:
  a. For **0 <= x <= r1**: output = **s1 / r1 * x**
  b. For **r1 < x <= r2**:
     output = **((s2 – s1)/(r2 – r1))*(x – r1) + s1**
  c. For **r2 < x <= L-1**:
     output = **((L-1 – s2)/(L-1 – r2))*(x – r2) + s2**

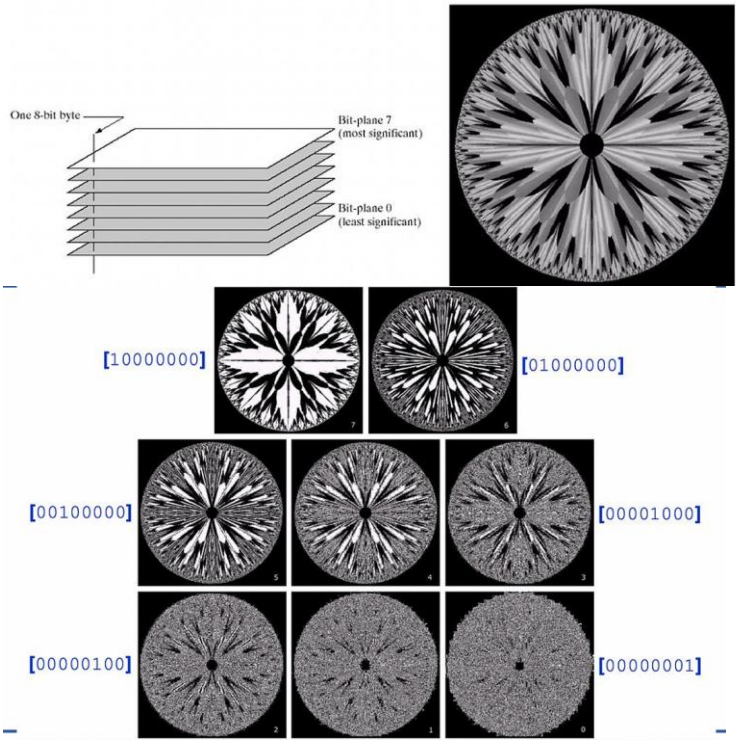




PIECEWISE TRANSFORMATION: BIT PLANE SLICING

• Bit-plane slicing is a method of decomposing an image into its constituent binary layers, where each layer corresponds to a bit position in the binary representation of pixel intensities.

• This technique is commonly used to analyze the significance of individual bits in forming the image and for applications like watermarking, compression, and image enhancement.



Bit-plane 7	Bit-plane 6	
Bit-plane 5	Bit-plane 4	Bit-plane 3
Bit-plane 2	Bit-plane 1	Bit-plane 0

IMAGE SMOOTHENING AND SHARPENING

Image Smoothing:

- Reduce noise and blur in an image by averaging out pixel intensity variations.
- This technique is often used to soften images or preprocess noisy data.

Image Sharpening:

- Enhance the edges and fine details of an image, making it appear more defined and clearer.
- This is often the reverse of smoothing.

FILTERING
- Filtering is a standard operation performed on digital images.
- Filters are normally used to remove noises from the image while keeping the image preserved.
- It's a technique used for modifying or enhancing the quality of the image.
- The selection of a filter is based on the nature of the task and the type and behavior of data.
- Some common filter applications are image smoothing, image sharpening and edge enhancement.
- The term "convolution" is used for the operation of applying a filter on an image

Different types
• Filters can be applied on digital signals in spatial domain as well as frequency domain. Thus, filters are of two types.
1.Spatial filters
• Spatial Filtering technique is used directly on pixels of an image. Filter mask is usually created in odd size so that it has specific center pixel. This mask is moved on the image such that the center of the mask traverses all image pixels.
2. Frequency filters
• This kind of filters are mainly focusing on the frequency of the image instead of the pixel intensity values.
• It is used for two kinds of operation namely smoothing and sharpening of images

• Also, **based on the method applied**, filters can be categorized as
1.Linear filters *(filters whose outputs mathematical equation can be written)*
• Linear filtering is the filtering method where the value of output pixel is linear combinations of the neighboring input pixels. It can be done with convolution operation. For example, mean/average filters or Gaussian filter.
2. Non-Linear filters *(filters whose outputs mathematical equation cannot be written)*
• Non-linear filter is a filter whose output is not a linear function of its input. Non-linear filtering cannot be done with convolution or Fourier multiplication. Median filter is a simple example of a non-linear filter.

LINEAR FILTER
• A linear filter is a transformation that processes an input signal x(t) (continuous) or x[n] (discrete) to produce an output signal y(t) or y[n], where the operation satisfies the principles of linearity:
1.**Additivity**: F(x1+x2)=F(x1)+F(x2)
2.**Homogeneity** (Scaling): F(a·x)=a·F(x)
• These filters perform convolution or correlation operations with the input signal.

Types of Linear Filters
Linear filters can be broadly categorized based on their purpose or domain of application:
1. Time Domain Filters
- **Moving Average Filter (Smoothing):**
  Used to reduce noise by averaging data points over a sliding window.
  Formula:

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$

  M: window size.
- **Difference Filter (Edge Detection):**
  Highlights abrupt changes in the signal, such as edges in images.

Example: **y[n]=x[n]−x[n−1]**

2. Frequency Domain Filters

Linear filters in the frequency domain are applied using the Fourier Transform.
Examples include:

- **Low-Pass Filter:** Allows low frequencies to pass and attenuates high frequencies.
- **High-Pass Filter:** Allows high frequencies to pass and attenuates low frequencies.
- **Band-Pass Filter**: Allows a specific range of frequencies to pass while attenuating others.

3. Spatial Domain Filters (For Images)

These are used for image processing:

- **Linear Convolution Filters**:
  Apply a kernel (small matrix) to an image.
  Example: Gaussian blur (low-pass filter).
- **Sharpening Filters:**
  Highlight fine details in an image.
  Example: Laplacian filter.

Mathematical Equation

The output y[n] of a linear filter can often be expressed using **convolution**:

$$y[n] = \sum_{k=-\infty}^{\infty} h[k] \cdot x[n-k]$$

• h[k]: Impulse response (filter coefficients).
• x[n]: Input signal.

In the frequency domain:

- **Y(f)=H(f)·X(f)**
- H(f): Frequency response of the filter.
- It is a linear frequency domain filter mechanism. Low pass filter removes the high frequency components and keeps low frequency components.
- It is used for smoothing the image.
- It's widely used in signal processing, image processing, and communication systems for noise reduction, signal smoothing, and anti-aliasing.
- A low pass filter can be represented as **G(x,y)=H(x,y)**.F(x,y) where F(x,y) is the Fourier Transform of original image and H(x,y) is the Fourier Transform of filtering mask.

MEAN FILTER

• The mean filter, also known as the box filter, is a simple and widely used linear spatial filter in image processing.
• Its primary purpose is to reduce noise and achieve image smoothing by replacing each pixel's value with the average of its neighbors. Below are the types of mean filter:

1.Averaging filter: It is used in reduction of the detail in image. All coefficients are equal.

2.Weighted averaging filter: In this, pixels are multiplied by different coefficients. Center pixel is multiplied by a higher value than average filter.

1.Averaging Filter

How the Mean Filter Works

1. A filter kernel (mask) is defined, typically a square matrix (e.g., 3x3, 5x5).
2. The kernel slides over the image, centering on each pixel.
3. For each position:
   - The pixel value at the center of the kernel is replaced by the average of the pixel values within the kernel.
   - 2.This averaging operation smooths the intensity variations

Example Kernel: A 3x3 mean filter kernel is represented as:

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

When applied to an image, this kernel averages the pixel values in a 3×3 neighborhood.

2.Weighted Average Filter

• The weighted average filter is an extension of the mean (box) filter, but instead of assigning equal weights to all pixels in the kernel, it assigns different weights based on their position relative to the center pixel.
• This filter smooths the image while emphasizing the contributions of certain pixels (e.g., center pixels more than edge pixels)

Steps in Weighted Average Filter

1. A weight matrix (kernel) is defined, with higher weights for pixels closer to the center and lower weights for those farther away.
2. The kernel slides across the image, centering on each pixel.
3. For each position, the output pixel value is calculated as the weighted sum of the pixel values in the kernel, divided by the sum of the weights

Example Weighted Kernel

• A common example is a Gaussian kernel, where weights are assigned based on the Gaussian distribution. For a 3×3 kernel:

$$\text{Gaussian kernel (3x3, } \sigma\text{=1): } \begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \times \frac{1}{16}$$

MEDIAN FILTER

The median filter is a widely used non-linear filter in image processing, primarily for removing noise, especially salt-and pepper noise, while preserving edges better than linear filters like the mean filter.

Steps:

1. A sliding window (kernel) is moved over the image.
2. For each position of the kernel:
   - The pixel values in the kernel are sorted in ascending order.
   - 2.The median value of the sorted list is taken as the new pixel value for the center of the kernel.
3. The process is repeated for all pixels in the image.

Example: Median Filter with a 3x3 Kernel

For a given kernel:

$$\begin{bmatrix}12 & 80 & 15\\ 40 & 255 & 60\\ 10 & 25 & 20\end{bmatrix}$$

• Sort the values: [10,12,15,20,25,40,60,80,255]
• Median = 40
• The center pixel (255) is replaced by 40.

GAUSSIAN KERNEL

- A Gaussian kernel is a widely used smoothing filter in image processing based on the Gaussian (normal) distribution.
- It is particularly effective for reducing noise and blurring an image while preserving edges better than a mean filter.
- The kernel weights are determined by the Gaussian function, which gives higher importance to the center pixels and gradually decreases the weights for neighboring pixels.

1D Gaussian Function

• The 1D Gaussian function is:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

x: The distance from the mean (typically 0 in standard form).

σ: The standard deviation, controlling the "spread" or "width" of the Gaussian curve.

$\frac{1}{\sqrt{2\pi}\sigma}$: A normalization factor ensuring that the area under the curve is 1.

Key Characteristics:
• **Bell-shaped curve:** Symmetrical around the mean.
• **Peak at x=0**: The maximum value occurs at the center (mean).
• **Width controlled by σ**: Larger σ values result in a wider curve, while smaller σ values result in a narrower curve.

2D Gaussian Function
• The 2D Gaussian function is an extension of the 1D Gaussian, applied to two variables x and y. It is defined as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

x, y: Distances from the center in the horizontal and vertical directions.

σ: The standard deviation, controlling the spread in both directions.

$\frac{1}{2\pi\sigma^2}$: Normalization factor ensuring the sum of all weights equals 1

Key Characteristics:
• **Symmetry**: The function is circularly symmetric around the center (mean), assuming equal variance in x and y.
• **Central Peak**: The highest value is at the center (x=0,y=0).
• **Weight Decay**: Values decrease as the distance from the center increases.

Adjusting the Gaussian Kernel
- **Size**:
  - The kernel size is typically odd (e.g., 3x3, 5x5).
  - Larger sizes result in more smoothing.
- **Sigma (σ)**:
  - Controls the spread of the Gaussian.
  - Larger σ values produce greater blurring.

Effect of Kernel Size and Sigma

Kernel Size	Sigma (σ)	Effect
Small	Small	Minimal blurring
Small	Large	Smooth with detail
Large	Small	Significant blur
Large	Large	Extensive blurring

Neighbours of a Pixel
- A pixel p at coordinates (x,y) has:
  - **4-Horizontal/Vertical Neighbours**:
    - $(x+1,y)$
    - $(x-1,y)$
    - $(x,y+1)$
    - $(x,y-1)$
    - These are called the **4-neighbours** of p: $N4(p)$.
  - **Diagonal Neighbours**:
    - $(x+1,y+1)$

        - $(x+1,y-1)$
        - $(x-1,y+1)$
        - $(x-1,y-1)$
        - These are called the **diagonal neighbours** of p: $ND(p)$.
- The **4-neighbours** and the **diagonal neighbours** of p are collectively known as the **8-neighbours** of p: $N8(p)$.

ADJACENCY BETWEEN PIXELS

- Let V be the set of intensity values used to define adjacency.
    - In a binary image, $V=\{1\}$ if referring to adjacency of pixels with value 1.
    - In a grayscale image, V typically contains more elements.
    - For example, in the adjacency of pixels with intensity values ranging from 0 to 255, set V could be any subset of these 256 values.

Types of Adjacency

1. **4-Adjacency**:
    - Two pixels p and q with values from V are 4-adjacent if q is in the set $N4(p)$.
2. **8-Adjacency**:
    - Two pixels p and q with values from V are 8-adjacent if q is in the set $N8(p)$.
3. **m-Adjacency (Mixed Adjacency)**:
    - Two pixels p and q with values from V are m-adjacent if:
        - q is in $N4(p)$, or
        - q is in $ND(p)$ and the set $N4(p) \cap N4(q)$ has no pixels whose values are from V.

CONNECTIVITY BETWEEN PIXELS

- **Importance**:
    - Connectivity is a crucial concept in digital image processing.
    - It is used for establishing boundaries of objects and components of regions in an image.
- **Definition**:
    - Two pixels are said to be connected if:
        - They are adjacent in some sense (neighbor pixels, 4/8/m-adjacency).
        - Their gray levels satisfy a specified criterion of similarity (equal intensity level).

Types of Connectivity Based on Adjacency

1. **4-Connectivity**:
    - Two or more pixels are said to be 4-connected if they are 4-adjacent to each other.
2. **8-Connectivity**:
    - Two or more pixels are said to be 8-connected if they are 8-adjacent to each other.
3. **m-Connectivity**:
    - Two or more pixels are said to be m-connected if they are m-adjacent to each other.

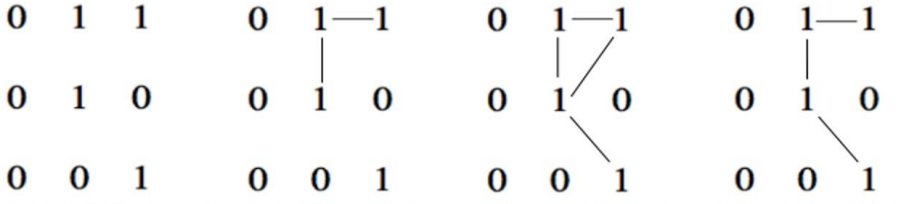


Fig: An arrangement of pixels | Fig: 4-connectivity of pixels | Fig: 8-connectivity of pixels | Fig: m-connectivity of pixels

---

Distance Measures

- For pixels p,q, and z with coordinates $(x,y),(s,t)$, and (v,w) respectively, D is a distance function or metric if:

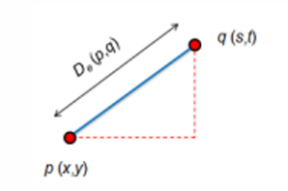
i. $D[p,q] \geq 0$ { $D[p,q]=0$ iff $p=q$ }
ii. $D[p,q]=D[q,p]$
iii. $D[p,q] \geq 0$ { $D[p,q]+D[q,z] \geq D[p,z]$ }

Euclidean Distance

- The Euclidean distance between p and q is defined as:

$$D_e(p,q)=[(x-s)^2+(y-t)^2]^{1/2}$$

- Pixels having a distance less than or equal to some value r from (x,y) are the points contained in a disk of radius r centered at (x,y).

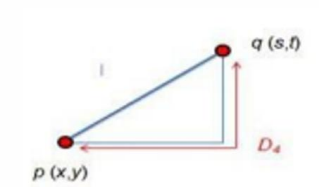


D4 Distance (City-Block Distance)

- The D4 distance between p and q is defined as:

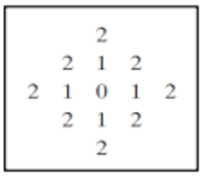
$$D_4(p,q)=|x-s|+|y-t|$$

- Pixels having a D4 distance from (x,y) less than or equal to some value r form a diamond centered at (x,y).



Example:

- The pixels with distance $D4 \leq 2$ from (x,y) form contours of constant distance. The pixels with $D4=1$ are the 4-neighbors of (x,y).

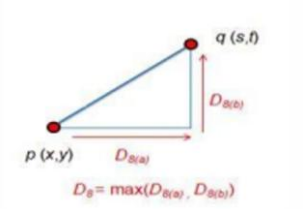


D8 Distance (Chessboard Distance)

- The D8 distance between p and q is defined as:

$$D_8(p,q)=\max(|x-s|,|y-t|)$$

- Pixels having a D8 distance from (x,y) less than or equal to some value r form a square centered at (x,y).



Example:

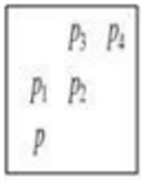
- $D8$ distance ≤ 2 from (x,y) forms contours of constant distance.

Dm Distance

- The Dm distance is defined as the shortest m-path between the points.
- In this case, the distance between two pixels will depend on the values of the pixels along the path, as well as the values of their neighbors.

Example:

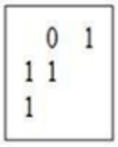
- Consider the following arrangement of pixels:
  - Assume p,$p2$, and $p4$ have value 1, and $p1$ and $p3$ can have values of 0 or 1.
  - Consider the adjacency of pixels values 1 (i.e. V = {1})



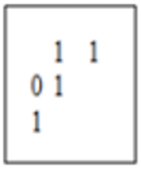
  - To compute the Dm between points pp and p4$p4$, we have 4 cases:

1. **Case 1**: If $p1$=0 and $p3$=0:
   - Length of the shortest m-path (the Dm distance) is 2: (p,$p2$,$p4$).

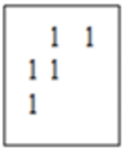
2. **Case 2**: If $p1$=1 and $p3$=0:
   - $p1$ and pp are no longer adjacent; the length of the shortest path will be 3: (p,$p1$,$p2$,$p4$).



3. **Case 3**: If $p1$=0 and $p3$=1:
   - The shortest m-path will also be 3: (p,$p2$,$p3$,$p4$).



4. **Case 4**: If $p1$=1 and $p3$=1:
   - Length of the shortest m-path will be 4: (p,$p1$,$p2$,$p3$,$p4$).



EDGE DETECTION

- **Definition**:
  - Edges are significant local changes of intensity in a digital image.
  - An edge can be defined as a set of connected pixels that forms a boundary between two disjoint regions.
- **Types of Edges**:
  - Horizontal edges
  - Vertical edges
  - Diagonal edges

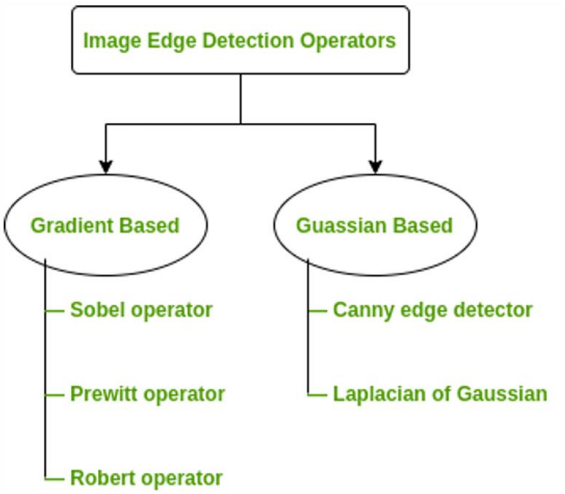
Importance of Edge Detection

- **Purpose**:
  - Edge detection is a method of segmenting an image into regions of discontinuity.
  - It is widely used in digital image processing for:
    - Pattern recognition
    - Image morphology
    - Feature extraction
- **Benefits**:

- Allows users to observe the features of an image for significant changes in gray level.
- Indicates the end of one region in the image and the beginning of another.
- Reduces the amount of data in an image while preserving its structural properties.

Edge Detection Operators

- **Types**:
  i. **Gradient-Based Operators**:
    - Compute first-order derivatives in a digital image.
    - Examples:
      - Sobel operator
      - Prewitt operator
      - Robert operator
  ii. **Gaussian-Based Operators**:
    - Compute second-order derivatives in a digital image.
    - Examples:
      - Canny edge detector
      - Laplacian of Gaussian



---

1. Sobel Operator

- **Description**:
  - The Sobel operator is a discrete differentiation operator that computes the gradient approximation of image intensity for edge detection.
  - It produces either the normal to a vector or the corresponding gradient vector at the pixels of an image.
  - Uses two 3x3 kernels or masks convolved with the input image to calculate vertical and horizontal derivative approximations.

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- **Advantages**:
  - Simple and time-efficient computation.
  - Effective at detecting smooth edges.
  - Good for thick and rough edges.
- **Limitations**:
  - Diagonal direction points are not always preserved.
  - Highly sensitive to noise.
  - Not very accurate in edge detection.

2.Prewitt Operator

- **Description**:
  - The Prewitt operator is similar to the Sobel operator and detects vertical and horizontal edges of an image.
  - It is effective for detecting the orientation and magnitude of an image.

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Advantages**:
  - Good performance in detecting vertical and horizontal edges.
  - Best operator for detecting the orientation of an image.
- **Limitations**:
  - The magnitude of coefficients is fixed and cannot be changed.
  - Diagonal direction points are not always preserved.

3.Robert Operator

- **Description**:
  - The Robert operator computes the sum of squares of the differences between diagonally adjacent pixels in an image through discrete differentiation.
  - It uses 2x2 kernels or masks for gradient approximation.

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad M_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

- **Advantages**:
  - Easy detection of edges and orientation.
  - Diagonal direction points are preserved.
- **Limitations**:
  - Very sensitive to noise.
  - Not very accurate in edge detection.

MODULE 6

MOTION ANALYSIS

Motion Analysis and Tracking

- **Definition:**
  Motion analysis and tracking in computer vision refer to the processes of detecting, analyzing, and following the movement of objects or regions of interest within video or image sequences.
- **Applications:**
  This field plays a crucial role in:
  - Surveillance
  - Autonomous navigation
  - Sports analysis
  - Robotics
  - And more

Components of Motion Analysis

Motion analysis is an overarching task that can be broken down into the following components:

1. Object Detection
2. Object Localisation
3. Object Segmentation
4. Tracking
5. Motion Detection
6. Pose Estimation

Input and Assumptions

- The usual input to a motion analysis system is a **temporal image sequence.**
- Assumptions and prior knowledge help to simplify analysis:
  - Information about camera motion (mobile or static)

- Time interval between consecutive images (short enough to represent continuous motion)
- This helps in selecting appropriate motion analysis techniques.

Categories of Motion-Related Problems

1. **Motion Detection**
   - Simplest problem
   - Registers any detected motion
2. **Moving Object Detection and Location**
   - Camera may be static while objects move, or vice versa
3. **3D Property Derivation from 2D Projections**
   - From images taken at different time instants

TYPES OF MOTION ANALYSIS

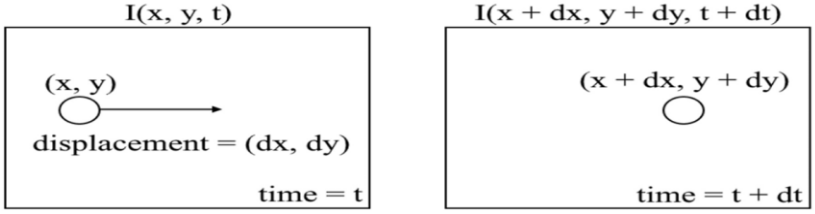
Description	Type
Detects whether motion has occurred in a scene (e.g., surveillance systems)	Motion Detection
Follows a specific object across multiple frames (e.g., tracking a car or person)	Object Tracking
Estimates the motion of each pixel or feature point from one frame to the next	Optical Flow
Separates moving foreground from static background	Background Subtraction
Tracks the path of an object over time	Trajectory Estimation
Analyzes body part motions (e.g., hand/body movements)	Gesture Recognition
Recognizes high-level events from motion (e.g., fall detection, fights)	Event Detection
Identifies complex motions like walking, running, jumping	Human Action Recognition

CORE TECHNIQUES USED

Usage	Technique
Pixel-wise motion estimation and tracking	Optical Flow
Predicts object positions during tracking	Kalman Filter
Color-based object tracking	Meanshift/Camshift
Complex motion and action recognition	Deep Learning (e.g., CNN + RNN, 3D CNNs)
Feature point tracking for action recognition	Dense Trajectories
Motion segmentation with static cameras	Background Subtraction (e.g., MOG2)

OPTICAL FLOW

- **Definition:**
  Optical flow represents the motion of objects between consecutive frames, caused by relative motion between the object and the camera.
- **Mathematical Representation:**
  - Let image intensity III be a function of space (x, y) and time (t).
  - If a pixel moves by (dx,dy) over time dt, then: I(x,y,t)=I(x+dx,y+dy,t+dt)



  - Assumptions:
    - Pixel intensities are constant across frames
    - Taylor series approximation is applied and common terms removed

$$I(x+\delta x, y+\delta y, t+\delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \dots$$

$$\Rightarrow \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

- **Optical Flow Equation:**

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

  - Where:
    - **u=dx/dt**
    - **v=dy/dt**
- **Challenge**:
  One equation with two unknowns – needs additional constraints (e.g., Lucas-Kanade method) to solve.

SPARSE VS DENSE OPTICAL FLOW

- **Sparse Optical Flow:**
  - Tracks motion vectors of select "interesting" pixels (e.g., edges, corners)
  - Faster, less computational load
- **Dense Optical Flow:**
  - Computes motion for every pixel in the frame
  - More accurate, but computationally expensive

Visual Comparison:

- **Left:** Sparse – a few key points
- **Right:** Dense – all pixels

Implementing Sparse Optical Flow

- Selects a sparse set of feature pixels (edges, corners) to track their motion vectors.
- Features are passed from frame to frame to ensure consistency in tracking.
- Common techniques:
  - **Shi-Tomasi Corner Detector**
  - **Lucas-Kanade Method**
  - **Horn–Schunck Method**
  - **Buxton–Buxton Method**

SHI-TOMASI CORNER DETECTOR

When selecting the pixels to track or implementing sparse optical flow, we only track the motion of a **feature set of pixels**. Features in images are **points of interest** that present rich image content, such as corners, which are invariant to translation, scale, rotation, and intensity changes.
The **Shi-Tomasi Corner Detector** is very similar to the popular **Harris Corner Detector** and can be implemented through the following steps:

Harris Corner Detector

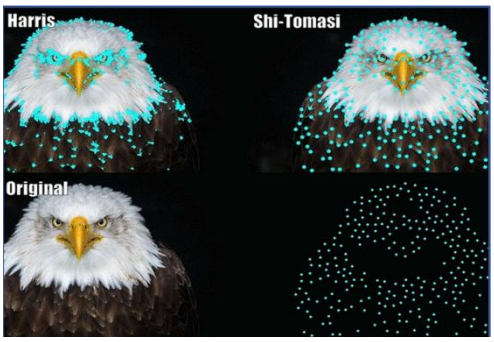
1. **Determine windows** (small image patches) with large gradients (variations in image intensity) when translated in both *x* and *y* directions.
2. For each window, **compute a score R.**
3. Based on the value of *R*, classify each window as **flat**, **edge**, or **corner**.

One of the most widely used **classical corner detection** methods.
Working Principle:

- Based on measuring how much the **image intensity changes** when a small window is moved in different directions.

- It uses the **second moment matrix**, also called the **structure tensor**:

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$



Where:

- I_x, I_y = image gradients in x and y directions (computed using Sobel filters or similar)
- The matrix is computed over a window around each pixel.

Corner Response Function (Harris Response):

$$R = \det(M) - k \cdot (\text{trace}(M))^2$$

Where:

- $\det(M) = I_x^2 I_y^2 - (I_x I_y)^2$
- $\text{trace}(M) = I_x^2 + I_y^2$

k is an empirical constant (typically between **0.04 and 0.06**)

Corner Classification Based on R Value

Region Type	R Value
Edge	R < 0
Flat region	R ≈ 0
Corner	R >> 0

In the Harris Corner Detector, the **R value** is a **corner response function** used to determine the type of region in the image.

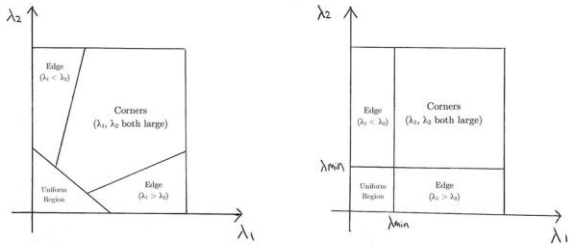
Shi-Tomasi Modification

Shi-Tomasi modified the scoring function of Harris. Instead of using the Harris score function, they proposed:

Shi-Tomasi Score: $R = \min(\lambda_1, \lambda_2)$

Where λ_1 and λ_2 are the **eigenvalues** of the structure tensor matrix *M*.

- If both λ_1 and λ_2 are large → **Corner**
- If one is large and the other is small → **Edge**
- If both are small → **Flat region**



← Harris & Shi-Tomasi

Only when both λ_1 and λ_2 are above a **minimum threshold (λ_min)** is the window classified as a **corner**.

Selective Object Tracking

You may want to **track a specific object** (e.g., a person) or a **category** of objects (e.g., 2-wheeler vehicles). Combine **object detection** with optical flow to **estimate the flow** of pixels within detected **bounding boxes**.

Shi-Tomasi Corner Detection: Step-by-Step

1. **Convert Image to Grayscale**
   Corner detection works on intensity; color is unnecessary.
2. **Compute Image Gradients**
   Compute partial derivatives (*Ix*, *Iy*) using Sobel operators or finite differences.
3. **Compute Structure Tensor Matrix (M)**
   For each pixel.

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Apply **Gaussian smoothing** to the components:

$$A = \text{Gaussian}(I_x^2), \quad B = \text{Gaussian}(I_y^2), \quad C = \text{Gaussian}(I_x I_y)$$

4. **Compute Eigenvalues (λ_1, λ_2) of M**
   Use these for the corner response function:

Shi-Tomasi Score = min(λ_1, λ_2)

5. **Threshold and Select Corners**
   - Apply a threshold on min(λ_1, λ_2).
   - Optionally, apply **non-maximum suppression** to retain only the strongest local corners.
6. **Return Top N Corners (Optional)**
   Useful in tracking applications like **Lucas-Kanade optical flow**.

LUCAS-KANADE: SPARSE OPTICAL FLOW

Lucas and Kanade proposed a method to estimate the motion of interesting features by comparing **two consecutive frames**. (works best for slow moving objects)

→ An Iterative Image Registration Technique with an Application to Stereo Vision

Assumptions of the Lucas-Kanade Method:

1. **Small time increment (dt)** between frames so that object motion is small.
2. The scene is **natural**, with textured objects having smooth grayscale variations.

Implementation Overview

- Take a **3x3 window** around detected features (e.g., from Shi-Tomasi).
- Assume all 9 points in the window share the **same motion**.

The optical flow equations are written for each of the *n* pixels (e.g., *n = 9* for 3x3 window):

$$\begin{aligned} &I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \\ &I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \\ &\vdots \\ &I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \end{aligned}$$

Let:

- $q_1, q_2, \ldots, q_n$ be the pixels in the window
- $Ix(q_i), Iy(q_i), It(q_i)$ be the partial derivatives for each pixel q_i

This creates a system of *n* equations.

Matrix Form: **A·v = b**

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \qquad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \qquad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

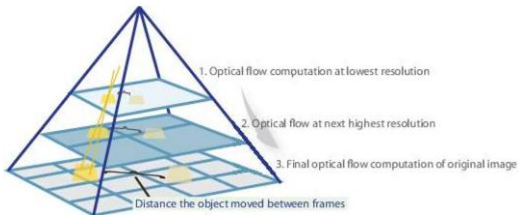
This system is over-determined (more equations than unknowns). Apply **least squares fitting** to reduce it to a **two-equation-two-unknown** problem:

Let:

- **Vx = u = dx/dt** (horizontal motion)
- **Vy = v = dy/dt** (vertical motion)

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix}$$

Solving these equations gives the **optical flow vector**.



Lucas-Kanade Method Summary

Assumptions:

1. **Pixel intensity remains constant** as it moves.
2. **Motion is small** and **locally constant** within a neighborhood.
3. Operates on a **small window** (e.g., 3x3 or 5x5) around each point.

Given two image frames at time t and $t + \Delta t$, and a pixel at (x, y):

The brightness constancy assumption:

$$I(x, y, t) = I(x + u, y + v, t + \Delta t)$$

Where:

- u, v are the **optical flow vectors** (displacement) in the x and y directions.

Using Taylor expansion, we get the **optical flow constraint equation:**

$$I_x \cdot u + I_y \cdot v + I_t = 0$$

But we have one equation and two unknowns → underdetermined.

So, we assume that all points within a small window (say n points) around the pixel have the **same motion vector** (u, v), and solve using **least squares:**

$$A \cdot \begin{bmatrix} u \\ v \end{bmatrix} = -b$$

Where:

- A is an $n \times 2$ matrix with image gradients I_x, I_y
- b is an $n \times 1$ vector of $-I_t$

Solution:

$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T b$$

LEAST SQUARES FITTING:

🌟 **Core Concept**

Least Squares Fitting finds the best-fitting curve (often a line) to a set of data points by **minimizing the sum of squared vertical differences (residuals)** between observed values yi and predicted values yi^

This process is preferred over minimizing absolute errors because squared differences are **smooth and differentiable,** enabling easier calculus-based optimization.

📈 **Linear Fit: f(x) = a + bx**

We aim to determine coefficients a (intercept) and b (slope) that minimize the residual sum of squares:

$$R^2 = \sum_{i=1}^{n} [y_i - (a + bx_i)]^2$$

✅ **Conditions for Minimization (First Derivatives = 0):**

$$\frac{\partial R^2}{\partial a} = -2 \sum_{i=1}^{n} [y_i - (a + bx_i)] = 0$$

$$\frac{\partial R^2}{\partial b} = -2 \sum_{i=1}^{n} [y_i - (a + bx_i)] x_i = 0$$

These yield the **normal equations:**

$$na + b \sum x_i = \sum y_i$$

$$a \sum x_i + b \sum x_i^2 = \sum x_i y_i$$

🧮 **Matrix Formulation**

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}$$

Solving this gives a and b.

🔁 **Shortcut Formulas Using Means**

Let:

- $\bar{x} = \frac{1}{n}\sum x_i$, mean of x-values
- $\bar{y} = \frac{1}{n}\sum y_i$, mean of y-values

Then:

$$b = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{ss_{xy}}{ss_{xx}}$$

$$a = \bar{y} - b\bar{x}$$

Where:

- ss_{xx}: sum of squares of x_i
- ss_{xy}: sum of product deviations

📊 **Goodness of Fit**

The coefficient of determination r2r^2r2 measures the fit:

$$r^2 = \frac{(ss_{xy})^2}{ss_{xx} \cdot ss_{yy}}$$

This tells you what fraction of the total variance in yyy is explained by the linear model.

📏 **Error Estimates**

Residuals: $e_i = y_i - \hat{y}_i$

Residual variance estimate:

$$s^2 = \frac{\sum e_i^2}{n-2}$$

Standard errors:

$$SE(b) = \frac{s}{\sqrt{ss_{xx}}}$$

$$SE(a) = s\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{ss_{xx}}}$$

These help assess uncertainty in the slope and intercept estimates.

🧠 **Practical Notes**

- Vertical fitting (minimizing y-differences) is preferred over perpendicular fitting because it gives simpler formulas and is appropriate when **x is known and y is measured.**
- Outliers can distort the fit because squaring emphasizes larger errors.
- Nonlinear fitting may involve **linearizing** the model or **iterative fitting** techniques.

DENSE OPTICAL FLOW

Dense optical flow gives the **flow vectors of the entire frame**, with **one flow vector per pixel**. It offers **higher accuracy** but is **computationally expensive.**

Key Points:

- Dense optical flow computes the **optical flow vector for every pixel** in each frame.
- While computation is slower, it yields more **accurate and denser results.**
- It is suitable for applications like:
  - Learning structure from motion
  - Video segmentation

FARNEBÄCK OPTICAL FLOW METHOD

What is it?

The **Farnebäck method** is a **dense optical flow algorithm** that estimates motion for **every pixel** in the image.

Differences from Sparse Methods:

- Unlike **Lucas-Kanade** (which tracks only selected points), Farnebäck gives a **full vector field of motion.**
- Ideal for:
  - Motion visualization
  - Object segmentation
  - Activity recognition

The algorithm approximates the local neighborhood of pixels with a **quadratic polynomial** using signal modeling:

📐 Local signal model:

Each neighborhood of the image is modeled as:

$$f(x) \approx x^T Ax + b^T x + c$$

Where:

- $f(x)$: intensity in the neighborhood
- A: symmetric matrix representing curvature
- b: vector representing slope
- c: scalar (constant term)

This is a **second-order polynomial** representation of pixel intensities in a small window.

How Motion is Estimated in Farnebäck Method

Step 1: Polynomial Approximation

- A **quadratic polynomial** is fit to every small window (e.g., 5x5) of the image.
- Describes local intensity variations.

Step 2: Displacement Matching

- Between two frames, find the **flow vector (displacement)** that aligns the polynomial models of corresponding neighborhoods.

Step 3: Coarse-to-Fine Estimation

- Uses an **image pyramid**:
  - Starts at low resolution (coarse) to estimate large motion.
  - Refines estimates progressively at higher resolutions (finer).

Gunnar Farnebäck Optical Flow

- Operates on **all pixels** (unlike Lucas-Kanade which works on corners detected by Shi-Tomasi).
- Detects **pixel intensity changes** between two frames.
- Converts results to **HSV format** for better visibility:
  - **Hue**: Direction of flow (angle)
  - **Value**: Magnitude of flow (distance)
- HSV strength set to **255** for optimal visibility.

KALMAN FILTER

Overview:

- A **Kalman filter** is an **optimal estimator** that infers parameters from **indirect, inaccurate, and uncertain observations.**
- It is **recursive**, allowing new measurements to be processed as they arrive.

Optimality:

- If noise is **Gaussian**, it **minimizes mean square error.**
- If not, it is still the **best linear estimator** given noise mean and standard deviation.

Why Kalman Filters Are Popular:

- Good practical results
- Supports **real-time online processing**
- Easy to implement with basic understanding
- **Measurement equations need not be inverted**

Real-World Examples:

- Tracking planetary orbits
- Aircraft/missile tracking via RADAR
- Robot localization and mapping

Why It's Called a "Filter":

- Finds the "best estimate" by **filtering out noise**
- Projects measurements onto the **state estimate**

What is a Covariance Matrix?

The covariance of two random variables x_1 and x_2 is

$$\begin{aligned} \mathrm{cov}(x_1,x_2) &\equiv E[(x_1-\bar{x}_1)(x_2-\bar{x}_2)] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x_1-\bar{x}_1)(x_2-\bar{x}_2)p(x_1,x_1)dx_1dx_2 \\ &\equiv \sigma^2_{x_1x_2} \end{aligned}$$

where p is the joint probability density function of x_1 and x_2.

The ***correlation coefficient*** is the normalised quantity

$$\rho_{12} \equiv \frac{\sigma^2_{x_1x_2}}{\sigma_{x_1}\sigma_{x_2}}, \quad -1 \le \rho_{12} \le +1$$

The covariance of a *column vector* $\mathbf{x}=[x_1 \,..\, x_n]'$ is defined as

$$\begin{aligned} \mathrm{cov}(\mathbf{x}) &\equiv E[(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})'] \\ &= \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})'\,p(\mathbf{x})\;dx_1..dx_n \\ &\equiv \mathbf{P}_{\mathbf{xx}} \end{aligned}$$

and is a *symmetric n* by *n* matrix and is *positive definite* unless there is a linear dependence among the components of $\boldsymbol{x}$.

The $(i,j)^{th}$ element of $\mathbf{P}_{\mathbf{xx}}$ is $\sigma^2_{x_ix_j}$

Interpreting a covariance matrix:

diagonal elements are the variances, off-diagonal encode correlations.

Diagonalising a Covariance Matrix

cov(**x**) is symmetric => can be ***diagonalised*** using an ***orthonormal*** basis.

By changing coordinates (pure rotation) to these unity orthogonal vectors we achieve ***decoupling*** of error contributions.

The basis vectors are the eigenvectors and form the axes of ***error ellipses***.

The lengths of the axes are the square root of the eigenvalues and correspond to standard deviations of the ***independent*** noise contribution in the direction of the eigenvector.

Example: Error ellipses for mobile robot odometry derived from covariance matrices:

1. Prediction Step

Predict the current state based on the previous state:

- **State Estimate:**

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

- **Covariance Estimate:**

$$P_k^- = AP_{k-1}A^T + Q$$

2. Update Step

Correct the prediction using the actual measurement:

- **Kalman Gain:**

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

- **Updated State Estimate:**

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

- **Updated Covariance Estimate:**

$$P_k = (I - K_k H)P_k^-$$

🧩 Components

- $\hat{x}_k$: Estimated state (e.g., position and velocity)
- A: State transition matrix
- B: Control matrix (if applicable)
- u_k: Control vector
- P: Error covariance matrix
- Q: Process noise covariance
- R: Measurement noise covariance
- H: Measurement matrix
- z_k: Measurement at time k

Kalman Filter: Step-by-Step

Step 1: Initialization

- Set initial **state estimate** (e.g., position, velocity)
- Set **initial error covariance** P0
- Define matrices:
  - A: State transition
  - B: Control input (if any)
  - H: Observation model
  - Q: Process noise covariance
  - R: Measurement noise covariance

Step 2: Prediction

- Predict the **next state** before receiving a measurement.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

$$P_k^- = AP_{k-1}A^T + Q$$

Where:

- $\hat{x}_k^-$: predicted state
- P_k^-: predicted error covariance
- u_k: control input (optional)

Step 3: Measurement

- Obtain new measurement zk (e.g., from a sensor).

Step 4: Kalman Gain Calculation

- Calculate **Kalman Gain K_k** to balance prediction and measurement.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

Step 5: Update (Correction)

- Update prediction using the actual measurement.

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

Where:

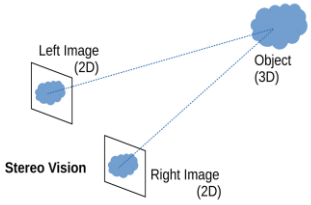
- $\hat{x}_k$: updated (corrected) state estimate
- P_k: updated uncertainty

Step 6: Repeat

- Loop through steps as new data arrives.

Use $\hat{x}_k$ and P_k as the new inputs for the next time step.

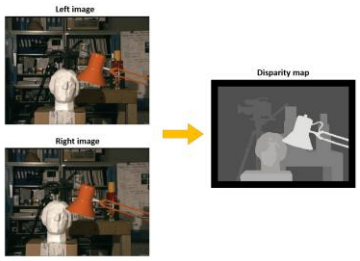
STEREO VISION



- Stereo vision, also known as **stereopsis** or **binocular vision**, is a technique used to perceive depth in a scene by capturing and analyzing images from two or more cameras placed slightly apart, mimicking the way human eyes work.
- The basic principle behind stereo vision is **triangulation**.
- When two cameras (or "stereo cameras") capture images of the same scene from slightly different viewpoints, the resulting pair of images, called **stereo pairs**,

contains disparities or differences in the positions of corresponding points in the two images.
- By analyzing these disparities, a computer vision system can **calculate the depth** information of objects in the scene.
- Objects that are closer to the cameras will have larger disparities, while objects farther away will have smaller disparities.
- Stereo vision algorithms often involve techniques such as feature matching, disparity mapping, and epipolargeometry to compute the depth map or a 3D representation of the scene.

DISPARITY MAP



The disparity is the apparent motion of objects between a pair of stereo images.
To experience this, try closing one eye and then rapidly open it while closing the other one.
You will note that closer objects will move significantly while objects further away will move very little.
This phenomenon is called **disparity**.

Disparity Map - Computation

- Given a pair of stereo images, to compute the **disparity map**:
  - First, **match every pixel** in the left image with its corresponding pixel in the right image.
  - Then, **compute the distance** for each pair of matching pixels.
  - Finally, represent these distance values as an **intensity image**, called the **disparity map**.

Disparity Map - Example

- Objects in the **foreground** (e.g., a lamp or statue) appear fairly shifted in stereo images and are thus marked with **brighter pixels** in the disparity map.
- Objects in the **background** have **low disparity** since their displacement between stereo images is very small.

Disparity and Depth

- Depth is **inversely proportional** to disparity.
- If the **geometric arrangement** of the cameras is known, the disparity map can be converted into a **depth map** using **triangulation**.

Important notes:

- When **disparity is near zero**, small disparity differences produce **large depth differences**.
- When **disparity is large**, small disparity differences **do not change** the depth significantly.
- Hence, **stereo vision systems** have **high depth resolution** only for objects **relatively near** the camera.

The Correspondence Problem

- Two cameras collect images from **slightly different positions** in 3D space, leading to differences in the images.
- For a large part of the scene seen in both images, **two pixels correspond** to the same 3D point in the real world.

The **Correspondence Problem** is defined as:

- Finding a match across the two images to determine **which part** of the scene on the right matches a given location on the left.

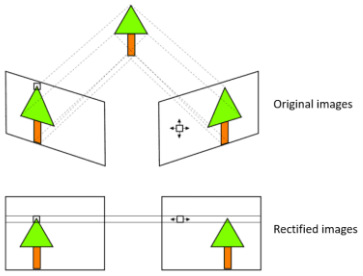
Earlier approaches:

- Relied on **search techniques** and **optimization**.

Modern approaches:

- Use **deep learning** for correspondence matching.
- **Matching** is the key computation in stereo vision.

Simplification by Rectification



- To compute the disparity map, we must solve the **correspondence problem.**
- Rectification of stereo images simplifies this:
  - After rectification, corresponding points lie on the **same horizontal line.**
  - Thus, the **2D stereo correspondence problem** is reduced to a **1D search** (only along the x-axis).

A basic approach for finding corresponding pixels is the *Block Matching algorithm*. It is based on comparing a small window around a point in the first image with multiple small blocks along the same horizontal line in the second image. For each pair of windows, a loss function is computed. The point $(\tilde{x}, y)$ in the second image with minimum loss value is the best match for the point (x, y) in the first image. Hence, the disparity value at the coordinate (x, y) will be $d(x, y) = (\tilde{x} - x)$.

Two different functions are usually used for finding the matching pixels. The first is the Sum of Absolute Differences (SAD) which computes the sum of elementwise absolute differences of two windows W^L and W^R, of size $M \times N$, extracted from the two stereo images:

$$\mathrm{SAD}(W^L, W^R) = \sum_{i=1}^{N} \sum_{j=1}^{M} |W_{ij}^L - W_{ij}^R|.$$

The second function is the Sum of Squared Differences (SSD). While the SAD computes a sum of absolute values, the SSD computes the sum of squared differences:

$$\mathrm{SSD}(W^L, W^R) = \sum_{i=1}^{N} \sum_{j=1}^{M} (W_{ij}^L - W_{ij}^R)^2$$

In general, SAD is preferable to SSD since it is faster and more robust to noise and outliers.

Disparity Mapping

Disparity Mapping is the process of computing a disparity map from a pair of stereo images (left and right views) taken from slightly different perspectives.
It shows the **pixel-wise horizontal displacement** of objects between the two images, and it is crucial for stereo vision and 3D reconstruction.

- **Input:** Stereo image pair (Left and Right images)
- **Output:** Disparity map (grayscale or color-coded image where intensity = disparity)

For corresponding points:

$$\mathrm{Disparity}(d) = x_{\mathrm{left}} - x_{\mathrm{right}}$$

To compute **depth (Z)** from disparity:

$$Z = \frac{f \cdot B}{d}$$

Where:

- f: focal length
- B: baseline (distance between cameras)
- d: disparity (in pixels)

Steps in Disparity Mapping

1. Stereo Image Acquisition

- Capture two images of the same scene from slightly different horizontal positions (left and right images).
- Use two cameras mounted side-by-side, just like human eyes.
- Cameras must be **synchronized** and **properly aligned** to minimize errors.

2. Camera Calibration

- Determine **intrinsic** (focal length, optical center) and **extrinsic** (rotation and translation) parameters of both cameras.

- Use a **calibration pattern** (like a chessboard) to compute:
    - Camera matrix (focal lengths, principal point)
    - Distortion coefficients
    - Rotation (R) and Translation (T) between cameras

Output: Camera matrices, distortion coefficients, rectification transforms.
Required for accurate 3D triangulation and depth computation.

3. Image Rectification

- Align stereo images so that corresponding points lie on the **same row** (epipolar lines become horizontal).
- Remove distortions and warp the images using **homography**.
- Search for matches is simplified to a **1D search** (along x-axis only).

4. Correspondence Matching (Stereo Matching)

- Find corresponding pixels in the left and right rectified images.
- For each pixel in the left image:
    - Search along the same row in the right image within a limited window.
    - Compare pixel windows using similarity measures:
        - Sum of Absolute Differences (SAD)
        - Sum of Squared Differences (SSD)
        - Normalized Cross-Correlation (NCC)
    - Pick the best match based on the **lowest difference** (or **highest correlation**).
- Store the **shift (disparity)** between the left and right match.

5. Disparity Map Generation

- Create a **2D map** where each pixel represents the **disparity value** (in pixels).
- Interpretation:
    - **Higher disparity value** → Object is **closer** to the camera.
    - **Lower disparity value** → Object is **farther** away.
- The map can be visualized in **grayscale** or using a **color map**.

6. Depth Map Computation

Using triangulation: **$Z=(f\times B)/d$**
Where:

- Z = Depth
- f = Focal length
- B = Baseline (distance between cameras)
- d = Disparity

This produces a **3D depth map**, useful for applications like **3D modeling** or **robot navigation**.

TRIANGULATION

- In stereo vision and 3D reconstruction, **triangulation** is the process of determining the **3D coordinates (depth)** of a point in the real world by using its projections onto two or more camera images taken from different viewpoints.
- Think of a triangle formed by:
    - Two camera centers (left and right),
    - The point in the 3D scene you're trying to locate,
    - And the lines (rays) from the cameras to that point.
- By knowing the geometry of the cameras (positions and directions) and where the point appears in each image, you can compute where the point lies in 3D space.

Basic Triangulation Formula

Let:

- x_L, x_R: x-coordinates of a corresponding point in the **left** and **right** image
- $d = x_L - x_R$: disparity
- f: focal length of the camera
- B: baseline (distance between the two cameras)

Then the **depth Z** is given by:

$$Z = \frac{f \cdot B}{d}$$

Once Z is known, you can compute **X** and **Y** using:

$$X = \frac{(x - c_x) \cdot Z}{f}, \quad Y = \frac{(y - c_y) \cdot Z}{f}$$

Where:

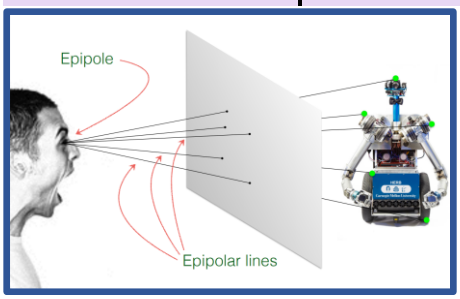
- (x, y): pixel location in the image
- (c_x, c_y): principal point (optical center) of the camera

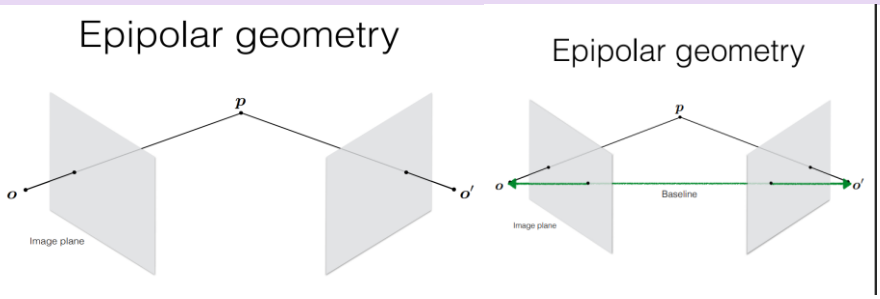
EPIPOLAR GEOMETRY

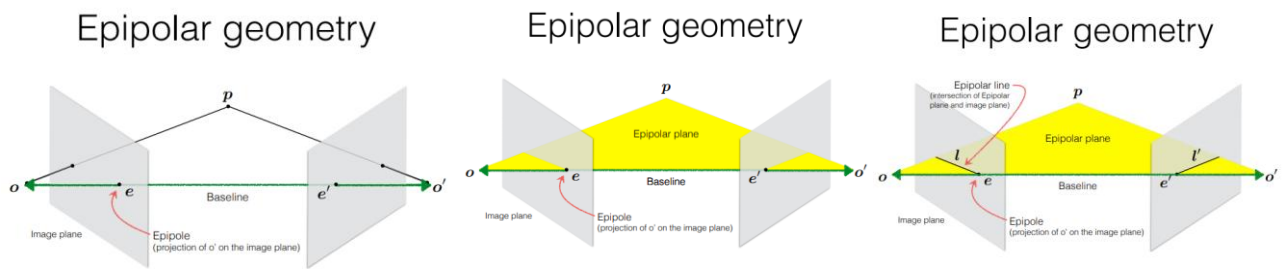
- **Epipolar Geometry** describes the geometric relationship between two views of the same scene captured by two cameras.
- It is the foundation of **stereo vision** and **3D reconstruction**, enabling efficient and accurate **correspondence matching**.

Explanation of Terms

Term	Description
Epipole	The point where the line joining the camera centers (baseline) intersects the image plane.
Epipolar Line	The line in the second image on which the corresponding point of a pixel in the first image must lie.
Epipolar Plane	The plane that contains a 3D point and the two camera centers.
Baseline	The line connecting the optical centers of the two cameras.
Corresponding Point	The same 3D point seen from two camera views.







The Fundamental Matrix F

- The **fundamental matrix** is the **algebraic representation** of **epipolar geometry**.
- It is derived from the mapping between a point and its corresponding **epipolar line**.
- Properties of the fundamental matrix are specified after deriving it.

The Fundamental matrix provides a correspondence $\boldsymbol{x}^T \boldsymbol{F} \boldsymbol{x}' = \mathbf{0}$, where $\boldsymbol{x}, \boldsymbol{x}'$ are 2D corresponding points in separate images. In other words,

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

IMAGE FUSION

What is Image Fusion?

- **Image fusion** is the process of integrating **complementary information** from multiple images of the same scene into a single, **enhanced image** that is more informative, accurate, and useful for human and machine interpretation.
- It combines the specifications or features from different images into one to create a more **accurate and informative image.**

How Image Fusion Works

- In image processing, different similar images are taken and their information is **gathered into one** image – this process is called **fusion.**
- Example:
  - Two images of the same scene: one **colored**, one **black & white.**
  - Some details visible in one image may not be visible in the other.
  - Image fusion combines the **best parts** of both to create a **fault-free, detailed** image.

Why Fuse Images?

Benefit	Application
Combine MRI (soft tissue) and CT (bone structure)	Medical imaging
Fuse high-res panchromatic with low-res multispectral images	Remote sensing
Merge infrared (night vision) with visible images	Security
Create all-in-focus images from multi-focus captures	Microscopy
Create HDR, panoramic, or depth-enhanced photos	Computational photography

Types of Image Fusion

1. ***Pixel-Level Fusion***
   - Combines raw pixel values.
   - Fundamental and requires **well-aligned (registered) images.**
2. ***Feature-Level Fusion***
   - Extracts **features** (edges, shapes, textures) from images.
   - Fusion is based on **feature descriptors.**
   - More **robust** to noise and illumination changes.
3. ***Decision-Level Fusion***
   - Each image is analyzed **independently.**
   - Final decisions are fused using **majority voting** or **rules.**
   - Common in **AI/ML, surveillance,** and **diagnostics.**

Detailed Explanation

- **Pixel-Level Fusion**: Analyzes corresponding pixels from multiple images.
- **Feature-Level Fusion**: Matches and fuses important object features.
- **Decision-Level Fusion**: Combines analyzed decisions from separate images.

Methods Used for Image Fusion

- **Multiplicative Algorithm**: Combines high-quality and low-quality images to create a more informative result.
- **Subtractive Method**: Overlapping based on subtractive algorithms; good color from colored image and detail from black & white.
- **PCA (Principal Component Analysis)**: Reduces data redundancy and modifies pixel values to maximize important information.

- **IHS (Intensity Hue Saturation) Method**: Sharpens images by transforming RGB into IHS space.
- **High Pass Filter Method**: Combines high-resolution B/W info with low-resolution color image (may lose some sharpness).
- **Brovey Transform Method**: Simple fusion of multi-sensor data, often increases image brightness.
- **Wavelet Method**: Applies color information to black & white images but may lose corner clarity.

Image Fusion Techniques

Simple Pixel-Level Methods

- Fast methods but may cause **blurring** or **loss of contrast**.

Method	Formula
Averaging	$I_f = \frac{I_1+I_2}{2}$
Maximum	$I_f(x,y) = \max[I_1(x,y), I_2(x,y)]$
Minimum	$I_f(x,y) = \min[I_1(x,y), I_2(x,y)]$

Transform Domain Techniques

- **Transform images → fuse coefficients → reconstruct**:

A. Wavelet Transform-Based Fusion

- Decompose images with Discrete Wavelet Transform (DWT).
- Fuse approximation and detail coefficients.
- Reconstruct using inverse DWT.

B. Laplacian Pyramid Fusion

- Build multi-level pyramids.
- Fuse coefficients at each level.
- Reconstruct final image.

C. Principal Component Analysis (PCA)

- Reduces dimensionality and fuses weighted principal components.

D. Discrete Cosine Transform (DCT)

- Used for compression-based fusion (similar to JPEG block processing).

Evaluation Metrics for Image Fusion

Description	Metric
Information content in the image. Higher = more detail	Entropy (H)
Structural similarity to source images	SSIM
Signal-to-noise ratio (fusion vs source)	PSNR
Shared information between fused and source images	Mutual Information (MI)
Fusion quality combining edge and gradient info	Q_AB/F
Measures contrast and intensity variation	Standard Deviation

Applications of Image Fusion

1. Object Identification

- Fusing high-res black & white with low-res colored images makes **object identification easier**.

2. Classification

- Fused images are **easier to categorize** based on features due to improved resolution.

3. Change Detection

- Fusion helps **detect changes** between images over time (e.g., distortion, missing areas).

Summary Table

Use Case	Technique Examples	Fusion Level
Fast, simple cases	Averaging, Max, PCA	Pixel-Level

Preserving important structures	Edge/Texture Fusion	Feature-Level
High-quality, scalable fusion	DWT, Laplacian, DCT	Transform-Level
AI-driven decision making	Majority Voting, AI-based methods	Decision-Level

MODULE 7

SCENE ANALYSIS

🔍 What is Scene Analysis?

Scene analysis is a branch of image processing and computer vision that focuses on understanding what's in an image or video frame. It involves identifying:

- **Objects** (what is in the scene)
- **Positions** (where they are)
- **Relationships** (how objects interact)
- **Environment** (context like indoor/outdoor)

It mimics **human visual understanding** and is key in areas like autonomous vehicles, surveillance, robotics, etc.

🎯 Goals of Scene Analysis

1. **Object Detection**: Find known/unknown objects.
2. **Object Recognition**: Classify detected objects (e.g., "car", "tree").
3. **Object Localization**: Determine where each object is.
4. **Scene Understanding**: Analyze interactions (e.g., person riding a bike).
5. **Semantic Segmentation**: Label every pixel with a class.
6. **3D Scene Reconstruction**: Build 3D models from 2D images.

🧩 Components of Scene Analysis

Component	Description
Low-Level Processing	Feature extraction (edges, textures, corners)
Mid-Level Processing	Group features into regions (segmentation)
High-Level Processing	Use AI to interpret relationships and context

🛠️ Techniques Used in Scene Analysis

Purpose	Technique
Feature Detection	SIFT, ORB
Segmentation	Graph Cuts, Watershed, U-Net
Object Detection	YOLO, Faster R-CNN
Scene Classification	CNN models
Depth Estimation	Stereo vision, depth sensors
Motion Tracking	Optical Flow

🔄 Typical Scene Analysis Pipeline

1. **Input**: Image or video frame.
2. **Preprocessing**: Resize, denoise, normalize.
3. **Feature Extraction**: Use SIFT, ORB, etc.
4. **Object Detection/Segmentation**: With YOLO, DeepLab, etc.
5. **Scene Interpretation**: AI rules or deep learning to describe the scene.

📏 Evaluation Metrics

Metric	Use
Accuracy	Correct classification
IoU	Overlap between prediction and ground truth
Precision/Recall	Quality of detection
F1-Score	Balance between precision and recall
Detection Time	Speed of detection

False Positive Rate	Incorrect detection of background as objects

🔎 DETECTION OF KNOWN OBJECTS (USING LINEAR FILTERS)

Linear filters are matrices used to enhance patterns (like edges or shapes) via convolution.

Steps:

1. Design a filter (e.g., edge detector).
2. Convolve with the image.
3. Apply thresholding.
4. Post-process (e.g., non-max suppression).

Example:

- **Vertical edge detection**: Sobel operator.
- **Template matching**: Cross-correlation with object template.

For convolution:

$$R(x,y) = \sum_i \sum_j I(x+i, y+j) \cdot K(i,j)$$

❓ DETECTION OF UNKNOWN OBJECTS

Focuses on **unusual or novel patterns**. It's important in:

- **Surveillance** (e.g., intruder detection)
- **Medical imaging** (e.g., tumors)
- **Manufacturing** (e.g., defects)
- **Autonomous vehicles** (e.g., unknown obstacles)

Techniques:

1. Blob Detection

Detects bright/dark regions with uniform texture.

- **LoG** (Laplacian of Gaussian): Detects circular blobs.

$$LoG(x,y) = \nabla^2(G(x,y) * I(x,y))$$

- **DoG** (Difference of Gaussian): Faster approximation of LoG, scale-invariant

$$DoG(x,y) = G_{\sigma_1}(x,y) - G_{\sigma_2}(x,y)$$

📌 **Applications**: Cell detection, bright spots in X-rays, SIFT keypoints (DoG).

2. Saliency Detection

Identifies **visually distinct areas** that stand out.

- Uses **contrast, color, orientation**.
- Generates a **saliency map**.
- Algorithms: Itti-Koch, Spectral Residual, SalGAN.

📌 **Applications**: Foreground detection, weakly supervised learning.

3. Region Growing

Starts with seed points and **grows regions** based on similarity.

- Criteria: intensity, texture, statistical similarity.

📌 **Applications**: Tumor segmentation, unknown object segmentation.

4. Clustering Techniques

Groups similar pixels using unsupervised learning.

- **K-Means**: Divides into K groups by color/intensity.
- **DBSCAN**: Finds dense clusters and identifies outliers.

📌 **Applications**: Satellite image segmentation, detecting foreign objects.

📷 Example: Surveillance Camera

Let's say a person walks into a normally empty frame:

- **Saliency Detection**: Highlights the person.
- **LoG/DoG**: Detects blob-like region.
- **Region Growing**: Defines the full human shape.
- **Clustering**: Segments the person as a separate group.

🎯 Even without training, the system detects an anomaly.

HOUGH TRANSFORM

The **Hough Transform** is a pivotal algorithm in **computer vision** and **image processing**, enabling the detection of geometrical shapes such as **lines**, **circles**, and **ellipses** within images.

Key Concepts:

- Transforms **image space** into **parameter space**.
- Uses a **voting mechanism** to identify shapes through **local maxima** in an accumulator array.
- Most commonly used to detect **lines** and **edges**, represented in **polar coordinates** using **rho (ρ)** and **theta (θ)**.

Applications:

- **Edge Detection**
- **Feature Extraction**
- **Object Detection**
- **Line and Circle Detection**
- **Generalized Shape Identification**

Why is it Needed?

- **Edge detectors** like Canny may miss points due to:
  - Noise
  - Imperfect image data
  - Gaps in edges
- **Hough Transform** helps group noisy edge points into shapes like lines or circles, even with **missing or broken lines**.

How Does It Work?

1. Edge Detection Preprocessing

- Use an edge detector (e.g., **Canny**) to detect strong edges in the image.
- Pixels with value **255** are considered edge pixels.

2. Representation in Polar Coordinates

- A line in polar coordinates:

$$\rho = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

- ρ: Perpendicular distance from origin
- θ: Angle from the horizontal axis

All points (x, y) on the same line yield the **same (ρ, θ)** values.

3. Creating the Hough Space

- Hough Space is a **2D matrix H[ρ, θ]**
- θ ranges from **0° to 180°**
- For every edge pixel (x, y):
  - Iterate θ from 0° to 180°
  - Calculate ρ
  - **Increment H[ρ, θ]** (vote)

4. Detecting Lines

- The points in **Hough Space** with votes **above a threshold** are considered **lines** in the image.

Advantages of Hough Transform:

- Detects lines of **any orientation**
- Robust to **noise**
- Works well even if edges are **broken** or **incomplete**

CORNER DETECTION

A **corner** in an image is a point where the **intensity changes significantly in two or more directions**.

It usually occurs at the **intersection of two edges** and can be visually identified as a **sharp turn** or **distinct point**, like the corner of a square or a chessboard.

Mathematical Definition:

- Corners are regions with **high gradient variations** in both the **x** and **y** directions.
- They are considered:
  - **Repeatable**
  - **Stable**
  - **Distinctive**

These properties make corners ideal for computer vision tasks such as **matching** and **tracking**.

Why Detect Corners?

Corners are:

- ✅ **Invariant** to translation, rotation, and small changes in scale
- ✅ **Good keypoints** for tracking and recognition
- ✅ **Easily localizable**

1. Harris Corner Detector

One of the most widely used **classical corner detection** methods.

Working Principle:

- Based on measuring how much the **image intensity changes** when a small window is moved in different directions.
- It uses the **second moment matrix**, also called the **structure tensor**:

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Where:

- I_x, I_y = image gradients in x and y directions (computed using Sobel filters or similar)
- The matrix is computed over a window around each pixel.

Corner Response Function (Harris Response):

$$R = \det(M) - k \cdot (\text{trace}(M))^2$$

Where:

- $\det(M) = I_x^2 I_y^2 - (I_x I_y)^2$
- $\text{trace}(M) = I_x^2 + I_y^2$
- k is an empirical constant (typically between **0.04 and 0.06**)

Interpretation of R:

- 📌 **R is large positive** → Likely a **corner**
- ⚠️ **R is small or negative** → **Not a corner**

2. Shi-Tomasi Corner Detector

An improvement over Harris; commonly used in **Kanade-Lucas-Tomasi (KLT)** trackers.

Key Concept:

- Instead of using the Harris response R, it considers the **minimum eigenvalue** of the matrix M:

Corner $\iff \min(\lambda_1, \lambda_2) > \text{threshold}$

Where:

- λ_1, λ_2 are eigenvalues of matrix MMM

Advantages of Shi-Tomasi:

- ✅ More **accurate** and **stable** than Harris
- ✅ Well-suited for **feature tracking**, especially in **video sequences**

IMAGE TAGGING

- **Image tagging** simply entails setting **keywords** for the elements contained in a visual.

- Example:
  A wedding photo may be tagged with terms like:
  - "wedding", "couple", "marriage"
  - As well as colors, objects, specific items, and abstract concepts like "love" or "relationship"

Definition:
Image tagging is the process of **automatically assigning descriptive labels or keywords (tags)** to an image based on its **visual content.**
Tags can describe:
- Objects (e.g., "dog", "car")
- Scenes (e.g., "beach", "city")
- Actions (e.g., "running", "eating")
- Emotions
- Semantic concepts

Applications of Image Tagging:
- 🔍 Image search and retrieval
- ⚠️ Content moderation
- 📁 Photo organization
- ♿ Accessibility tools (e.g., **alt-text** generation)

Approaches to Image Tagging
1. Manual Tagging
- Performed by **human annotators**
- **Time-consuming** and **not scalable**
- Commonly used to create **ground truth datasets** for training ML models

2. Rule-Based Systems (Traditional Computer Vision)
Pre-deep learning era methods using **handcrafted features** and classical classifiers.
🔍 **Features:**
- Color histograms
- Texture descriptors (e.g., **GLCM**, **LBP**)
- Shape features (e.g., **edges**, **contours**)

📊 **Classifiers:**
- Support Vector Machines (SVM)
- k-NN, Decision Trees
- Naive Bayes

❌ **Limitations:**
- Poor performance in **complex, real-world images**
- Not robust to **scale**, **occlusion**, or **lighting variation**

3. Auto Tagging (AI-Powered)
- Also known as **auto-tagging**
- Adds **contextual information** to images, videos, and live streams
- Enables easier and more robust **discovery** and **organization**

4. Deep Learning-Based Tagging (Modern Approach)
🔥 **Why Deep Learning?**
- Learns **hierarchical representations** directly from raw images
- Offers **greater accuracy** and **robustness**

Deep Learning Methods for Image Tagging

1. Convolutional Neural Networks (CNNs)
Pipeline:
Input image → CNN → Fully connected layers → Multi-label output (sigmoid)
Popular Pretrained Networks:

- VGGNet
- ResNet
- EfficientNet
- Inception

These are often **fine-tuned** on datasets like **MS-COCO**, **ImageNet**, or **Open Images**.

2. Multi-Label Classification

- Unlike single-label classification, tagging often involves **multiple labels per image**.
- **Output:**
  - Each tag has a **sigmoid-activated neuron**
  - Tags are treated as **independent** (not softmax)

3. Attention Mechanisms

Purpose:

Focus the model on **important regions** relevant to each tag.

Techniques:

- CAM (Class Activation Mapping)
- Grad-CAM
- Self-attention (Transformers)

4. Image Captioning + Tag Extraction

- Some systems first **generate captions** using Vision + NLP models
- Then, **extract tags** from captions

Example Pipelines:

- CNN + RNN
- Vision Transformer + GPT

5. Vision-Language Pretrained Models (VLPMs)

These models jointly understand **images and language**, enabling **zero-shot tagging**.

Examples:

- CLIP (Contrastive Language–Image Pretraining, OpenAI)
- BLIP
- ALIGN
- Flamingo