Lab Report 3 : Histogram Equalization, Transformation Function(CDF), Histogram Specification using Digital Image Processing

Course Title: Digital Image Processing Lab

Course Code: CSE-406

Date of Performance: September 8, 2024

Date of Submission: September 15, 2024

Submitted to:

Dr. Morium Akter
Professor

Dr. Md. Golam Moazzam
Professor

Department of Computer Science and Engineering
Jahangirnagar University

Submitted by:

Class Roll	Exam Roll	Name
370	202182	Rubayed All Islam

Department of Computer Science and Engineering
Jahangirnagar University
Savar, Dhaka, Bangladesh

Experiment Name

a) Illustration of histogram equalization.

b) Transformation function or CDF of dark image.

c) Histogram specification of an image.

Objectives

a) **Image Equalization**: To enhance the contrast of an image by applying histogram equalization, and to observe the effect on both the image and its histogram.

b) **CDF Calculation**: To compute and visualize the Cumulative Distribution Function (CDF) of an image and understand its role in image processing.

c) **Histogram Specification**: To modify an image to match a specified histogram and analyze the effects on the image and its histogram.

Experiment 1: Illustration of histogram equalization

MATLAB Code:

Listing 1: histogram_equalization.m

```
% Read the image
img = imread('shell.jpg');

% Convert the image to grayscale if it's RGB
if size(img, 3) == 3
    img_gray = rgb2gray(img);
else
    img_gray = img;
end

% Perform histogram equalization
img_eq = histeq(img_gray);

% Display the original image and its histogram
figure;
subplot(2, 2, 1);
imshow(img_gray);
title('Original Image');

subplot(2, 2, 2);
imhist(img_gray);
title('Histogram of Original Image');

% Display the equalized image and its histogram
subplot(2, 2, 3);
```

```
imshow(img_eq);
title('Equalized Image');

subplot(2, 2, 4);
imhist(img_eq);
title('Histogram of Equalized Image');
```

Python Code:

Listing 2: histogram_equalization.py

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

# Read the image
img = cv2.imread('shell.jpg')

# Convert the image to grayscale if it's RGB
if len(img.shape) == 3:
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
else:
    img_gray = img

# Perform histogram equalization
img_eq = cv2.equalizeHist(img_gray)

# Plot the original image and its histogram
plt.figure(figsize=(10, 6))

plt.subplot(2, 2, 1)
plt.imshow(img_gray, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.hist(img_gray.ravel(), 256, [0, 256])
plt.title('Histogram of Original Image')

# Plot the equalized image and its histogram
plt.subplot(2, 2, 3)
plt.imshow(img_eq, cmap='gray')
plt.title('Equalized Image')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.hist(img_eq.ravel(), 256, [0, 256])
plt.title('Histogram of Equalized Image')

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

Output:

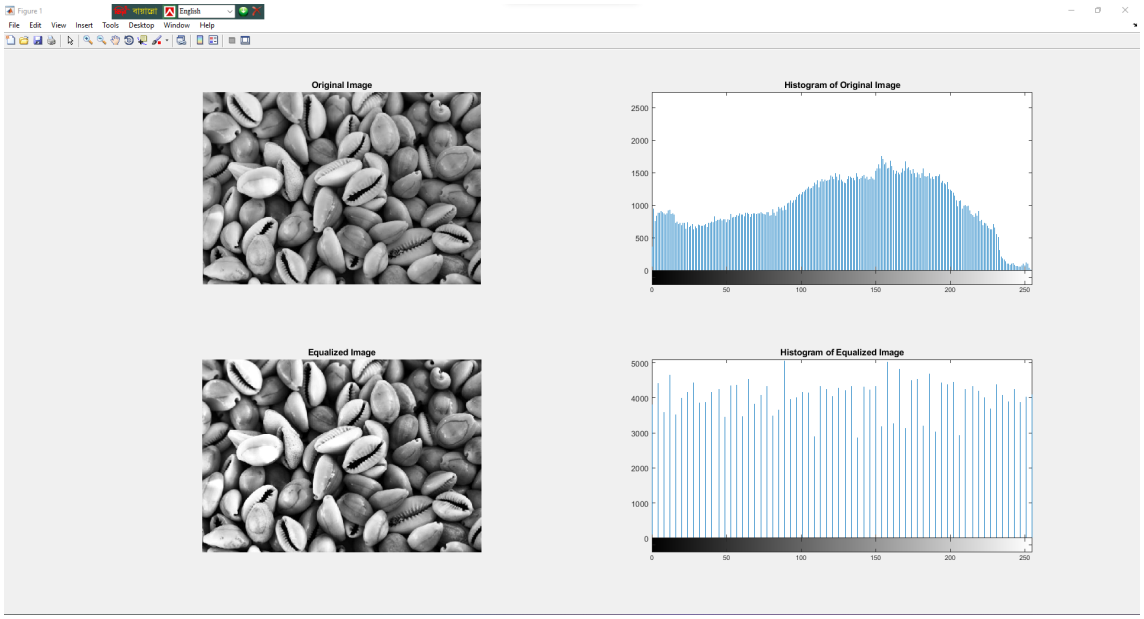

Figure 1: Histogram Equalization

Experiment 2: Transformation function or CDF of dark image

MATLAB Code:

Listing 3: cumulative_distribution_function(CDF).m

```
% Read the image
img = imread('shell.jpg');  % Replace with your image file

% Convert the image to grayscale if it's RGB
if size(img, 3) == 3
    img_gray = rgb2gray(img);
else
    img_gray = img;
end

% Calculate the histogram of the grayscale image
[counts, binLocations] = imhist(img_gray);

% Normalize the histogram to get the PDF (probability
   distribution function)
pdf = counts / sum(counts);

% Calculate the CDF (cumulative sum of the PDF)
cdf = cumsum(pdf);

% Create a figure with 3 subplots
figure;

% Show the grayscale image
subplot(1, 3, 1);
imshow(img_gray);
title('Grayscale Image');

% Plot the histogram
subplot(1, 3, 2);
bar(binLocations, counts);
title('Histogram of Grayscale Image');
xlabel('Intensity Values');
ylabel('Number of Pixels');

% Plot the CDF
subplot(1, 3, 3);
plot(binLocations, cdf, 'LineWidth', 2);
title('Transformation function of the Image');
xlabel('Intensity Values');
ylabel('Cumulative Probability');
grid on;

% Adjust layout
```

```
set(gcf, 'Position', [100, 100, 1200, 400]);  % Adjust window
    size for better visualization
```

Python Code:

Listing 4: cumulative_distribution_function(CDF).py

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

# Read the image
img = cv2.imread('shell.jpg')  # Replace with your image file

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

# Calculate the histogram
hist, bin_edges = np.histogram(img_gray, bins=256, range=(0,
    255))

# Normalize the histogram to get PDF
pdf = hist / np.sum(hist)

# Calculate the CDF or transformation function
cdf = np.cumsum(pdf)

# Create a figure with 3 subplots
plt.figure(figsize=(12, 4))

# Show the grayscale image
plt.subplot(1, 3, 1)
plt.imshow(img_gray, cmap='gray')
plt.title('Grayscale Image')
plt.axis('off')

# Plot the histogram
plt.subplot(1, 3, 2)
plt.bar(bin_edges[:-1], hist, width=1, edgecolor='black')
plt.title('Histogram of Grayscale Image')
plt.xlabel('Intensity Values')
plt.ylabel('Number of Pixels')

# Plot the CDF
plt.subplot(1, 3, 3)
plt.plot(bin_edges[:-1], cdf, color='r', linewidth=2)
plt.title('Transformation function of the Image')
plt.xlabel('Intensity Values')
plt.ylabel('Cumulative Probability')
plt.grid(True)

```

```
# Show all plots
plt.tight_layout()
plt.show()
```

Output:

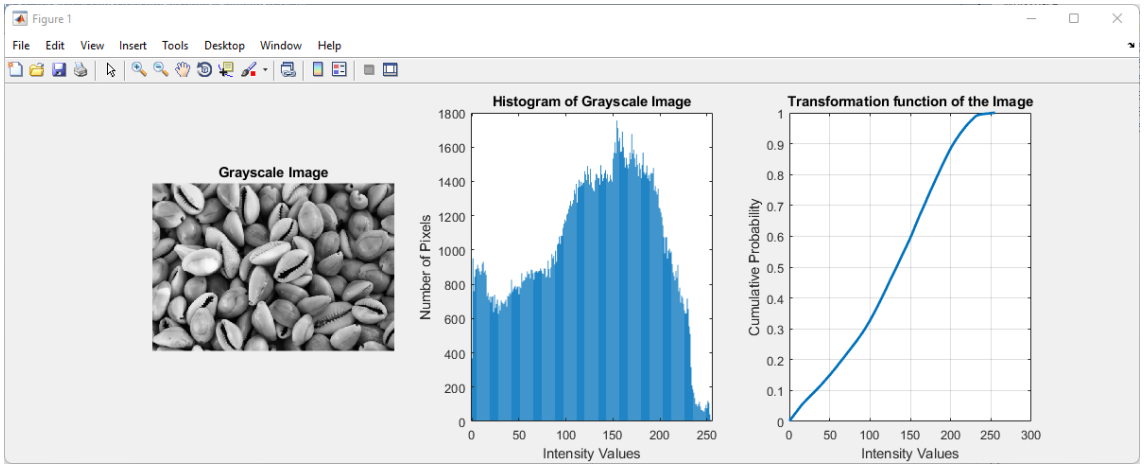

Figure 2: Transformation function or CDF of dark image

Experiment 3: Histogram specification of an image

MATLAB Code:

Listing 5: histogram_specification.m

```
% Read the original image
originalImage = imread('mars.png');
if size(originalImage, 3) == 3
    originalImage = rgb2gray(originalImage); % Convert to
       grayscale if it's a color image
end

% Display the original image
figure;
subplot(2,2,1);
imshow(originalImage);
title('Original Image');

% Display the histogram of the original image
subplot(2,2,2);
imhist(originalImage);
title('Original Histogram');

% Create a desired histogram (For example, a uniform histogram)
desiredHist = ones(256,1) / 256;

% Compute the cumulative distribution function (CDF) of the
   original image
[originalCounts, ~] = imhist(originalImage);
originalCDF = cumsum(originalCounts) / numel(originalImage);

% Compute the CDF for the desired histogram
desiredCDF = cumsum(desiredHist);

% Create a lookup table for mapping the original image to the
   desired histogram
lookupTable = zeros(256,1);
for i = 1:256
    [~, idx] = min(abs(originalCDF(i) - desiredCDF));
    lookupTable(i) = idx - 1;
end

% Apply the histogram specification
specifiedImage = lookupTable(double(originalImage) + 1);
specifiedImage = uint8(specifiedImage);

% Display the image after histogram specification
subplot(2,2,3);
imshow(specifiedImage);
title('Specified Image');

```

```
% Display the histogram of the specified image
subplot(2,2,4);
imhist(specifiedImage);
title('Specified Histogram');
```

Python Code:

Listing 6: histogram_specification.py

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

# Read the original image
original_image = cv2.imread('mars.png', cv2.IMREAD_GRAYSCALE)

# Display the original image
plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)
plt.imshow(original_image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

# Display the histogram of the original image
plt.subplot(2, 2, 2)
plt.hist(original_image.ravel(), bins=256, range=[0, 256],
    density=True)
plt.title('Original Histogram')

# Create a desired histogram (For example, a uniform histogram)
desired_hist = np.ones(256) / 256

# Compute the cumulative distribution function (CDF) of the
    original image
original_hist, bin_edges =
    np.histogram(original_image.flatten(), bins=256, range=[0,
    256], density=True)
original_cdf = np.cumsum(original_hist)
original_cdf = original_cdf / original_cdf[-1]  # Normalize

# Compute the CDF for the desired histogram
desired_cdf = np.cumsum(desired_hist)

# Create a lookup table for mapping the original image to the
    desired histogram
lookup_table = np.zeros(256)
for i in range(256):
    lookup_table[i] = np.searchsorted(desired_cdf,
        original_cdf[i]) - 1

```

```
# Apply the histogram specification
specified_image = lookup_table[original_image]

# Display the image after histogram specification
plt.subplot(2, 2, 3)
plt.imshow(specified_image, cmap='gray')
plt.title('Specified Image')
plt.axis('off')

# Display the histogram of the specified image
plt.subplot(2, 2, 4)
plt.hist(specified_image.ravel(), bins=256, range=[0, 256],
    density=True)
plt.title('Specified Histogram')

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

Output:

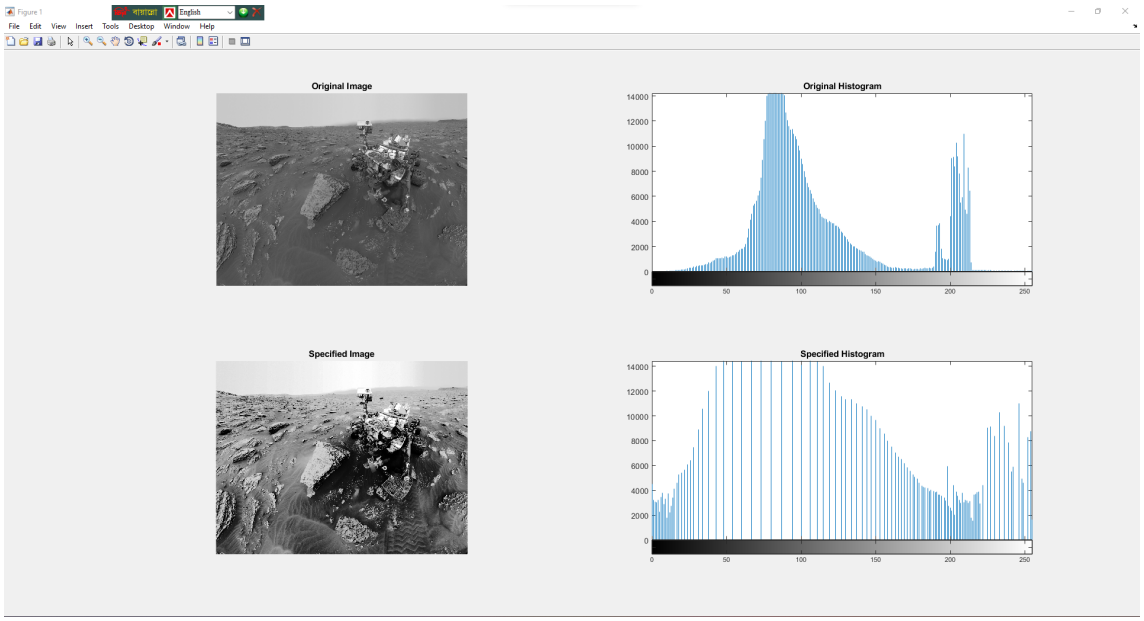

Figure 3: Histogram specification of an image

Results

Image Equalization

MATLAB Results:

- The original grayscale image and its histogram were displayed.
- The equalized image and its histogram were displayed, showing an enhanced contrast compared to the original.

Python Results:

- The original grayscale image and its histogram were plotted.
- The equalized image and its histogram were plotted, demonstrating improved contrast in the equalized image.

Observations:

- Histogram equalization effectively redistributes the intensity levels of the image, leading to better contrast and visibility of details in the image.

Transformation Function or CDF Calculation

MATLAB Results:

- The grayscale image was displayed.
- The histogram and CDF of the grayscale image were plotted, showing the distribution of pixel intensities and their cumulative probability.

Python Results:

- The grayscale image was displayed.
- The histogram and CDF of the grayscale image were plotted, providing a visual representation of the image's intensity distribution and cumulative probability.

Observations:

- The CDF provides insight into the distribution of pixel intensities and is useful for understanding image contrast and brightness.

Histogram Specification

MATLAB Results:

- The original grayscale image and its histogram were displayed.
- An image with a specified (uniform) histogram was created and displayed, along with its histogram.

Python Results:

- The original grayscale image and its histogram were plotted.
- An image with a specified (uniform) histogram was created and displayed, along with its histogram.

Observations:

- Histogram specification modifies the image to match a desired histogram, which can be used to achieve specific visual effects or to enhance contrast according to predefined criteria.