Digital Image Processing

CLASS NOTES

Mahanth Yalla
M. Tech-AI, IISc

Preface

These notes are based on the lectures delivered by **Prof. Rajiv Soundararajan (ECE) and Prof. Soma Biswas (EE)** in the course **E9 241 - Digital Image Processing** at Indian Institute of Science (IISc) Bengaluru - Aug Semester 2025. The notes are intended to be a concise summary of the lectures and are not meant to be a replacement for the lectures.

Disclaimer

These notes are not official and may contain errors. Please refer to the official course material for accurate information.

"I cannot guarantee the correctness of these notes. Please use them at your own risk".

- Mahanth Yalla

Contribution

If you find any errors or have any suggestions, please feel free to open an issue (Create New Issue) or a pull request on this GitHub repository, I will be happy to incorporate them.

Feedback

If you have any feedback or suggestions on the notes, please feel free to reach out to me via social media or through mail mahanthyalla [at] {iisc [dot] ac [dot] in , gmail [dot] com}.

Acknowledgements: I would like to thank Prof. Rajiv Soundararajan (ECE) and Prof. Soma Biswas (EE) for delivering the lectures and providing the course material. I would also like to thank the TAs for their help and support.

Contents

Chapter 1

Binary Image Processing

1.1 Introduction

Definition 1.1.1: Image Definition

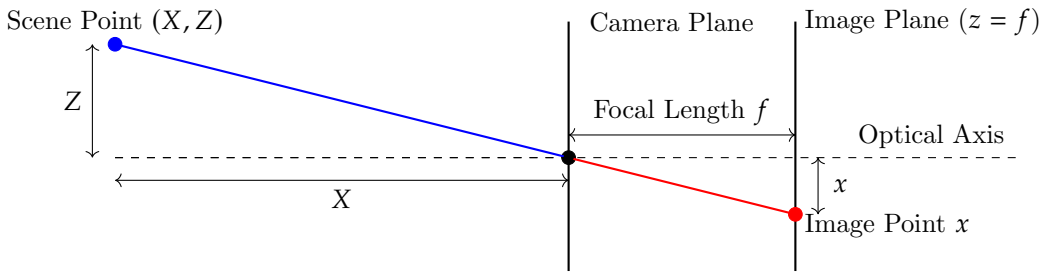
An **image** is a projection of a 3D object onto a 2D surface. This dimensional reduction causes loss of certain 3D information, which is generally very hard to recover.

1.2 Image Formation

The process of **image formation** models how a 3D scene is mapped to a 2D image plane through a camera model. A pinhole camera serves as the simplest abstraction to study this process.

1.2.1 Pinhole Camera Model (2D)

In the 2D case, the relation between a point (X, Z) in the scene and its projection (x) on the image plane is derived by similar triangles:



from the diagram, we have:

$$\frac{x}{f} = \frac{X}{Z} \quad \Rightarrow \quad x = f \cdot \frac{X}{Z}$$

Here, f is the focal length of the pinhole camera.

Note:-

we use Capital letters for 3D coordinates and lowercase for 2D image coordinates and observe that the Z component is lost.

1.2.2 Pinhole Camera Model (3D)

Extending to 3D coordinates (X, Y, Z), the projection onto the image plane gives:

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

This shows how 3D geometry is mapped to 2D via perspective projection.

1.2.3 Homogeneous Coordinates

Homogeneous coordinates are used to express projection as a matrix multiplication:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Note:-

Homogeneous coordinates are crucial because they unify perspective projection, translation, and rotation into matrix multiplications.

1.2.4 Intrinsic Matrix

The **intrinsic parameters** capture camera-specific properties such as focal length and principal point offset:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where (f_x, f_y) are focal lengths in pixel units and (c_x, c_y) is the principal point.

1.2.5 Extrinsic Matrix

The **extrinsic parameters** describe the camera's position and orientation in the world:

$$M = [R|t]$$

where R is a 3×3 rotation matrix and t is a 3×1 translation vector.

Note:-

The rotation matrix R can be defined using an angle α in 2D as:

$$R(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

This matrix rotates a point by α radians in the plane. In 3D, rotation can be performed about each axis using three matrices:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad R_y(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}, \quad R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A general 3D rotation can be represented by multiplying these matrices:

$$R = R_z(\psi) R_y(\phi) R_x(\theta)$$

where θ, ϕ, and ψ are rotation angles about the x, y, and z axes, respectively.

1.2.6 Projection Matrix

The complete mapping from 3D world coordinates to 2D image plane is:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KM \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

with scaling factor s.

Claim 1.2.1 Projection Matrix

The projection matrix $P = K@M$ completely defines the mapping from world coordinates to image coordinates, which depends on the intrinsic and extrinsic parameters of the camera. (i.e., focal length, principal point, rotation, and translation vectors.)

Definition 1.2.1: Camera Calibration

The process of estimating the intrinsic and extrinsic parameters of a camera from observed images of a known calibration pattern.

Calibration ensures accurate geometric measurements from images.

1.3 Image Representation

1.3.1 Pixel Values

A digital image is represented as a 2D array of **pixels (picture elements)**, each encoding intensity (grayscale) or color.

Definition 1.3.1: Image & Pixel

An **image** is a 2D array

$$I : \{0, \ldots, M-1\} \times \{0, \ldots, N-1\} \to \{0, \ldots, K-1\}$$

Each element $I[i, j]$ is a **pixel** with **intensity** (gray level) in $\{0, \ldots, K-1\}$, where, 0 represents black and $K-1$ represents white. For a 8 bit storage, $B = 8$, then maximum intensity can be calculated as $K = 2^B = 2^8 = 256$, 0 represents black and 255 represents white.

For normalized intensity, use

$$I_n[i, j] = \frac{I[i, j]}{(K-1)} \in [0, 1]$$

.

1.3.2 Color Spaces

Different **color spaces** represent pixel values differently:

- RGB: additive primary colors.
- HSV: hue, saturation, value (closer to human perception).
- YCbCr: luminance and chrominance separation. (where Y is intensity)

1.3.3 Feature Extraction and Descriptors

Features capture essential information in images such as edges, corners, or textures. **Descriptors** encode features into numerical representations (e.g., SIFT, HOG) for matching and recognition.

1.3.4 Sampling and Quantization

The ideal irradiance signal is sampled on a discrete grid and quantized to a finite set of gray levels.

- **Sampling:** Selecting discrete spatial points to represent an image.
  choose integer pixel sites (i, j); the image becomes $I[i, j]$ on an $M \times N$ grid.
- **Quantization:** Mapping continuous intensity values into discrete levels
  map real intensities to $\{0, 1, \ldots, K-1\}$, e.g., $K = 256$ for 8-bit grayscale.

Binary images: special case $K = 2$ with intensities $\{0, 1\}$ (or $\{0, 255\}$ in 8-bit storage).

Note:-

Undersampling causes aliasing; inadequate quantization leads to loss of detail.

1.3.5 Image Formats

Typical resolutions include 256×256, 512×512, **1920x1080**, etc.As we can calculate the number of Bytes required to store these images, we find:

- For 256×256 images: $256 \times 256 \times 1 = 65,536$ Bytes (assuming 8-bit grayscale).
- For 512×512 images: $512 \times 512 \times 1 = 262,144$ Bytes (assuming 8-bit grayscale).
- For 1920×1080 images: $1920 \times 1080 \times 3 = 6,220,800$ Bytes (assuming 24-bit RGB).

which is nearly 6.3 MB for a full HD image (1920x1080 3-channel RGB image). Hence, image storage can be quite substantial, necessitating efficient compression techniques.

Few common **Formats** include:

- JPEG: lossy compressed format.
  The most common for photographs to save space, as the human eye is less sensitive to high-frequency details. The compression is achieved by discarding some image data, but the benefit is a significantly reduced file size. (e.g. the full HD image (1920x1080) can be compressed from **6.3 MB** to around less than a **1 MB**)
- PNG: lossless compression, supports transparency.
- BMP: uncompressed raster format.
- TIFF: flexible format supporting various compressions.
- GIF: supports animation and transparency (limited color palette).
- WEBP: modern format providing lossy and lossless compression.
- HEIF: high efficiency image format, supports advanced features.
- AVIF: image format based on AV1 compression, offering high quality at smaller file sizes.
- EXR: high dynamic range (HDR) image format, supports wide color gamuts and high bit depths.
- DNG: raw image format for digital photography, preserves original sensor data.
- PPM: portable pixmap format, simple uncompressed color image format.

1.3.6 Binary Images and Thresholding

Definition 1.3.2: Binary Image

A **binary image** is an image that consists of only two colors, typically black and white. Each pixel in a binary image is represented by a single bit, where 0 represents black and 1 represents white.

The simplest method to obtain binary images is **thresholding**:

$$B(x,y) = \begin{cases} 1 & I(x,y) \geqslant T \\ 0 & I(x,y) < T \end{cases}$$

where T is the threshold.

1.3.7 Gray Level Histograms

The histogram of grayscale values is a fundamental tool to analyze and design thresholding algorithms.

Claim 1.3.1 Histogram-based Thresholding

If the histogram shows two well-separated peaks, the optimal threshold lies near the valley between them.

1.4 Histogram of an Image

Definition 1.4.1: Gray-Level Histogram

A **gray-level histogram** is a representation of the distribution of pixel intensities in a grayscale image. It counts the number of pixels for each intensity level, providing insights into the image's contrast and brightness.

Mathematical Formulation Let a grayscale image be

$$I : \{0,\dots,M-1\} \times \{0,\dots,N-1\} \to \{0,\dots,K-1\}$$

. The *histogram* counts occurrences at each gray level, given as a function as

$$H : \{0,\dots,K-1\} \to \{0,\dots,MN\}$$

such that

$$H(k) = \text{no of occurrences of gray level } k$$

where $k \in \{0,\dots,K-1\}$

$$H(k) \;=\; \#\{(i,j) : I[i,j] = k\} \qquad k = 0,\dots,K-1$$

also,

$$\sum_{k=0}^{K-1} H(k) = MN.$$

The *normalized histogram* (PMF) is

$$p(k) = \tfrac{H(k)}{MN}$$

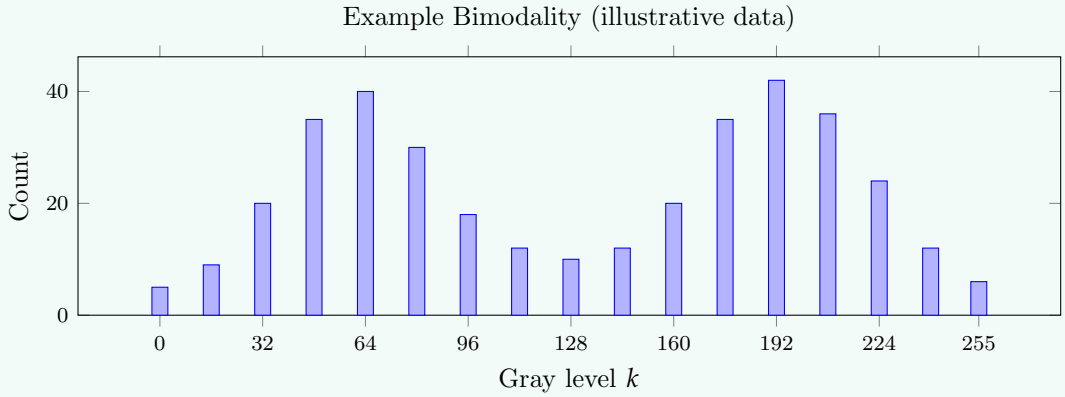
and

$$\sum_{k} p(k) = 1$$

Example 1.4.1 (Interpreting Histogram Shapes)

Given an image whose histogram exhibits a tall peak near intensity 40 (dark shades), and another smaller, broad peak near 200 (bright shades), we deduce the image features a dark background with a bright foreground—ideal for segmentation via thresholding.

- **Dark image:** $p(k)$ concentrated near $k \approx 0$.
- **Bright image:** $p(k)$ concentrated near $k \approx K-1$.
- **Bimodal image:** two peaks (e.g., dark foreground on bright background) $\Rightarrow$ suitable for a *single* global threshold.



Note:-

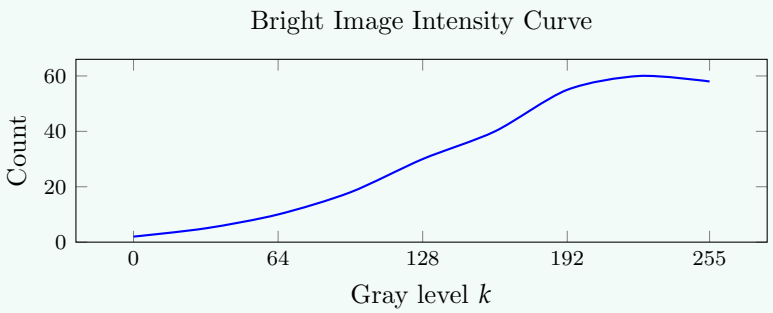
Understanding the histogram profile assists in identifying whether an image is underexposed, overexposed, well-contrasted, or subject to other illumination artifacts.

Types of Histograms and Their Interpretation Different histogram profiles relate directly to the visual impression and underlying properties of an image:

- **Bright Images:** Most pixel values clustered towards the higher end of the gray level range.
- **Dark Images:** Values crowded in the lower end, leading to overall darker visual output.
- **Dual Peak Model:** Exhibits two pronounced peaks, often corresponding to distinct foreground and background regions; common in images fit for binarization.
- **Flat Histogram:** Pixels uniformly distributed across gray levels—rare in natural images, may occur in images heavily corrupted with noise.
- **Equal Histograms:** Result from histogram equalization operations to improve contrast.

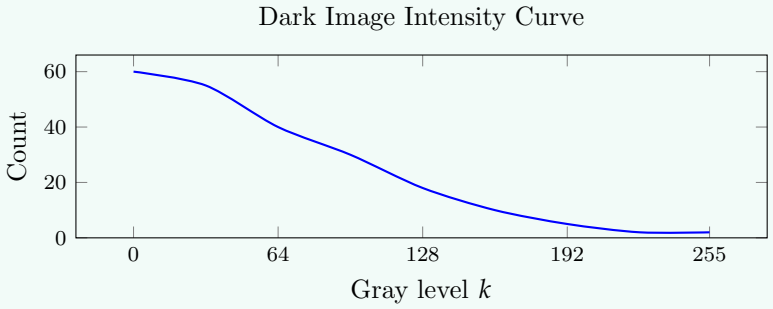
Example 1.4.2 (Histogram Interpretation)

- A **bright image** shows histogram concentrated on high intensity values.



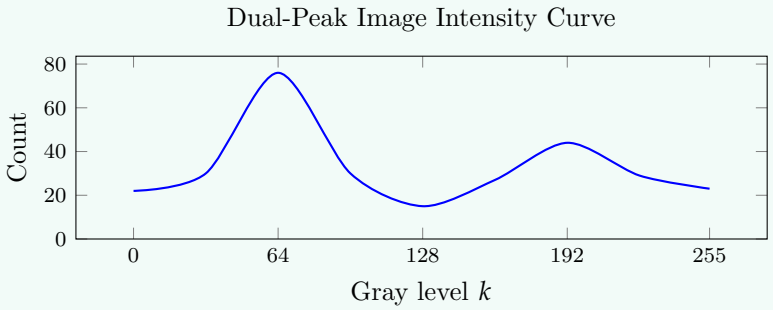
Intensity profile of a bright image: curve shifted towards high gray levels

- A **dark image** shows histogram concentrated on low values.



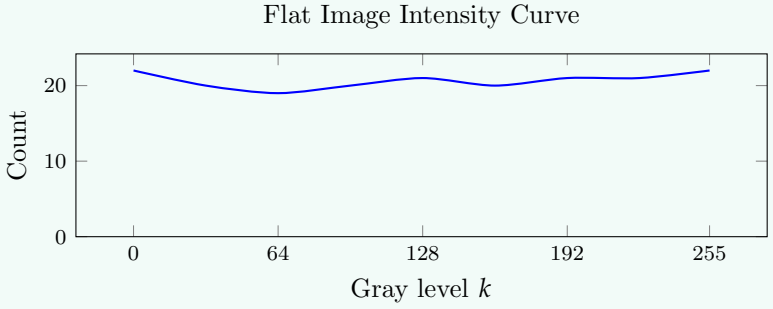
Intensity profile of a dark image: curve shifted towards low gray levels

- A **dual-peak image** indicates presence of both dark (background) and bright (foreground) regions.



Intensity profile of a dual-peak image: two distinct peaks at low and high gray levels

- A **flat histogram** corresponds to uniformly distributed intensities.



Intensity profile of a flat image: uniform distribution across all gray levels

At a Glance:

Histogram Type	Typical Scene	Segmentation Implication
Dark-skewed	Low-light or dark objects	Threshold near lower gray levels
Bright-skewed	Bright background/lighting	Threshold near higher gray levels
Bimodal	Foreground vs. background	Global threshold often effective
Flat/noisy	Low contrast/high noise	Consider contrast stretch or adaptive threshold

1.5 Binarization

Binarization is the process of converting a grayscale image into a binary image, where each pixel is assigned a value of either 0 (black) or 1 (white). This is typically done by applying a threshold to the image, such that pixels above the threshold are set to 1 and those below are set to 0.

$$B(x,y) = \begin{cases} 1 & I(x,y) \geqslant T \\ 0 & I(x,y) < T \end{cases}$$

where T is the threshold.

Choosing an appropriate threshold is crucial for effective binarization. we can use histogram-based methods to select the threshold, by first order statistics or second order statistics (Otsu's method).

1.5.1 Probability and Statistical Prerequisites

Basic Probability Concepts Consider an image histogram with L gray levels $(0, 1, ..., L-1)$. Let p_k denote the normalized probability for gray level k:

$$p_k = \frac{n_k}{N}$$

where n_k is the number of pixels with gray level k, and N is the total pixel count.

Class Probabilities, Means, and Variances

For a given threshold T:
Class Probability

- Probability that the pixel belongs to class 0 (background)

$$\omega_0(T) = \sum_{k=0}^{T} p_k \quad \text{(Probability of class 0 - background - black pixels)}$$

- Probability that the pixel belongs to class 1 (foreground)

$$\omega_1(T) = \sum_{k=T+1}^{L-1} p_k \quad \text{(Probability of class 1 - foreground - white pixels)}$$

Class Means

- Probability that the pixel takes values k given that it belongs to class 0

$$\mu_0(T) = \sum_{k=0}^{T} k \cdot p(k|C_0) \quad \text{(Mean of class 0)}$$

$$\text{where} \quad p(k|C_0) = \frac{p(k, C_0)}{p(C_0)} = \frac{p_k p(C_0|k)}{p(C_0)}$$

$$\text{for} k \in \{0 \dots T\} \quad p(k|C_0) = \frac{p_k}{\omega_0(T)}$$

$$\mu_0(T) = \frac{1}{\omega_0(T)} \sum_{k=0}^{T} k p_k$$

$$\text{also, for} k \in \{T+1 \dots L-1\} \quad p(k|C_0) = 0$$

$$\mu_0(T) = \frac{1}{\omega_0(T)} \sum_{k=0}^{L-1} k p_k$$

- Probability that the pixel takes values k given that it belongs to class 1

$$\mu_1(T) = \sum_{k=T+1}^{L-1} k \cdot p(k|C_1) \quad \text{(Mean of class 1)}$$

$$\text{where} \quad p(k|C_1) = \frac{p(k, C_1)}{p(C_1)} = \frac{p_k p(C_1|k)}{p(C_1)}$$

$$\text{for} k \in \{T+1 \dots L-1\} \quad p(k|C_1) = \frac{p_k}{\omega_1(T)}$$

$$\mu_1(T) = \frac{1}{\omega_1(T)} \sum_{k=T+1}^{L-1} k p_k$$

$$\text{also, for} k \in \{0 \dots T\} \quad p(k|C_1) = 0$$

$$\mu_1(T) = \frac{1}{\omega_1(T)} \sum_{k=0}^{L-1} k p_k$$

- Overall Image Mean

$$\begin{aligned} \mu_T &= \sum_{k=0}^{L-1} k p_k \\ &= \sum_{k=0}^{T} k p_k + \sum_{k=T+1}^{L-1} k p_k \\ &= \mu_0(T)\omega_0(T) + \mu_1(T)\omega_1(T) \end{aligned}$$

Class Variances

- Variance of class 0

$$\begin{aligned} \sigma_0^2(T) &= \sum_{k=0}^{T} (k - \mu_0(T))^2 p(k|C_0) \\ &= \sum_{k=0}^{T} (k - \mu_0(T))^2 \frac{p_k}{\omega_0(T)} \end{aligned}$$

- Variance of class 1

$$\begin{aligned}\sigma_1^2(T) &= \sum_{k=T+1}^{L-1}(k-\mu_1(T))^2 p(k|C_1) \\ &= \sum_{k=T+1}^{L-1}(k-\mu_1(T))^2 \frac{p_k}{\omega_1(T)}\end{aligned}$$

- Total Image Variance

$$\begin{aligned}\sigma^2(T) &= \sum_{k=0}^{L-1}(k-\mu_T)^2 p_k \\ &= \sum_{k=0}^{T}(k-\mu_T)^2 p_k + \sum_{k=T+1}^{L-1}(k-\mu_T)^2 p_k \\ &= \sigma_0^2(T) + \sigma_1^2(T) + \omega_0(T)[\mu_0(T)-\mu_T]^2 + \omega_1(T)[\mu_1(T)-\mu_T]^2\end{aligned}$$

Definition 1.5.1: Within Class Variance $\sigma_w^2(T)$

The variance of pixel intensities within class is given by:

$$\sigma_w^2(T) = \omega_0(T)\sigma_0^2(T) + \omega_1(T)\sigma_1^2(T)$$

Also known as intra-class variance, $\sigma_w^2(T)$ measures the compactness of pixel intensities within each class. **maximizing** $\sigma_w^2(T)$ leads to better class separability.

Definition 1.5.2: Between Class Variance $\sigma_b^2(T)$

The variance of pixel intensities between class is given by:

$$\begin{aligned}\sigma_b^2(T) &= \sigma^2 - \sigma_w^2(T) \\ &= \sigma^2 - \big(\omega_0(T)\sigma_0^2(T) + \omega_1(T)\sigma_1^2(T)\big) \\ &= \omega_0(T)\omega_1(T)\left[\mu_0(T) - \mu_1(T)\right]^2\end{aligned}$$

Also known as inter-class variance, $\sigma_b^2(T)$ measures the separability of pixel intensities between classes. **minimizing** $\sigma_w^2(T)$ leads to better class compactness.

Derivation of $\sigma_b^2(T)$ from definition of $\sigma_w^2(T)$

Derivation of $\sigma_b^2(T)$.

$$\begin{aligned}\sigma^2 &= \sigma_w^2(T) + \sigma_b^2(T) \\ \sigma_b^2(T) &= \sigma^2 - \sigma_w^2(T) \\ &= \sigma^2 - \big(\omega_0(T)\sigma_0^2(T) + \omega_1(T)\sigma_1^2(T)\big)\end{aligned}$$

Now, expand the total variance σ^2. Let $\mu = \omega_0\mu_0 + \omega_1\mu_1$ be the global mean. let's also ignore (T) for simplicity. Then

$$\begin{aligned}\sigma^2 &= \sum_i \omega_i\big(\sigma_i^2 + \mu_i^2\big) - \mu^2 \\ &= \omega_0(\sigma_0^2 + \mu_0^2) + \omega_1(\sigma_1^2 + \mu_1^2) - \mu^2.\end{aligned}$$

Substitute this back:

$$\begin{aligned}\sigma_b^2(T) &= \left[\omega_0(\sigma_0^2+\mu_0^2)+\omega_1(\sigma_1^2+\mu_1^2)-\mu^2\right]-\left(\omega_0\sigma_0^2+\omega_1\sigma_1^2\right)\\ &= \omega_0\mu_0^2+\omega_1\mu_1^2-\mu^2.\end{aligned}$$

Since $\mu = \omega_0\mu_0 + \omega_1\mu_1$, we expand:

$$\begin{aligned}\sigma_b^2(T) &= \omega_0\mu_0^2+\omega_1\mu_1^2-\left(\omega_0\mu_0+\omega_1\mu_1\right)^2\\ &= \omega_0\mu_0^2+\omega_1\mu_1^2-\left(\omega_0^2\mu_0^2+\omega_1^2\mu_1^2+2\omega_0\omega_1\mu_0\mu_1\right)\\ &= \omega_0(1-\omega_0)\mu_0^2+\omega_1(1-\omega_1)\mu_1^2-2\omega_0\omega_1\mu_0\mu_1\\ &= \omega_0\omega_1\mu_0^2+\omega_0\omega_1\mu_1^2-2\omega_0\omega_1\mu_0\mu_1\\ &= \omega_0\omega_1(\mu_0-\mu_1)^2.\end{aligned}$$

😊

1.5.2 Thresholding using First Order statistics

Thresholding using first-order statistics involves selecting a threshold based on the mean or median intensity values of the image histogram. The basic idea is to compute a global threshold that separates the foreground from the background by analyzing the intensity distribution.

Mean Thresholding The threshold is set to the mean intensity value of the image.

$$T = \frac{1}{N}\sum_{x,y} I(x,y)$$

where N is the total number of pixels.

Median Thresholding The threshold is set to the median intensity value, which is more robust to outliers.

$$T = \text{median}(I(x,y))$$

These methods are simple and computationally efficient but may not perform well for images with complex backgrounds or varying illumination.

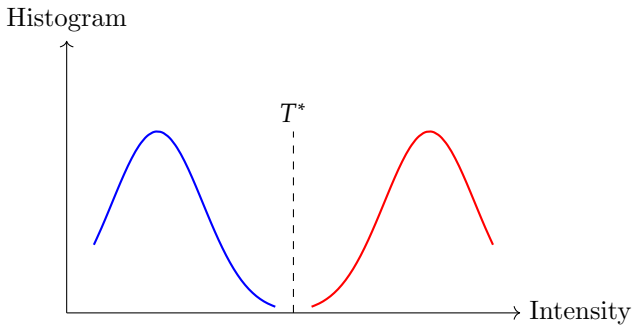
1.5.3 Otsu's Binarization

Otsu's Binarization method uses Second Order Statistics to find the optimal threshold for image binarization.

Definition 1.5.3: Otsu's Binarization

The process of determining the threshold T^* that maximizes the inter-class variance between foreground and background, thus optimally segmenting a bimodal histogram image.

Optimization Criteria Otsu's method selects a threshold T^* to partition the image pixels into two classes (often foreground and background) so that the variance between these classes (**inter-class variance**) is maximized, or equivalently, the variance within each class (**intra-class variance**) is minimized. These criteria are mathematically dual—maximizing $\sigma_b^2(T)$ is identical to minimizing $\sigma_w^2(T)$ since their sum is the total variance σ^2.



A typical bimodal histogram: T^* chosen to best separate two peaks.

Claim 1.5.1 Equivalence of Maximization and Minimization Criteria

Maximizing the between-class variance $\sigma_b^2(T)$ is equivalent to minimizing within-class variance $\sigma_w^2(T)$, as

$$\arg\max_T \sigma_b^2(T) = \arg\min_T \sigma_w^2(T)$$

since σ^2 is fixed for a given image histogram.

Note:-

In practice, Otsu's method typically maximizes $\sigma_b^2(T)$, but formulations based on minimizing $\sigma_w^2(T)$ yield the same optimal threshold T^*.

1.5.4 Method 1: Maximization of Inter-Class Variance

Threshold T^* is selected to maximize the between-class variance $\sigma_b^2(T)$.

$$T^* = \arg\max_T \sigma_b^2(T) = \arg\max_T \left[\omega_0(T)\omega_1(T)\left(\mu_0(T) - \mu_1(T)\right)^2\right]$$

The algorithm for Otsu's binarization by maximizing $\sigma_b^2(T)$ is as follows:

1. Compute the histogram and probabilities $p_i = n_i/N$ for each gray level i.
2. For each possible threshold T (from gray level 1 to $L-2$):
   - Calculate class weights (probabilities): $\omega_0(T)$ and $\omega_1(T)$.
   - Calculate class means: $\mu_0(T)$ and $\mu_1(T)$.
   - Compute between-class variance:

$$\sigma_b^2(T) = \omega_0(T)\,\omega_1(T)\left[\mu_0(T) - \mu_1(T)\right]^2$$

3. Find the threshold T^* that maximizes $\sigma_b^2(T)$.

Example 1.5.1 (Numerical Example)

Suppose an image has normalized histogram values: $p_0 = 0.4$, $p_1 = 0.2$, $p_2 = 0.2$, $p_3 = 0.2$ Try threshold $T = 1$:

$$\omega_0(1) = p_0 + p_1 = 0.6$$
$$\omega_1(1) = 0.4$$
$$\mu_0(1) = \frac{0 \cdot 0.4 + 1 \cdot 0.2}{0.6} = 0.333$$
$$\mu_1(1) = \frac{2 \cdot 0.2 + 3 \cdot 0.2}{0.4} = 2.5$$

Now,

$$\sigma_b^2(1) = 0.6 \times 0.4 \times (0.333 - 2.5)^2 \approx 0.6 \times 0.4 \times 4.694 \approx 1.126$$

similarly for $T = 2$:

$$\begin{aligned}
\omega_0(2) &= p_0 + p_1 + p_2 = 0.8 \\
\omega_1(2) &= 0.2 \\
\mu_0(2) &= \frac{0 \cdot 0.4 + 1 \cdot 0.2 + 2 \cdot 0.2}{0.8} = 0.75 \\
\mu_1(2) &= \frac{3 \cdot 0.2}{0.2} = 3
\end{aligned}$$

Now,

$$\sigma_b^2(2) = 0.8 \times 0.2 \times (0.75 - 3)^2 \approx 0.8 \times 0.2 \times 5.0625 \approx 0.81$$

T	$\omega_0(T)$	$\omega_1(T)$	$\mu_0(T)$	$\mu_1(T)$	$\sigma_b^2(T)$
1	0.6	0.4	0.3333	2.5	**1.1267**
2	0.8	0.2	0.75	3	0.81

Maximum σ_b^2 occurs at $T = 1$, $\sigma_b^2(1) \approx$ **1.1267**.

1.5.5 Method 2: Minimization of Within-Class Variance

Threshold T^* is selected to minimize the within-class variance $\sigma_w^2(T)$.

$$T^* = \arg\min_T \sigma_w^2(T) = \arg\min_T \left[\omega_0(T)\sigma_0^2(T) + \omega_1(T)\sigma_1^2(T)\right]$$

This approach explicitly minimizes intra-class variance $\sigma_w^2(T)$.

1. For each threshold T, calculate:

$$\sigma_w^2(T) = \omega_0(T)\sigma_0^2(T) + \omega_1(T)\sigma_1^2(T)$$

   where $\sigma_k^2(T)$ are variances for classes $k = 0, 1$, based on histogram statistics within each class.

2. Find T^* that minimizes $\sigma_w^2(T)$.

Claim 1.5.2 Proof of Equivalence

Since $\sigma^2 = \sigma_w^2(T) + \sigma_b^2(T)$ for all T, maximizing $\sigma_b^2(T)$ is the same as minimizing $\sigma_w^2(T)$.
Now, because σ^2 is independent of T, for two thresholds T_1, T_2 we have

$$\sigma_b^2(T_1) > \sigma_b^2(T_2) \quad \Longleftrightarrow \quad \sigma_w^2(T_1) < \sigma_w^2(T_2),$$

since subtracting the same constant σ^2 from both sides reverses neither inequalities nor signs but simply gives

$$\sigma^2 - \sigma_b^2(T_1) < \sigma^2 - \sigma_b^2(T_2) \quad \Longleftrightarrow \quad \sigma_w^2(T_1) < \sigma_w^2(T_2).$$

Therefore

$$\arg\max_T \sigma_b^2(T) \;=\; \arg\min_T \sigma_w^2(T)$$

Equivalently, using the explicit form

$$\sigma_b^2(T) = \omega_0(T)\,\omega_1(T)\big(\mu_0(T) - \mu_1(T)\big)^2$$

maximizing this expression over T yields the same T that minimizes $\sigma_w^2(T)$ because

$$\sigma_w^2(T) = \sigma^2 - \sigma_b^2(T).$$

Hence any T that increases the between-class separation $\omega_0\omega_1(\mu_0-\mu_1)^2$ necessarily reduces the within-class scatter and vice versa.
Thus,

$$\text{Maximize } \sigma_b^2(T) \Leftrightarrow \text{Minimize } \sigma_w^2(T)$$

Note:-

- The identity relies on σ^2 being fixed (global histogram is fixed). If the data used to compute σ^2 changed with T, the equivalence would not hold.
- Since $\omega_0, \omega_1 \geq 0$ and μ_0, μ_1 are real, $\sigma_b^2(T) \geq 0$. Maximizing σ_b^2 therefore finds the threshold giving the largest possible separation between class means (weighted by class sizes), which is precisely the threshold that minimizes the residual within-class variance.

1.5.6 Algorithm: Otsu's Binarization (most used)

1. Compute image histogram.
2. For each threshold T:
   - Partition pixels into two classes.
   - Compute $\omega_k(T)$, $\mu_k(T)$, $\sigma_k^2(T)$ as needed.
   - Compute either $\sigma_b^2(T)$ or $\sigma_w^2(T)$.
3. Determine T^* by maximizing $\sigma_b^2(T)$ or minimizing $\sigma_w^2(T)$.
4. Binarize the image: pixels $\leqslant T^*$ are assigned to class 0; others to class 1.

Note:-

Otsu's method is most effective for bimodal histograms, but may be suboptimal for multi-modal or unimodal histograms. Variations and generalizations (multi-level, local thresholding) exist for more complex cases.

1.6 Connected Component Analysis

A *connected component* in a binary image is a maximal group of adjacent foreground pixels, where adjacency is defined by connectivity (typically, 4-connected or 8-connected).

Definition 1.6.1: Connected Component

A subset of foreground pixels (p, q) such that for any pixel p in the subset, there exists a sequence of pixels in the subset linking p to q via connected neighbors.

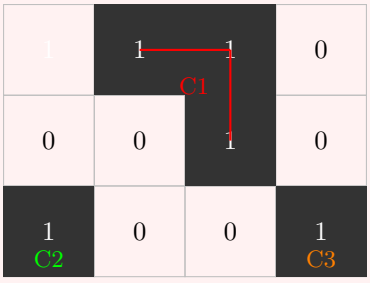
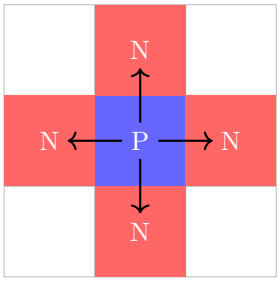


Illustration of 4-connected components in a binary image

Connectivity

- **4-Connectivity:** Each pixel is connected to its immediate horizontal and vertical neighbors.
  Neighbors for a pixel (x, y) are

$$\{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}$$



Plus-shaped 4-neighborhood of pixel P

- **8-Connectivity:** Includes diagonal neighbors as well.
  Neighbors for a pixel (x, y) are

$$\{(x-1, y), (x+1, y), (x, y-1), (x, y+1), (x-1, y-1), (x-1, y+1), (x+1, y-1), (x+1, y+1)\}$$

Definition 1.6.2: Path

Path from (x, y) to (s, t) is a sequence of pixels $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ where $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$, and each consecutive pair of pixels are connected according to the chosen connectivity (4 or 8).

Definition 1.6.3: Connectedness

the pixels (x, y) and (s, t) are connected if there exists a path between them consisting entirely of same polarity pixels (all foreground or all background).

A **4-Connected Component** is a set of pixels that are connected to each other through 4-connectivity paths.

1.6.1 Extraction of Connected Components

Claim 1.6.1 Labeling Algorithms

Every pixel belonging to a connected component can be assigned a unique label using the connected component labeling algorithm.

Definition 1.6.4: Region index array

Region index array is a matrix of the same size as the binary image, where each pixel is assigned a label (integer) corresponding to the connected component it belongs to. Background pixels are typically labeled as 0.

The goal is to assign distinct labels to all connected components in a binary image.

Labeling Algorithm

1. Scan the image pixel by pixel in a defined order (e.g., left-to-right, top-to-bottom).
2. For each foreground pixel, examine its neighbors (depending on the connectivity rule).
3. Assign a new label if no neighbor is labeled; otherwise assign the neighbor's label.

4. If multiple labeled neighbors, assign one and record equivalences.
5. After complete pass, resolve label equivalences: assign unique labels to all connected components.

Example 1.6.1 (Connected Component Labeling)
Say, Given the binary image (rows indexed from 1):

$$B = \begin{matrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{matrix}$$

We use *4-connectivity* and the standard two-pass algorithm (scan row-major). Denote left neighbour by N_L and up neighbour by N_U. Maintain a provisional label counter L (start $L \leftarrow 1$) and an equivalence table (for union-find).

First pass (scan, assign provisional labels and record equivalences).

Scan order (only foreground pixels shown). For each foreground pixel we inspect N_L, N_U and apply the standard rules:

- if both neighbours background, $\Rightarrow$ assign new label L, increment L;
  if $I(i-1,j) = I(i,j-1) = 0$ then, $R(i,j) = L$ and $L++$.
- if only one neighbour is labelled $\Rightarrow$ copy that label;
  if $I(i-1,j) = M$ or $I(i,j-1) = M$ then, $R(i,j) = M$.
- if both neighbours labelled with different labels $\Rightarrow$ assign one of them and record their equivalence.
  if $I(i-1,j) = M$ and $I(i,j-1) = N$ then, $R(i,j) = M$ and record equivalence $M \sim N$.

Step	Pixel	N_L	N_U	Action	Result / equivalences
1	$(1,2)$	0	none	both background $\Rightarrow$ new label	assign 1. $L \leftarrow 2$.
2	$(1,3)$	1	none	left labelled	assign 1.
3	$(2,3)$	0	1	up labelled	assign 1.
4	$(3,1)$	none	0	both background	assign 2. $L \leftarrow 3$.
5	$(3,4)$	0	0	both background	assign 3. $L \leftarrow 4$.

Note: no step produced conflicting labels from N_L and N_U, hence no equivalences were recorded (equivalence sets remain $\{1\},\{2\},\{3\}$).
Provisional labelled image (Region) after first pass:

$$R^{(1)} = \begin{matrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 3 \end{matrix}$$

Equivalence resolution (union-find / flattening).
Equivalence table (pairs recorded during first pass): none in this example. Therefore canonical representatives are

$$\text{repr}(1) = 1, \quad \text{repr}(2) = 2, \quad \text{repr}(3) = 3.$$

Second pass (replace provisional labels by their canonical representatives).
Apply

$$R_{\text{final}}(x,y) \leftarrow \text{repr}\big(R^{(1)}(x,y)\big)$$

Since representatives are identity, the final labelled image is

$$R_{\text{final}} = \begin{matrix} 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ \mathbf{2} & 0 & 0 & \mathbf{3} \end{matrix}$$

Components (explicit pixel sets).

$$C_1 = \{(1,2),(1,3),(2,3)\}, \quad C_2 = \{(3,1)\}, \quad C_3 = \{(3,4)\}.$$

Note:-

Correctly extracting connected components is crucial for counting objects, segmenting regions, and preparing for further analysis such as shape feature extraction.

1.7 Binary Image Filters

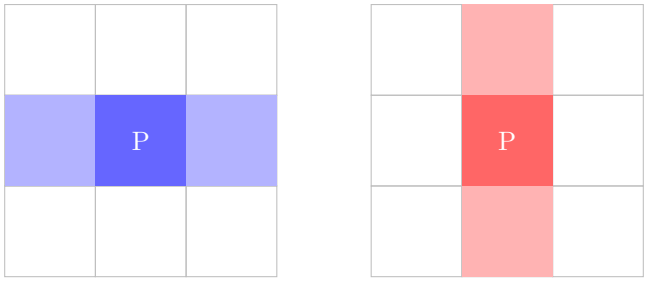
Binary images require specialized filtering techniques to process and enhance them, typically via mathematical morphology.

1.7.1 Window

A **window** or **kernel** in filtering is a small submatrix (usually square, e.g., 3×3) that moves systematically over the image, determining how each pixel should be transformed based on its neighborhood.

1D Window

In 1D signals, a window is a segment of the signal used for processing. For example, a 3-point window might include samples $x[m, n-1], x[m, n], x[m, n+1]$ for a row window and $x[m-1, n], x[m, n], x[m+1, n]$ for a column window.



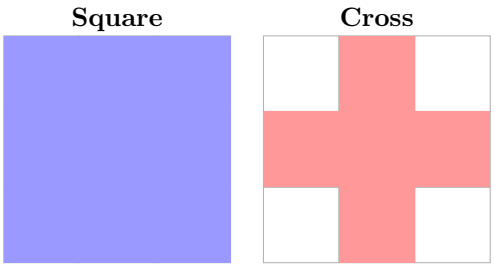
Row Window Column Window

2D Window

Windows are mostly non-casual, means pixels can be influenced by their neighbors in both dimensions. A window can be defined as a rectangular region centered around a pixel, encompassing its surrounding pixels.

Few shapes include square, Cross, circle, diamond, etc.

Common Window Shapes



1.7.2 Binary Morphology

A set of techniques used in image processing for analyzing and processing binary images based on shapes. It involves operations that probe and transform the structure of objects in the image using a defined window (structuring element).

Window Notation Let,

$$B_i = (p_i, q_i) \quad \text{s.t.} \quad (p_i, q_i) \in B$$

then, A window is given as,

$$B = \{B_1, B_2, B_3, \ldots, B_{2p+1}\}$$

Example 1.7.1 (Window Example)
Consider a ROW window with p pixels centered at $(0, 0)$:

$$\text{ROW}(2p+1) = \{(-p, 0), (-p+1, 0), \ldots, (0, 0), \ldots, (p-1, 0), (p, 0)\}$$

Consider a SQUARE 3×3 window centered at pixel $(0, 0)$:

$$\text{SQUARE}(9) = \{(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 0), (0, 1), (1, -1), (1, 0), (1, 1)\}$$

Window Set Given a image I and a Window B, the window set is given as

$$B \downarrow I(i, j) = \{I(i+p, j+q) | (p, q) \in B\}$$

1.7.3 Image Edges and Window Boundaries

Applying filters near image edges requires care, as the window may extend outside the image domain. Strategies include:

- **Padding:** Extend image with zeros or mirrored edges.
- **Ignore:** Only process central pixels with complete neighborhoods.
- **Wrap-around:** For mathematical convenience (rare in practice).

1.7.4 Binary Filter

Definition 1.7.1: Binary Filter

A transformation applied to a binary image using a specified window (kernel), resulting in output pixels determined by logical operations (AND, OR, etc.) on the neighborhood.

Let G be a binary operation on a windowed set, then the binary filtered image is given as:

$$J(i, j) = G(B \downarrow I(i, j)) \qquad \forall (i, j) \in I$$

Note:-

In morphological operations, careful handling of edges is necessary to avoid artifacts such as unwanted erosion or dilation at the image boundaries.

Fundamental Morphological Operations

Mathematical morphology operates using set theory to process shapes in binary images. The two core operations are **dilation** and **erosion**, with further derived operations.

Definition 1.7.2: Dilation

For a binary image I and structuring element B, the dilation $J = \text{DILATE}(I, B) = I \oplus B$ is defined as:

$$J(i,j) = OR\{B \downarrow I(i,j)\}$$

where

$$OR\{x_1, x_2, \ldots, x_n\} = \bigvee_k x_k = x_1 + x2 + \ldots + x_n = \max(x_1, x_2, \ldots, x_n)$$

is the logical OR operation or the Maximum pixel value picker over the set.

Definition 1.7.3: Erosion

For a binary image I and structuring element B, the erosion $J = \text{ERODE}(I, B) = I \ominus B$ is:

$$J(i,j) = AND\{B \downarrow I(i,j)\}$$

where

$$AND\{x_1, x_2, \ldots, x_n\} = \bigwedge_k x_k = x_1 \cdot x_2 \cdot \ldots \cdot x_n = \min(x_1, x_2, \ldots, x_n)$$

is the logical AND operation or the Minimum pixel value picker over the set.

Algorithmic Steps for binary image

- **Dilation:** For each pixel, if any neighbor within B is foreground, set pixel to 1.
- **Erosion:** For each pixel, if all neighbors within B are foreground, set pixel to 1; else 0.

Note:-

Dilation expands object boundaries, while erosion shrinks them. The choice of structuring element B (shape and size) significantly affects the outcome.

Duals

Dilation and erosion are dual operations in mathematical morphology. This means that the dilation of the complement of an image is equal to the complement of the erosion of the image:

$$\text{DILATE}(I^c, B) = (\text{ERODE}(I, B))^c$$

where I^c is the complement of the image I.

This comes straight from the definitions of dilation and erosion, as the operations are fundamentally linked through the concept of set complementation, i.e., $(x_1 \bar{+} x_2) = \bar{x_1} \cdot \bar{x_2}$.

Derived Morphological Operations

Majority Filter The majority filter is a simple yet effective morphological operation used to remove noise from binary images. It works by replacing each pixel's value with the majority value of its neighbors within a specified window.

$$J(i,j) = \text{MAJORITY}\{B \downarrow I(i,j)\}$$

This can help to smooth out small irregularities in the image.

Boundary Detection The boundary of objects in a binary image can be extracted as:

$$\text{Boundary}(I) = \text{XOR}\{I, \text{ERODE}(I, B)\}$$

Claim 1.7.1 Morphological Property

- Dilation expands, erosion shrinks, and their combinations (opening, closing) provide sophisticated noise removal and shape manipulation.
- Morphological operations are inherently non-linear, making them robust to noise and variations in object shape.
- Majority filter effectively reduces salt-and-pepper noise by enforcing local consistency.
- Boundary detection highlights object edges, aiding in shape analysis and recognition.

Chapter 2

Gray Scale Image Processing

2.1 Point Operations

Point operations are image transformations where each output pixel is computed solely from its corresponding input pixel, without regard for neighboring values.

Definition 2.1.1: Point Operation

A transformation applied to each pixel individually, $J(x, y) = T[I(x, y)]$, where $I(x, y)$ is the input image and T is a point-wise mapping function.

These operations can be linear or non-linear, and are different from local operators/spatial filters (i.e., those that consider neighboring pixels such as ERODE, DILATE, etc.).

No explicit neighborhood information is used in point operations, and no explicit modifications based on spatial context. however, they can still be influenced by global image properties (e.g., overall brightness, contrast).

2.1.1 Image Offset (Brightness Adjustment)

The simplest point operation is the offset, where a constant value c is added to all pixel intensities:

$$J(x, y) = I(x, y) + c$$

This increases (or decreases) the brightness of the image uniformly.

- If $c > 0$, the image becomes brighter.
- If $c < 0$, the image becomes darker.

Example 2.1.1 (Image Offset Application)

Suppose $I_1, I_2, \ldots, I_N$ are N images of the same scene taken under different lighting conditions (different exposures). If we wish to equalise all their average intensities to $\frac{K}{2}$ (where K is number of intensity levels), a point operation can be used to normalize brightness across these images by applying an offset to each image:

$$J_i(x, y) = I_i(x, y) + c_i$$

where c_i is the offset computed as:

$$c_i = \frac{K}{2} - \text{mean}(I_i)$$

2.1.2 Image Scaling (Contrast Adjustment)

Image scaling multiplies all pixel intensities by a constant factor a:

$$J(x, y) = a \cdot I(x, y)$$

A scaling factor $a > 1$ increases contrast, while $0 < a < 1$ darkens the image.

2.1.3 Offset and Scaling

Combining offset and scaling operations allows for more flexible image adjustments. The combined operation can be expressed as:

$$J(x, y) = a \cdot I(x, y) + c$$

Where a is the scaling factor and c is the offset. This allows for both contrast adjustment (via scaling) and brightness adjustment (via offset) in a single operation.

Example 2.1.2 (Negative Image)

The negative of an image is obtained by inverting the pixel values by choosing :

- image offset $c = K - 1$, Where K is the number of intensity levels.
- image scaling $a = -1$

Thus,

$$J(x, y) = K - 1 - I(x, y)$$

This operation effectively reverses the brightness levels, making dark areas light and vice versa.

Example 2.1.3 (Full Scale Contrast stretch)

Let A & B be the minimum and maximum intensity values in the image respectively. We wish to stretch the contrast to full scale $[0, K-1]$. This can be achieved from $aA + c = 0$ and $aB + c = K - 1$, we can solve for a and c, we get:

- image scaling $a = \frac{K-1}{B-A}$
- image offset $c = -\frac{K-1}{B-A} \cdot A$

Thus,

$$\begin{aligned} J(x, y) &= \frac{K-1}{B-A} \cdot I(x, y) - \frac{K-1}{B-A} \cdot A \\ &= \frac{K-1}{B-A} \cdot (I(x, y) - A) \end{aligned}$$

This allows us to stretch the pixel values to cover the full range of intensities, hence enhances the overall contrast of the image.

2.1.4 Non-linear Point Operations

Non-linear point operations are techniques that apply non-linear transformations to the pixel values of an image. These operations can enhance certain features or suppress others, depending on the specific transformation applied.

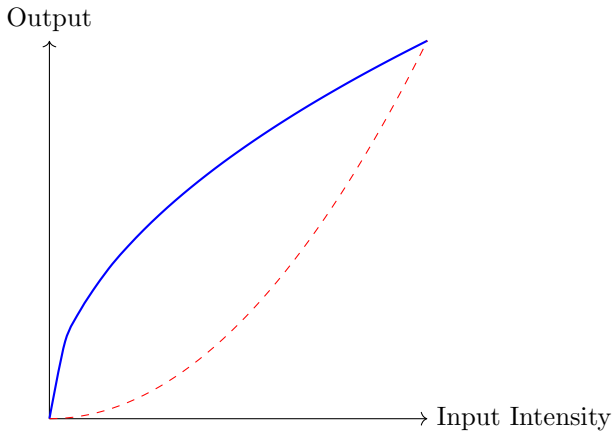
Common Examples

- **Gamma Correction:**

  $$J(x, y) = c \cdot I(x, y)^{\gamma} \qquad \gamma > 0$$

  Where c is a scaling factor and γ is the gamma value.

  - $0 < \gamma < 1$: Enhances dark regions (gamma correction).
  - $\gamma > 1$: Enhances bright regions (inverse gamma correction).

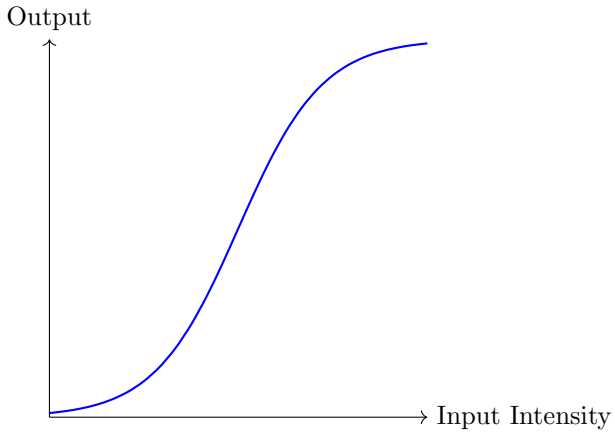


Gamma Correction (Blue : $\gamma = 0.5 \rightarrow I^{\frac{1}{2}}$, Red : $\gamma = 2 \rightarrow I^2$)

- **Sigmoid Correction:**

$$J(x,y) = \frac{K}{1 + \exp(-\beta[I(x,y) - \alpha])}$$

Where K is maximal intensity, α the mid-point, and β the slope. Useful for contrast enhancement around a chosen intensity.



Sigmoid Correction Curve

- **Other Non-linear Operations:**
  - *Logarithmic Correction:* $g(x,y) = c \cdot \log(1 + f(x,y))$
  - *Exponential Correction:* $g(x,y) = a \cdot \exp(f(x,y)/b)$
  - *Piece-wise Linear Transformations:* used for specific intensity mapping (e.g., contrast stretching).

Note:-

Non-linear point operations are often employed to address unique characteristics of image acquisition (camera response, display systems) or to prepare images for further processing steps like thresholding or segmentation.

2.2 Histogram Equalization

Histogram equalization is a method to adjust image intensities to enhance contrast. The histogram of an image is "flattened" (equalized), redistributing pixel values as uniformly as possible over the intensity range.

Definition 2.2.1: Histogram Equalization

A point-wise image transformation that remaps the intensity distribution so that the output histogram is (approximately) uniform. This increases the dynamic range and enhances global contrast.

Given a gray scale image with intensities r in $[0, K-1]$ and histogram $h(r)$, histogram equalization computes a transformation HE as the cumulative distribution function (CDF):

$$HE(r) = \sum_{k=0}^{r} \frac{h(k)}{MN}$$

where MN is the total number of pixels.

Definition 2.2.2: Histogram Equalization Transformation

The transformation $s = HE(r)$ maps each input intensity r to an output intensity s so that s is approximately uniformly distributed over $[0, K-1]$.

2.3 Spatial Smoothing Filters

Spatial smoothing filters are a concept in image processing used to reduce noise and smooth out variations in an image by averaging or combining pixel values in a local neighborhood.

Types of Filters:

1. **Averaging Filter:** Replaces each pixel with the average of its neighbours.

$$\omega_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

2. **Gaussian Filter:** Weights the neighbors according to a Gaussian distribution, giving higher weight to pixels closer to the center.

$$h(m,n) = K\, e^{-(m^2+n^2)}, \qquad m,n \in \{-1,0,1\}.$$

K is chosen such that

$$\sum_{m,n} h(m,n = 1$$

or

$$K = \left(\sum_{m=-1}^{1} \sum_{n=-1}^{1} e^{-(m^2+n^2)} \right)^{-1}.$$

3. **Median Filter:** A nonlinear filter that replaces each pixel with the median of its neighborhood. Useful for removing *salt-and-pepper noise*.

4. **Weighted Average Filter:** Similar to the averaging filter, but assigns more importance to the central pixel. Example:

$$\omega_3 = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

2.4 Applications of Spatial Smoothing Filters

Spatial smoothing filters are widely used in image processing tasks. Some common applications include:

1. **Denoising:** By averaging neighboring pixels, smoothing filters reduce the impact of random noise, making the image cleaner for further processing.

2. **Preprocessing before Binarization:** Smoothing helps suppress small intensity variations so that threshold-based binarization produces cleaner object boundaries.

3. **Edge Detection Preprocessing:** Many edge detection algorithms (such as Canny) first apply Gaussian smoothing to reduce noise. This ensures that false edges caused by random pixel fluctuations are minimized.

4. **Image Pyramid / Downsampling:** Before reducing image resolution, Gaussian smoothing is applied to avoid aliasing. This is a fundamental step in building image pyramids for multi-scale analysis (e.g., object detection and image compression).

2.5 Spatial Sharpening Filters

$$\frac{\partial f}{\partial x} = f(m+1,n) - f(m,n) = \nabla f_x(m,n) \rightarrow \textit{Image gradient in x-direction}$$

$$\frac{\partial f}{\partial y} = f(m,n+1) - f(m,n) = \nabla f_y(m,n) \rightarrow \textit{Image gradient in y-direction}$$

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} &= \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) = (f(m+1,n) - f(m,n)) - (f(m,n) - f(m-1,n)) \\ &= f(m+1,n) - 2f(m,n) + f(m-1,n) \end{aligned}$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial y}\right) = f(m,n+1) - 2f(m,n) + f(m,n-1)$$

2.6 Laplacian of an Image

The Laplacian is a second-order derivative operator that highlights regions of rapid intensity change in an image. It works by comparing a pixel's value with its neighbors using convolution masks, making edges stand out while smooth regions stay near zero. At edges, it produces strong positive or negative responses, and zero-crossings indicate the actual edge locations. Because it amplifies rapid changes, the Laplacian is highly sensitive to noise. To reduce this effect, it is often combined with Gaussian smoothing, leading to the Laplacian of Gaussian (LoG) operator. The equation for the Laplacian is given by

$$\begin{aligned} \nabla^2 f &= \nabla^2 f_x + \nabla^2 f_y \\ &= \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \\ &= f(m+1,n) + f(m-1,n) + f(m,n+1) + f(m,n-1) - 4f(m,n) \end{aligned}$$

- 3×3 **filter (Isotropic):**

$$h(m,n) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- **Another variant:**

$$h(m,n) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow \textit{Smooth regions will have zero convolved value}$$

Application of Laplacian

- Highlight intensity discontinuity.
- Deemphasize regions with slowly varying intensity levels.

Image Sharpening using Laplacian Filters

The sharpened image is given by:

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

where $c = -1$ if the central coefficient of the Laplacian filter is negative.

In terms of convolution:

$$\begin{aligned} g(m,n) &= f(m,n) + c\,[f(m,n) * h(m,n)] \\ &= f(m,n) + f(m,n) * h'(m,n) \end{aligned}$$

Expanding the discrete Laplacian:

$$g(m,n) = f(m,n) + \Big[\, 4f(m,n) - \big(f(m+1,n) + f(m-1,n) + f(m,n+1) + f(m,n-1)\big) \,\Big]$$

Unsharp Masking and High-Boost Filtering

Let $\bar{f}(x,y)$ be a blurred (unsharp/smoothed) version of the original image $f(x,y)$.

$$g_{\text{mask}}(x,y) = f(x,y) - \bar{f}(x,y)$$

The sharpened image is obtained as:

$$g(x,y) = f(x,y) + k\, g_{\text{mask}}(x,y)$$

where

$$\begin{aligned} k = 1 &\implies \text{Unsharp Masking (basic sharpening)} \\ k > 1 &\implies \text{High-Boost Filtering (stronger sharpening effect)} \end{aligned}$$

2.7 Geometric Operation

We have an input image $I(i,j)$ and the output image is defined as

$$J(i,j) = I(a_1(i,j), a_2(i,j)).$$

- **Image Translation**

$$\begin{aligned} a_1(i,j) = i - b_1, \quad a_2(i,j) = j - b_2 \\ J(i,j) = I(i - b_1,\, j - b_2) \end{aligned}$$

For example, if $b_1 = b_2 = 2$:

$$J(2,2) = I(0,0), \quad J(0,0) = I(-2,-2) = 0 \quad (\text{outside image} \to 0).$$

- **Image Rotation**
  A counter-clockwise rotation by angle θ is given by:

$$a_1(i,j) = i\cos\theta - j\sin\theta$$

Rotation of axis in counter-clockwise direction

$$a_2(i,j) = i\sin\theta + j\cos\theta$$

Bibliography