Lecture Notes on Digital Image Processing (DIP)

Mr. Rahimul Islam Mazumdar

Module 1: Introduction & Fundamentals

What is Digital Image Processing (DIP)?

Digital Image Processing is the use of computer algorithms to perform processing, enhancement, analysis, and interpretation of digital images. It involves converting images into a numerical form (pixels) and applying mathematical techniques to extract useful information, improve image quality, or make decisions based on image content.

Applications of DIP

1. Medical Imaging

- X-ray, MRI, CT scans, Ultrasound – Image enhancement, noise removal, segmentation of tissues/organs.
- Cancer detection – Identifying tumors in mammograms, brain scans.
- Surgical assistance – Image-guided robotic surgery.
- Disease diagnosis – Automated detection of diabetic retinopathy, heart diseases.

2. Satellite Imaging

- Remote sensing – Extracting land cover, vegetation, and water bodies.
- Weather forecasting – Processing satellite images of clouds and storms.
- Disaster management – Flood, earthquake, and wildfire detection.
- Urban planning – Monitoring population density, road mapping.

3. Biometrics

- Face recognition – Security and surveillance systems.
- Fingerprint recognition – Authentication in devices and forensic science.
- Iris recognition – High-security authentication systems.
- Gait recognition – Identifying individuals by walking patterns.

4. Robotics & Computer Vision

- Object detection & recognition – Robots identifying and interacting with objects.
- Navigation & path planning – Autonomous vehicles using visual sensors.
- Industrial automation – Quality inspection in manufacturing.
- Human-robot interaction – Gesture and facial expression recognition.

What is Digital Image ?

A **digital image** is a representation of a real-world visual scene in a **numerical (discrete) form** that a computer can process.
It is composed of tiny elements called **pixels (picture elements)**, each storing an intensity or color value. When arranged in rows and columns, these pixels form the complete image.

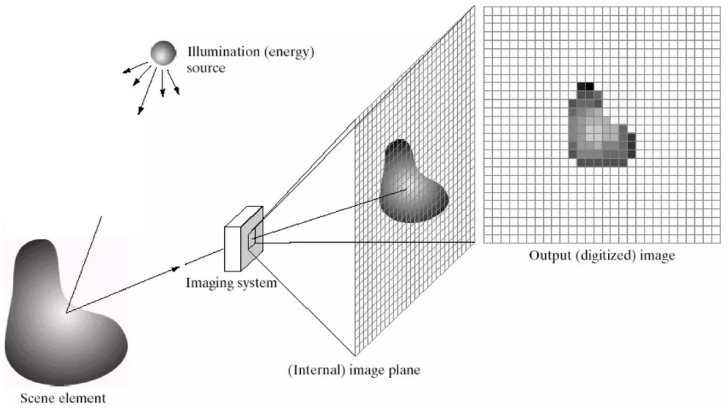
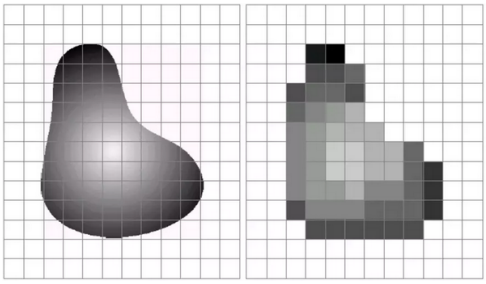
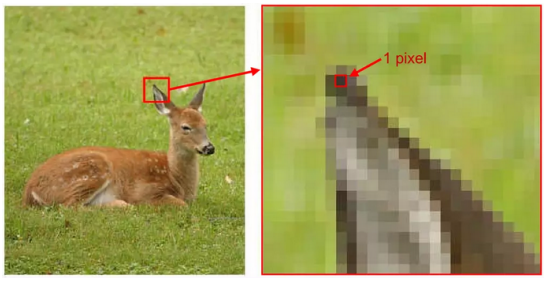


Figure 1: Representation of real world scene in digital pixel form

Pixel values typically represent gray levels, colours, heights, opacities etc **Remember digitization implies that a digital image is an approximation of a real scene.**



(a) Scene to pixel



(b) Real scene to pixel assignment

Figure 2: Digital image representation from a real scene

Formation of a Digital Image

1. Image Acquisition (Sensing)

- The first step is to capture a real-world scene using a sensor (e.g., camera, satellite sensor, medical scanner).
- The sensor converts **light energy (photons)** into **electrical signals**.

2. Sampling

- The continuous image is divided into a finite grid of points (pixels).
- This process is called **spatial sampling**.
- Example: breaking a 512×512 continuous image into 262,144 discrete pixels.

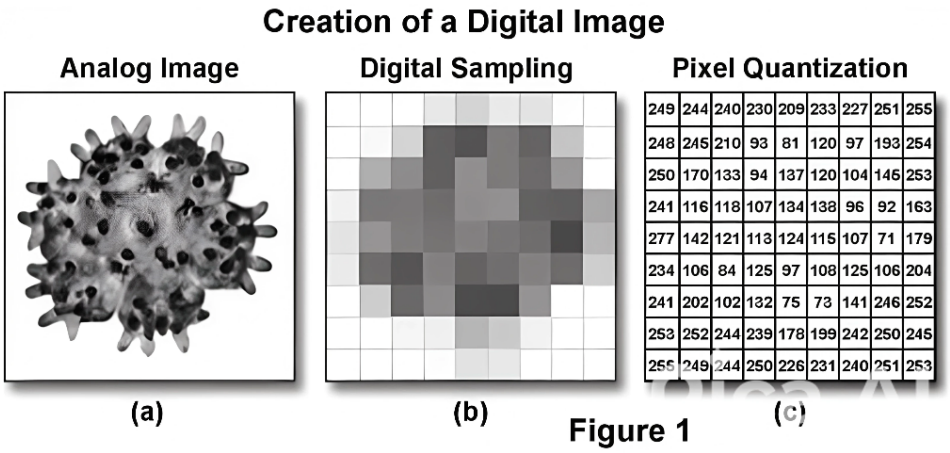


Figure 3: Formation of digital image

3. Quantization

- Each sampled pixel's intensity value is mapped into a finite set of levels.
- For example:
  - **1-bit image** → 2 levels (black & white).
  - **8-bit image** → 256 levels (0–255).
  - **24-bit color image** → 16.7 million possible colors (256 for each RGB channel).

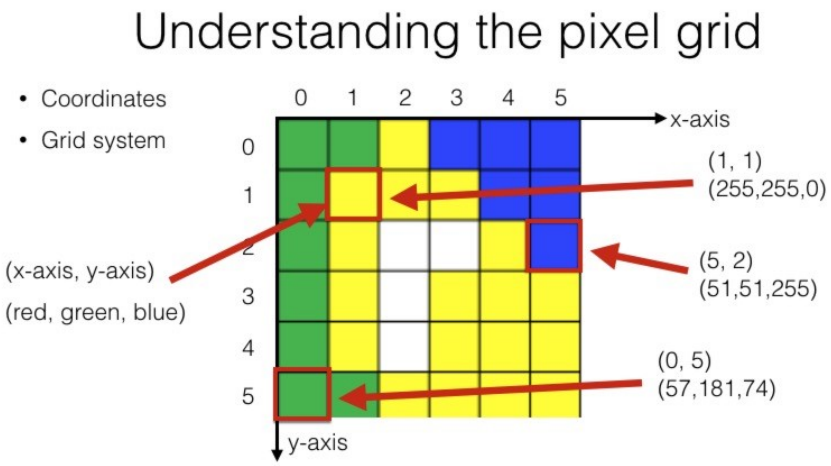


Figure 4: Understanding the pixel grid for a colour image

4. Storage / Representation

- The digital image is stored in memory as a matrix of pixel values.
- Example: A grayscale digital image can be represented as a 2D array:

$$\begin{bmatrix} 0 & 120 & 255 \\ 34 & 200 & 180 \\ 90 & 50 & 0 \end{bmatrix}$$

In short:
Analog Image (continuous) → Sensing → Sampling → Quantization → Digital Image (discrete).

→ RGB Color Image (24-bit)

An **RGB color image** is a digital image where each pixel is represented by three color components:

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

These three channels combine to produce a wide range of colors based on the **additive color model**.

Bit Depth

- In a **24-bit RGB image**, each color channel (R, G, B) is represented using **8 bits**.
- So:

$$8 \text{ bits (R)} + 8 \text{ bits (G)} + 8 \text{ bits (B)} = 24 \text{ bits per pixel}$$

Number of Colors

- Each 8-bit channel can represent values from 0 to 255 (256 levels).
- Total number of possible colors:

$$256 \times 256 \times 256 = 16,777,216 \text{ colors } (\approx 16.7 \text{ million})$$

- Therefore, a 24-bit RGB image is also called a **True Color Image**.

Pixel Representation

- Each pixel is stored as a triplet (R, G, B).
- Examples:
  - (255, 0, 0) → Pure Red
  - (0, 255, 0) → Pure Green
  - (0, 0, 255) → Pure Blue
  - (255, 255, 255) → White
  - (0, 0, 0) → Black

Storage Example

- For an image of size 512×512:
  - Number of pixels = $262,144$.
  - Memory required = $262,144 \times 3$ bytes $\approx$ 786 kB.

Summary: A **24-bit RGB image** uses 8 bits for each color channel, giving about **16.7 million colors**, sufficient to represent natural scenes with high visual quality.

→ Key Points

- A digital image is a **2D function** $f(x, y)$, where:
  - $x, y \rightarrow$ spatial coordinates

  - $f(x, y) \rightarrow$ intensity (gray level or color value) at that point
- In practice, x, y, and $f(x, y)$ are all **finite and discrete** (unlike analog/continuous images).
- **Types of digital images:**
  - **Binary image:** only 0 (black) and 1 (white).
  - **Grayscale image:** intensity values (0–255 for 8-bit images).
  - **Color image:** usually represented in RGB (Red, Green, Blue) channels.
  - **Multispectral/Hyperspectral images:** captured beyond visible range (e.g., satellite images).

Example

- A grayscale image of size 512×512 has $262,144$ pixels.
- Each pixel can hold a value between 0 (black) and 255 (white).

Process Level	Input	Output	Examples
Low-Level Process (Image Processing)	Image	Image/ features	Noise removal, image sharpening
Mid-Level Process (Computer Vision)	Image	Attributes	Object recognition, segmentation
High-Level Process (Computer Vision)	Attributes	Understanding	Scene understanding, autonomous navigation

Table 1: Comparison of Low, Mid, and High-Level Processes in Image Analysis

→Image Processing- works on pixels, Computer vision-works on Attributes/ Features. or
→Classical (low-level) segmentation → Image Processing task.
→Semantic/Instance/panoptic segmentation → Computer Vision task.

Listing 1: Reading and displaying an image using OpenCV

```
import cv2
# Read the image (provide the correct path to your image file)
image = cv2.imread('sample_image.jpg')
or,
image = cv2.imread(r'C:\Users\mazum\Indian Institute of
   Technology Guwahati\Anirban Dasgupta - Manual Distraction\
   test_image.jpg')

# Display the image in a window
cv2.imshow('Displayed Image', image)

# Wait until a key is pressed
cv2.waitKey(0)

# Close all OpenCV windows
cv2.destroyAllWindows()
```

Module 2: Image Enhancement in Spatial Domain

Intensity Transformations: Negative, Log, and Power-Law

Image enhancement in the spatial domain is performed by directly manipulating the pixel intensities of an image. Among the commonly used intensity transformation techniques are:

1. **Image Negative:** The negative transformation is defined as:

$$s = L - 1 - r$$

   where:
   - L = total number of gray levels (e.g., 256 for an 8-bit image)
   - r = input pixel intensity
   - s = output pixel intensity

   This technique is useful for enhancing white or gray details embedded in dark regions of an image.

1. **Log Transformation:** The log transformation is defined as:

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

   where:
   - c is a scaling constant
   - r is the input intensity (normalized)

   This transformation expands the values of darker pixels while compressing the higher intensity levels. It is useful in situations where the dynamic range of the image is large, such as Fourier spectrum visualization.

2. **Power-Law (Gamma) Transformation:** The power-law transformation is defined as:

$$s = c \cdot r^{\gamma}$$

   where:
   - γ (gamma) is the exponent factor that controls enhancement.
   - $\gamma < 1$ brightens the image, while $\gamma > 1$ darkens it.

   This method is widely used in *gamma correction* for display devices and image enhancement.

These intensity transformations help improve visual appearance or prepare the image for further analysis in various computer vision and image processing applications.

Contrast Stretching, Histogram Equalization & Specification

Image enhancement techniques such as contrast stretching, histogram equalization, and histogram specification are widely used to improve the visual quality and feature representation of images. Their mathematical foundations are described below.

1. Contrast Stretching

Contrast stretching enhances the dynamic range of an image by linearly mapping the intensity values from a narrow range to the full intensity range $[0, L-1]$, where L is the number of gray levels (e.g., $L = 256$ for an 8-bit image).

If r is the input pixel intensity and s is the output intensity, the mapping function is:

$$s = \frac{(r - r_{\min})}{(r_{\max} - r_{\min})} \cdot (L-1)$$

where: - $r_{\min}$: Minimum intensity in the input image - $r_{\max}$: Maximum intensity in the input image This operation stretches the low contrast range $[r_{\min}, r_{\max}]$ to $[0, L-1]$, improving the image contrast.
Note: Contrast stretching spreads values linearly from 0 to 9.

Example: Image Enhancement Techniques Let the original 3×3 grayscale image I be:

$$I = \begin{bmatrix} 52 & 55 & 61 \\ 59 & 79 & 61 \\ 66 & 64 & 58 \end{bmatrix}$$

where the pixel intensity range is $L = 0$ to 255 (8-bit image). The transformation is defined as:

$$s = \frac{(r - r_{\min})}{(r_{\max} - r_{\min})} \cdot (L-1)$$

For this image:

$$r_{\min} = 52, \quad r_{\max} = 79$$

For pixel intensity $r = 52$:

$$s = \frac{(52 - 52)}{(79 - 52)} \cdot 255 = 0$$

For pixel intensity $r = 79$:

$$s = \frac{(79 - 52)}{27} \cdot 255 = 255$$

For pixel intensity $r = 61$:

$$s = \frac{(61 - 52)}{27} \cdot 255 \approx 85$$

The resulting stretched image is:

$$I_{\text{CS}} = \begin{bmatrix} 0 & 28 & 85 \\ 66 & 255 & 85 \\ 132 & 122 & 57 \end{bmatrix}$$

2. Histogram Equalization

Histogram equalization redistributes the intensity levels so that the histogram is approximately uniform.
1. Compute the probability distribution:

$$p(r_k) = \frac{n_k}{MN}$$

2. Compute the cumulative distribution function (CDF):

$$c(r_k) = \sum_{j=0}^{k} p(r_j)$$

3. Map each pixel using:

$$s_k = (L-1) \cdot c(r_k)$$

Note: Histogram equalisation redistributes values nonlinearly, so crowded intensities (like 4's) get more spread out, giving better visibility in dense regions.

Example: Histogram Equalization Given the original 3×3 grayscale image:

$$I = \begin{bmatrix} 52 & 55 & 61 \\ 59 & 79 & 61 \\ 66 & 64 & 58 \end{bmatrix}$$

Let $L = 256$ (8-bit), total pixels $MN = 9$.

Step 1: Unique intensities and histogram counts Unique intensities in ascending order: $\{52, 55, 58, 59, 61, 64, 66, 79\}$.

r_k	n_k
52	1
55	1
58	1
59	1
61	2
64	1
66	1
79	1

Step 2: PDF (probability of occurrence)

$$p(r_k) = \frac{n_k}{MN} \quad \Rightarrow$$

r_k	$p(r_k)$
52	$\frac{1}{9} \approx 0.111$
55	$\frac{1}{9} \approx 0.111$
58	$\frac{1}{9} \approx 0.111$
59	$\frac{1}{9} \approx 0.111$
61	$\frac{2}{9} \approx 0.222$
64	$\frac{1}{9} \approx 0.111$
66	$\frac{1}{9} \approx 0.111$
79	$\frac{1}{9} \approx 0.111$

Step 3: CDF (cumulative distribution function)

$$c(r_k) = \sum_{j \le k} p(r_j)$$

r_k	$c(r_k)$
52	0.111
55	0.222
58	0.333
59	0.444
61	0.666
64	0.777
66	0.888
79	0.999

Step 4: Equalization mapping

$$s_k = (L-1)\, c(r_k) = 255\, c(r_k)$$

(Round to nearest integer.)

r_k	s_k
52	$\lfloor 255 \cdot 0.111 \rceil = 28$
55	$\lfloor 255 \cdot 0.222 \rceil = 57$
58	$\lfloor 255 \cdot 0.333 \rceil = 85$
59	$\lfloor 255 \cdot 0.444 \rceil = 113$
61	$\lfloor 255 \cdot 0.666 \rceil = 170$
64	$\lfloor 255 \cdot 0.777 \rceil = 198$
66	$\lfloor 255 \cdot 0.888 \rceil = 226$
79	$\lfloor 255 \cdot 0.999 \rceil = 255$

Step 5: Apply mapping to each pixel

$$I_{\text{HE}} = \begin{bmatrix} 28 & 57 & 170 \\ 113 & 255 & 170 \\ 226 & 198 & 85 \end{bmatrix}$$

Result: The output I_{HE} has intensities spread more uniformly over $[0, 255]$, improving global contrast.

3. Histogram Specification (Matching)

Histogram specification (or matching) modifies the intensity distribution of an image to match a target histogram. This is particularly useful when a certain image appearance or style is desired.

Steps: 1. Perform histogram equalization on the input image to get a uniform distribution:

$$s = T(r)$$

2. Perform histogram equalization on the target histogram to get the mapping:

$$v = G(z)$$

3. Find the inverse mapping G^{-1} to match the output intensity:

$$z = G^{-1}(s)$$

Here: - $T(r)$: CDF of the input image - $G(z)$: CDF of the target histogram - z: Final intensity level in the output image

This ensures that the processed image has an intensity distribution similar to the target histogram.

Example: Histogram Specification (Matching) Given the original 3×3 grayscale image:

$$I = \begin{bmatrix} 52 & 55 & 61 \\ 59 & 79 & 61 \\ 66 & 64 & 58 \end{bmatrix}$$

Let $L = 256$ (8-bit), total pixels $MN = 9$. We want to match the histogram of I to a desired histogram (target). **Step 1: Compute the histogram of the original image** Unique intensities in ascending order: $\{52, 55, 58, 59, 61, 64, 66, 79\}$

r_k	n_k	$p_r(r_k) = n_k/9$
52	1	0.111
55	1	0.111
58	1	0.111
59	1	0.111
61	2	0.222
64	1	0.111
66	1	0.111
79	1	0.111

Step 2: Define the desired (target) histogram Suppose the desired histogram (uniform distribution for simplicity) is:

z_q	n_q	$p_z(z_q)$
0	1	0.111
36	1	0.111
72	1	0.111
108	1	0.111
144	1	0.111
180	1	0.111
216	1	0.111
255	2	0.222

Step 3: Compute CDF of the original and target histograms Original CDF $c_r(r_k)$:

r_k	$c_r(r_k)$
52	0.111
55	0.222
58	0.333
59	0.444
61	0.666
64	0.777
66	0.888
79	0.999

Target CDF $c_z(z_q)$:

z_q	$c_z(z_q)$
0	0.111
36	0.222
72	0.333
108	0.444
144	0.555
180	0.666
216	0.777
255	0.999

Step 4: Match each original intensity to target intensity For each original $c_r(r_k)$, find the target intensity z_q with closest $c_z(z_q)$.

r_k	$c_r(r_k)$		$MappedIntensity$
52	0.111	0.111	0
55	0.222	0.222	36
58	0.333	0.333	72
59	0.444	0.444	108
61	0.666	0.666	180
64	0.777	0.777	216
66	0.888	0.999	255
79	0.999	0.999	255

Step 5: Generate the output image Replace each pixel in I with the mapped intensity:

$$I_{\text{HS}} = \begin{bmatrix} 0 & 36 & 180 \\ 108 & 255 & 180 \\ 255 & 216 & 72 \end{bmatrix}$$

Result:

The output image I_{HS} now follows the specified target histogram distribution, demonstrating histogram specification (matching).

Spatial filtering: smoothing (mean, median), sharpening (Laplacian, high-boost).

Practical: Contrast adjustment, histogram equalization, noise reduction.

Table 2: Summary of Contrast Stretching, Histogram Equalization and Histogram Matching

Method	**Mapping Rule**	**Output Sequence**
Original	Values confined between 2–6 (low contrast)	[2, 3, 4, 3, 4, 5, 4, 5, 6]
Contrast Stretching (improves global contrast)	Linear scaling: $$I_{\text{new}} = \frac{I-2}{6-2} \times 9$$	[0, 2, 5, 2, 5, 7, 5, 7, 9] Pixel values span full 0–9 range
Histogram Equalization (improves local contrast)	Nonlinear CDF-based redistribution into 0–9	[1, 3, 5, 3, 5, 7, 5, 7, 8]
Histogram Matching (matches to reference distribution)	Transform intensities so histogram resembles reference (here uniform in 0–9)	[1, 3, 5, 3, 5, 7, 5, 7, 9]

Module 3: Image Enhancement in Frequency Domain

- Fourier Transform & 2D DFT basics.
- Frequency domain filtering: Low-pass (blurring), High-pass (sharpening).
- Homomorphic filtering (illumination correction).

Practical: Frequency filtering using FFT.

Module 4: Image Restoration

- Image degradation models.
- Noise models: Gaussian, Salt & Pepper, Speckle.
- Noise removal filters: Inverse filtering, Wiener filtering.
- Motion blur restoration.

Practical: Adding noise & applying restoration filters.

Module 5: Color Image Processing

- Color models: RGB, CMYK, HSV, YCbCr.
- Pseudocolor & false color processing.
- Color transformations & corrections.

Practical: Convert RGB to HSV, enhance images by channels.

Module 6: Morphological Image Processing

- Binary image basics.
- Dilation, erosion, opening, closing.
- Boundary extraction, skeletonization.
- Morphological segmentation.

Practical: Apply morphological operations for object detection.

Module 7: Image Segmentation

- Edge detection: Sobel, Prewitt, Canny.
- Thresholding: Global (Otsu), Adaptive.
- Region-based segmentation.
- Watershed segmentation.

Practical: Edge detection, Otsu thresholding.

Module 8: Representation & Description

- Boundary descriptors: Chain codes, shape numbers.
- Region descriptors: Area, perimeter, moments.
- Texture analysis: Statistical, structural, spectral.
- Feature extraction & dimensionality reduction (PCA).

Practical: Extract features from objects in an image.

Module 9: Image Compression

- Redundancy in images.
- Lossless compression: Run-length, Huffman, LZW.
- Lossy compression: Transform coding, JPEG, JPEG2000.
- Video compression basics: MPEG, H.264.

Practical: Compress images using JPEG & analyze quality loss.

Module 10: Advanced Topics (Introductory)

- Image recognition & classification (pattern recognition).
- Machine learning & Deep learning in image processing (CNNs).
- Image segmentation with deep networks (U-Net, Mask R-CNN).
- Applications: Face recognition, medical imaging, driver monitoring.

Practical: Use pre-trained deep learning models for image classification.

Module 11: Project & Case Studies

- Medical image analysis (tumor detection).
- Remote sensing image classification.
- Biometric recognition (face, iris, fingerprint).
- Driver drowsiness detection (eye state recognition).

Suggested Textbooks

- R. C. Gonzalez & R. E. Woods – *Digital Image Processing.*
- A. K. Jain – *Fundamentals of Digital Image Processing.*
- Pratt – *Digital Image Processing.*

Tools for Practice

- MATLAB Image Processing Toolbox.
- Python (OpenCV, NumPy, scikit-image, TensorFlow/Keras).