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

Elective 3

Laboratory Activity No. 3
Image Enhancement

Score

Submitted by:
Alambra, Joseph Nathaniel A.
Aragon, Patrick Laurence M.
Banal, Daryll L.
Sentasas, David Bryan L.
Tutanes, Allen Christopher O.
SAT 7:00AM – 4:00PM / CPE 0332.1-1

Date Submitted
03-08-2024

Submitted to:

Engr. Maria Rizette H. Sayo

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

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.

C:\Users\daryl\Downloads\Lab_Activity3_Python.py

untitled0.py* | Lab_Activity3_Python.py

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Aug  3 01:27:10 2024

@author: Alambra, Joseph Nathaniel A.
Aragon, Patrick Laurence M.
Banal, Daryll L.
Sentasas, David Bryan L.
Tutanes, Allen Christopher O.

"""


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)
```

C:\Users\daryl\Downloads\Lab_Activity3_Python.py

untitled0.py* | Lab_Activity3_Python.py

```python
# 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)

# Filtering using average filter but different values
img_avg_filtered2 = cv2.blur(img_gray, (10, 10))

# Display the median filtered image
cv2.imshow('Filtered Image (Using Average but Different Values)', img_avg_filtered2)

# Filtering using median filter but different values
img_median_filtered2 = cv2.medianBlur(img_gray, 1)

# Display the median filtered image
cv2.imshow('Experimented Filtered Image (Median)', img_median_filtered2)

# 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')
```

C:\Users\daryl\Downloads\Lab_Activity3_Python.py

untitled0.py* | Lab_Activity3_Python.py

```python
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')

# Histogram (Experimented Median Filtered)
hist6 = cv2.calcHist([img_median_filtered2], [0], None, [256], [0, 256])
plt.figure(6)
plt.title('Histogram of Experimented Median Filtered')
plt.plot(hist6, color='black')

plt.tight_layout()
plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

III. Results

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

PYTHON RESULTS

picture file: flower.jpg

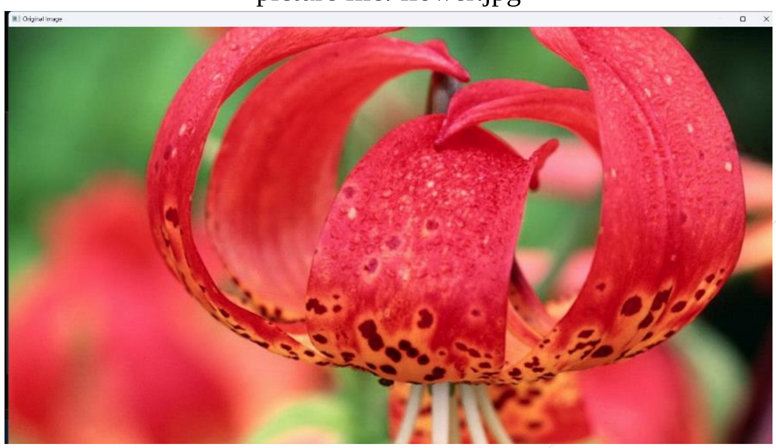


Figure 1A: Acquire an Image of a Flower

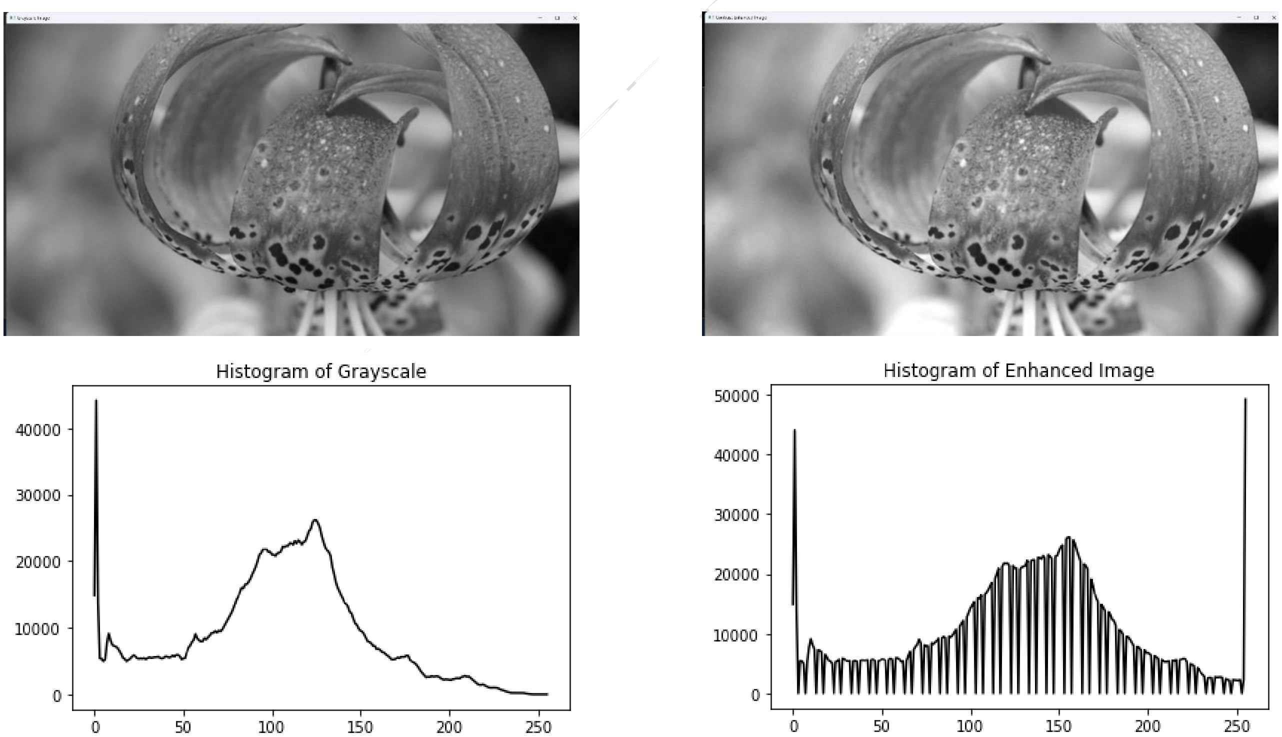


Figure 2: Grayscale, Contrast Enhancement, and its Histogram

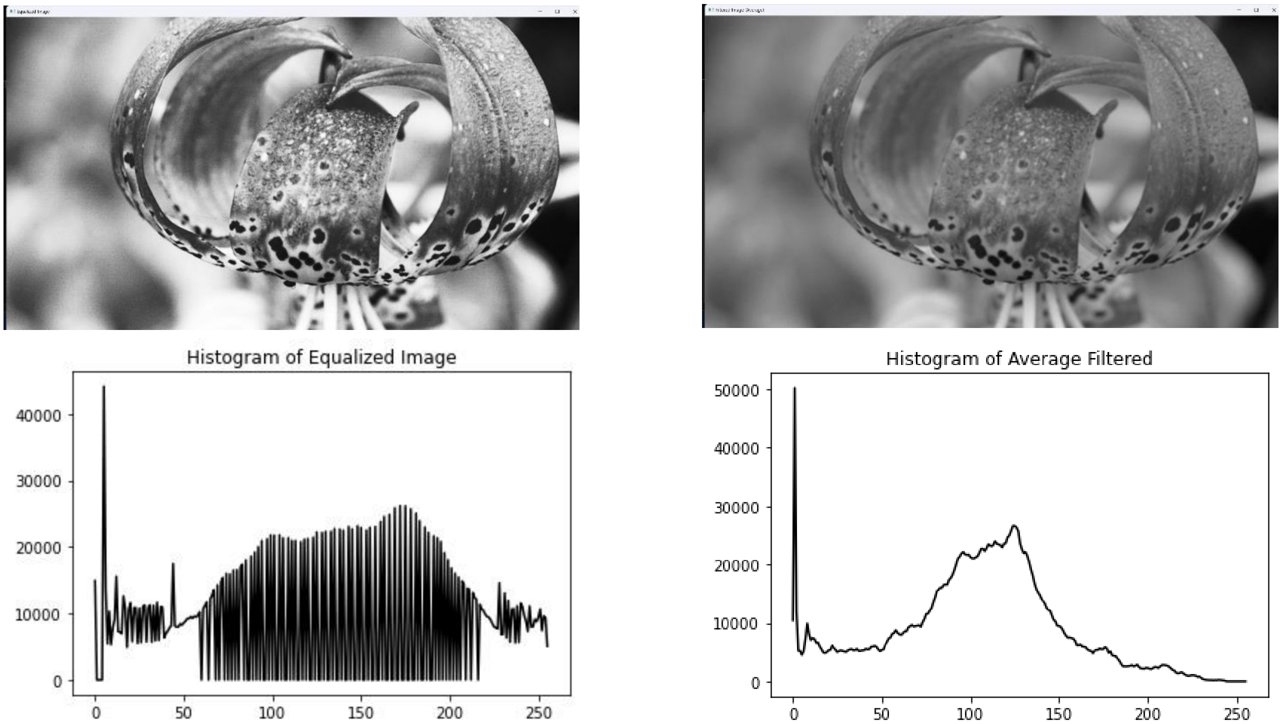


Figure 3: Histogram Equalized and Average Filtered Image and Its Histogram

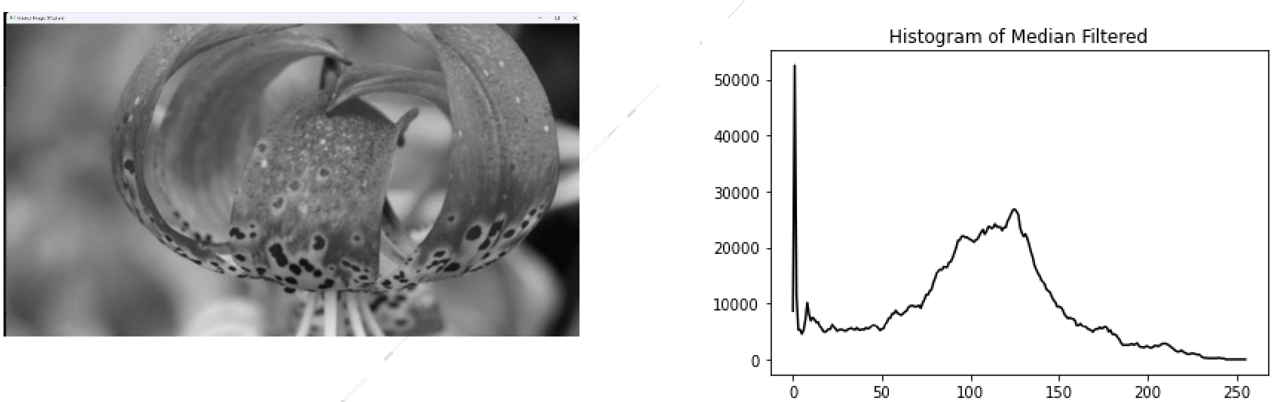


Figure 4: Median Filtered Image and Its Histogram

OCTAVE RESULTS

picture file: flower.jpg

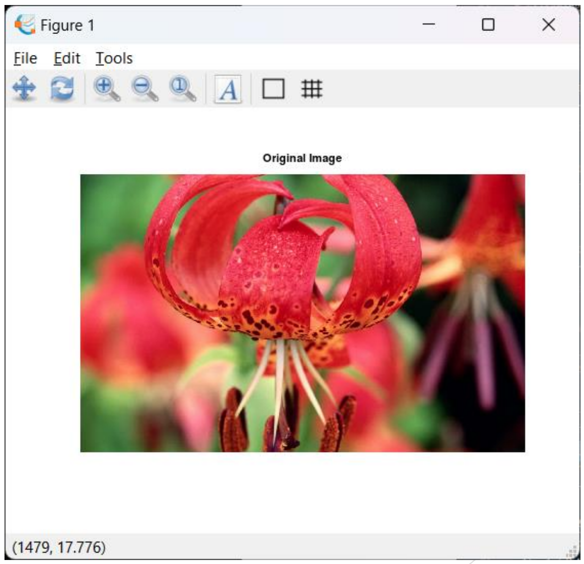
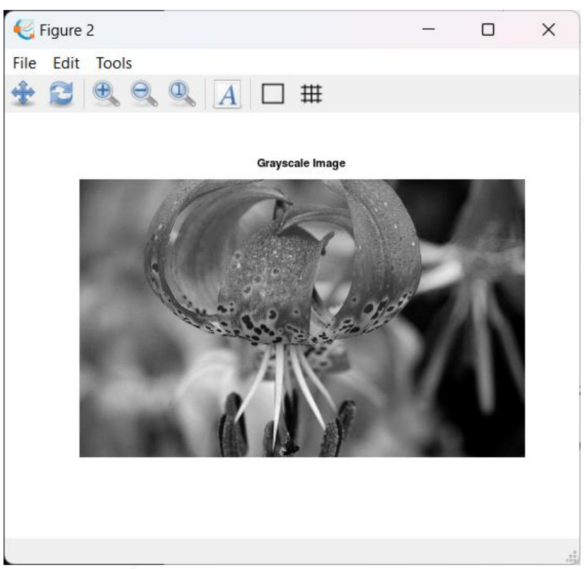
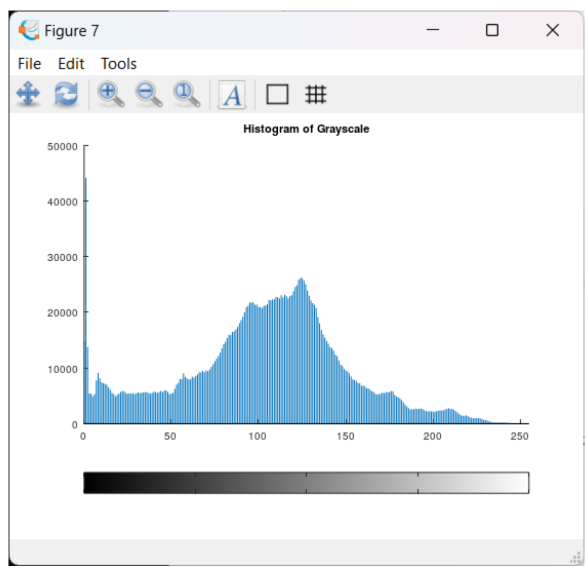


Figure 1B: (OCTAVE) Acquire an Image of a Flower





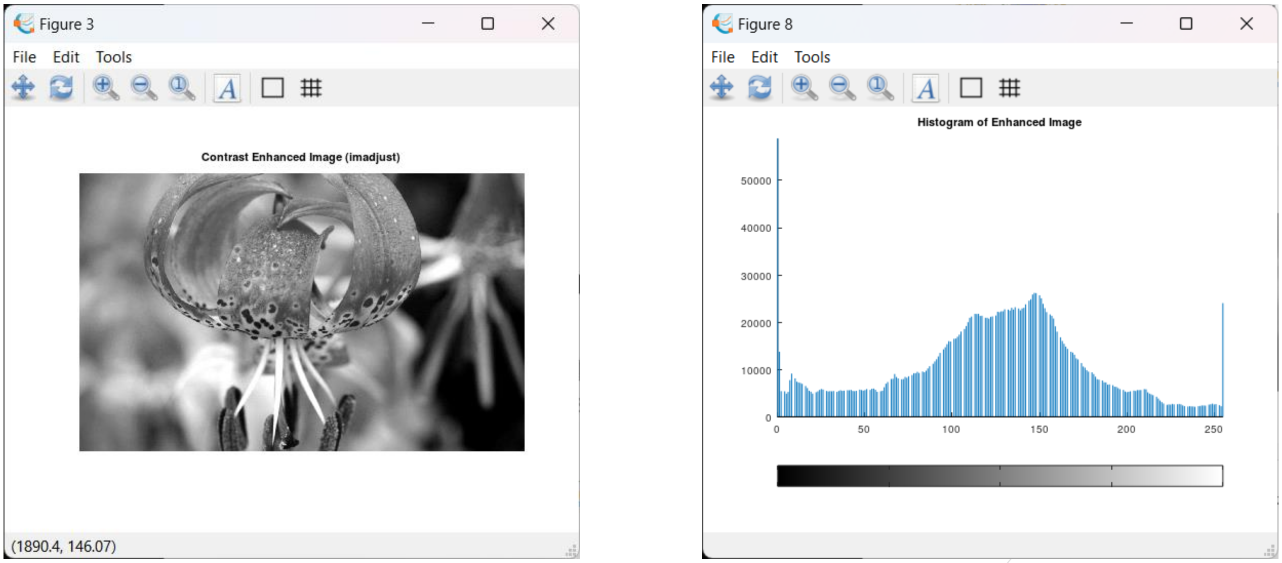


Figure 2B: (OCTAVE) Grayscale, Contrast Enhancement, and its Histogram

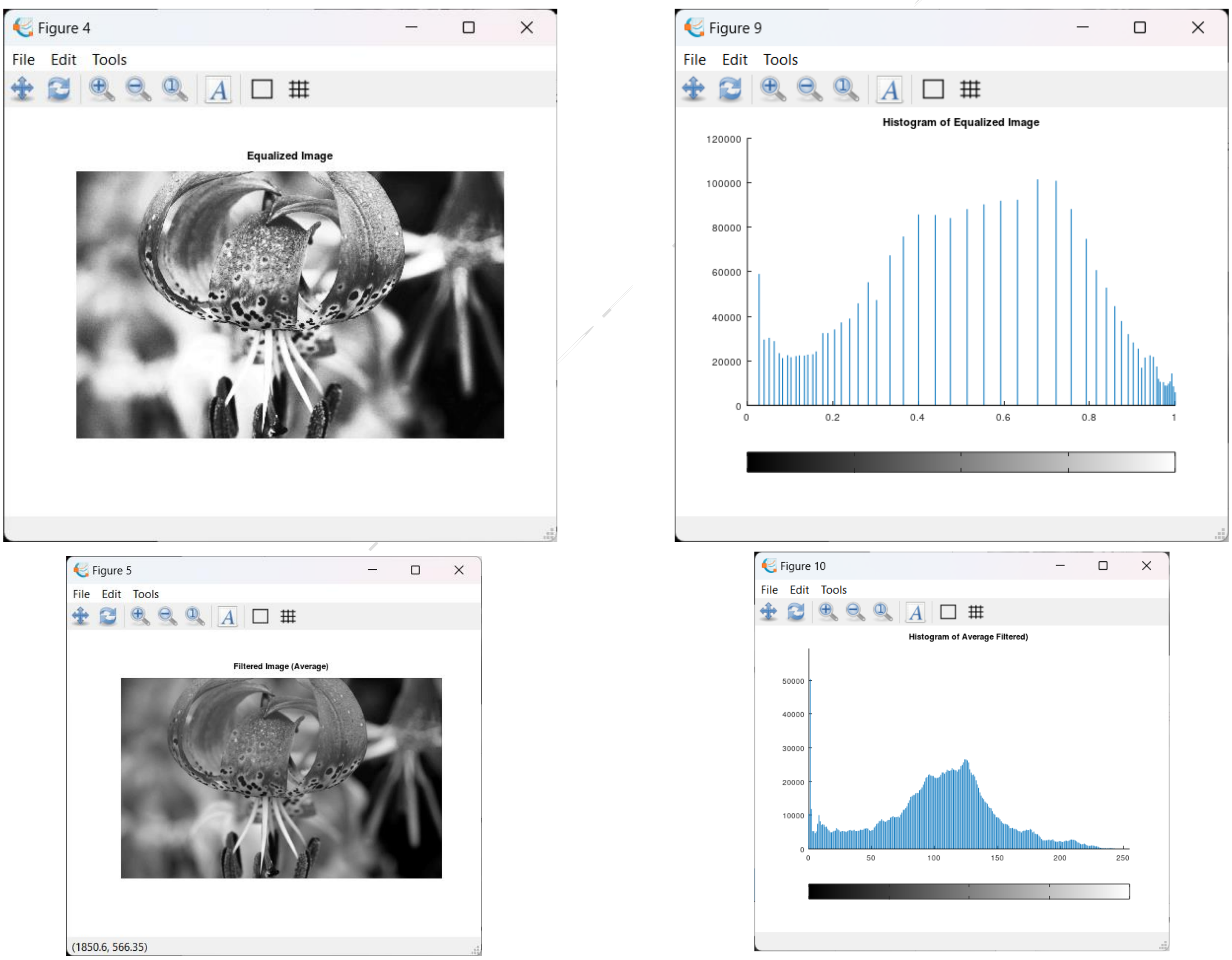


Figure 3B: (OCTAVE) Histogram Equalized and Average Filtered Image and Its Histogram

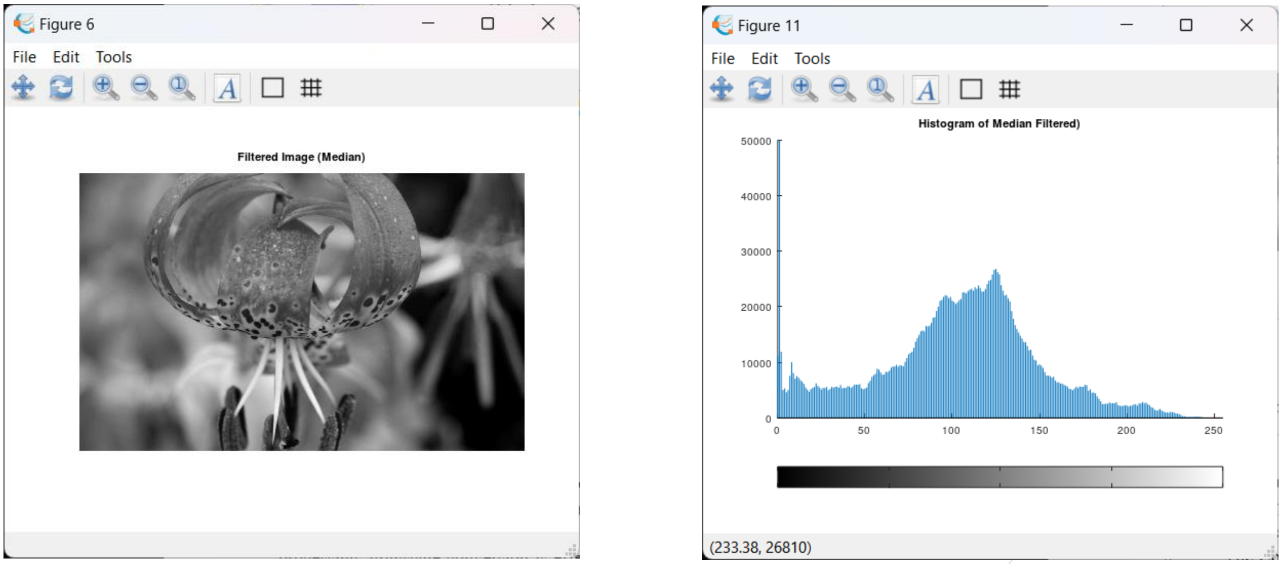


Figure 4B: (OCTAVE) Median Filtered Image and Its Histogram

MATLAB RESULTS

picture file: flower.jpg

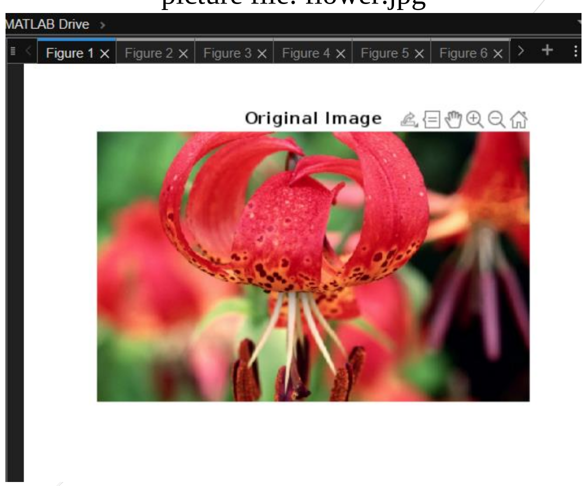
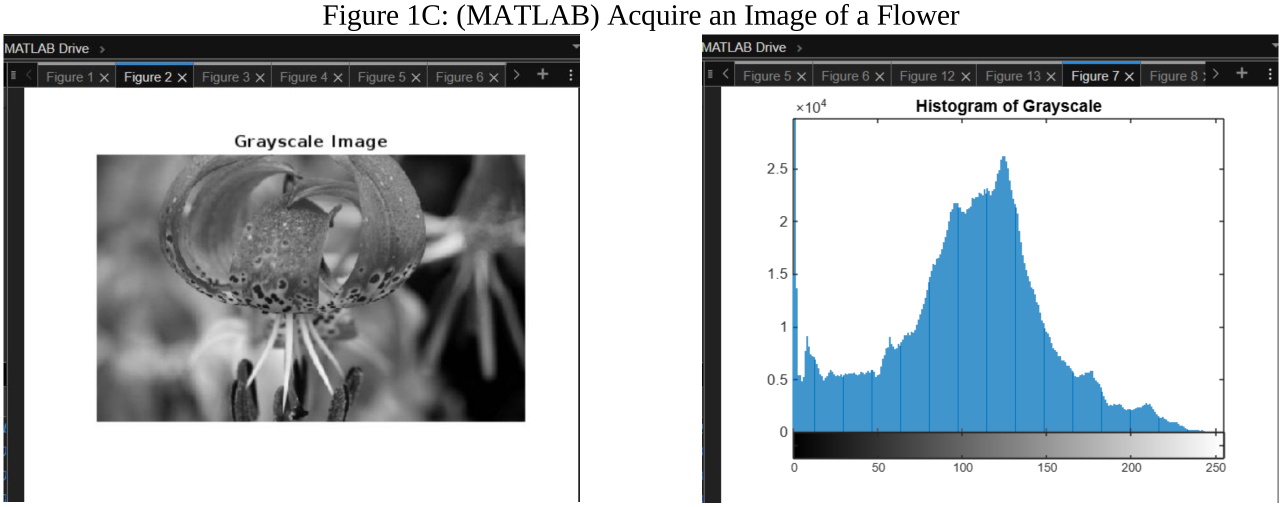


Figure 1C: (MATLAB) Acquire an Image of a Flower



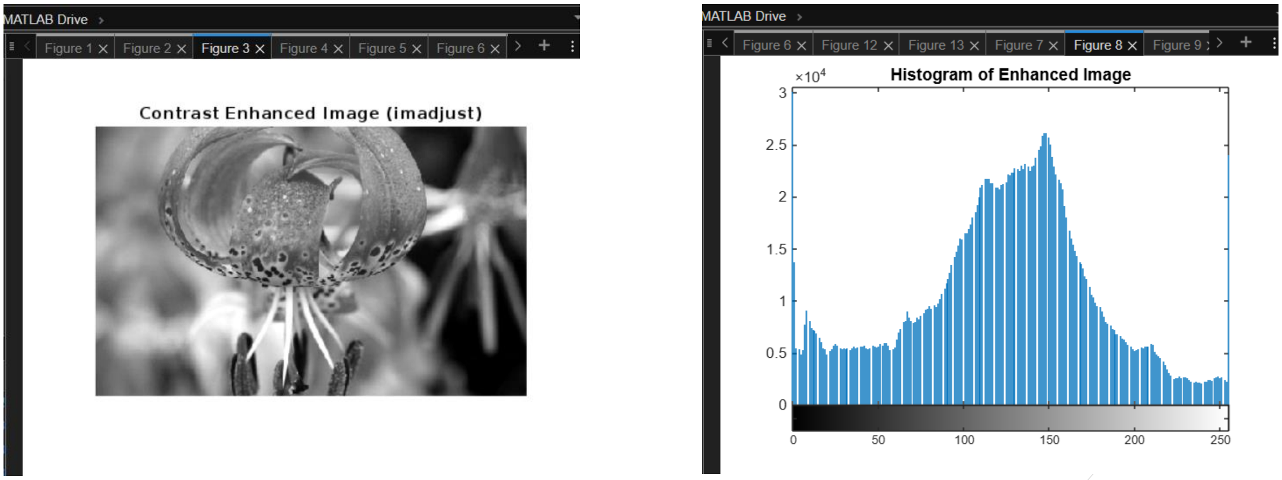


Figure 2C: (MATLAB) Grayscale, Contrast Enhancement, and its Histogram

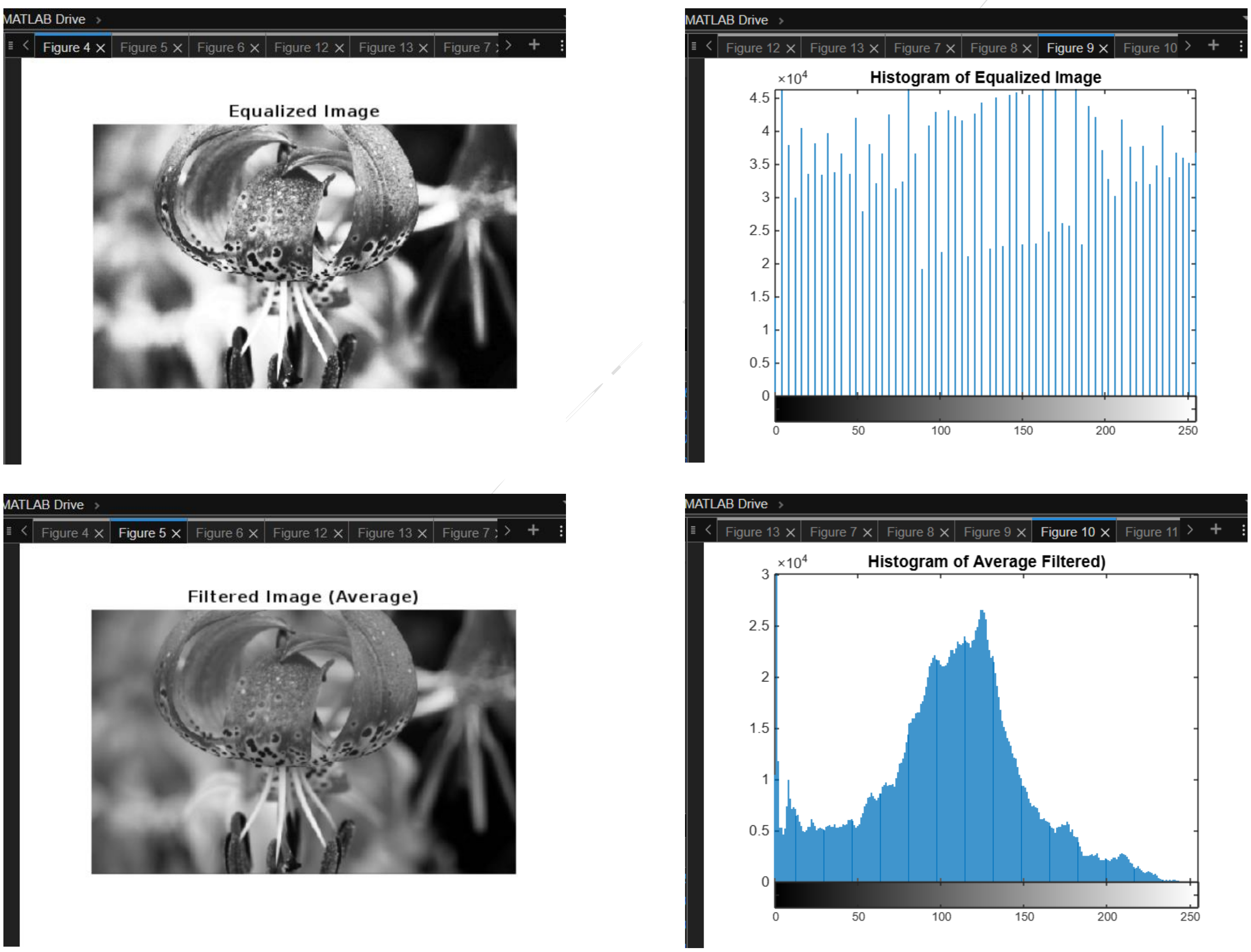
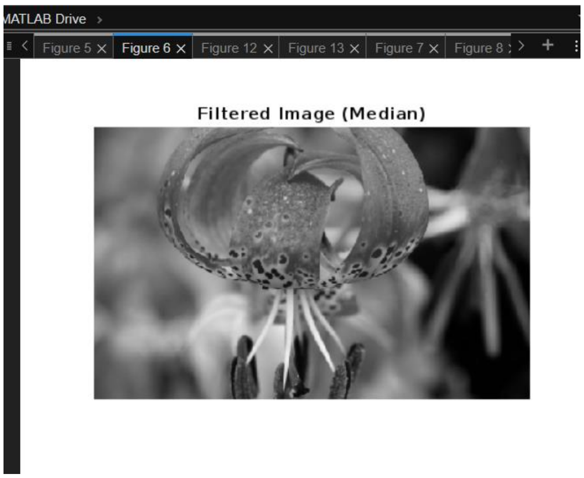


Figure 3C: (MATLAB) Histogram Equalized and Average Filtered Image and Its Histogram



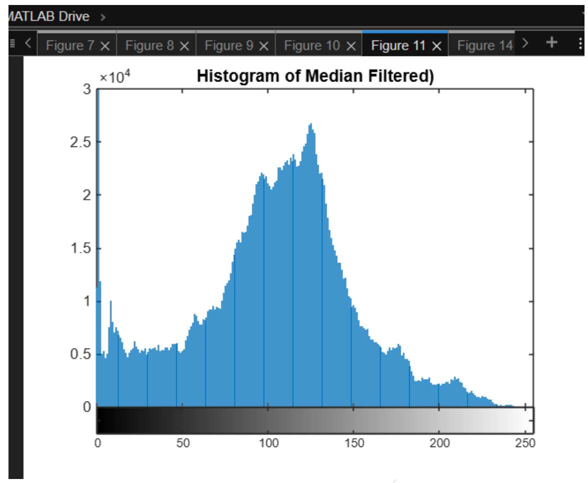


Figure 4C: (MATLAB) Median Filtered Image and Its Histogram

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)');
```

PYTHON RESULTS

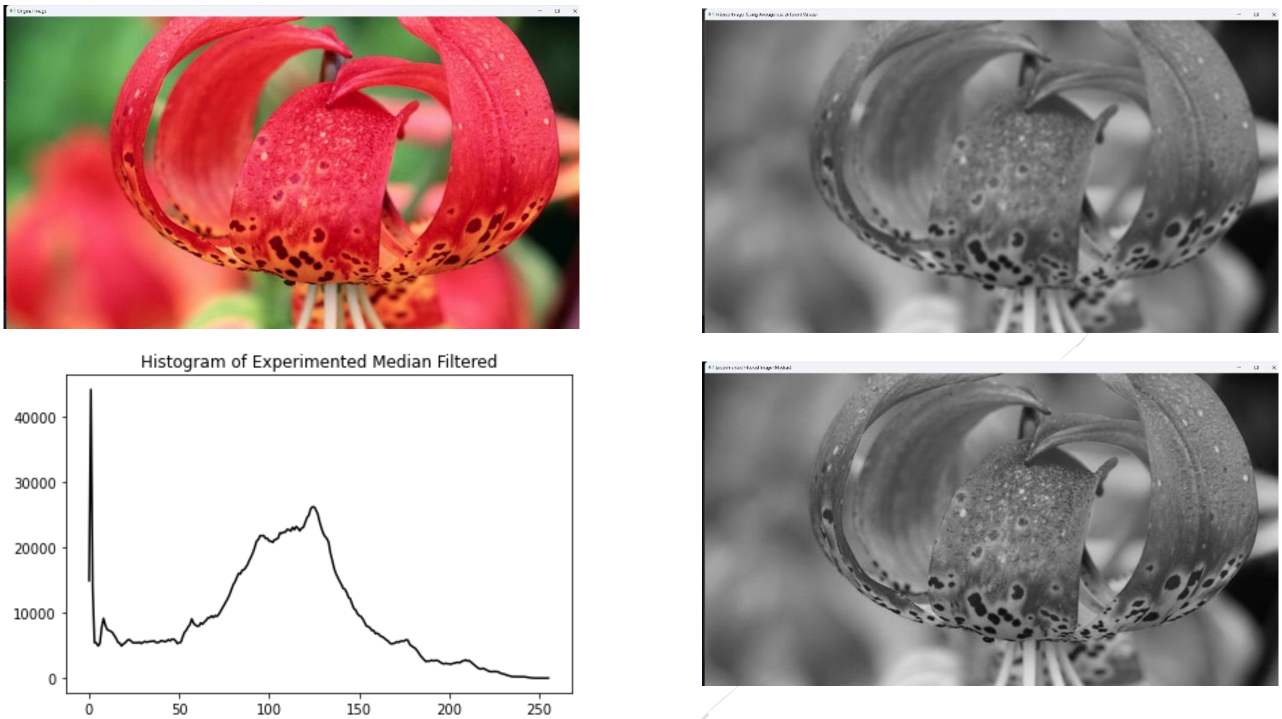


Figure 5A: (PYTHON) Parameters Modification and Its Histogram

OCTAVE RESULTS

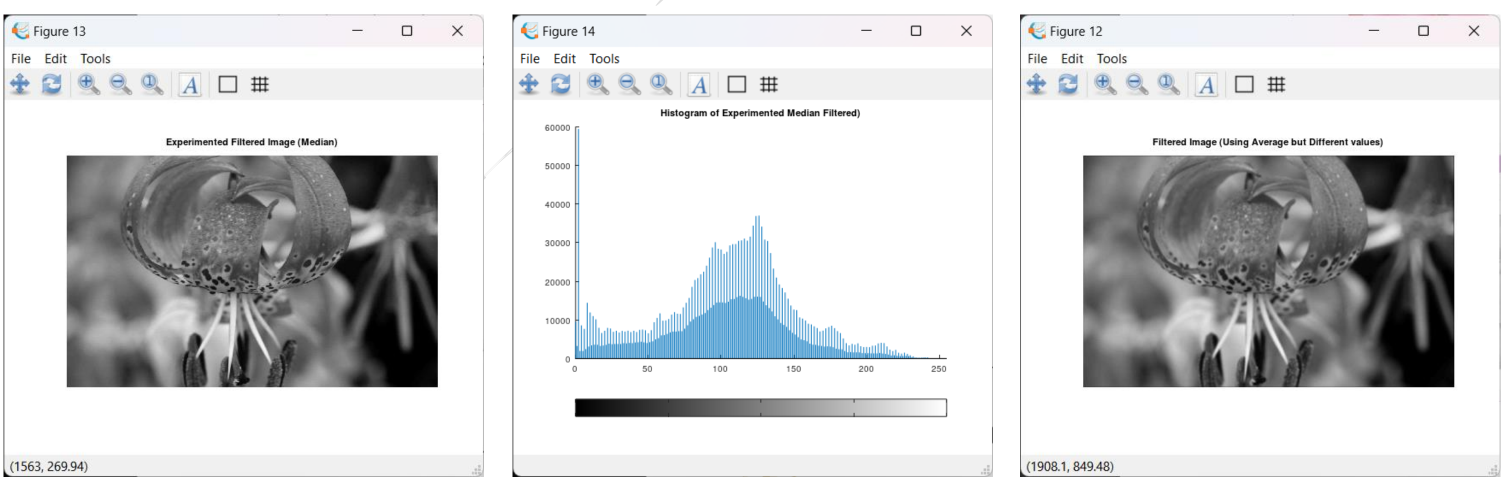


Figure 5B: (OCTAVE) Parameters Modification and Its Histogram

MATLAB RESULTS

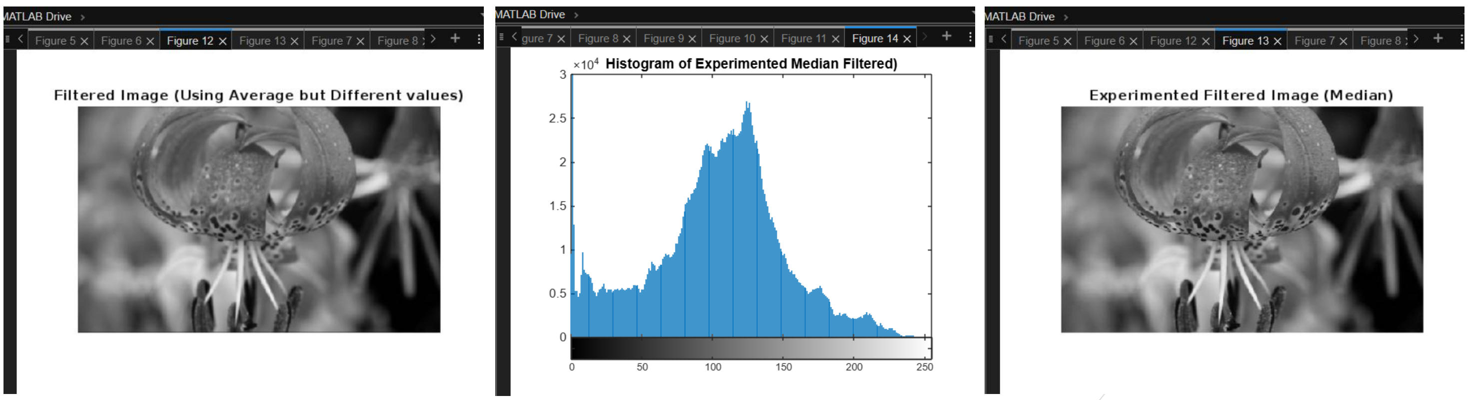


Figure 5C: (MATLAB) Parameters Modification and Its Histogram

1. Visualize the results, analyze and interpret:

Grayscale conversion simplifies the image and reduces computational complexity, preparing the image for further processing steps such as histogram equalization and filtering. Contrast enhancement improves the visibility of details in the image, with the ***convertScaleAbs*** method effectively scaling pixel values to enhance contrast, making features more distinguishable.

Histogram equalization further enhances the image by equalizing the histogram of the grayscale image, resulting in better contrast and spreading out the most frequent intensity values to enhance the global contrast. In the spatial domain, average filtering smooths the image by averaging the pixel values within a specified kernel size, effectively reducing noise but also blurring edges. Median filtering, on the other hand, is more effective in reducing noise while preserving edges compared to averaging, as it replaces each pixel value with the median of the neighboring pixel values. Additional filtering experiments, involving different kernel sizes and values, help in understanding the impact of filter parameters on image quality and noise reduction.

The histograms visualize and interpret the distribution of pixel intensities before and after applying the enhancement techniques. The original image's histogram, often concentrated in certain regions, indicates low contrast. In contrast, the histograms of the contrast-enhanced and histogram-equalized images show a more spread-out distribution, indicating improved contrast. The histograms of the filtered images demonstrate how noise reduction affects pixel intensity distribution.

IV. Conclusion

This laboratory activity effectively demonstrates the principles and techniques of image enhancement using Python and OpenCV. The grayscale conversion simplifies the image and prepares it for further processing steps such as histogram equalization and filtering. Contrast enhancement and histogram equalization significantly improve the visibility and contrast of image details, as evidenced by the spread-out histograms. These techniques redistribute the intensity values across the image, making subtle features more prominent and the overall image more visually appealing.

Spatial domain filtering, including average and median filtering, plays a crucial role in noise reduction. Average filtering smooths the image by averaging the pixel values within a specified kernel size, reducing noise but also resulting in a loss of edge sharpness. Median filtering, however, is particularly effective in preserving edges while reducing noise, as it replaces each pixel value with the median of the neighboring pixel values. This makes median filtering superior in scenarios where edge preservation is critical.

The additional filtering experiments, involving different kernel sizes and values, provide valuable insights into how varying parameters can impact image quality and noise reduction. By experimenting with these parameters, we can fine-tune the filtering process to achieve the desired level of noise reduction and detail preservation.

The group conducted a comprehensive analysis of the image enhancement techniques, using a combination of grayscale conversion, contrast enhancement, histogram equalization, and spatial domain filtering. The histograms created for each stage of the process allowed the group to visualize the distribution of pixel intensities and assess the effectiveness of each technique. The group's systematic approach to experimenting with different filtering parameters further enhanced their understanding of the impact of these techniques on image quality.

Overall, the applied algorithms achieved the desired outcomes of image enhancement, demonstrating their practical effectiveness for various applications in image processing. The comprehensive analysis and visualization through histograms underscored the effectiveness of each technique, providing a clear understanding of their impact on the image.

References

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

OpenCV Documentation. OpenCV: Open Source Computer Vision Library. Available at: https://docs.opencv.org/

GeeksforGeeks. (2024, July 30). OpenCV tutorial in Python. https://www.geeksforgeeks.org/opencv-python-tutorial/