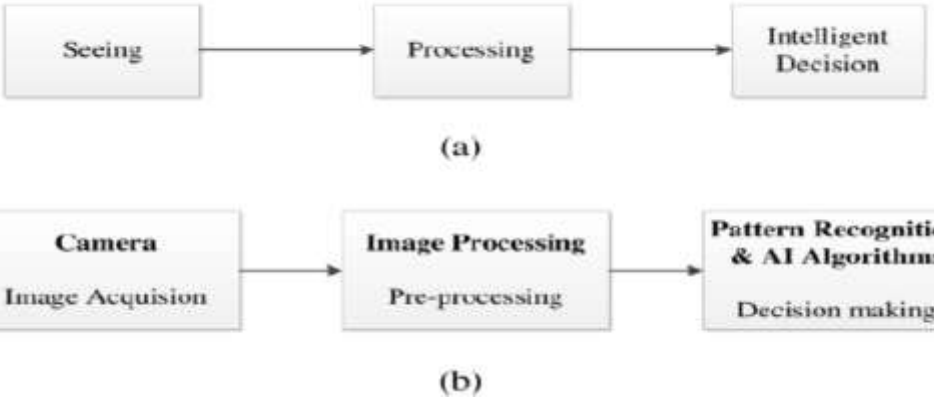
Human Visual System and Computer Vision



The basic principle of these two systems is almost same, i.e., conversion of light into useful signals/information from which accurate models of the physical world are constructed. Similarly, when considered at a high level, the structures of human and computer vision are somewhat similar, i.e., both have light sensors which convert photons into a signal (image), a processing step, and finally a mechanism to interpret the signal (object recognition).

Computer vision, image processing computer graphics and Image formation

Computer Vision

- **Goal:** Enable computers to *understand* and interpret images or video as humans do.
- **Input/Output:** Takes real-world images or video streams and outputs high-level information such as object labels, scene understanding, or automated decisions.
- **Examples:** Facial recognition, object detection, medical image diagnosis, autonomous vehicle perception.

Image Processing

- **Goal:** *Enhance* or transform images at the pixel or signal level for better quality or feature extraction.
- **Input/Output:** Starts with existing images and produces improved images or extracted features (e.g., edges, textures).
- **Examples:** Noise reduction, contrast enhancement, sharpening, edge detection, image compression.

Computer Graphics

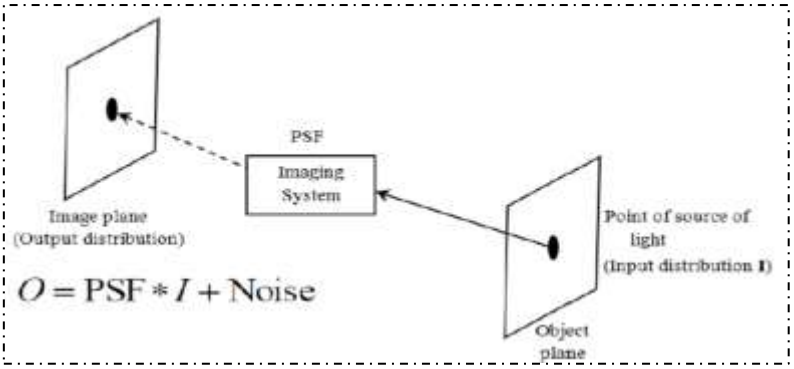
- **Goal:** *Create* or synthesize visual content from mathematical models or data.
- **Input/Output:** The input is any scene of a real world and the output is an image. Uses geometric models, textures, and lighting instructions to generate entirely new images or animations.
- **Examples:** Video game rendering, movie CGI, virtual reality environments, architectural visualization.

IMAGE FORMATION

- The image formation process can be mathematically represented as:

Image = PSF * Object function + Noise

- The object function is an object or a scene that is being imaged. The light from a source is incident on the scene or the object surface, and it is reflected back to the camera or the imaging system-**its ideal intensity distribution**
- The point spread function (PSF) is the impulse response when the inputs and outputs are the intensity of light in an imaging system, i.e., it represents the response of the system to a point source. PSF indicates the spreading of the object function, and it is a characteristic of the imaging instrument or the camera. A good or sharp imaging system generally has a narrow PSF, whereas a poor imaging system has a broad PSF. For a broad PSF, blurred images are formed by the imaging system.
- The symbol * represents **convolution**, a mathematical operation that “smears” or “blurs” the object function according to the PSF.
- Noise - Represents random disturbances added during image capture, such as sensor noise, photon shot noise, or electronic interference. It further degrades the image quality.
- The object function (light reflected from the object) is transformed into the image data by convolving it with the PSF. So, image formation can be considered as a process which transforms an input distribution into an output distribution.



Radiometry

In computer vision, **radiometry** is the branch of physics that deals with measuring and describing the *physical energy of electromagnetic radiation*, especially visible light, as it interacts with surfaces and sensors.
It provides the mathematical foundation for how light is emitted, reflected, transmitted, and captured—essential for tasks like 3-D reconstruction, photometric analysis, and realistic rendering.

- Lines, patches tilted with respect to the viewing or camera direction appear smaller apparent lengths, areas, respectively, to the camera/viewer. This is called a **foreshortening effect**. Let us consider that the surface area A is tilted under some angle θ between the surface normal and the line of observation. In this case, the solid angle is reduced by a foreshortening factor of cosθ This can be expressed as:

$$d\Omega = \frac{dA\cos\theta}{r^2}$$

- Radiant energy Q (measured in joules, J) is proportional to the number of photons, *i.e.*, total energy emitted by a light source or received by a detector.
- Radiant flux/power Φ (measured in watts, W) is the total radiant energy emitted, reflected, transmitted or received per unit time by a light source or a detector.
- Radiant exitance M (measured in Wm^{-2}) is the power emitted by a light source per unit surface in all directions.
- Radiant intensity I (measured in Wsr^{-1}) is the power leaving a point on a surface into unit solid angle, *i.e.*, exitant flux per unit solid angle, *i.e.*,

$$I = \frac{d\phi}{d\Omega}$$

- Irradiance E (measured in Wm^{-2}) is the light arriving at a point on a surface from all visible directions, *i.e.*,

$$E = \frac{d\phi}{dA}$$

- Radiance L (measured in $Wm^{-2}sr^{-1}$) is the power leaving unit projected surface area into unit solid angle, *i.e.*, the flux exitant from the surface is termed as radiance, which is the flux emitted per unit foreshortened surface area per unit solid angle, *i.e.*,

$$L = \frac{d^2\phi}{dA\cos\theta d\Omega}$$

- Radiosity B (measured in Wm^{-2}) is the radiant flux leaving (emitted, reflected and transmitted) by a surface per unit area.

Albedo $\rho(\lambda)$ can be defined as the ratio of light reflected by an object to light received. Let $E_i(\lambda)$ denote the incoming irradiance caused by the illumination on the surface, and $E_r(\lambda)$ is the energy flux per unit area reflected by the surface. The ratio

$$\rho(\lambda) = \frac{E_r(\lambda)}{E_i(\lambda)}$$

is called the reflectance coefficient or albedo. For simplicity, colour properties of the surface may be neglected, and in this case, albedo does not depend on the wavelength λ.

In a Phong model, the radiance leaving a specular surface is considered and it is proportional to $\cos^n(\delta\theta) = \cos^n(\theta_0 - \theta_s)$, where θ_0 is the exist angle, θ_s is the specular direction, and n is a parameter. Large values of n produce a small and narrow specular lobe (sharp specularities), whereas small values of n give wide specular lobes and large specularities (ambiguous boundaries).

$$L_o(P, \theta_o, \phi_o) = \rho_1 \int_\Theta L_i(P, \theta_i, \phi_i) \cos\theta_i d\Omega + \rho_2 L_i(P, \theta_s, \phi_s) \cos^n(\theta_0 - \theta_s)$$

Point operations depend on a pixel's value, and it is context-free (memoryless). The pixel value is changed by using a transformation T as shown in Figure 2.3. The new gray level (colour) value in a spatial location (x, y) in the resulting image depends only on the gray level (colour) in the same spatial location (x, y) in the original image. The point operation may also be nonlinear (logarithmic point operation). In general, the point operation can be mathematically represented as:

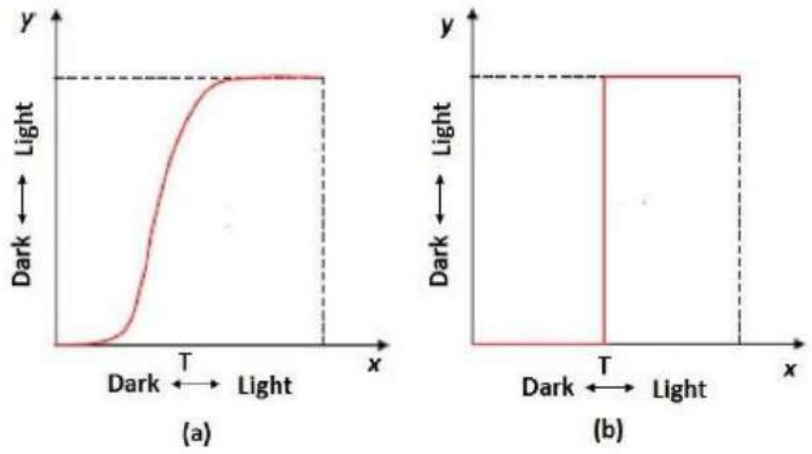
$$g(x, y) = T(f(x, y)) \tag{2.1}$$

A point operation can be defined as a mapping function, where a given gray level $x \in [0, L]$ is mapped to another gray level $y \in [0, L]$ according to a transformation $y = T(x)$. The transformation $y = X$ is called "lazy man operation" or "identity," as it has no influence on visual quality at all.

Contrast stretching: Contrast stretching is a linear mapping function used to manipulate the contrast of an image. In this process, the values of the input image are mapped to the values of the output image. As illustrated in Figure 2.5(a), higher contrast can be produced than the original by darkening the levels below T in the original image and brightening the levels above T in the original. Thresholding produces a binary image, and the corresponding transformation is shown in Figure 2.5(b).

The piecewise contrast stretching operation (Figure 2.6) can be represented as:

$$y = \begin{cases} \alpha x & 0 \le x < a \\ \beta(x - a) + c_1 & a \le x < b \\ \gamma(x - b) + c_2 & b \le x < L \end{cases}$$



Log transformation: The logarithmic (log) transformation is used to compress the dynamic range of an image. This transformation maps a narrow range of dark input values into a wider range of output levels, and the opposite is true for higher values of input grayscale levels. In other words, values of dark pixels in an image are expanded, while the high grayscale values are compressed. The inverse log transformation does the opposite. As shown in Figure 2.8, log transformation is used to view Fourier transformed images, as their dynamic ranges are very high. Log transformations show the details that are not visible due to a large dynamic range of pixel values. The inverse log transformation is used to expand the higher value pixels in an image. However, it compresses darker-level pixel values.

The log function is mathematically represented as:

$$y = c \log_{10}(1 + x)$$

Gray-level slicing: Gray-level slicing highlights a specific range of gray levels in an image, *i.e.*, displays high value for gray levels in the range of interest and low value for all other gray levels. The transformation shown in Figure 2.7 (a) highlights range $[a, b]$ and reduces all others to a constant level. On the other hand, the transformation as shown in Figure 2.7 (b) highlights range $[a, b]$, but preserves all other levels.

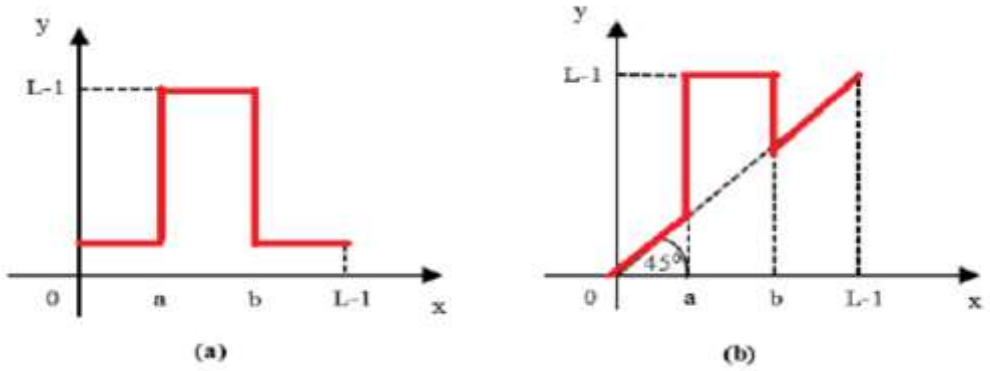


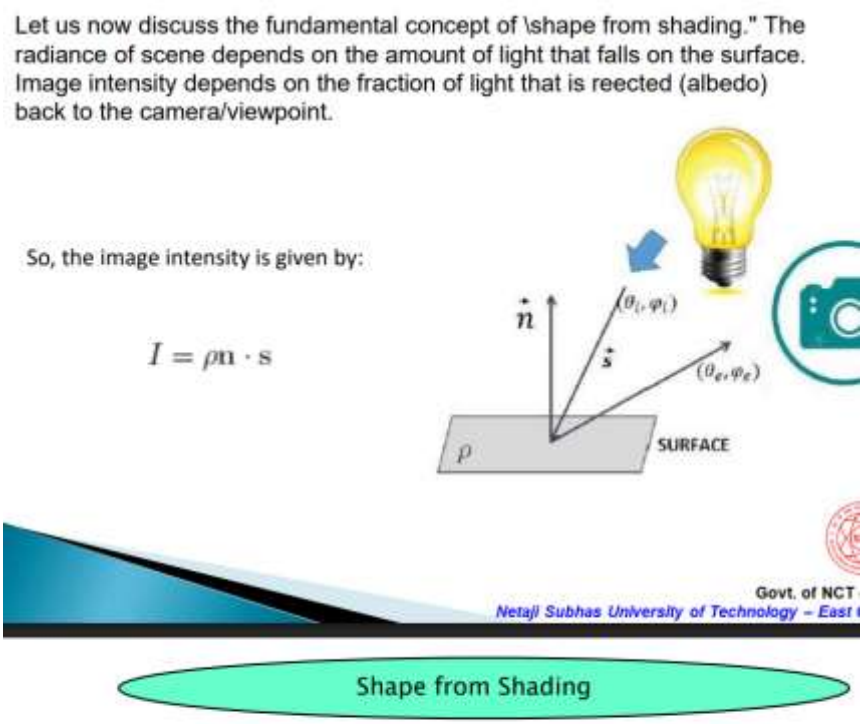
FIGURE 2.7: Gray-level slicing.

Histogram processing: Histogram of an image represents the relative frequency of occurrence of various gray levels in the image. The histogram of the 2D image $f(x, y)$ plots the population of pixels with each gray level. The image histogram indicates the characteristics of an image. Histogram of an image with gray levels in the range $[0, L-1]$ is defined as:

$$n_k = h(r_k)$$

where, r_k is the k^{th} gray level, n_k is the number of pixels in the image having gray level r_k, and $h(r_k)$ is the histogram of the image having gray level r_k. To get a normalized image histogram, each of the components of the histogram is divided by the total number of pixels (N^2) in the $N \times N$ image.

$$p(r_k) = \frac{n_k}{N^2}$$



Here, ρ is the albedo of the surface, and it lies between 0 and 1. Also, the n and s give the direction of the surface normal and the light source direction, respectively. The brightness of the surface, as seen from the camera, is linearly correlated to the amount of light falling on the surface. In this case, Lambertian surface is considered, which appears equally bright from all the viewing or camera directions. Lambertian surfaces reflect lights without absorbing.

1. Edge Detection

Goal:

Edges are places in an image where the intensity changes sharply. Detecting edges helps in recognizing object boundaries, textures, and important features.

Basic idea:

- Compute the gradient of the image intensity.
- High gradient → likely an edge.

Common techniques:

- **Sobel Operator:** Computes approximate gradients in x and y directions.
- **Prewitt Operator:** Similar to Sobel but with slightly different filters.
- **Canny Edge Detector:** More advanced; uses Gaussian smoothing, gradient computation, non-maximum suppression, and hysteresis thresholding.

Mathematical concept:

If $I(x, y)$ is the image:

$$\text{Gradient magnitude} = \sqrt{(I_x)^2 + (I_y)^2}$$

where I_x and I_y are partial derivatives.

2. Laplacian of Gaussian (LoG)

Goal:

Detect edges and spots (regions of rapid intensity change) with better noise suppression.

Steps:

1. Smooth the image using a **Gaussian filter** to reduce noise.
2. Apply the **Laplacian operator** (second derivative) to detect regions where intensity changes rapidly.

Formula:

$$\text{LoG}(x, y) = \nabla^2(G_\sigma * I) = \frac{\partial^2}{\partial x^2}(G_\sigma * I) + \frac{\partial^2}{\partial y^2}(G_\sigma * I)$$

- G_σ = Gaussian kernel with standard deviation σ
- $*$ = convolution
- ∇^2 = Laplacian operator

Key property:

- LoG detects edges as **zero-crossings** (where the output changes sign).

Visual intuition:

- Smooth out noise → highlight edges → find zero-crossings to locate edges precisely.

3. Difference of Gaussian (DoG)

Goal:

Approximate LoG but more efficient.

Idea:

- Subtract two Gaussian-blurred versions of the image with slightly different σ values:

$$\text{DoG}(x, y) = G_{\sigma_1} * I(x, y) - G_{\sigma_2} * I(x, y), \quad \sigma_2 > \sigma_1$$

- It behaves like a band-pass filter emphasizing structures at a specific scale.
- Computationally cheaper than LoG because it avoids computing the Laplacian explicitly.

Applications:

- Key step in **SIFT (Scale-Invariant Feature Transform)** for detecting keypoints.

Summary / Comparison

Method	Noise Handling	Computation	Detection Type	Notes
Gradient/Sobel	Low	Simple	First-order edges	Sensitive to noise
LoG	High	Moderate	Second-order edges (zero-crossings)	Detects fine edges, suppresses noise
DoG	High	Efficient	Approximate LoG	Used in SIFT, fast scale-space edge detection

1. Continuous Image Representation

Definition:

A continuous image is modeled as a **function over a continuous domain**, usually representing intensity or color at every point in space.

$$I(x, y) : \mathbb{R}^2 \to \mathbb{R} \quad (\text{grayscale})$$

$$I(x, y) : \mathbb{R}^2 \to \mathbb{R}^3 \quad (\text{color})$$

- x, y = continuous spatial coordinates
- $I(x, y)$ = intensity or color at point (x, y)

Properties:

- Defined **for all points** in a region.
- Can represent **smooth changes** in intensity.
- Useful for **theoretical analysis, differential operations, and filtering**.

Example operations:

- Computing **gradients**: $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$
- **Continuous transformations** like rotations, scaling, and convolution with continuous kernels.

Visual intuition:

Think of a **mathematical surface**, where the height at each (x, y) is the intensity.

↓

2. Discrete Image Representation

Definition:

A discrete image is represented as a **grid of pixels**, sampled from the continuous image at regular intervals.

$$I[m, n], \quad m = 0, \ldots, M-1; \quad n = 0, \ldots, N-1$$

- $M \times N$ = number of pixels (rows × columns)
- Each $I[m, n]$ = quantized intensity value (e.g., 0–255 for 8-bit grayscale)

Properties:

- Only defined at **specific sample points**.
- **Digital images** used in computers are always discrete.
- Intensity values are **quantized**, not continuous.

Example operations:

- Convolution with discrete kernels (Sobel, Gaussian blur)
- Image transformations (rotation, scaling) using interpolation
- Storage in formats like PNG, JPEG, BMP

Visual intuition:

Think of a **chessboard**, where each square represents a pixel value.

11. Gabor Filters

Definition:

Gabor filters are **bandpass filters** used to capture **specific frequency and orientation information** from images. They are widely used for **texture analysis, feature extraction, and pattern recognition**.

Mathematical Form:

A 2D Gabor filter is:

$$g(x, y) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cdot \cos\left(2\pi \frac{x'}{\lambda} + \phi\right)$$

where:

- $x' = x\cos\theta + y\sin\theta$
- $y' = -x\sin\theta + y\cos\theta$
- λ = wavelength of sinusoidal factor
- θ = orientation
- σ = Gaussian envelope standard deviation
- ϕ = phase offset
- γ = aspect ratio

Properties:

- Sensitive to **edges, textures, and frequencies** at a given orientation.
- Mimics **human visual cortex responses**.

Applications:

- Texture segmentation
- Face recognition
- Fingerprint recognition

12. Shape Representation

Definition:

Shape representation is a method to describe **geometric properties of objects** in an image. This is essentia for **object recognition, classification, and analysis**.

Main approaches:

a) Boundary-based Representation

- Only the **contour** or edge of the object is used.
- Examples:
  - **Chain code:** Encodes contour as sequence of directions.
  - **Polygon approximation:** Approximates contour by vertices.
  - **Fourier descriptors:** Uses Fourier transform of boundary coordinates.

b) Region-based Representation

- Uses the **entire area** of the object.
- Examples:
  - **Moment invariants:** Capture shape features invariant to translation, rotation, scaling.
  - **Area, centroid, eccentricity, compactness**: Simple shape descriptors.

c) Skeleton / Medial Axis

- Represents shape as **thin lines or skeleton** inside the object.
- Useful for **pattern recognition and shape matching**.

Applications:

- Character recognition (OCR)
- Object classification
- Medical imaging (organ or tumor shapes)

13. B-Spline Curves

Definition:

B-Splines are **piecewise polynomial curves** used for **smooth curve representation** in computer vision and graphics. They generalize Bezier curves and allow flexible shape modeling.

Mathematical Form:

$$C(u) = \sum_{i=0}^{n} N_{i,k}(u) P_i$$

where:

- P_i = control points
- $N_{i,k}(u)$ = B-spline basis function of degree k
- u = parameter, $0 \leq u \leq 1$

Properties:

- **Local control:** Moving a control point affects only a portion of the curve.
- **Smoothness:** C^k continuous (smooth up to k-th derivative).
- Can represent **complex shapes efficiently**.

Applications in Computer Vision:

- Shape modeling and reconstruction
- Object contour representation
- Curve fitting to edge points
- Medical imaging (organ contours)
