Lab Report: 03
Title: Histogram Equalization, Cumulative Distribution Function (CDF)
&
Histogram Matching in Digital Image Processing

Course title: Digital Image Processing Laboratory
Course code: CSE-406
4th Year 1st Semester Examination 2023

Date of Submission: 15/09/2024

Submitted to-
Dr. Md. Golam Moazzam
Professor
Department of Computer Science and Engineering
Jahangirnagar University
&
Dr. Morium Akter
Professor
Department of Computer Science and Engineering
Jahangirnagar University
Savar, Dhaka-1342

Class Roll	Exam Roll	Name
353	202165	Shanjida Alam

Experiment Name: Histogram Equalization

Objectives:

1. Improve Image Contrast
2. Enhance Image Details
3. Reduce Noise and Artifacts

Code-01: Python

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
input_image = cv2.imread('nature.jpeg', cv2.IMREAD_GRAYSCALE)
plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1)
plt.imshow(input_image, cmap='gray')
plt.title('Input Image')
plt.axis('off')
plt.subplot(2, 2, 2)
plt.hist(input_image.ravel(), bins=256, range=[0, 256])
plt.title('Histogram of Input Image')
equalized_image = cv2.equalizeHist(input_image)
plt.subplot(2, 2, 3)
plt.imshow(equalized_image, cmap='gray')
plt.title('Histogram-Equalized Image')
plt.axis('off')
plt.subplot(2, 2, 4)
plt.hist(equalized_image.ravel(), bins=256, range=[0, 256])
plt.title('Histogram of Equalized Image')
plt.tight_layout()
plt.show()
import cv2
import numpy as np
from matplotlib import pyplot as plt
input_image = cv2.imread('your_image.jpg', cv2.IMREAD_GRAYSCALE)
plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1)
plt.imshow(input_image, cmap='gray')
plt.title('Input Image')
plt.axis('off')
plt.subplot(2, 2, 2)
```

```
plt.hist(input_image.ravel(), bins=256, range=[0, 256])
plt.title('Histogram of Input Image')
equalized_image = cv2.equalizeHist(input_image)
plt.subplot(2, 2, 3)
plt.imshow(equalized_image, cmap='gray')
plt.title('Histogram-Equalized Image')
plt.axis('off')
plt.subplot(2, 2, 4)
plt.hist(equalized_image.ravel(), bins=256, range=[0, 256])
plt.title('Histogram of Equalized Image')
plt.tight_layout()
plt.show()
```

Output:

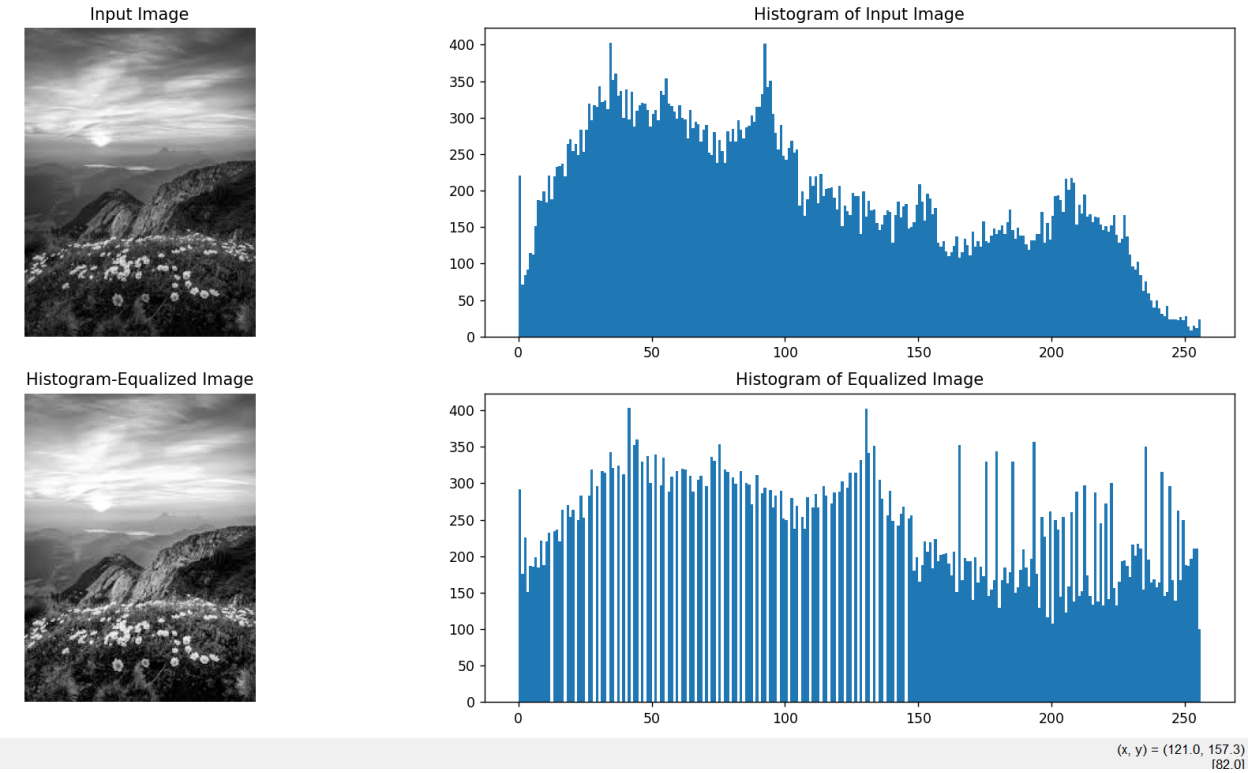

Figure 1.1: Showing the histogram equalization in python

Explanation:

1. Importing libraries
2. Reading the input image
3. Displaying the Input Image and Its Histogram
4. Performing Histogram Equalization
5. Displaying the Equalized Image and Its Histogram
6. Displaying the Final Plot

Code-02: MATLAB

```
 input_image = imread('flower.jpeg');
if size(input_image, 3) == 3
    input_image = rgb2gray(input_image);
end
figure;
subplot(2, 2, 1);
imshow(input_image);
title('Input Image');
subplot(2, 2, 2);
imhist(input_image);
title('Histogram of Input Image');
equalized_image = histeq(input_image);
subplot(2, 2, 3);
imshow(equalized_image);
title('Histogram-Equalized Image');
subplot(2, 2, 4);
imhist(equalized_image);
title('Histogram of Equalized Image');
```

Output:

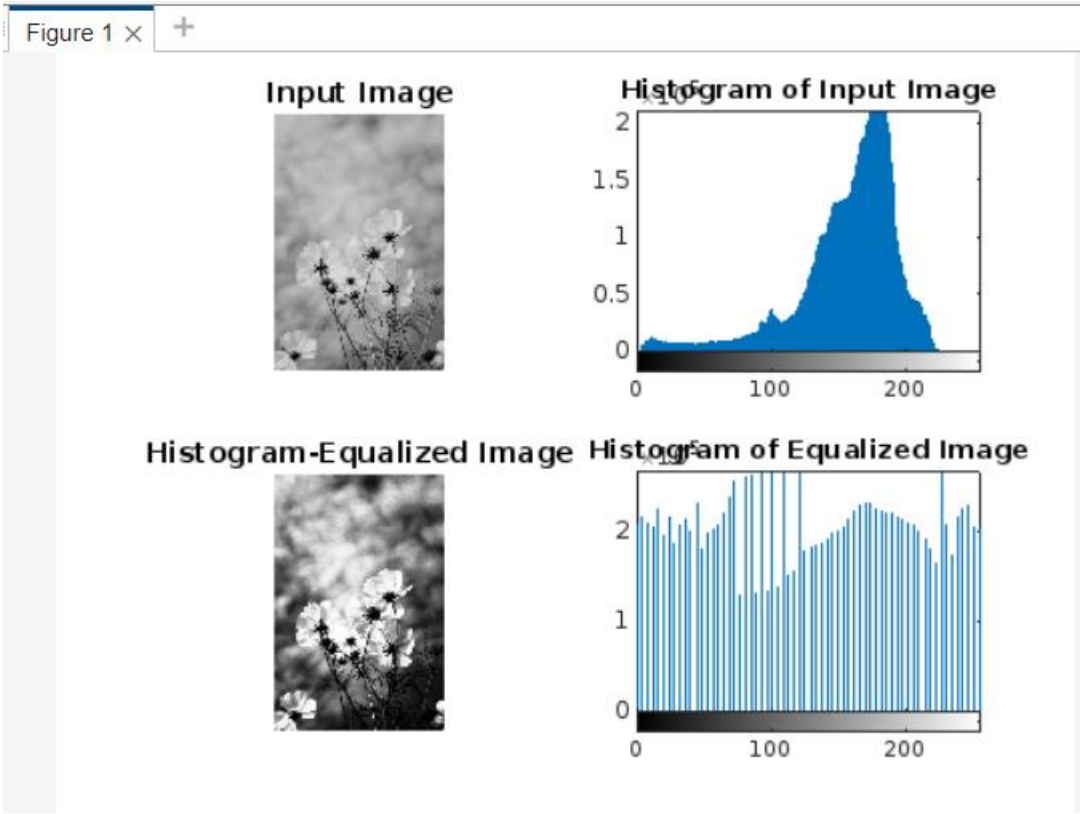

Figure 1.2: Showing the histogram equalization in MATLAB

Explanation:

1. Reading the Input Image
2. Converting the Image to Grayscale (If Necessary)
3. Displaying the Input Image and Its Histogram
4. The code uses histeq() to perform histogram equalization on the input image.
5. The code displays the equalized image and its histogram in the third and fourth subplots, respectively.

Experiment Name: Cumulative Distributed Function (CDF)

Objectives:

1. Image Normalization
2. Histogram Equalization
3. Image Segmentation

Code-01: Python

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
input_image = cv2.imread('nature.jpeg')
if len(input_image.shape) == 3:
    input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)
def compute_cdf(image):
    hist, bins = np.histogram(image.flatten(), 256, [0, 256])
    cdf = hist.cumsum()
    cdf_normalized = cdf / cdf.max()
    return cdf_normalized
plt.figure(figsize=(12, 6))
plt.subplot(2, 3, 1)
plt.imshow(input_image, cmap='gray')
plt.title('Input Image')
plt.axis('off')
plt.subplot(2, 3, 2)
plt.hist(input_image.ravel(), bins=256, range=[0, 256], color='black')
plt.title('Histogram of Input Image')
cdf_input = compute_cdf(input_image)
plt.subplot(2, 3, 3)
plt.plot(cdf_input, color='black', linewidth=2)
plt.title('CDF of Input Image')
plt.xlabel('Pixel Intensity Values')
plt.ylabel('CDF')
equalized_image = cv2.equalizeHist(input_image)
plt.subplot(2, 3, 4)
plt.imshow(equalized_image, cmap='gray')
plt.title('Histogram-Equalized Image')
plt.axis('off')
plt.subplot(2, 3, 5)
plt.hist(equalized_image.ravel(), bins=256, range=[0, 256], color='black')
```

```
plt.title('Histogram of Equalized Image')
cdf_equalized = compute_cdf(equalized_image)
plt.subplot(2, 3, 6)
plt.plot(cdf_equalized, color='black', linewidth=2)
plt.title('CDF of Equalized Image')
plt.xlabel('Pixel Intensity Values')
plt.ylabel('CDF')
plt.tight_layout()
plt.show()
```

Output:

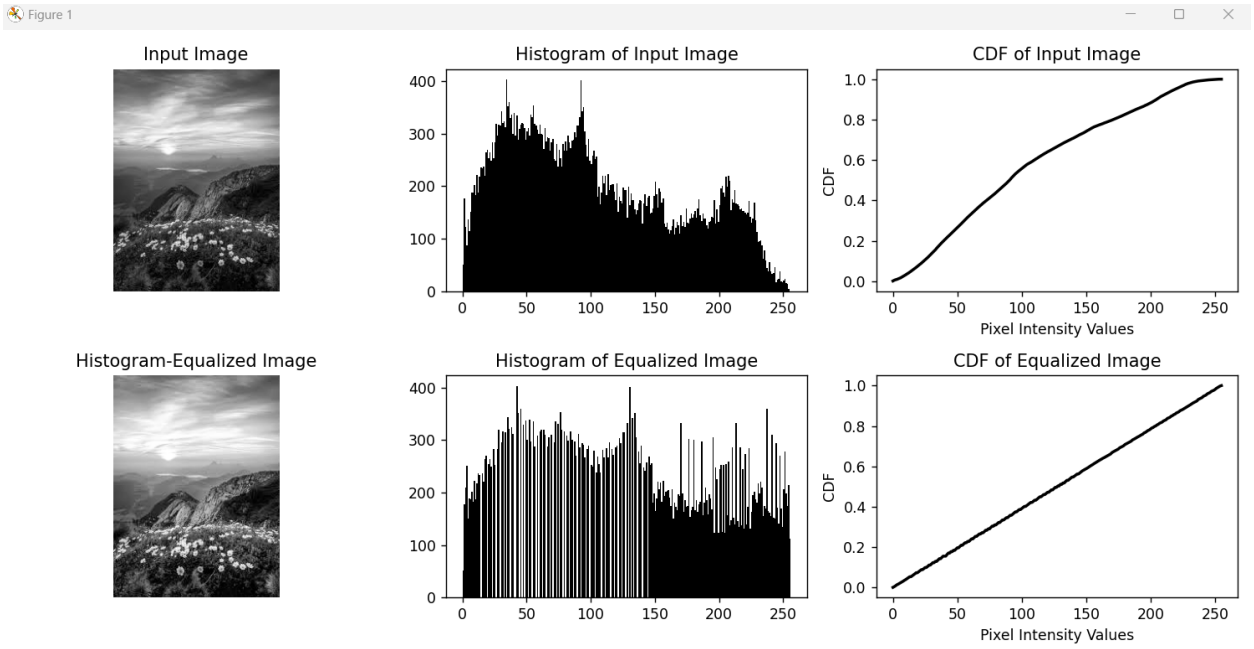

Figure 2.1: Showing the CDF using python code

Explanation:

1. Importing Libraries such as cv2, numpy, matplotlib.pyplot
2. Loading the Input Image
3. The code applies histogram equalization to the input image using OpenCV's equalizeHist function.
4. The code creates another three subplots
5. The code uses tight_layout to ensure the subplots fit nicely in the figure, and finally, it displays the figure using show.

Code-02: MATLAB

```
input_image = imread('flower.jpeg');
if size(input_image, 3) == 3
   input_image = rgb2gray(input_image);
end
figure;
subplot(2, 3, 1);
imshow(input_image);
title('Input Image');
subplot(2, 3, 2);
imhist(input_image);
title('Histogram of Input Image');
[counts, binLocations] = imhist(input_image);
cdf_input_image = cumsum(counts) / numel(input_image);
subplot(2, 3, 3);
plot(binLocations, cdf_input_image, 'LineWidth', 2);
title('CDF of Input Image');
xlabel('Pixel Intensity Values');
ylabel('CDF');
equalized_image = histeq(input_image);
subplot(2, 3, 4);
imshow(equalized_image);
title('Histogram-Equalized Image');
subplot(2, 3, 5);
imhist(equalized_image);
title('Histogram of Equalized Image');
[counts_eq, binLocations_eq] = imhist(equalized_image);
cdf_equalized_image = cumsum(counts_eq) / numel(equalized_image);
subplot(2, 3, 6);
plot(binLocations_eq, cdf_equalized_image, 'LineWidth', 2);
title('CDF of Equalized Image');
xlabel('Pixel Intensity Values');
ylabel('CDF');
```

Output:

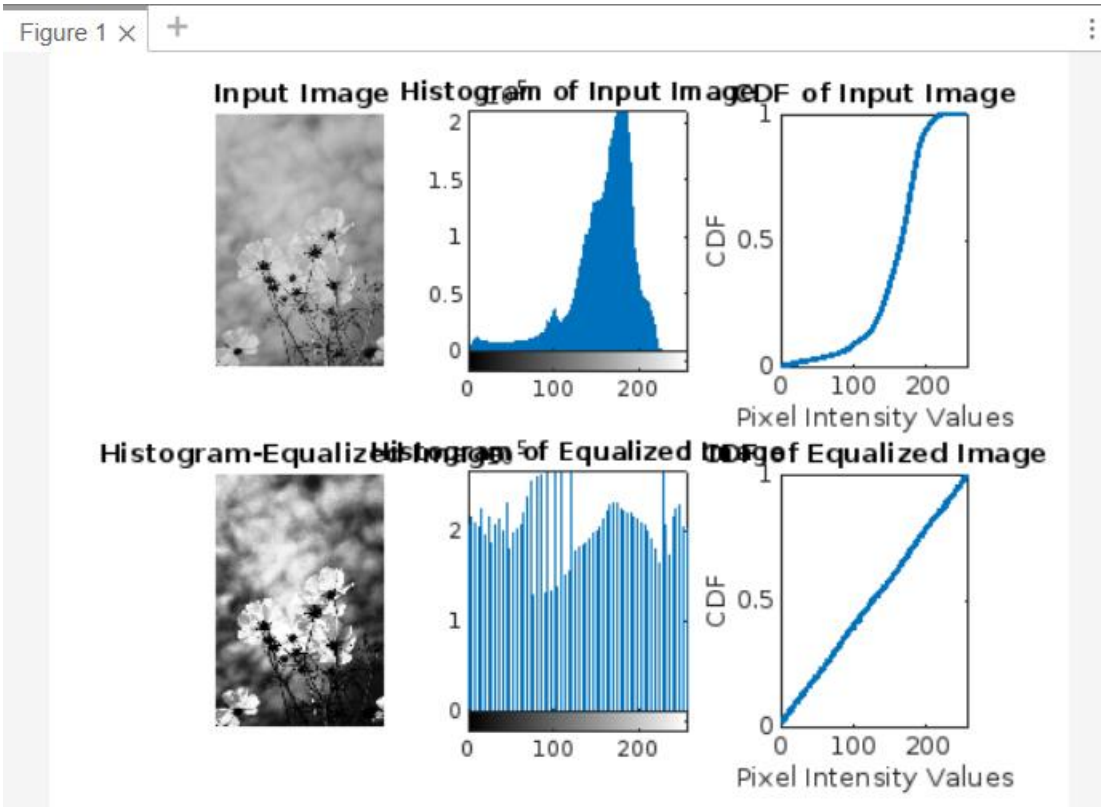

Figure 2.2: Showing the CDF using MATLAB

Explanation:

1. Reading the Input Image
2. Converting the Image to Grayscale
3. The input image is displayed using imshow.
4. The histogram of the input image is plotted using imhist.
5. Calculating and Plotting the CDF of the Input Image

Experiment Name: Histogram Matching in Digital Image Processing

Objectives:

1. Improve Image Contrast
2. By stretching the histogram, histogram specification can reveal hidden details in the image that were previously not visible.
3. By adjusting the histogram, histogram specification can improve the segmentation of objects in an image.
4. Histogram specification can also be used to enhance the color of an image by adjusting the histogram of each color channel (e.g., RGB).

Code-01: Python

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from skimage.exposure import match_histograms
input_image = cv2.imread('nature.jpeg', cv2.IMREAD_GRAYSCALE)
reference_image = cv2.imread('flower.jpeg', cv2.IMREAD_GRAYSCALE)
matched_image = match_histograms(input_image, reference_image)
plt.figure(figsize=(12, 6))
plt.subplot(2, 3, 1)
plt.imshow(input_image, cmap='gray')
plt.title('Input Image')
plt.axis('off')
plt.subplot(2, 3, 2)
plt.imshow(reference_image, cmap='gray')
plt.title('Reference Image')
plt.axis('off')
plt.subplot(2, 3, 3)
plt.imshow(matched_image, cmap='gray')
plt.title('Histogram-Matched Image')
plt.axis('off')
plt.subplot(2, 3, 4)
plt.hist(input_image.ravel(), bins=256, range=[0, 256], color='black')
plt.title('Histogram of Input Image')
plt.subplot(2, 3, 5)
plt.hist(reference_image.ravel(), bins=256, range=[0, 256], color='black')
plt.title('Histogram of Reference Image')
```

```
plt.subplot(2, 3, 6)
plt.hist(matched_image.ravel(), bins=256, range=[0, 256], color='black')
plt.title('Histogram of Matched Image')
plt.tight_layout()
plt.show()
```

Output:

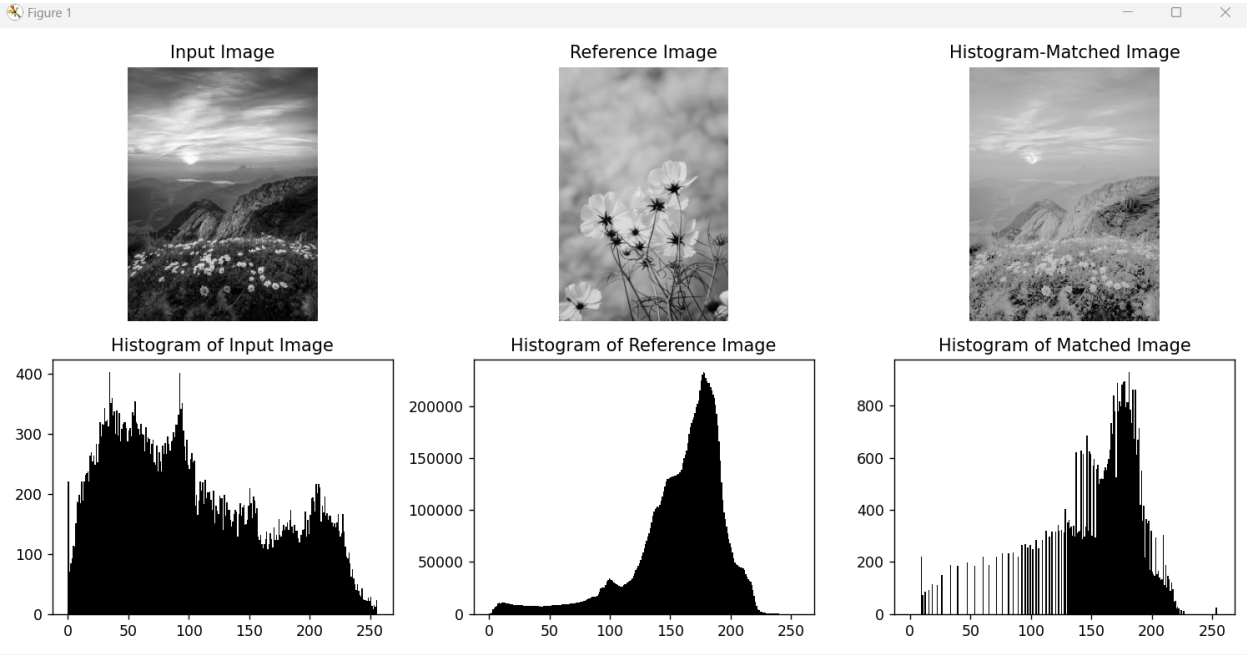

Figure 3.1: Showing the histogram specification using python code

Explanation:

1. Importing Libraries
2. input_image (the image to be transformed)
3. reference_image (the image whose histogram will be matched)
4. The code uses the match_histograms function from scikit-image to match the histogram of the input_image to the histogram of the reference_image. This function returns the histogram-matched image, which is stored in the matched_image variable.
5. The code plots the histograms of the three images using hist.

Code-02: MATLAB

```
input_image = imread('nature.jpeg');
reference_image = imread('flower.jpeg');
if size(input_image, 3) == 3
   input_image = rgb2gray(input_image);
end
if size(reference_image, 3) == 3
   reference_image = rgb2gray(reference_image);
end
matched_image = imhistmatch(input_image, reference_image);
figure;
subplot(2, 3, 1);
imshow(input_image);
title('Input Image');
subplot(2, 3, 2);
imshow(reference_image);
title('Reference Image');
subplot(2, 3, 3);
imshow(matched_image);
title('Histogram-Matched(I)');
subplot(2, 3, 4);
imhist(input_image);
title('Input Image');
subplot(2, 3, 5);
imhist(reference_image);
title('Reference Image');
subplot(2, 3, 6);
imhist(matched_image);
title('Matched Image');
```

Output:

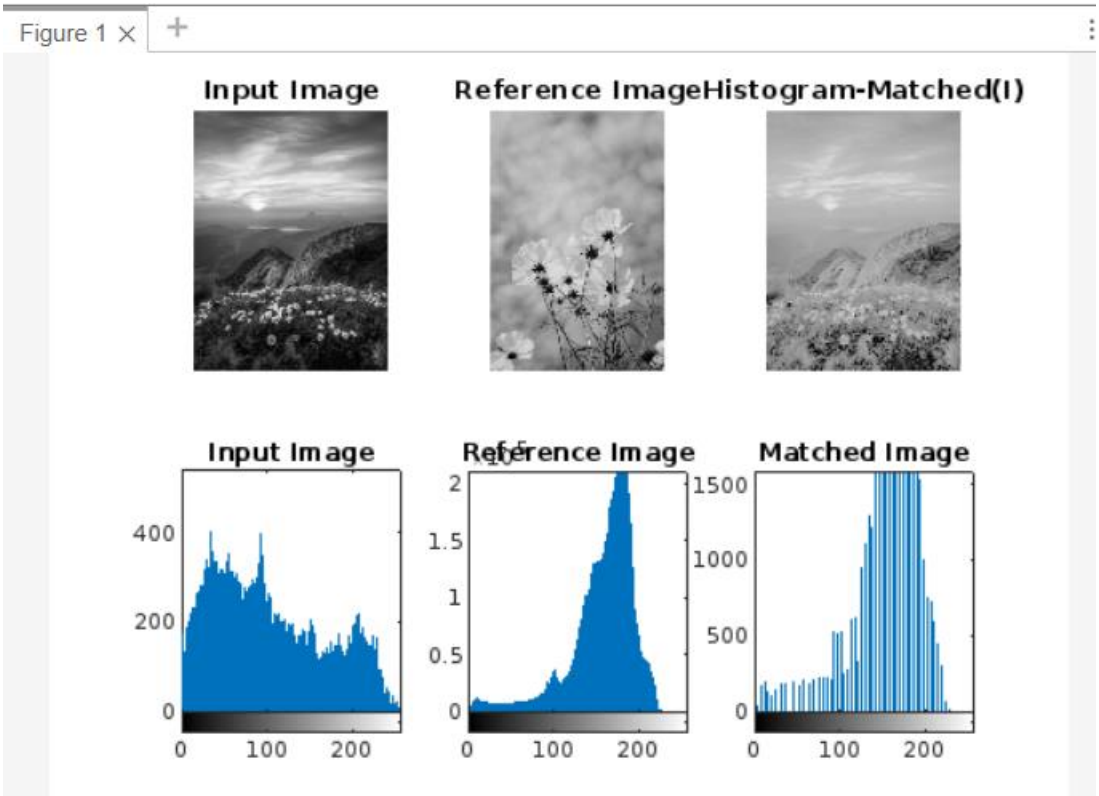

Figure 3.2: Showing the histogram specification in MATLAB

Explanation:

1. Loading Images
2. Converting to Grayscale
3. Performing Histogram Matching
4. Displaying Images and Histograms