Digital Image Processing Lab Report 1

Shufan Shahi
ID: 210041210

November 24, 2025

1 Task 1: Grayscale Histogram Generation

1.1 Problem Statement

Develop a function to compute and display the histogram of a grayscale image.

1.2 Solution Approach

The histogram function counts the frequency of each intensity level (0-255) in the grayscale image by iterating through all pixels.

1.3 Implementation

```
def my_histogram(gray_image):
    histogram = [0] * 256
    for row in gray_image:
        for pixel in row:
            histogram[pixel] += 1
    return histogram
```

1.4 Parameters

- `gray_image`: 2D numpy array or list with pixel values in range [0, 255]

1.5 Output

Returns a list of length 256 where each index represents an intensity level and its value represents the pixel count. The histogram is visualized using a bar plot with black bars, showing the distribution of pixel intensities across the image.

2 Task 2: Color Channel Separation and Display

2.1 Problem Statement

Extract and display the three color channels (Red, Green, Blue) of a color image, with each channel displayed in its respective color.

2.2 Solution Approach

Each channel is extracted from the RGB image and displayed by creating a zero-initialized 3-channel image where only the target channel contains the original values.

2.3 Implementation

```
def channel_show(color_image):
    R = color_image[:, :, 0]
    G = color_image[:, :, 1]
    B = color_image[:, :, 2]

    red_channel_img = np.zeros_like(color_image)
    red_channel_img[:, :, 0] = R

    green_channel_img = np.zeros_like(color_image)
    green_channel_img[:, :, 1] = G

    blue_channel_img = np.zeros_like(color_image)
    blue_channel_img[:, :, 2] = B
```

2.4 Parameters

- `color_image`: 3D numpy array of shape (H, W, 3) representing an RGB image

2.5 Output

Three separate images displayed side-by-side showing the red, green, and blue channels in their respective colors. The intensity in each channel corresponds to the original channel values from the color image.

3 Task 3: Color Histogram Analysis and Dominant Color Detection

3.1 Problem Statement

Generate histograms for all three color channels of a color image and determine which color is most dominant.

3.2 Solution Approach

The `my_histogram` function from Task 1 is applied to each color channel separately. Dominance is determined by computing the total weighted intensity for each channel:

$$\text{Total Intensity}_{\text{channel}} = \sum_{i=0}^{255} i \times \text{histogram}[i] \tag{1}$$

3.3 Implementation

```
def color_histogram_analysis(color_image):
    R = color_image[:, :, 0]
    G = color_image[:, :, 1]
    B = color_image[:, :, 2]

    hist_R = my_histogram(R)
    hist_G = my_histogram(G)
    hist_B = my_histogram(B)

    total_R = sum(i * hist_R[i] for i in range(256))
```

```
    total_G = sum(i * hist_G[i] for i in range(256))
    total_B = sum(i * hist_B[i] for i in range(256))

    totals = {"Red": total_R, "Green": total_G, "Blue": total_B}
    dominant_color = max(totals, key=totals.get)

    return dominant_color
```

3.4 Parameters

- `color_image`: 3D numpy array of shape (H, W, 3) representing an RGB image

3.5 Output

Three histograms displayed in red, green, and blue colors respectively, along with numerical total intensity values for each channel. The dominant color is determined by comparing these totals and printed to console.

4 Task 4: Color Channel Enhancement

4.1 Problem Statement

Implement a function to enhance a specific color channel by a given enhancement value between 0 and 1.

4.2 Solution Approach

The selected channel is multiplied by $(1 + \text{enhancement\_val})$, increasing its intensity. Values are clipped to [0, 255] to prevent overflow.

4.3 Implementation

```
def color_enhance(color_image, channel_number, enhancement_val):
    enhanced_image = color_image.copy().astype(np.float32)
    enhanced_image[:, :, channel_number] *= (1 + enhancement_val)
    enhanced_image = np.clip(enhanced_image, 0, 255)
    enhanced_image = enhanced_image.astype(np.uint8)
    return enhanced_image
```

4.4 Parameters

- `color_image`: 3D numpy array of shape (H, W, 3) representing an RGB image
- `channel_number`: Integer (0 for Red, 1 for Green, 2 for Blue)
- `enhancement_val`: Float in range [0, 1] representing the enhancement factor

4.5 Output

An enhanced RGB image where the specified channel has increased intensity. The function returns a numpy array that can be displayed alongside the original for comparison. In the example implementation, the red channel was enhanced by 50% (`enhancement_val=0.5`).

5 Task 5: Grayscale Image Negation

5.1 Problem Statement

Create a function to compute the negative of a grayscale image using the formula:

$$g(x, y) = L_{\max} - f(x, y) \tag{2}$$

where $L_{\max} = 255$ for 8-bit images.

5.2 Solution Approach

Each pixel intensity is subtracted from 255 to produce the negative image, effectively inverting the brightness levels.

5.3 Implementation

```
def invert_gray(gray_image):
    gray_image = np.array(gray_image, dtype=np.uint8)
    Lmax = 255
    negative_image = Lmax - gray_image
    return negative_image
```

5.4 Parameters

- `gray_image`: 2D numpy array or list with pixel values in range [0, 255]
- `Lmax`: Fixed at 255 for 8-bit images

5.5 Output

A negative grayscale image where dark regions become bright and bright regions become dark. The output is displayed using a grayscale colormap.

6 Task 6: Color Image Negation

6.1 Problem Statement

Extend the grayscale negation function to work with color images.

6.2 Solution Approach

The `invert_gray` function is directly applied to color images. Since NumPy operations work element-wise, the negation formula is applied to all three channels simultaneously.

6.3 Implementation

```
def invert_color(color_image):
    negative_image = invert_gray(color_image)
    return negative_image
```

6.4 Parameters

- `color_image`: 3D numpy array of shape (H, W, 3) representing an RGB image

6.5 Output

A color negative image where each channel is independently inverted. Colors appear complementary to the original (e.g., red becomes cyan, green becomes magenta, blue becomes yellow).