

International Islamic University Chittagong.

Department of Computer Science and Engineering.



Course Code: CSE-4876

Course Title: Pattern Recognition & Image Processing

Submitted By:

Name: Farida Nusrat

ID: C201242

Semester: 8th

Section: 8AF

Submitted To:

Mohammad Mahadi Hassan

Associate Professor, CSE.

Lab No: 05

Lab Title:

1. Histogram processing
2. Histogram Equalization

Methodology:

1. Histogram processing(code)

```
import cv2
import matplotlib.pyplot as plt

# Load the image
image = cv2.imread('/content/boy.jfif')

# Convert the image to grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Calculate histogram
hist = cv2.calcHist([gray_image], [0], None, [256], [0,256])

# Total number of pixels in the image
n = gray_image.shape[0] * gray_image.shape[1]

# Normalize the histogram
normalized_hist = hist / n

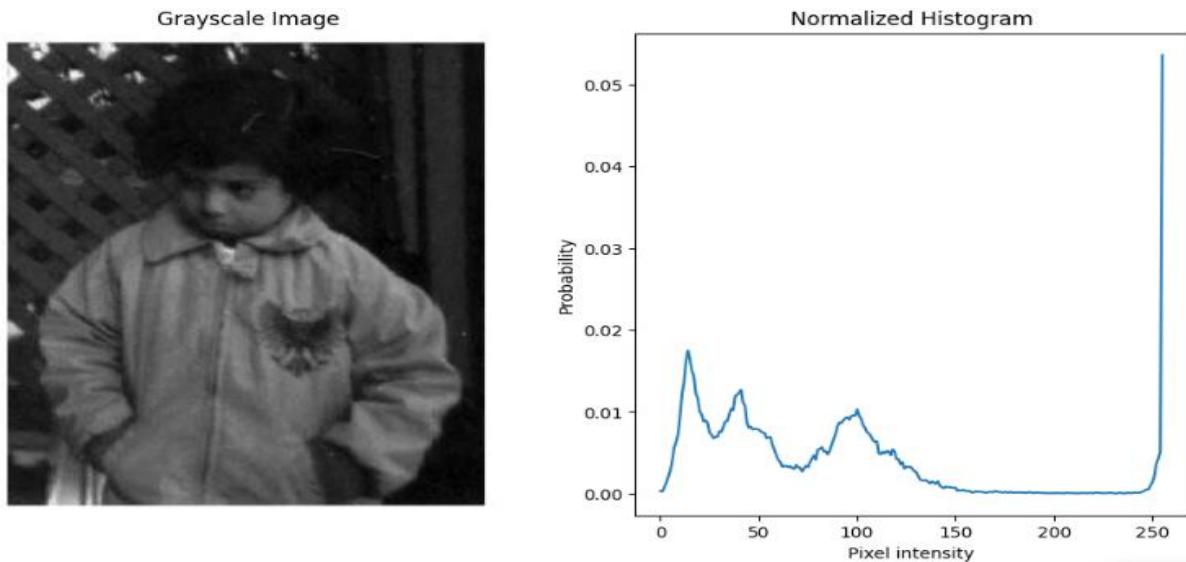
# Display the grayscale image and its normalized histogram
plt.figure(figsize=(12, 6))

# Grayscale image
plt.subplot(1, 2, 1)
plt.imshow(gray_image, cmap='gray')
plt.title('Grayscale Image')
plt.axis('off')

# Normalized histogram
plt.subplot(1, 2, 2)
plt.plot(normalized_hist)
plt.title('Normalized Histogram')
plt.xlabel('Pixel intensity')
plt.ylabel('Probability')

plt.show()
```

Output:



2. Histogram Equalization(code)

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

# Load the image
image = cv2.imread('/content/boy.jfif', cv2.IMREAD_GRAYSCALE)
# Calculate histogram for original image
hist_original = cv2.calcHist([image], [0], None, [256], [0,256])

# Compute cumulative distribution function (CDF)
cdf = hist_original.cumsum()

# Total number of pixels in the image
n = image.shape[0] * image.shape[1]

# Compute the transformation function N(g)
l = 8 # Number of bits per pixel
N = np.maximum(0, np.round((2 ** l * cdf) / n) - 1)

# Apply the transformation to the image
processed_image = N[image].astype(np.uint8)

# Calculate histogram for processed image
hist_processed = cv2.calcHist([processed_image], [0], None, [256], [0,256])

# Display only the processed image and its histogram
fig, axs = plt.subplots(1, 2, figsize=(15, 5))

# Processed Image
axs[0].imshow(processed_image, cmap='gray')
axs[0].set_title('Equalized Image')

# Processed Image Histogram
axs[1].plot(hist_processed, color='r')
```

```
axs[1].set_title('Equalized Image Histogram')  
plt.show()
```

Output:

