Assignment for Chapter 03: Image Enhancement Techniques

Objective:

To understand and apply image enhancement techniques for improving image quality by adjusting brightness, contrast, and removing noise using spatial filtering.

Assignment Tasks

Part 1: Theory

1. Explain the importance of image enhancement in real-world applications.
2. Define and differentiate the following terms:
    - Brightness
    - Contrast
    - Dynamic Range
3. Explain **Histogram Equalization** and its role in image enhancement.
   Provide the mathematical formula for histogram equalization:
   $$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{N},$$
   where:
    - r_k = Intensity level
    - $T(r_k)$ = Transformation function
    - n_j = Number of pixels with intensity j
    - N = Total number of pixels.
4. What is the difference between:
    - **Spatial Filtering** and **Frequency Domain Filtering**.
    - **Low-Pass Filters** and **High-Pass Filters**.

Part 2: Practical Tasks

1. **Brightness and Contrast Adjustment**
    - Write a program to manually adjust brightness and contrast of an image using:
      $g(x, y) = \alpha f(x, y) + \beta,$

where:
    - α = Contrast scaling factor
    - β = Brightness adjustment.

2. **Histogram Equalization**
    - Apply histogram equalization to a grayscale image and compare the results with the original image.
3. **Smoothing Filters**
    - Apply the following filters to reduce noise:
        - Average Filter
        - Gaussian Filter
        - Median Filter
4. **Edge Detection**
    - Detect edges in an image using:
        - Sobel Filter
        - Prewitt Filter
        - Laplacian Filter
5. **Sharpening Filters**
    - Apply a Laplacian-based sharpening filter to an image and display the result.

Sample Solutions

Part 1: Theory Solutions

1. **Importance of Image Enhancement**
   Image enhancement improves visual appearance, highlighting features or details critical for tasks like medical imaging, surveillance, or photography.
2. **Brightness, Contrast, and Dynamic Range**
    - **Brightness**: Overall lightness/darkness of an image.
    - **Contrast**: Difference between the darkest and brightest areas.
    - **Dynamic Range**: Range of intensity levels.

3. **Histogram Equalization Formula**

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{N}$$

Part 2: Practical Solutions

1. **Brightness and Contrast Adjustment**

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
image_path = "/path/to/image.jpg"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Apply brightness and contrast adjustment
alpha = 1.2  # Contrast
beta = 50    # Brightness
adjusted_image = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)

# Display Original and Adjusted Images
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap="gray")
plt.title("Original Image")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(adjusted_image, cmap="gray")
plt.title("Brightness & Contrast Adjusted")
plt.axis("off")

plt.tight_layout()
plt.show()
```

2. **Histogram Equalization**

```
# Apply Histogram Equalization
equalized_image = cv2.equalizeHist(image)

# Display Histogram Equalized Image
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap="gray")
plt.title("Original Image")
plt.axis("off")
```

```
plt.subplot(1, 2, 2)
plt.imshow(equalized_image, cmap="gray")
plt.title("Histogram Equalized Image")
plt.axis("off")

plt.tight_layout()
plt.show()
```

3. **Smoothing Filters**

```
# Apply Smoothing Filters
average_filter = cv2.blur(image, (5, 5))
gaussian_filter = cv2.GaussianBlur(image, (5, 5), 0)
median_filter = cv2.medianBlur(image, 5)

# Display Filtered Images
plt.figure(figsize=(18, 6))
plt.subplot(1, 3, 1)
plt.imshow(average_filter, cmap="gray")
plt.title("Average Filter")
plt.axis("off")

plt.subplot(1, 3, 2)
plt.imshow(gaussian_filter, cmap="gray")
plt.title("Gaussian Filter")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.imshow(median_filter, cmap="gray")
plt.title("Median Filter")
plt.axis("off")

plt.tight_layout()
plt.show()
```

4. **Edge Detection**

```
# Sobel, Prewitt, and Laplacian Filters
sobel_edges = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
prewitt_kernel = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
prewitt_edges = cv2.filter2D(image, -1, prewitt_kernel)
laplacian_edges = cv2.Laplacian(image, cv2.CV_64F)

# Display Edge Detection Results
plt.figure(figsize=(18, 6))
plt.subplot(1, 3, 1)
plt.imshow(sobel_edges, cmap="gray")
plt.title("Sobel Edge Detection")
plt.axis("off")
```

```
plt.subplot(1, 3, 2)
plt.imshow(prewitt_edges, cmap="gray")
plt.title("Prewitt Edge Detection")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.imshow(laplacian_edges, cmap="gray")
plt.title("Laplacian Edge Detection")
plt.axis("off")

plt.tight_layout()
plt.show()
```

5. **Sharpening Filter**

```
# Apply Laplacian-based Sharpening
kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
sharpened_image = cv2.filter2D(image, -1, kernel)

# Display Sharpened Image
plt.figure(figsize=(8, 6))
plt.imshow(sharpened_image, cmap="gray")
plt.title("Sharpened Image")
plt.axis("off")
plt.show()
```

Submission Instructions

1. Submit your Python code as a `.py` or Jupyter notebook (`.ipynb`).
2. Include a PDF report with:
   - Answers to theory questions.
   - Screenshots of outputs from practical tasks.
   - Explanations and observations.
3. Zip all files into a single folder named `Chapter03_Assignment_YourName.zip`.