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

Elective 3

Laboratory Activity No. 3
Image Enhancement

Score

Submitted by:
Cacal, Chad R.
Dela Cruz, Alfonso Rafael C.
Palabay, Joven Carl B.
Suyu, Chester T.
SAT 7:00AM – 4:00PM / 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 filterh
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

- Write a Python program that will implement the output in Method A.

```
import cv2
import matplotlib.pyplot as plt
# Acquire the image
img = cv2.imread('flower.jpg')

# Display the original image
cv2.imshow('Original Image', img)

# Convert to grayscale if the image is RGB
if img.shape[2] == 3:
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
else:
    img_gray = img

# Display the grayscale image
cv2.imshow('Grayscale Image', img_gray)

# Contrast enhancement using cv2.convertScaleAbs
img_contrast_enhanced = cv2.convertScaleAbs(img_gray, alpha=1.25, beta=0)

# Display the contrast-enhanced image
cv2.imshow('Contrast Enhanced Image', img_contrast_enhanced)

# Histogram equalization
img_histeq = cv2.equalizeHist(img_gray)

# Display the histogram equalized image
cv2.imshow('Equalized Image', img_histeq)

# Filtering using average
img_avg_filtered = cv2.blur(img_gray, (5, 5))

# Display the average filtered image
cv2.imshow('Filtered Image (Average)', img_avg_filtered)

# Filtering using median
img_median_filtered = cv2.medianBlur(img_avg_filtered, 5)

# Display the median filtered image
cv2.imshow('Filtered Image (Median)', img_median_filtered)
```

```
# Display histograms for comparison

# Grayscale histogram
hist1 = cv2.calcHist([img_gray], [0], None, [256], [0, 256])
plt.figure(1)
plt.title('Histogram of Grayscale')
plt.plot(hist1, color='black')

# Enhanced histogram
hist2 = cv2.calcHist([img_contrast_enhanced], [0], None, [256], [0, 256])
plt.figure(2)
plt.title('Histogram of Enhanced Image')
plt.plot(hist2, color='black')

# Equalized histogram
hist3 = cv2.calcHist([img_histeq], [0], None, [256], [0, 256])
plt.figure(3)
plt.title('Histogram of Equalized Image')
plt.plot(hist3, color='black')

# Histogram (Average Filtered)
hist4 = cv2.calcHist([img_avg_filtered], [0], None, [256], [0, 256])
plt.figure(4)
plt.title('Histogram of Average Filtered')
plt.plot(hist4, color='black')

# Histogram (Median Filtered)
hist5 = cv2.calcHist([img_median_filtered], [0], None, [256], [0, 256])
plt.figure(5)
plt.title('Histogram of Median Filtered')
plt.plot(hist5, color='black')
plt.tight_layout();plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

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

MATLAB Results

Figure 1a. Acquire an Image of a Flower

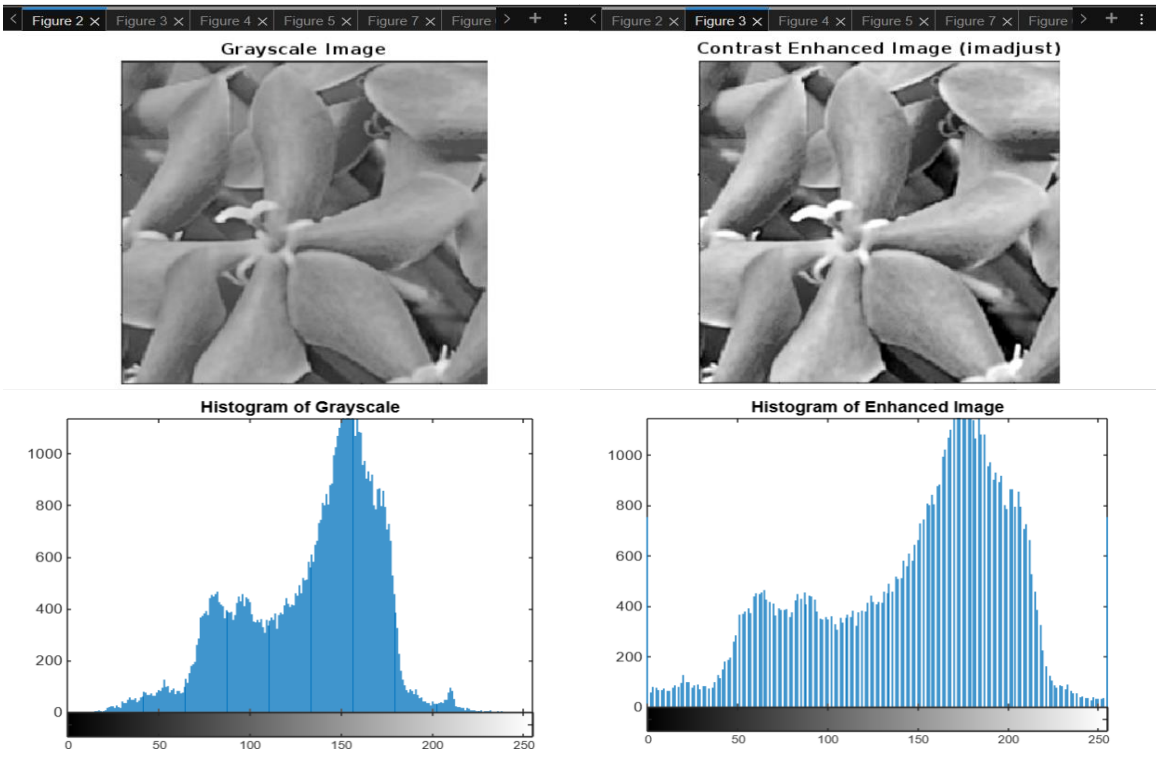

Figure 2a. Grayscale, Contrast Enhancement, and its Histogram

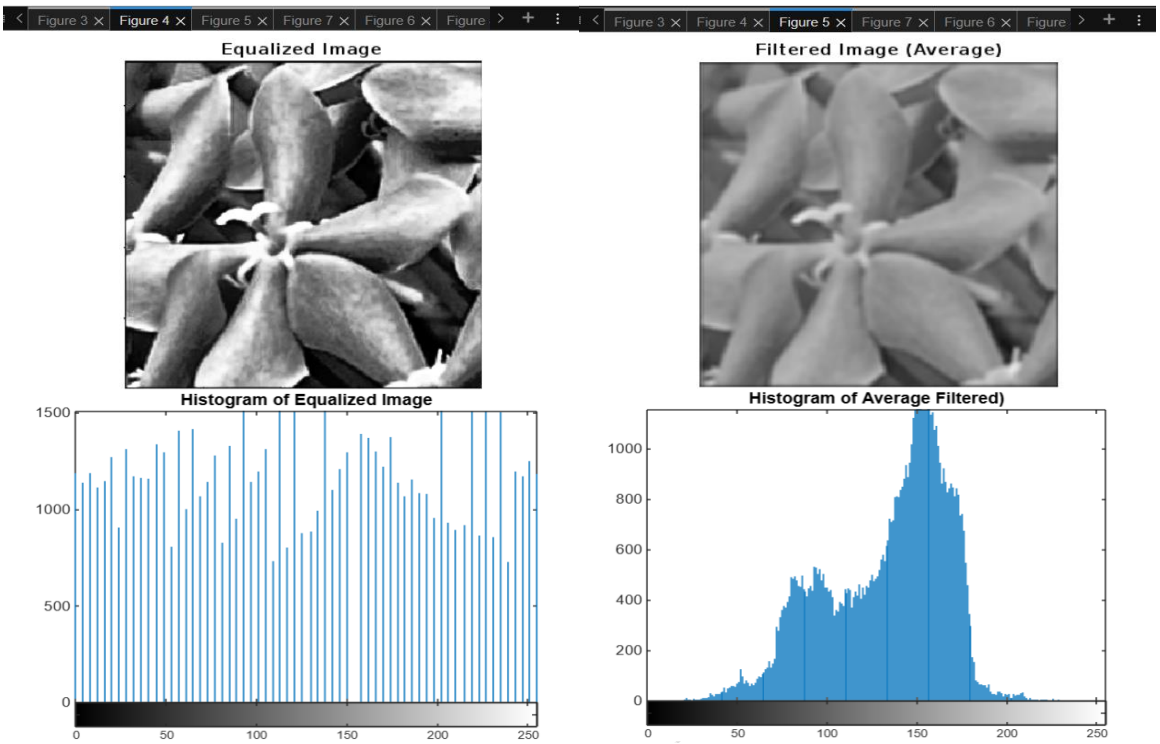

Figure 3a. Histogram Equalized and Average Filtered Image and Its Histogram

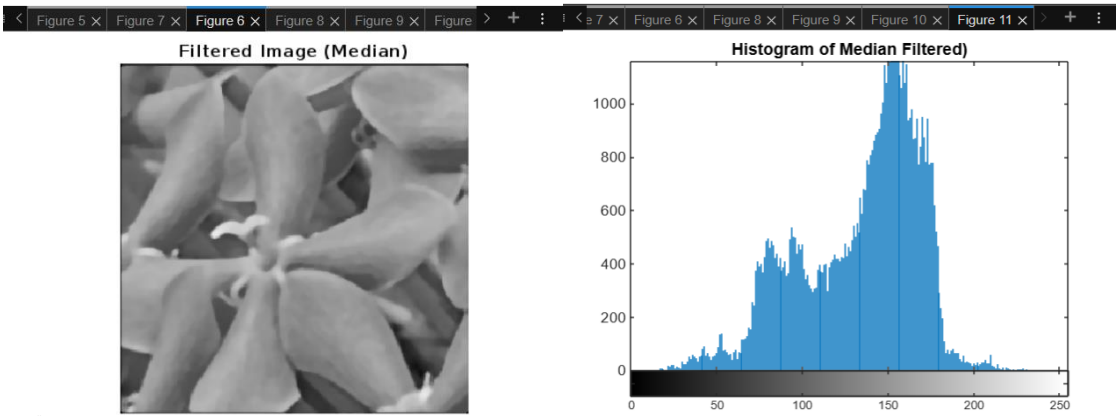

Figure 4a. Median Filtered Image and Its Histogram

Octave Results

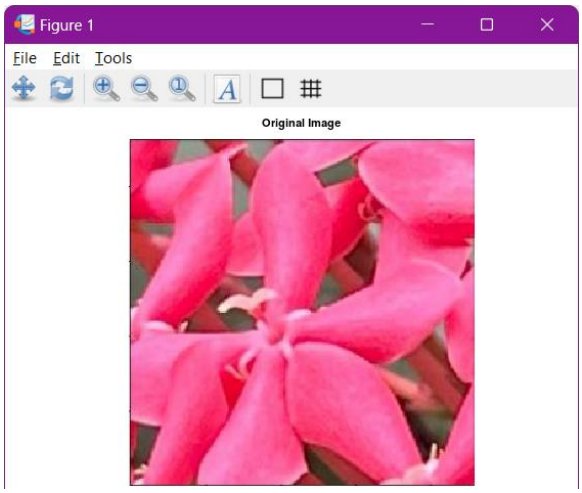

Figure 1b. Acquire an Image of a Flower

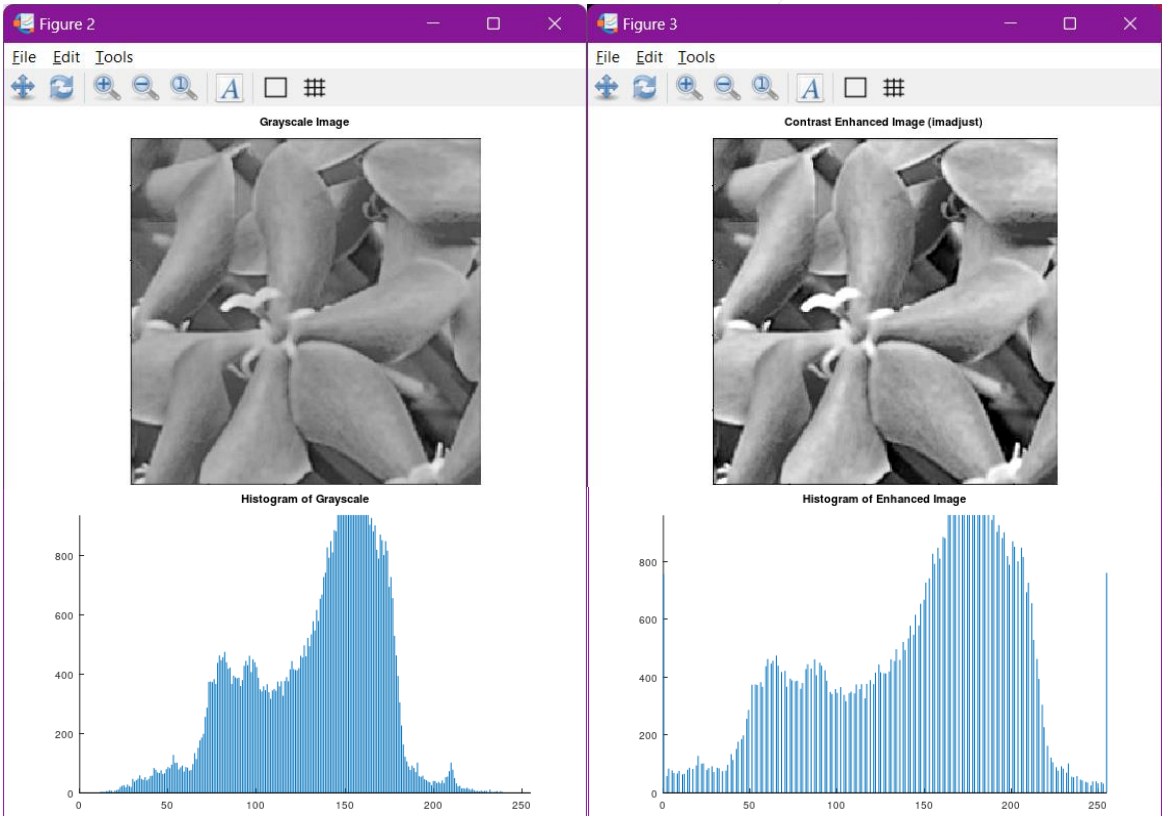

Figure 2b. Grayscale, Contrast Enhancement, and its Histogram

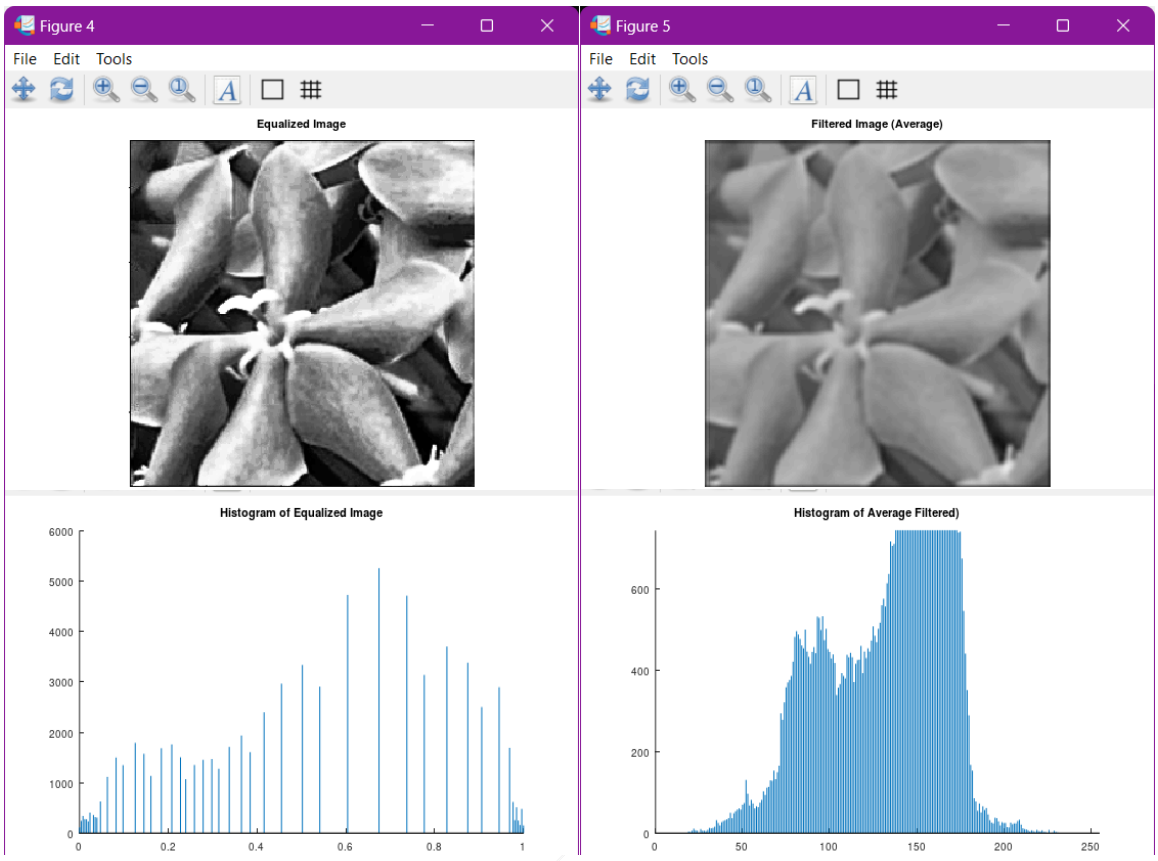

Figure 3b. Histogram Equalized and Average Filtered Image and Its Histogram

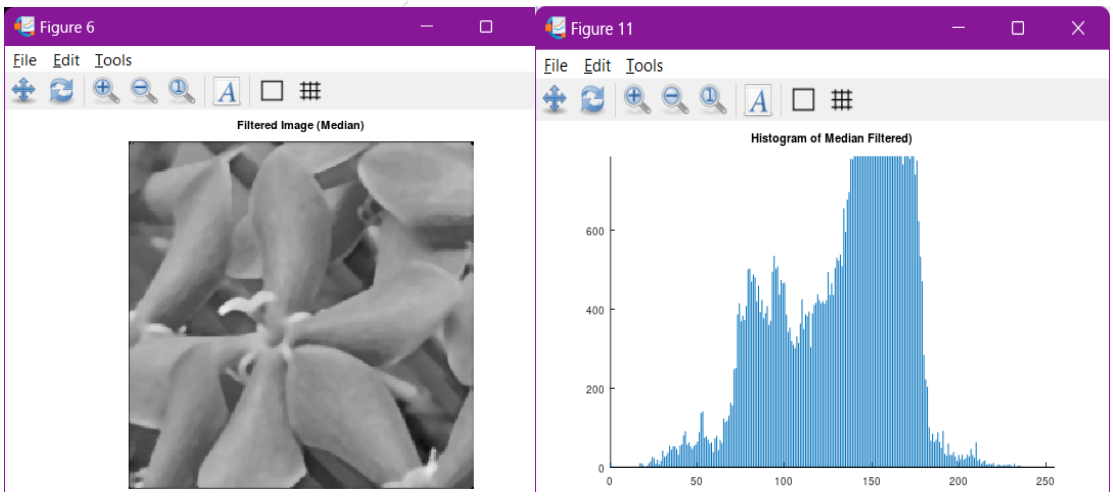

Figure 4b. Median Filtered Image and Its Histogram

Python Results

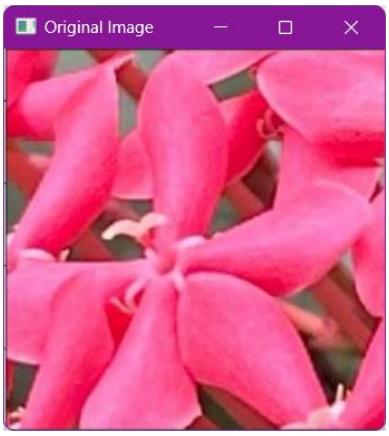

Figure 1c. Acquire an Image of a Flower

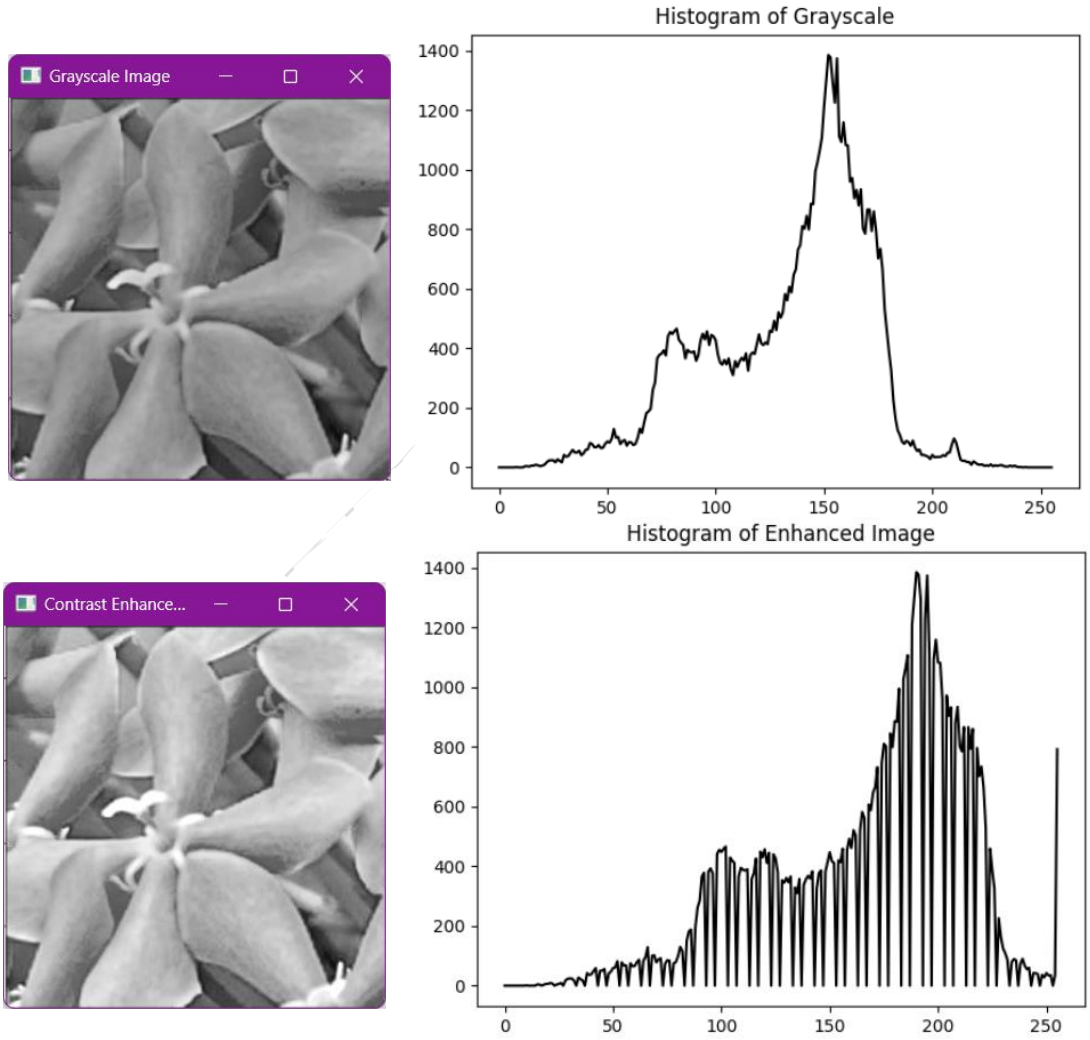

Figure 2c. Grayscale, Contrast Enhancement, and its Histogram

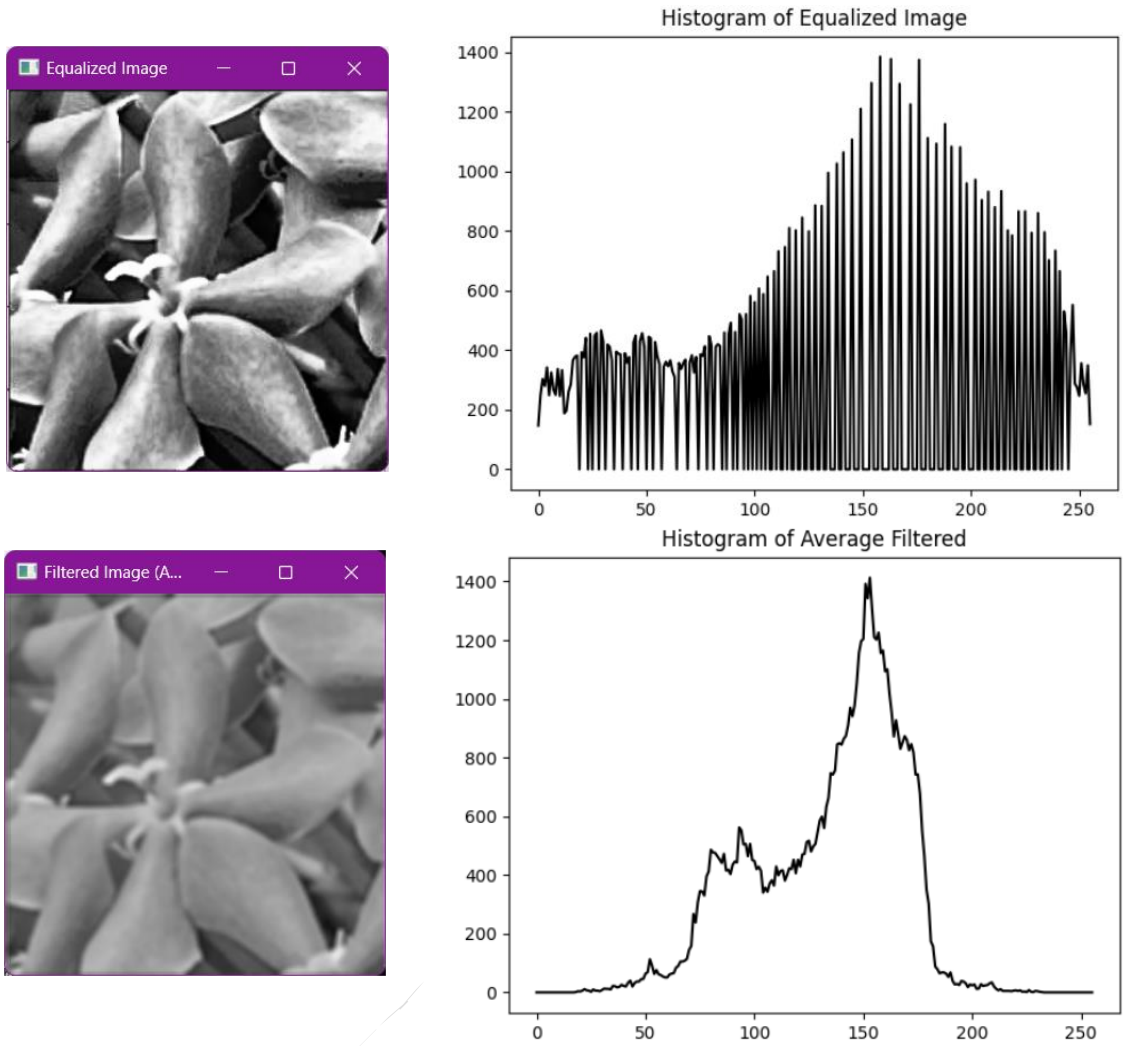

Figure 3c. Histogram Equalized and Average Filtered Image and Its Histogram

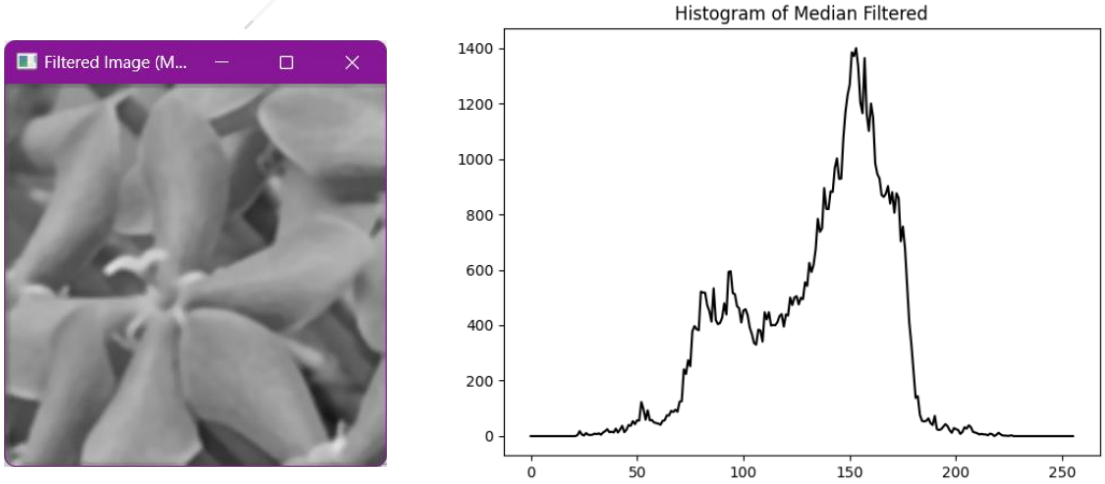

Figure 4c. Median Filtered Image and Its Histogram

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

<You can modify it to explore other functionalities>

```
% 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)');
```

MATLAB Results

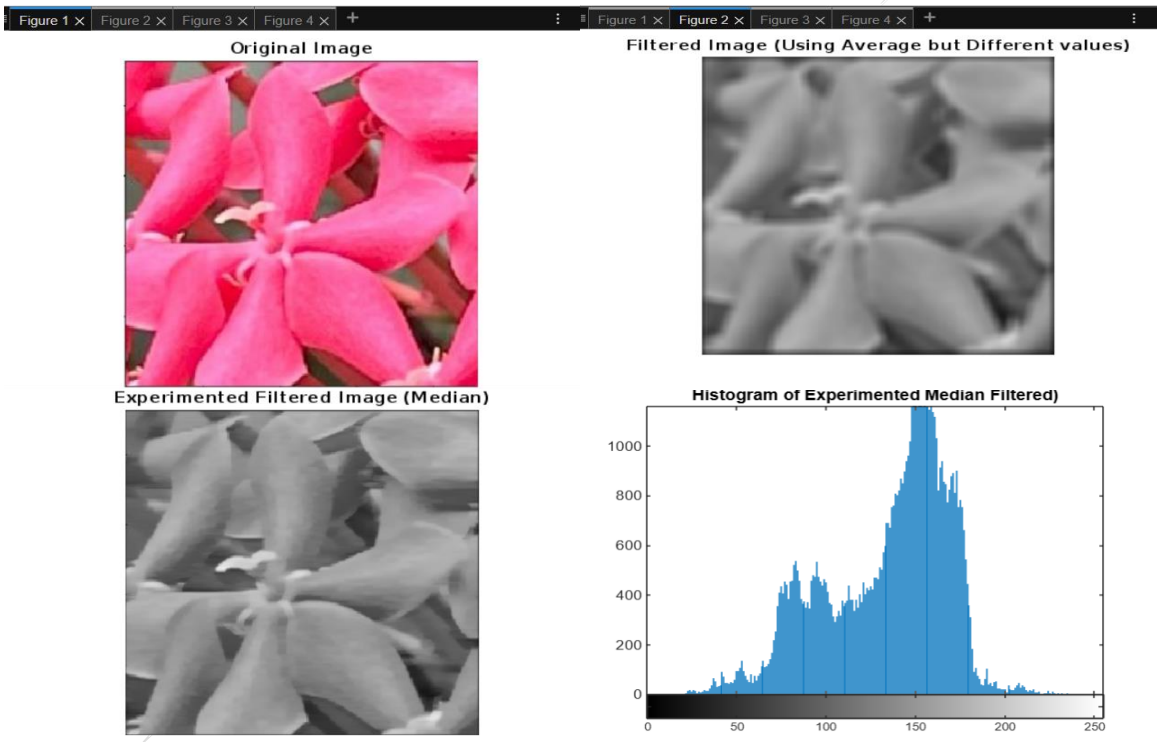

Figure 5a. Parameters Modification and Its Histogram

Octave Results

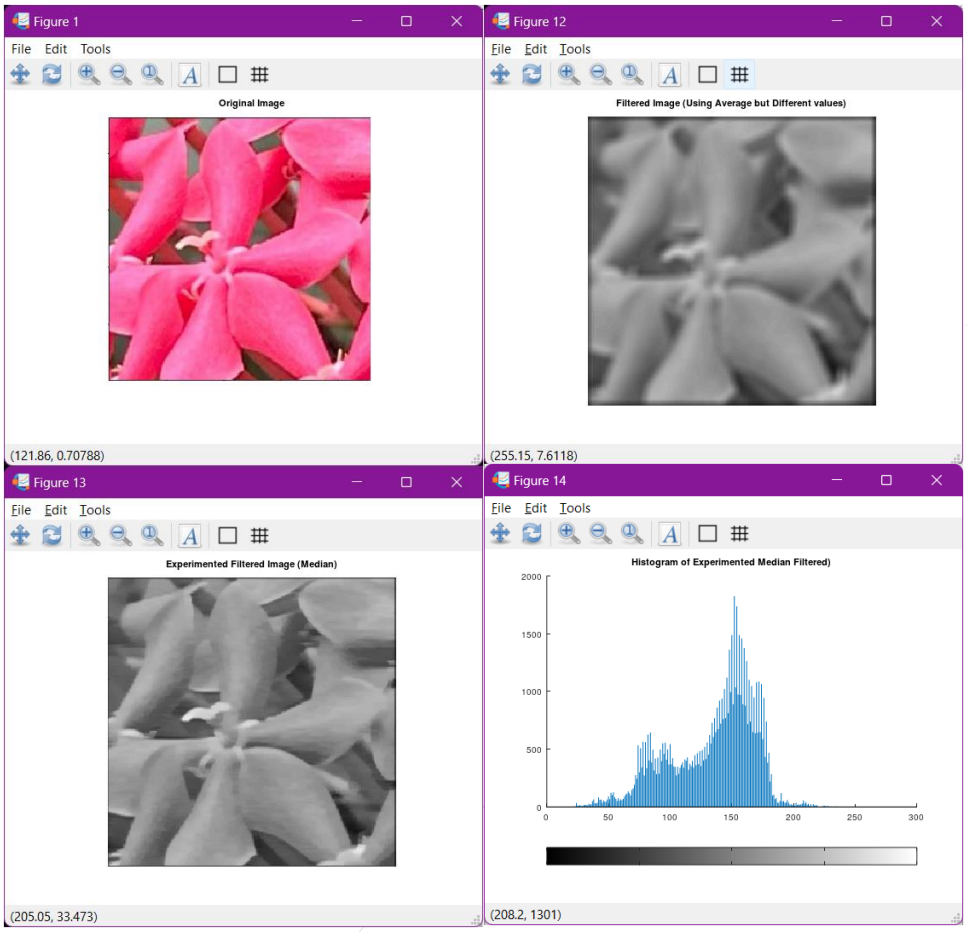

Figure 5b. Parameters Modification and Its Histogram

Python Results

Figure 5c. Parameters Modification and Its Histogram

2. Visualize the results, analyze and interpret:

< Discuss the effects of the applied algorithm on the image and its effectiveness in achieving the desired outcome. >

Beginning with the acquisition of the image, all the applied algorithms seamlessly read the image, consistent with the procedures followed in previous activities. However, when performing histogram equalization on the image, we observed notable discrepancies in the histograms generated by Octave, Python, and MATLAB. Specifically, the histogram displayed by MATLAB appeared more evenly distributed, suggesting a more uniform spread of pixel intensities. In contrast, the histograms produced by Octave and Python were skewed to the right, indicating that a majority of the pixel values had higher intensities. This variation can be attributed to the differences in the implementation of histogram equalization across these platforms.

It is plausible that the underlying algorithms for histogram equalization in MATLAB differ in their approach, resulting in a more balanced distribution of pixel intensities. Conversely, Octave and Python might employ methods that lead to a concentration of pixel values in the higher intensity range. This disparity underscores the importance of understanding the specific mechanics of each platform's implementation when performing image processing tasks.

Moving forward, we observed that all the applied algorithms were effective in stretching the image to utilize the entire range of intensities during contrast enhancement. This process ensured that the images displayed a broader spectrum of intensity values, thereby enhancing their visual quality as evidenced in the results presented above. Additionally, the algorithms used for filtering in the spatial domain, including both average and median filtering, consistently yielded the intended outcomes. These filters effectively smoothed the images and reduced noise, demonstrating their reliability and utility in spatial domain image processing.

IV. Conclusion

In conclusion, this laboratory activity demonstrates the practical application and comparative analysis of image enhancement techniques across MATLAB/Octave and OpenCV with Python. While all platforms effectively implemented the core principles of image enhancement, the differences in histogram equalization outcomes emphasize the need to understand each platform’s unique algorithmic approach. Overall, the successful application of contrast enhancement and spatial filtering techniques across platforms reaffirms their importance and utility in the field of image processing.

References

<This is in a separate page>

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

Geeks for Geeks. (2023, February 15). *Image Enhancement Techniques using OpenCV – Python.*
https://www.geeksforgeeks.org/image-enhancement-techniques-using-opencv-python/

Geeks for Geeks. (2024, July 30). *Histograms Equalization in OpenCV.*
https://www.geeksforgeeks.org/histograms-equalization-opencv/

Wood, K. (2023, June 15). *OpenCV Python Average Filtering [Youtube].*
https://www.youtube.com/watch?v=_MDV3pBGsfY

Wood, K. (2023, June 16). *OpenCV Python Median Filtering [Youtube].*
https://www.youtube.com/watch?v=7H2ixXwd7T0

Geeks for Geeks. (2023, January 03). *Python OpenCV – cv2.calcHist method.*
https://www.geeksforgeeks.org/python-opencv-cv2-calchist-method/