PAMANTASAN NG LUNGSOD NG MAYNILA
(University of the City of Manila)
Intramuros, Manila

Elective 3

Laboratory Activity No. 3
Image Enhancement

Score

Submitted by:
Bolocon, Joniel R.

De Guzman, Jastine V.

De Juan, Lord Welchie P.

Mercado, Ed-Vir G.

Puno, Harold Dennis R.

Saturday 7:00 AM – 4:00 PM / CPE 0332.1-1

Date Submitted
02-08-2024

Submitted to:

Engr. Maria Rizette H. Sayo

I. Objectives

This laboratory activity aims to implement the principles and techniques of image enhancement through MATLAB/Octave and open CV using Python

1. Acquire the image.
2. Show histogram equalization.
3. Show contrast enhancement.
4. Show filtering in the spatial domain (average and median).

II. Methods

A. Perform a task given in the presentation

- Copy and paste your MATLAB code

```
% Read an image
img = imread('E:\PLM CET SUBJECTS\Digital Image Processing\flower.jpg');

% Display the original image
figure;
imshow(img);
title('Original Image');

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


% Display the grayscale image
figure;
imshow(img_gray);
title('Grayscale Image');

% Contrast enhancement using imadjust
img_contrast_enhanced = imadjust(img_gray);
```

```
% Display the contrast-enhanced image
figure;
imshow(img_contrast_enhanced);
title('Contrast Enhanced Image (imadjust)');

% Histogram equalization
img_histeq = histeq(img_gray);

% Display the histogram equalized image
figure;
imshow(img_histeq);
title('Equalized Image');

% Filtering using average filter
h_avg = fspecial('average', [5, 5]);
img_avg_filtered = imfilter(img_gray, h_avg);

% Display the average filtered image
figure;
imshow(img_avg_filtered);
title('Filtered Image (Average)');

% Filtering using median filter
img_median_filtered = medfilt2(img_gray, [5, 5]);

% Display the median filtered image
figure;
imshow(img_median_filtered);
title('Filtered Image (Median)');

% Display histograms for comparison

% Grayscale histogram
figure;
imhist(img_gray);
title('Histogram of Grayscale');


% Enhanced histogram (imadjust)
figure;
imhist(img_contrast_enhanced);
title('Histogram of Enhanced Image');


% Equalized histogram
figure;
imhist(img_histeq);
```

```
title('Histogram of Equalized Image');

% Histogram (Average Filtered)
figure;
imhist(img_avg_filtered);
title('Histogram of Average Filtered)');

% Histogram (Median Filtered)
figure;
imhist(img_median_filtered);
title('Histogram of Median Filtered)');
```

B. Supplementary Activity

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

#Figure 1: Acquire an Image of a Flower
# Read an image
img = cv2.imread('flower.jpg')

# Display the original image
plt.figure()
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.title('Original Image')
plt.show()

#Figure 2: Grayscale, Contrast Enhancement, and its Histogram
# Convert to grayscale if the image is RGB
if len(img.shape) == 3:
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
else:
    img_gray = img

# Display the grayscale image
plt.figure()
plt.imshow(img_gray, cmap='gray')
plt.axis('off')
plt.title('Grayscale Image')
plt.show()

# Contrast enhancement
img_contrast_enhanced = cv2.normalize(img_gray, None, alpha=0, beta=255,
```

```
norm_type=cv2.NORM_MINMAX)

# Display the contrast-enhanced image
plt.figure()
plt.imshow(img_contrast_enhanced, cmap='gray')
plt.axis('off')
plt.title('Contrast Enhanced Image')
plt.show()

#Figure 3: Histogram Equalized and Average Filtered Image and Its
Histogram
# Histogram equalization
img_histeq = cv2.equalizeHist(img_gray)

# Display the histogram equalized image
plt.figure()
plt.imshow(img_histeq, cmap='gray')
plt.axis('off')
plt.title('Equalized Image')
plt.show()

# Filtering using average filter
h_avg = np.ones((5, 5), np.float32) / 25
img_avg_filtered = cv2.filter2D(img_gray, -1, h_avg)

# Display the average filtered image
plt.figure()
plt.imshow(img_avg_filtered, cmap='gray')
plt.axis('off')
plt.title('Filtered Image (Average)')
plt.show()

#Figure 4: Median Filtered Image and Its Histogram
# Filtering using median filter
img_median_filtered = cv2.medianBlur(img_gray, 5)

# Display the median filtered image
plt.figure()
plt.imshow(img_median_filtered, cmap='gray')
plt.axis('off')
plt.title('Filtered Image (Median)')
plt.show()

# Display histograms for comparison
```

```
# Grayscale histogram
plt.figure()
plt.hist(img_gray.ravel(), 256, [0, 256])
plt.title('Histogram of Grayscale')
plt.show()

# Enhanced histogram (contrast adjustment)
plt.figure()
plt.hist(img_contrast_enhanced.ravel(), 256, [0, 256])
plt.title('Histogram of Enhanced Image')
plt.show()

# Equalized histogram
plt.figure()
plt.hist(img_histeq.ravel(), 256, [0, 256])
plt.title('Histogram of Equalized Image')
plt.show()

# Histogram (Average Filtered)
plt.figure()
plt.hist(img_avg_filtered.ravel(), 256, [0, 256])
plt.title('Histogram of Average Filtered Image')
plt.show()

# Histogram (Median Filtered)
plt.figure()
plt.hist(img_median_filtered.ravel(), 256, [0, 256])
plt.title('Histogram of Median Filtered Image')
plt.show()
```

III. Results

Steps:

1. Copy/crop and paste your results. Label each output (Figure1, Figure2, Figure3, Figure 4, and Figure 5)

picture file: flower.jpg

Figure 1: Acquire an Image of a Flower (MATLAB)

Figure 1.1: Acquire an Image of Flower (PYTHON)

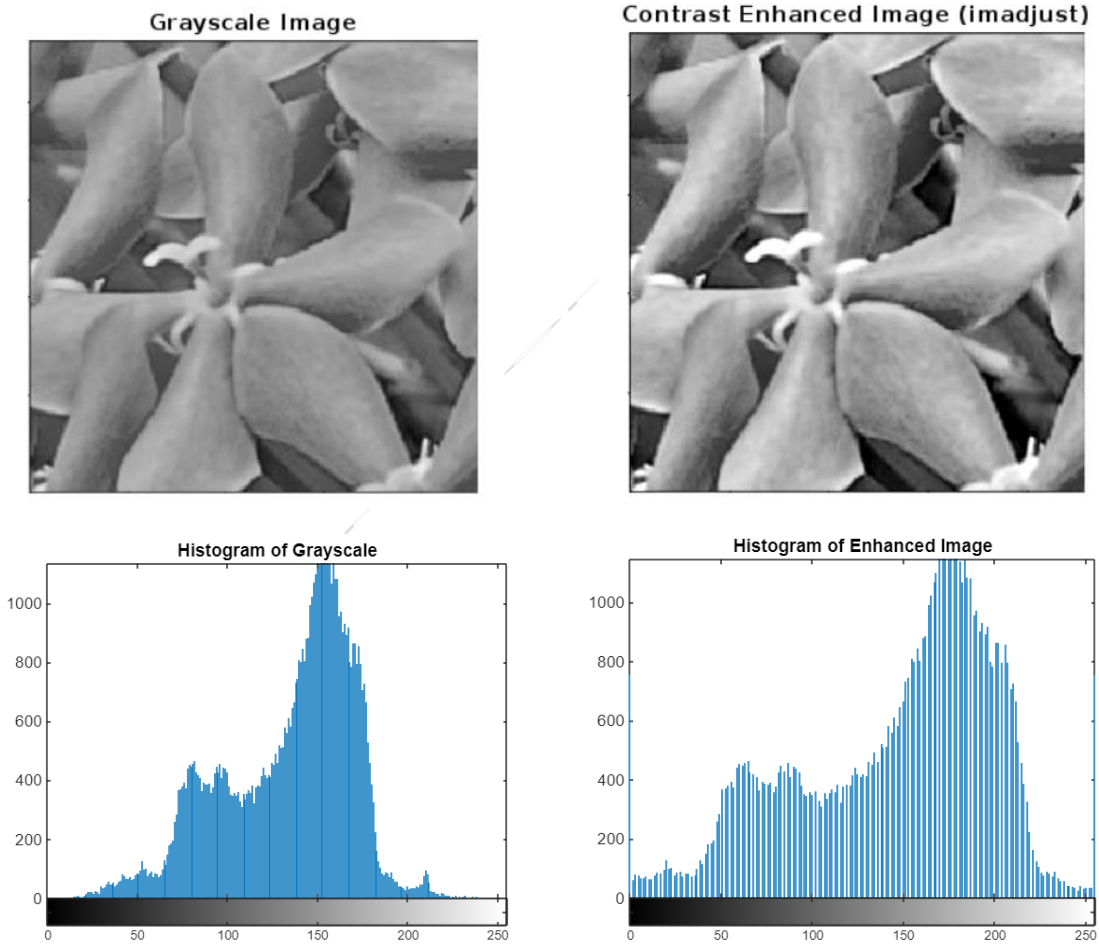

Figure 2: Grayscale, Contrast Enhancement, and its Histogram (MATLAB)

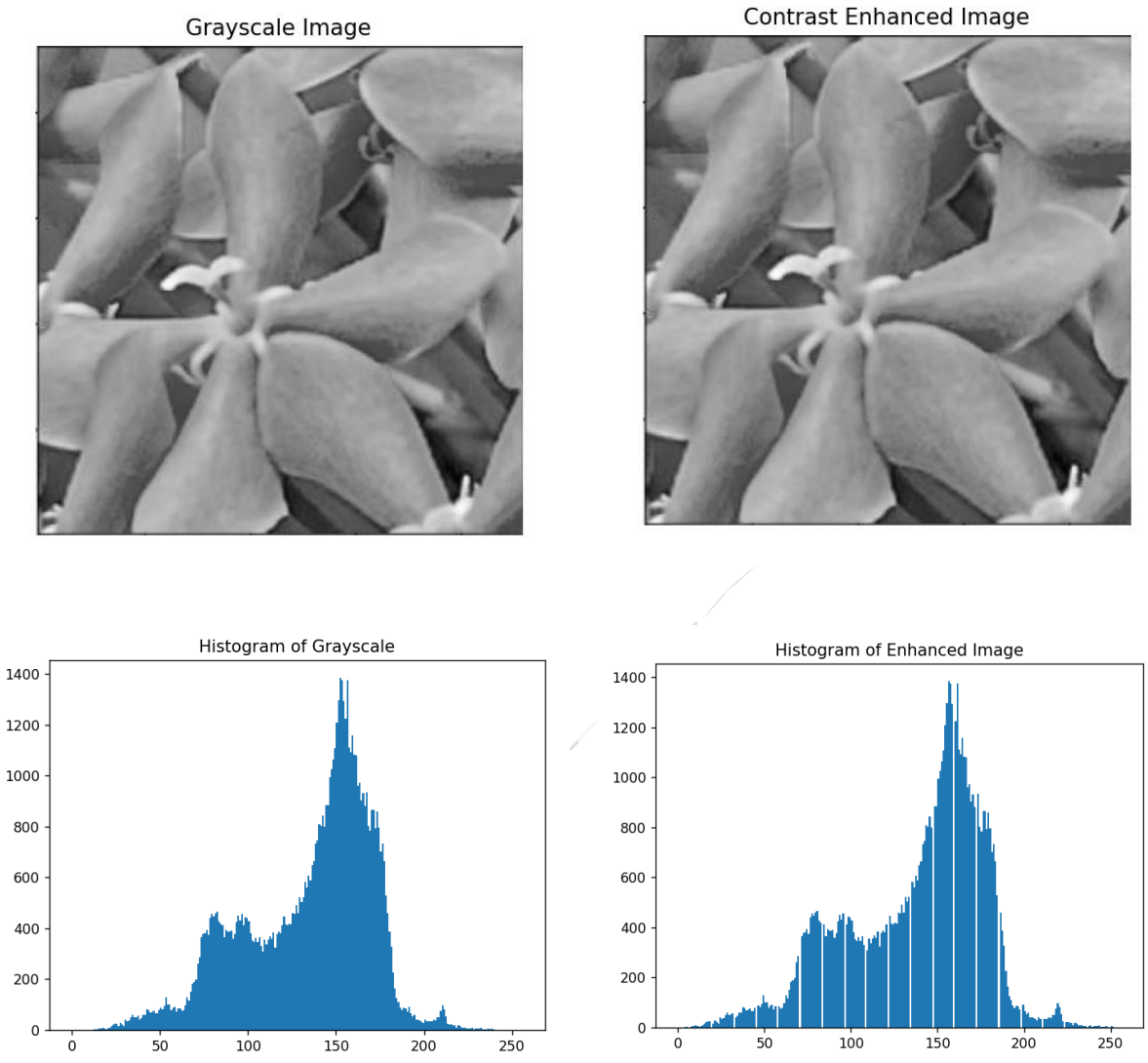

Figure 2.1: Grayscale, Contrast Enhancement, and its Histogram (PYTHON)

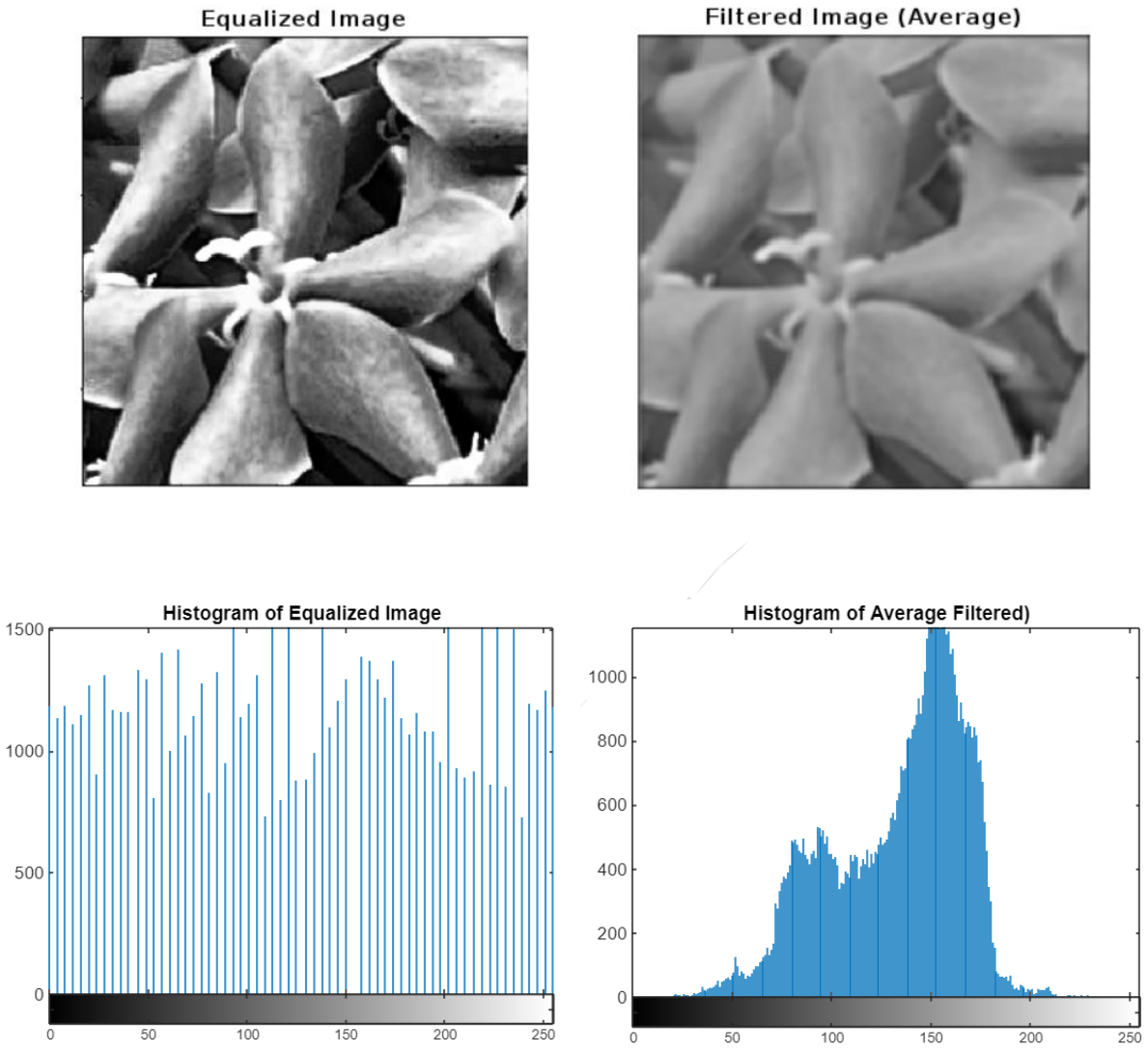

Figure 3: Histogram Equalized and Average Filtered Image and Its Histogram (MATLAB)

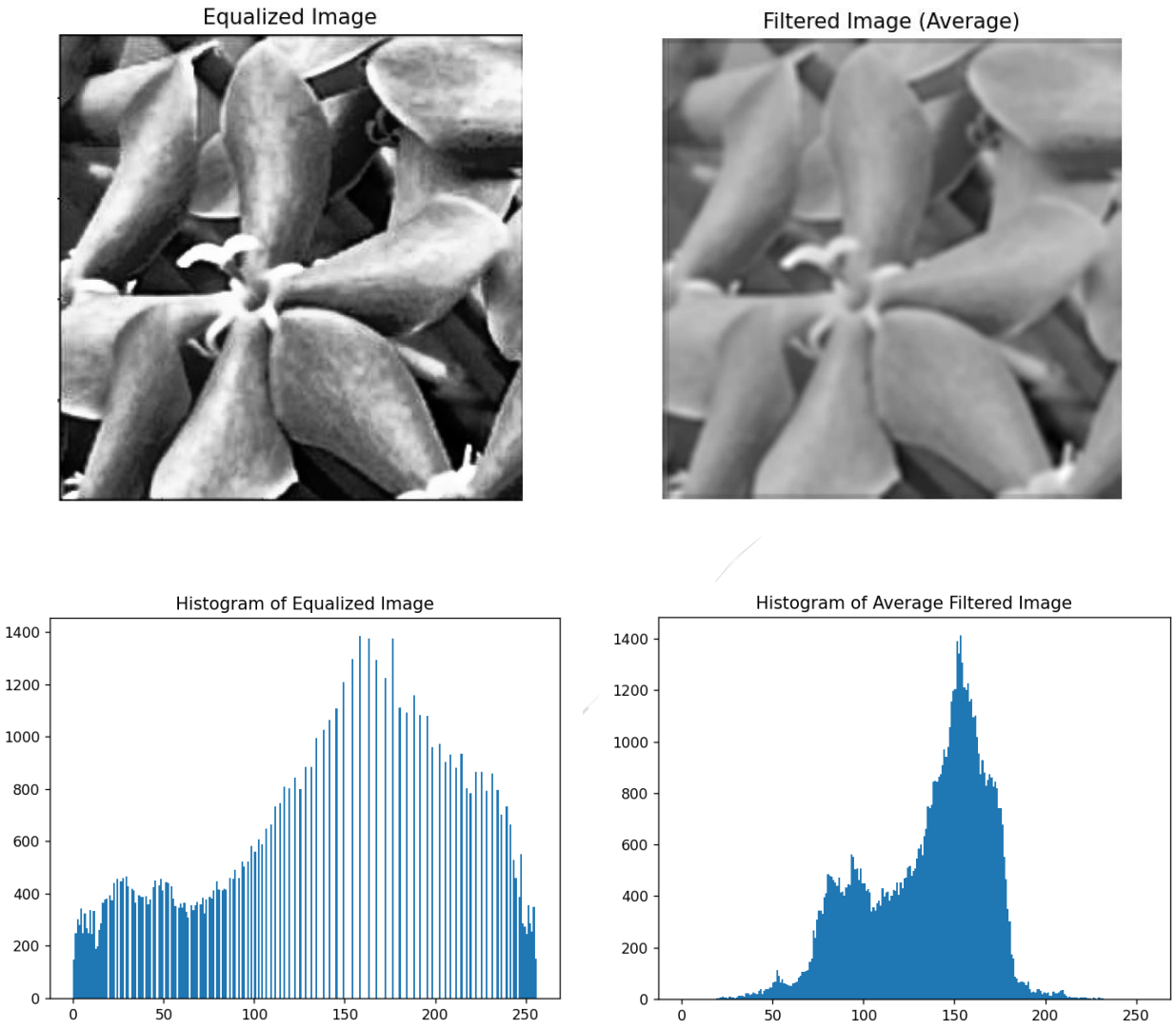

Figure 3.1: Histogram Equalized and Average Filtered Image and Its Histogram (PYTHON)

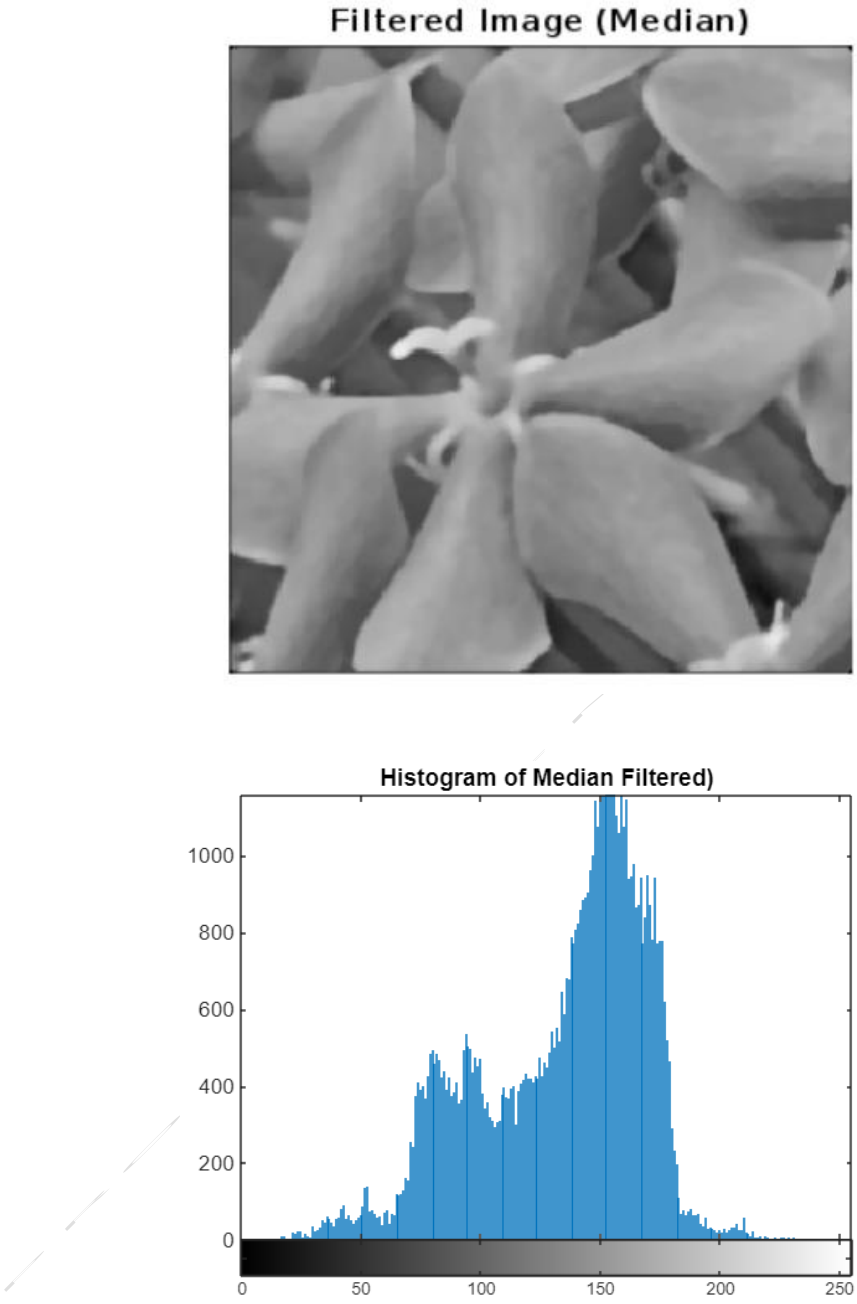

Figure 4: Median Filtered Image and Its Histogram (MATLAB)

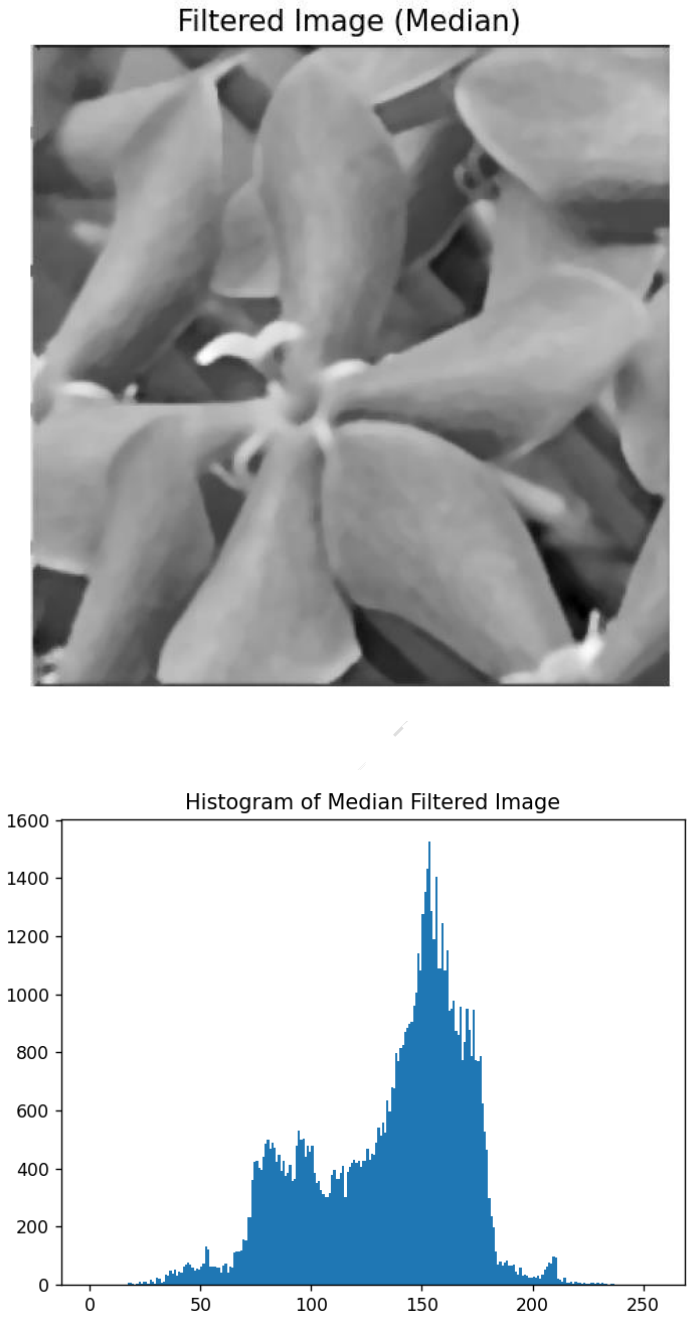

Figure 4.1: Median Filtered Image and Its Histogram (PYTHON)

These codes perform the following:

1. Grayscale conversion, which converts a color image (RGB) to a single-channel grayscale image. Colors are lost, but information about brightness is preserved. This depends on the desired outcome. If color information isn't crucial and you want to focus on brightness variations or prepare the image for further processing, grayscale conversion is effective. So in our image our original image is bright hence using the grayscale conversion is effective for our image that will be applied to other functions.
2. The Contrast Enhancement, which uses the function imadjust, stretches the contrast of the image by adjusting pixel values. Darker pixels become darker, and brighter pixels become brighter. This can make details in low-contrast areas more visible. The imadjust is effective for improving the visibility of features in images with low contrast. However, it can sometimes create an unnatural appearance or exaggerate noise in the image.
3. The Histogram Equalization uses the function histeq, which redistributes the pixel intensities in the image to create a flat histogram. This aims to achieve a more even distribution of brightness across the image. It is effective for images with uneven lighting or where specific features are obscured due to a concentration of pixels in a certain brightness range. It can enhance overall contrast and detail. However, it may sometimes create an overly artificial look or introduce artifacts.
4. Average filtering uses the function imfilter which replaces each pixel with the average value of its surrounding pixels which reduces noise in the image by blurring sharp edges and details. The average filter is effective for reducing random noise but can also blur important image features. It's good for removing minor noise while preserving larger structures.
5. Median filtering uses the function medfilt2 which replaces each pixel with the median value of its surrounding pixels. Similar to the average filter, it reduces noise but is less prone to blurring edges. It's particularly effective for removing salt-and-pepper noise (random black and white pixels). The median filter offers a good balance between noise reduction and edge preservation.

And lastly, each image uses a histogram through a function hist, which helps visualize the distribution of pixel intensities. Visualizing histograms allows you to understand the original contrast distribution (grayscale) and how it's affected by the applied algorithms (contrast enhancement, equalization, filtering). This helps assess the effectiveness of each step.

Parameter Modification

MATLAB:

```
% Convert to grayscale if the image is RGB
if size(img, 3) == 3
    img_gray = rgb2gray(img);
else
    img_gray = img;
end
% Filtering using average filter but different values
h_avg = fspecial('average', [10, 10]); % Original is [5,5]
img_avg_filtered = imfilter(img_gray, h_avg);

% Show the experimented image
figure;
imshow(img_avg_filtered);
title('Filtered Image (Using Average but Different values)');
```

```
% Filtering using median filter
img_median_filtered = medfilt2(img_gray, [1, 10]); % Original is [5,5]

% Display the median filtered image
figure;
imshow(img_median_filtered);
title('Experimented Filtered Image (Median)');

% Show the Histogram
figure;
imhist(img_median_filtered);
title('Histogram of Experimented Median Filtered)');
```

PYTHON:

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

# Filtering using average filter with different values
h_avg = np.ones((10, 10), np.float32) / 100
img_avg_filtered = cv2.filter2D(img_gray, -1, h_avg)

# Display the average filtered image with different values
plt.figure()
plt.imshow(img_avg_filtered, cmap='gray')
plt.axis('off')
plt.title('Filtered Image (Using Average but Different Values)')
plt.show()

# Filtering using median filter with different values
img_median_filtered = cv2.medianBlur(img_gray, 11)

# Display the median filtered image with different values
plt.figure()
plt.imshow(img_median_filtered, cmap='gray')
plt.axis('off')
plt.title('Experimented Filtered Image (Median)')
plt.show()

# Histogram (Median Filtered)
plt.figure()
plt.hist(img_median_filtered.ravel(), 256, [0, 256])
plt.title('Histogram of Experimented Median Filtered)')
plt.show()
```

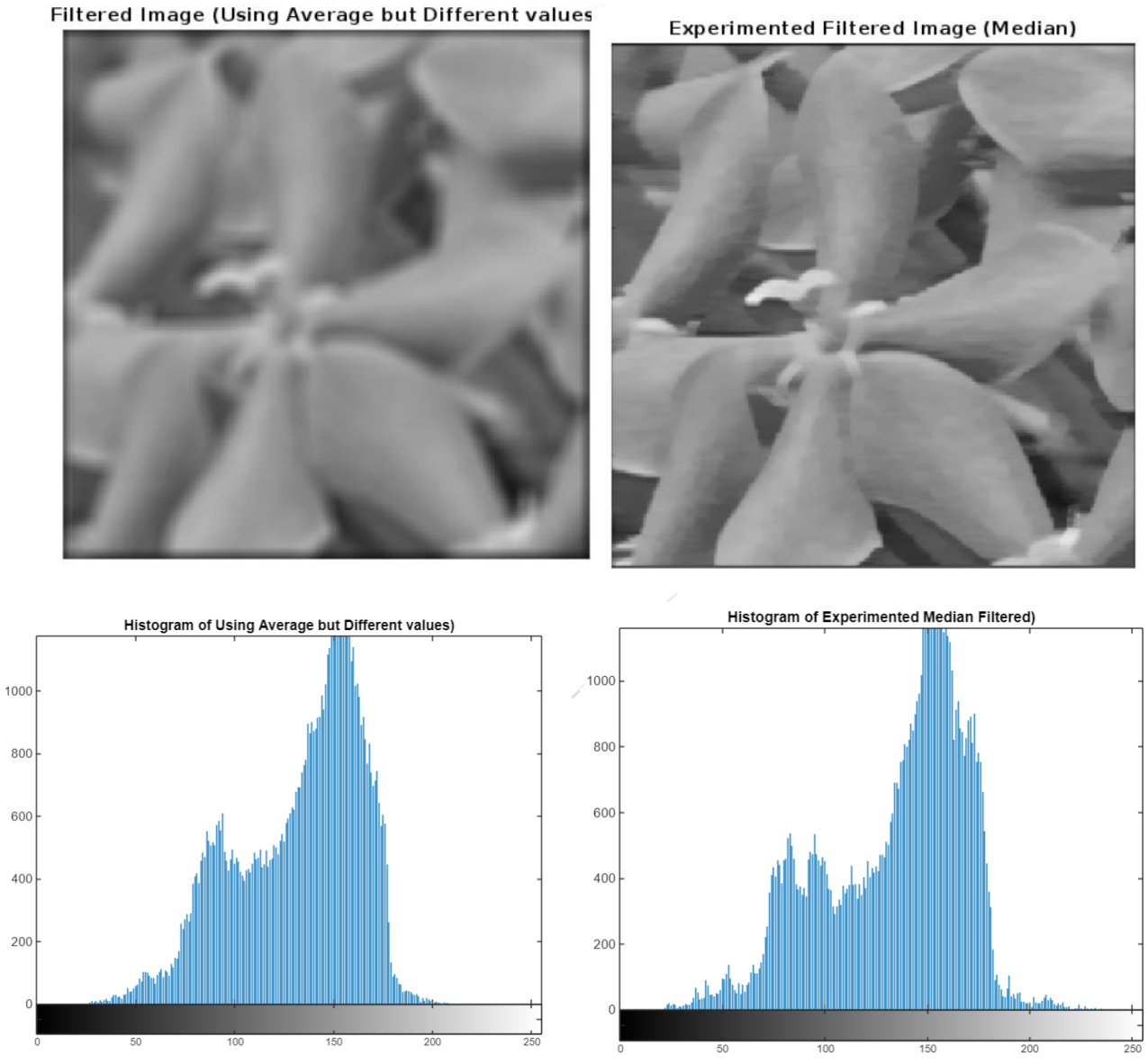

Figure 5: Parameters Modification and Its Histogram (MATLAB)

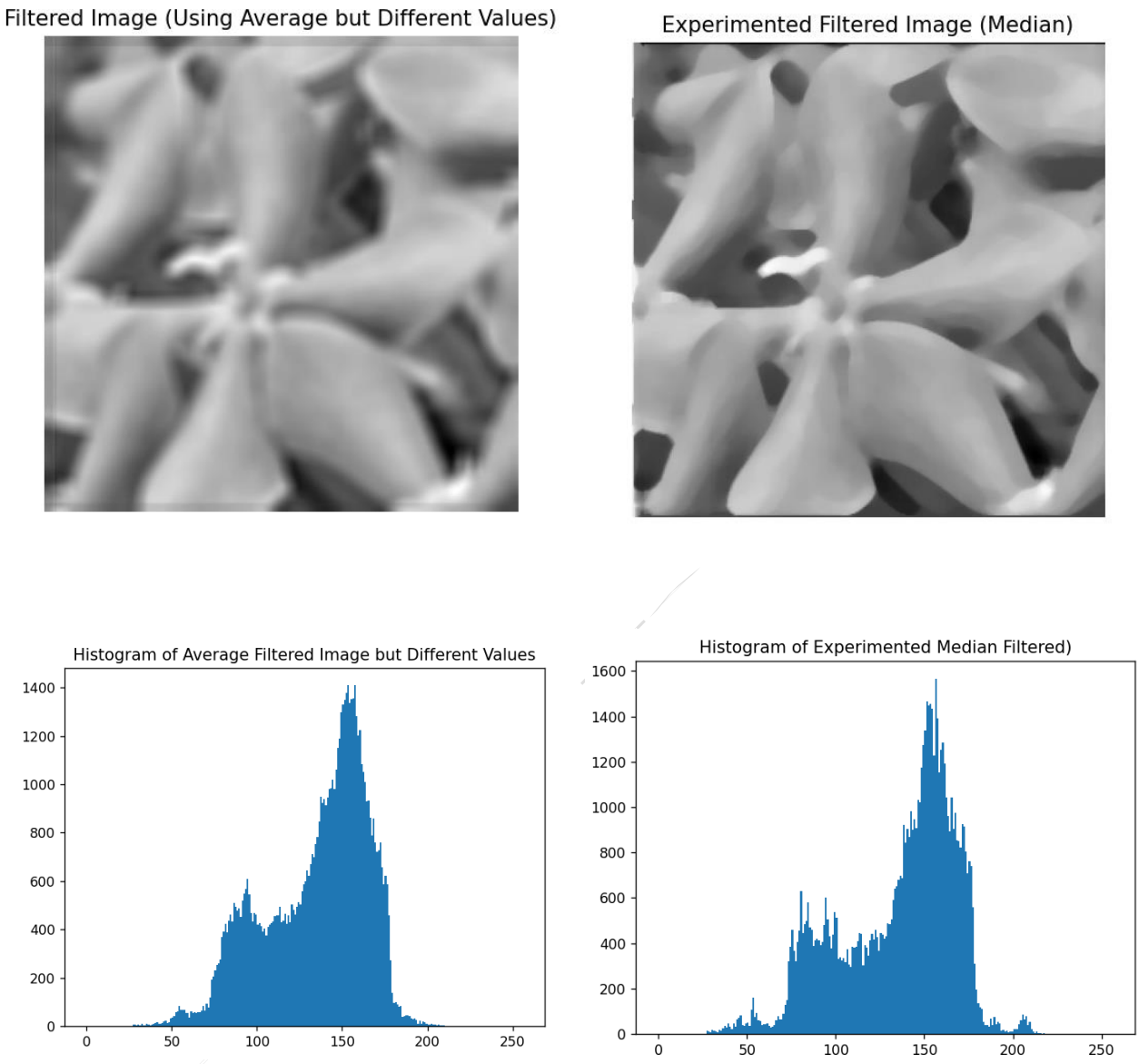

Figure 5.1: Parameters Modification and Its Histogram (PYTHON)

2. Visualize the results, analyze and interpret:

In comparing image enhancement techniques between MATLAB and Python, both platforms demonstrated consistency in grayscale conversion, histogram equalization, and filtering. MATLAB's contrast enhancement using imadjust produced brighter images with higher overall light intensity but slightly exaggerated noise, while Python's cv2.normalize yielded more subdued images with sharper focus and clearer details. Histogram equalization and filtering results were similar across both platforms, with MATLAB occasionally introducing a more artificial look in equalized images. Overall, MATLAB tended to enhance brightness and vibrancy, whereas Python focused on detail clarity and subtlety.

IV. Conclusion

In conclusion, after successfully performing this laboratory exercise, we have applied various image processing techniques, including image acquisition and color application, like previous laboratory exercises. While earlier activities focused on image acquisition and color application, this exercise include additional image processing techniques such as displaying histograms, contrast enhancement, filtering in spatial (i.e., average and median) and lastly, image blurring. Moreover, this activity utilizes same tools as the previous laboratory exercise such as MATLAB and Python. Both displayed the image appropriately.

References

[1] D.J.D. Sayo. "University of the City of Manila Computer Engineering Department Honor Code," PLM-CpE Departmental Policies, 2020.

<This is in a separate page>