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

Elective 3

Laboratory Activity No. 4
Image Restoration

Score

Submitted by:
Abarientos, Ramuel R.
Cazon, Iyhana Nicole A.
Reyes, John Alfred J.
Verzosa, Cristina Andrea B.
Vicente, Honesto E.

Saturday – 7:00 am – 4:00pm / CPE 0332.1-1

Date Submitted
10-08-2024

Submitted to:

Engr. Maria Rizette H. Sayo

I. Objectives

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

1. Acquire the image.
2. Show Gaussian filter for Image Restoration.
3. Show Deblurring (motion blur removal).

II. Methods

A. Perform a task given in the presentation

- Copy and paste your MATLAB code (use the original picture file: flower.jpg)

```
% Read the image
img = imread('original image'); % Replace with the path to your image file

% 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');

% Add blur to the image
len = 21;
theta = 11;
psf = fspecial('motion', len, theta);
img_blur = imfilter(img_gray, psf, 'conv', 'circular');

% Show the image
figure;
imshow(img_blur);
title('Motion Blurred Image');

% Filtering Techniques

% Gaussian filtering
h_gaussian = fspecial('gaussian', [5, 5], 1);
```

```
img_gaussian_filtered = imfilter(img_blur, h_gaussian);

% Display the Gaussian filtered
image figure;
imshow(img_gaussian_filtered);
title('Filtered Image (Gaussian)');

% Sharpening using unsharp masking
img_sharpened = imsharpen(img_blur);

% Display the sharpened
image figure;
imshow(img_sharpened);
title('Sharpened Image');

% Add Gaussian noise and remove it using median filter
img_noisy = imnoise(img_gray, 'gaussian', 0.02);
img_noisy_removed = medfilt2(img_noisy, [5, ]);

% Display the noise image
figure;
imshow(img_noisy);
title('Noisy');

% Display the noise-removed
images figure;
imshow(img_noisy_removed);
title('Noise Removed');

% Deblurring
estimated_nsr = 0.01;
img_deblurred = deconvwnr(img_blur, psf, estimated_nsr);
figure;
imshow(img_deblurred);
title("Deblurred Image");
```

Lab4.m

```
% Read the image
img = imread('C:\Users\user\Downloads\Lab4\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');

% Add blur to the image
len = 21;
theta = 11;
psf = fspecial('motion', len, theta);
img_blur = imfilter(img_gray, psf, 'conv', 'circular');

% Show the blurred image
figure;
imshow(img_blur);
title('Motion Blurred Image');

% Gaussian filtering
h_gaussian = fspecial('gaussian', [5, 5], 1);
img_gaussian_filtered = imfilter(img_blur, h_gaussian);

% Display the Gaussian filtered image
figure;
```

Lab4.m

```
% Display the Gaussian filtered image
figure;
imshow(img_gaussian_filtered);
title('Filtered Image (Gaussian)');

% Sharpening using unsharp masking
img_sharpened = imsharpen(img_blur);

% Display the sharpened image
figure;
imshow(img_sharpened);
title('Sharpened Image');

% Add Gaussian noise and remove it using median filter
img_noisy = imnoise(img_gray, 'gaussian', 0.02);
img_noisy_removed = medfilt2(img_noisy, [5, 5]);

% Display the noisy image
figure;
imshow(img_noisy);
title('Noisy Image');

% Display the noise-removed image
figure;
imshow(img_noisy_removed);
title('Noise Removed Image');

% Deblurring
estimated_nsr = 0.01;
img_deblurred = deconvwnr(img_blur, psf, estimated_nsr);

% Display the deblurred image
figure;
imshow(img_deblurred);
title('Deblurred Image');
```

MATLAB

lab_act_3_modif.m | labact3_octave.m | lab4_octave.m

```
pkg load image

% Read the image
img = imread('C:\Users\user\Downloads\Lab4\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');

% Add blur to the image
len = 21;
theta = 11;
psf = fspecial('motion', len, theta);
img_blur = imfilter(img_gray, psf, 'conv', 'circular');

% Show the blurred image
figure;
imshow(img_blur);
title('Motion Blurred Image');

% Gaussian filtering
h_gaussian = fspecial('gaussian', [5, 5], 1);
img_gaussian_filtered = imfilter(img_blur, h_gaussian);

% Display the Gaussian filtered image
figure;
imshow(img_gaussian_filtered);
title('Filtered Image (Gaussian)');

% Sharpening using unsharp masking
img_sharpened = imsharpen(img_blur);

% Display the sharpened image
figure;
imshow(img_sharpened);
title('Sharpened Image');

% Add Gaussian noise and remove it using median filter
```

```
% Add Gaussian noise and remove it using median filter
img_noisy = imnoise(img_gray, 'gaussian', 0.02);
img_noisy_removed = medfilt2(img_noisy, [5, 5]);

% Display the noisy image
figure;
imshow(img_noisy);
title('Noisy Image');

% Display the noise-removed image
figure;
imshow(img_noisy_removed);
title('Noise Removed Image');

% Deblurring
estimated_nsr = 0.01;
img_deblurred = deconvwnr(img_blur, psf, estimated_nsr);

% Display the deblurred image
figure;
imshow(img_deblurred);
title('Deblurred Image');
```

OCTAVE

B. Supplementary Activity

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import restoration
from scipy.ndimage import gaussian_filter, median_filter

# Read the image
img = cv2.imread('flower.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Display the original image
plt.figure()
```

```
plt.imshow(img_rgb)
plt.title('Original Image')
plt.axis('off')
plt.show()

# 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.title('Grayscale')
plt.axis('off')
plt.show()

# Add blur to the image
len = 21
theta = 11
psf = np.zeros((len, len))
psf[len//2, :] = 1
psf = cv2.warpAffine(psf, cv2.getRotationMatrix2D((len/2, len/2), theta, 1.0),
(len, len))
psf = psf / psf.sum()
img_blur = cv2.filter2D(img_gray, -1, psf)

# Show the blurred image
plt.figure()
plt.imshow(img_blur, cmap='gray')
plt.title('Motion Blurred Image')
plt.axis('off')
plt.show()

# Gaussian filtering
img_gaussian_filtered = gaussian_filter(img_blur, sigma=1)

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

# Sharpening using unsharp masking
img_sharpened = cv2.addWeighted(img_blur, 1.5, cv2.GaussianBlur(img_blur, (0, 0),
1), -0.5, 0)

# Display the sharpened image
```

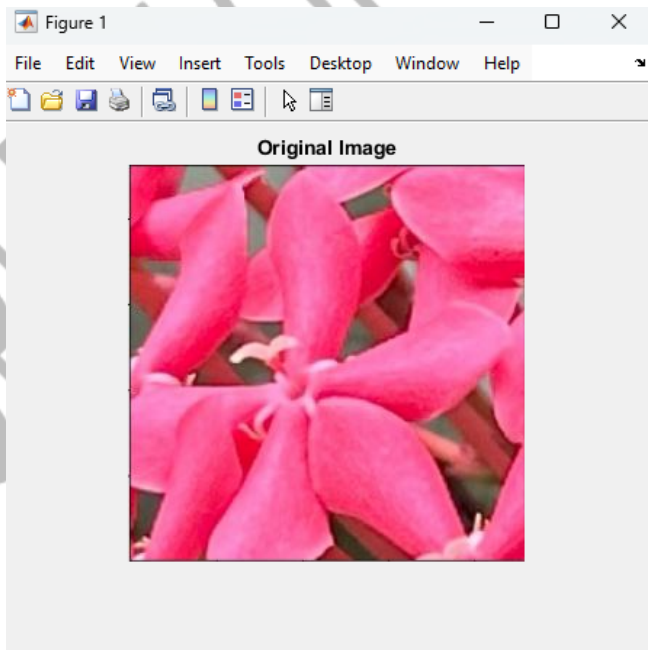
```
plt.figure()
plt.imshow(img_sharpened, cmap='gray')
plt.title('Sharpened Image')
plt.axis('off')
plt.show()

# Add Gaussian noise and remove it using median filter
img_noisy = img_gray + np.random.normal(0, 25, img_gray.shape)
img_noisy = np.clip(img_noisy, 0, 255).astype(np.uint8)
img_noisy_removed = median_filter(img_noisy, size=5)

# Display the noisy image
plt.figure()
plt.imshow(img_noisy, cmap='gray')
plt.title('Noisy')
plt.axis('off')
plt.show()

# Display the noise-removed image
plt.figure()
plt.imshow(img_noisy_removed, cmap='gray')
plt.title('Noise Removed')
plt.axis('off')
plt.show()

# Deblurring
estimated_nsr = 0.01
img_deblurred = restoration.wiener(img_blur, psf, estimated_nsr)

# Display the deblurred image
plt.figure()
plt.imshow(img_deblurred, cmap='gray')
plt.title('Deblurred Image')
plt.axis('off')
plt.show()
```

III. Results

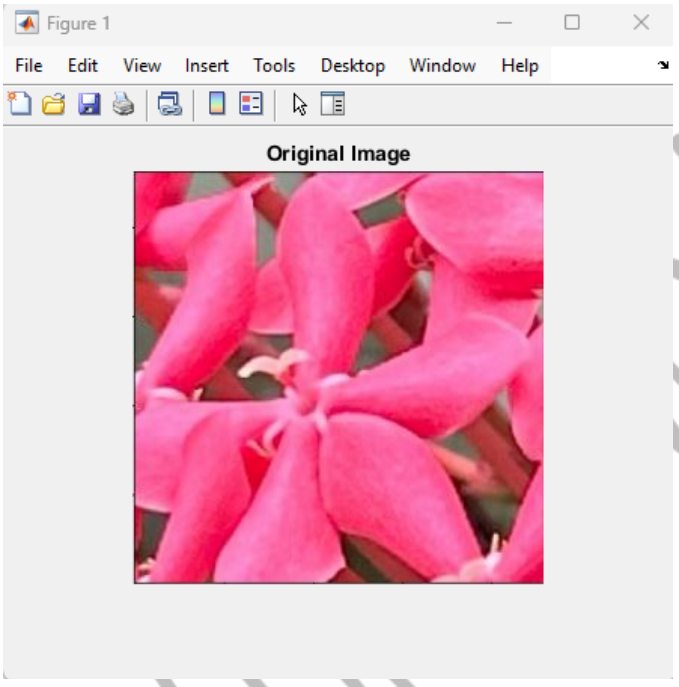
Steps:

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

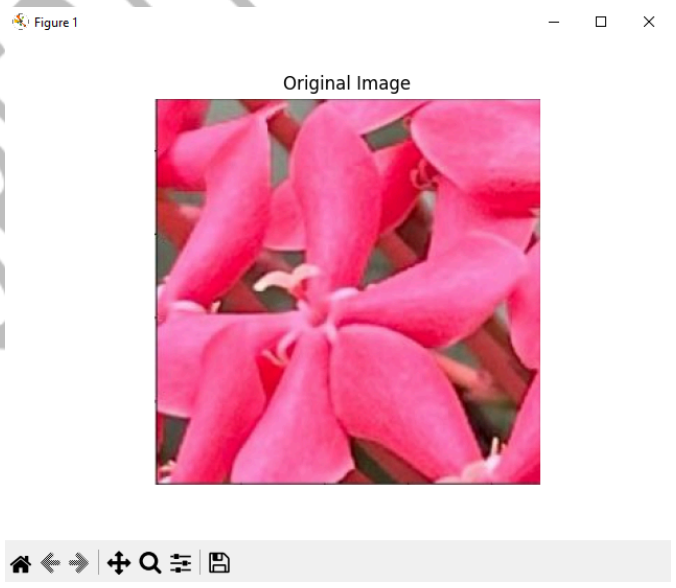
picture file: flower.jpg



MATLAB

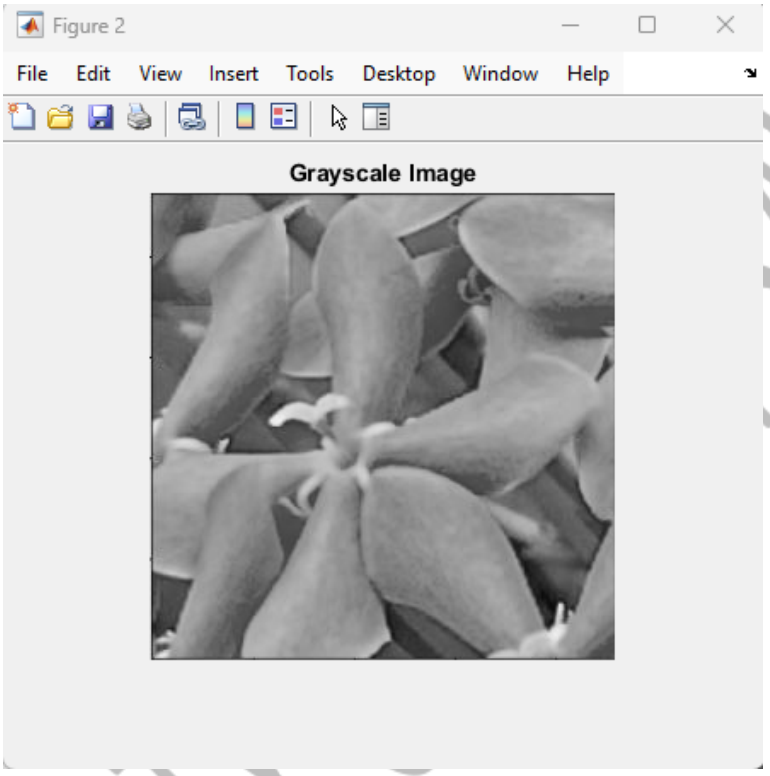


OCTAVE

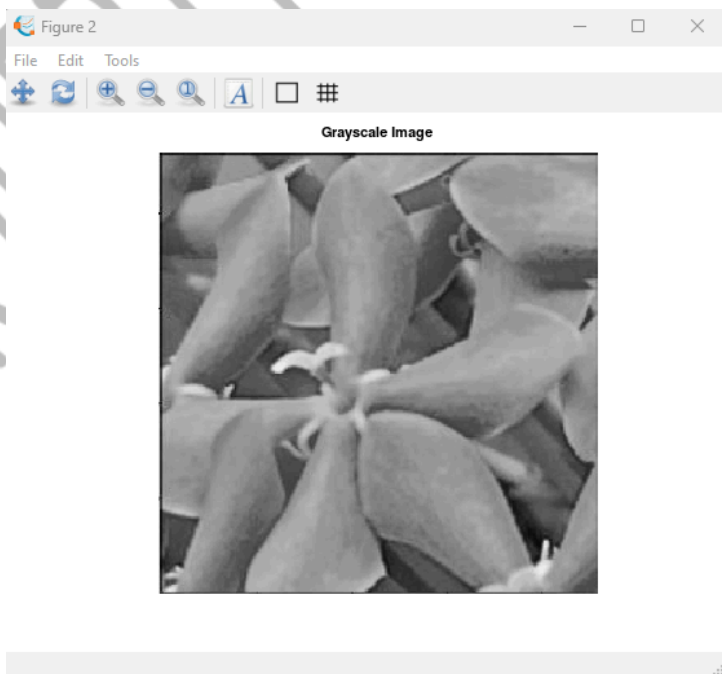


PYTHON

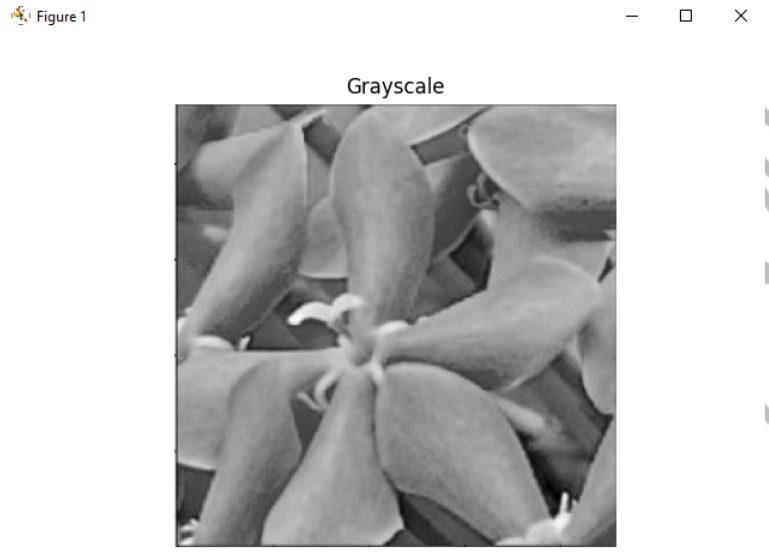
Figure 1:Original Image



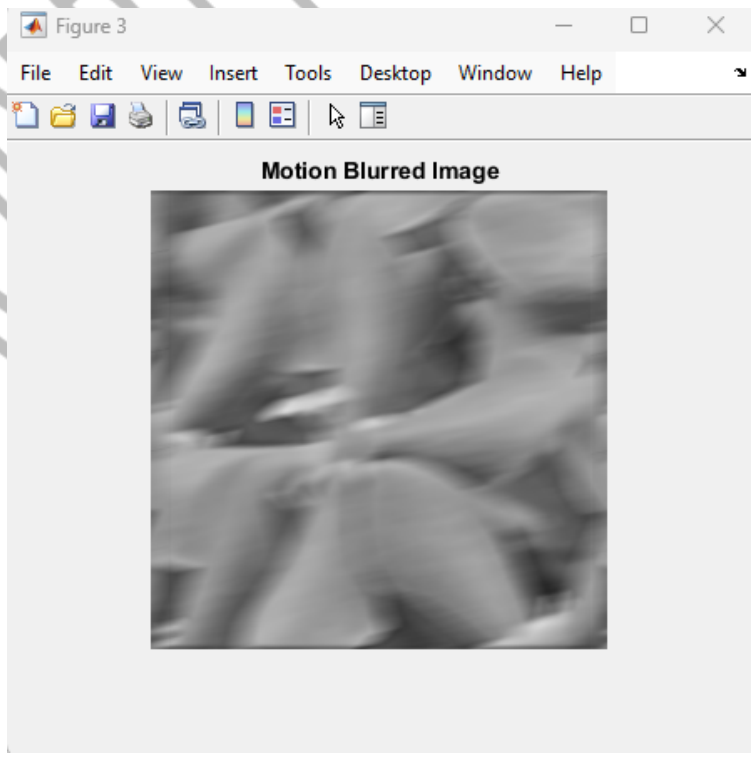
MATLAB



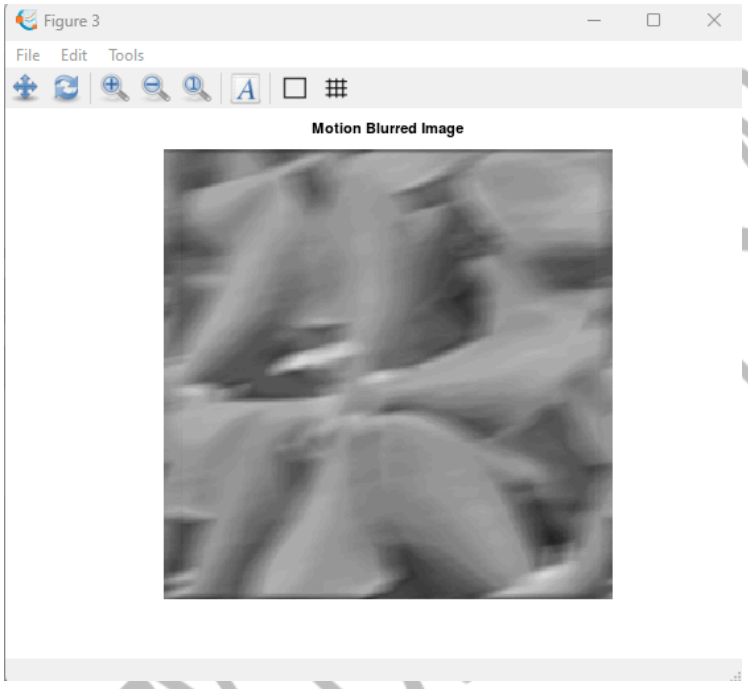
OCTAVE



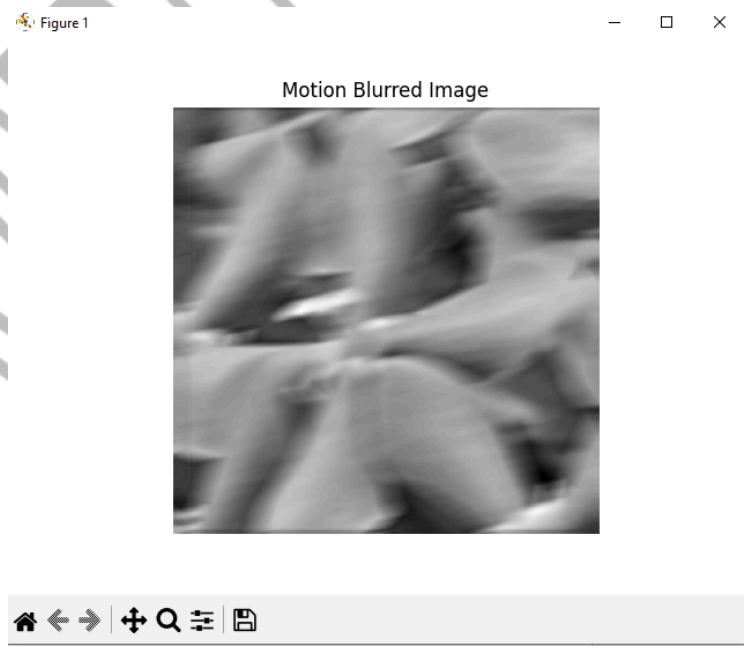
PYTHON
Figure 2: Grayscale



MATLAB

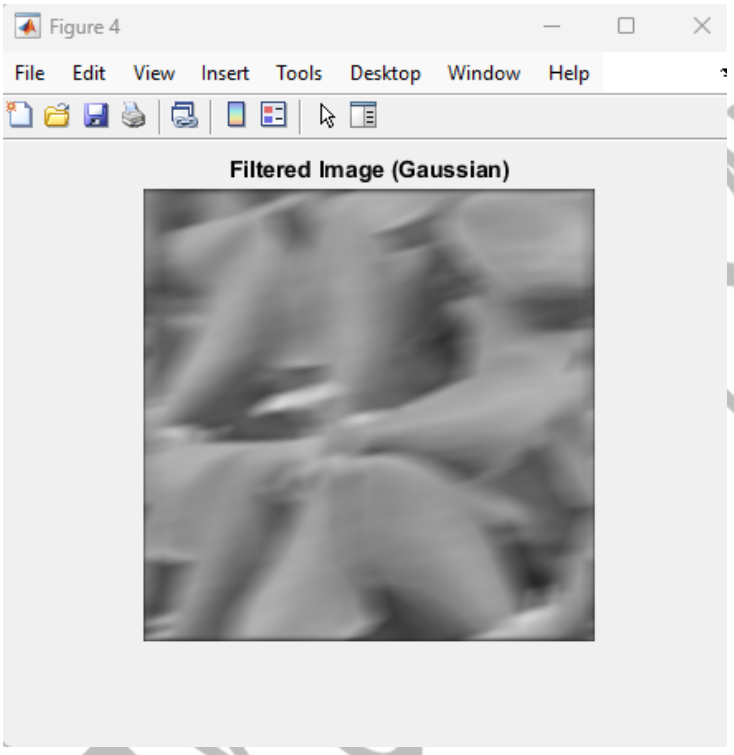


OCTAVE

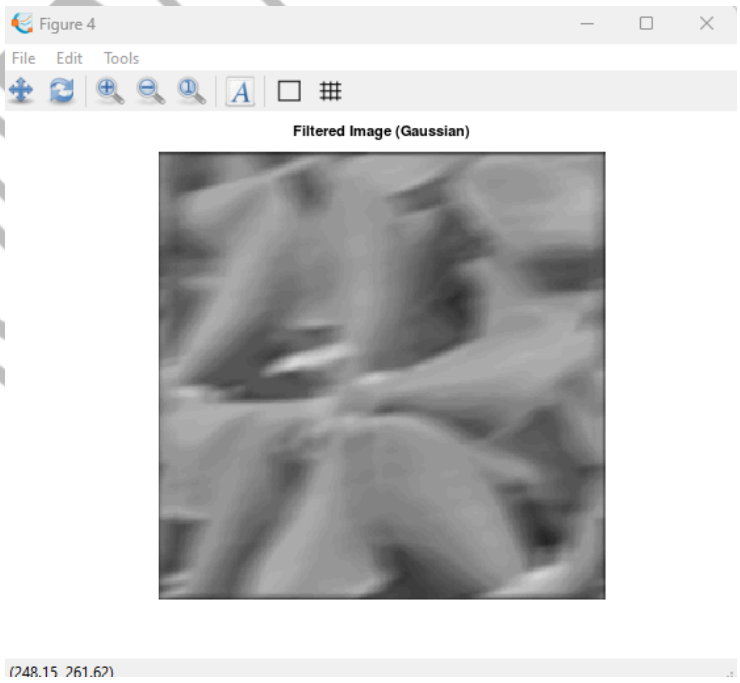


PYTHON

Figure 3: Motion Blurred Image



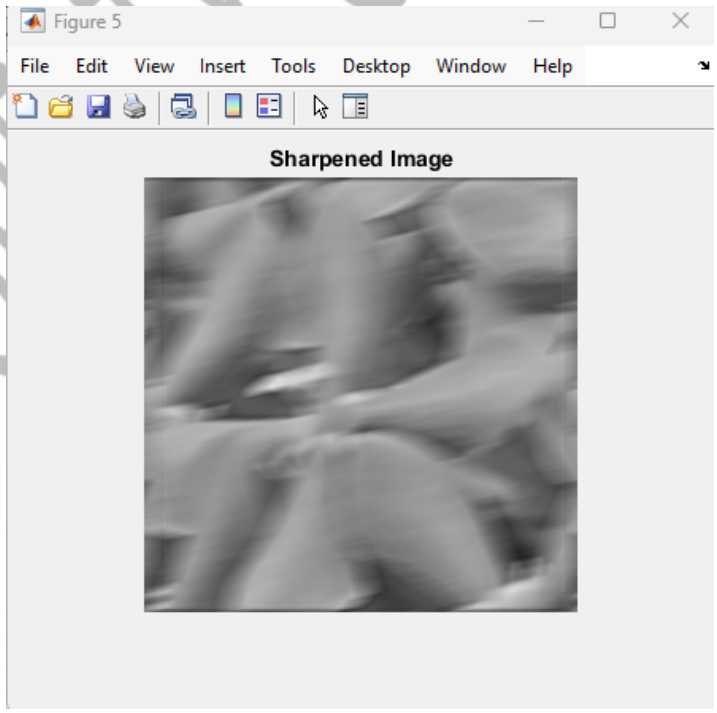
MATLAB



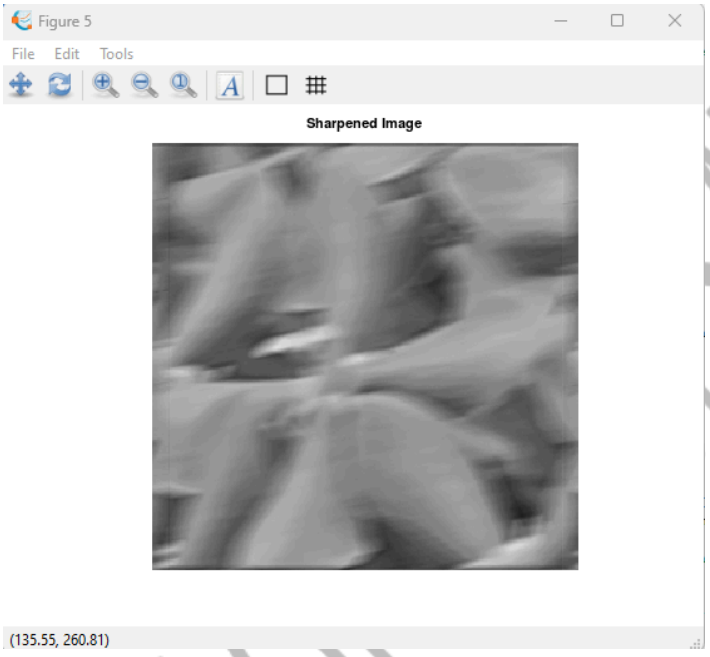
OCTAVE

PYTHON

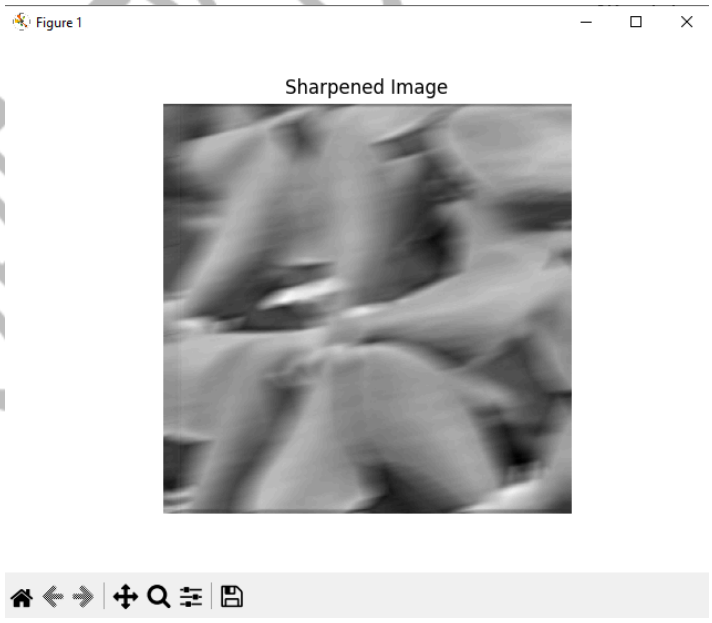
Figure 4: Gaussian-filtered Image



MATLAB

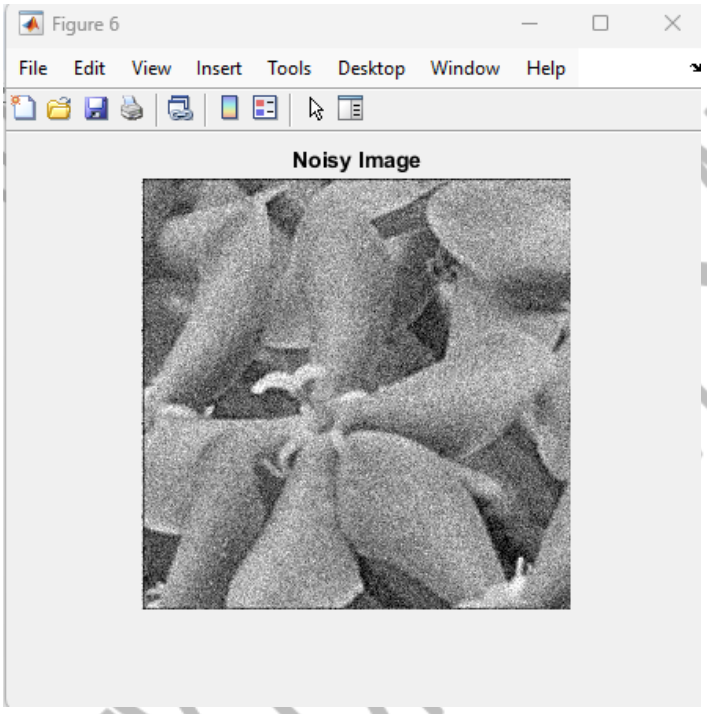


OCTAVE

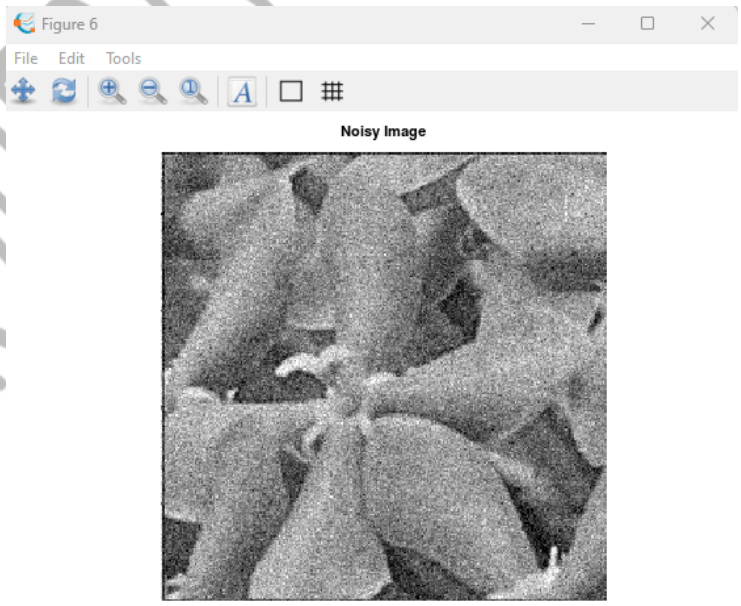


PYTHON

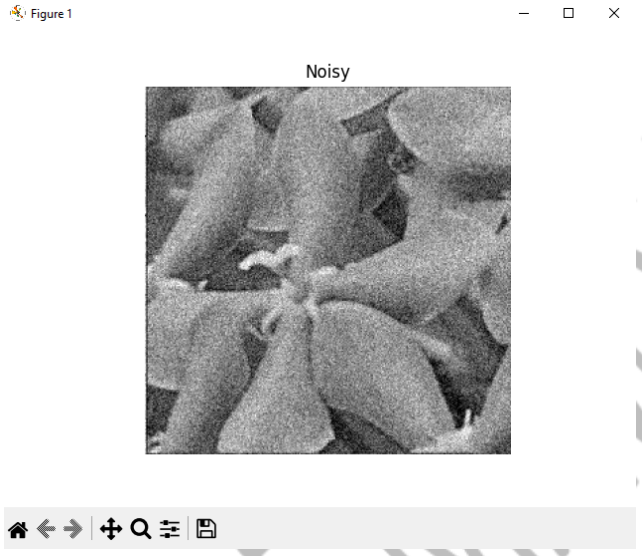
Figure 5: Sharpen Image



MATLAB

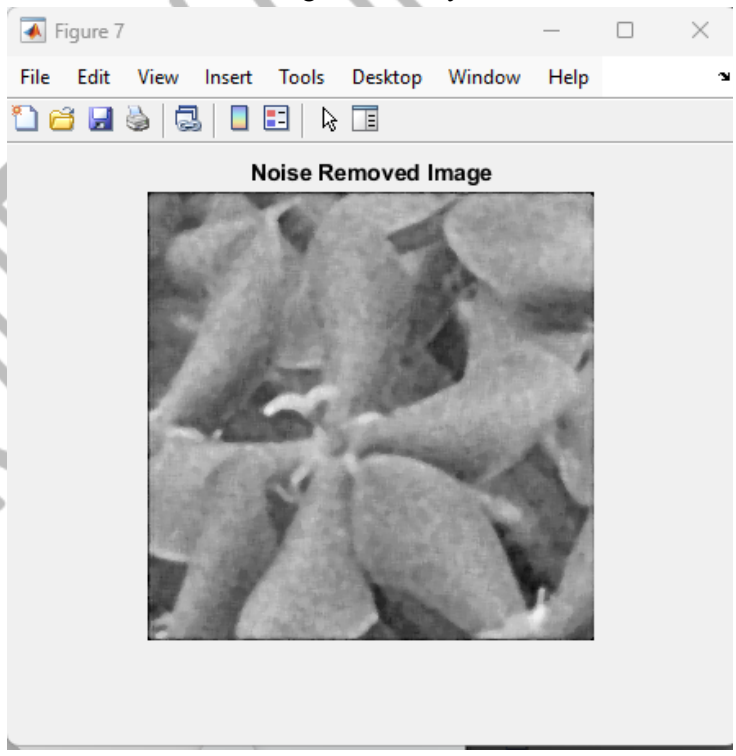


OCTAVE

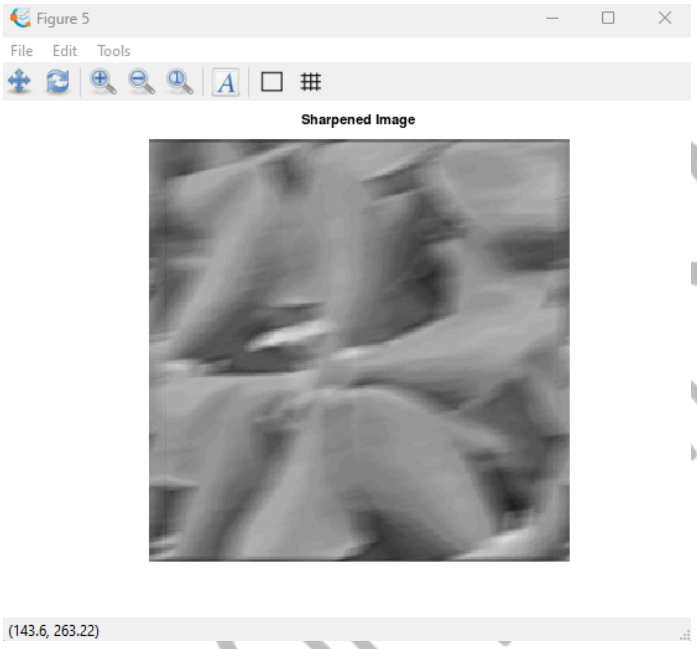


PYTHON

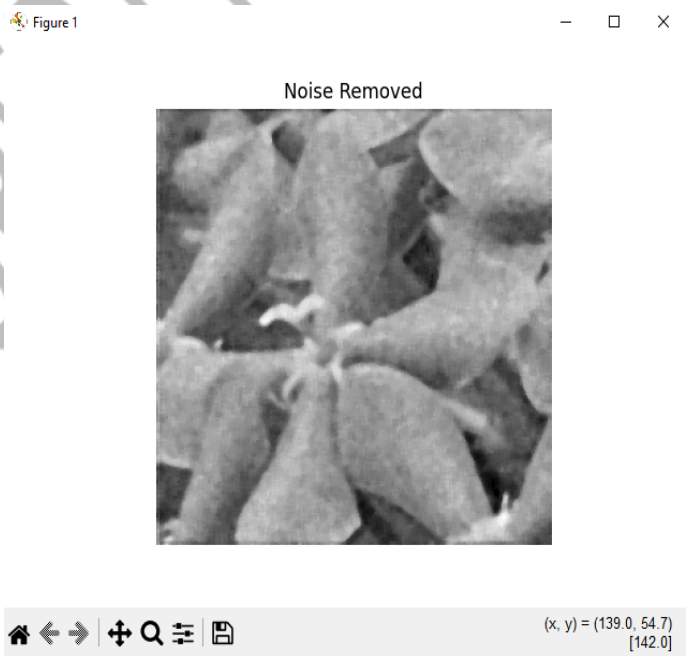
Figure 6: Noisy



MATLAB

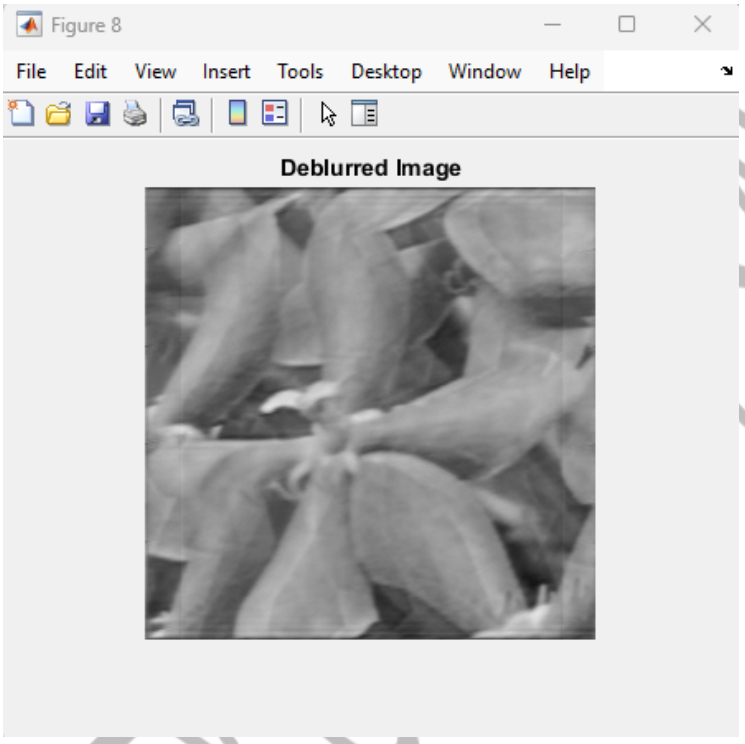


OCTAVE

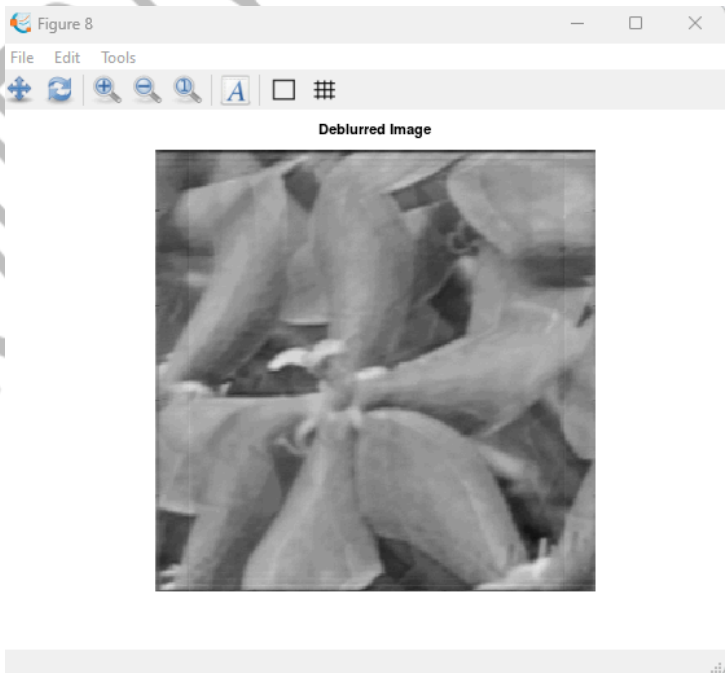


PYTHON

Figure 7: Noise Remove



MATLAB



OCTAVE

PYTHON

Figure 8: Deblurred Image

These codes perform the following:

- Grayscale Conversion: The code first converts the image to grayscale if it's colored (RGB format). This simplifies the image by removing color information, making it easier for subsequent algorithms to process.
- Motion Blur: A motion blur filter is applied, simulating the effect of camera movement during image capture. This can blur sharp edges and details in the original image.
- Gaussian Filtering: A Gaussian filter is used to smooth out the image further. This reduces noise introduced by the motion blur but can also blur sharp details remaining from the original image.
- Sharpening: Unsharp masking is applied to enhance edges in the image. This counteracts the blurring effect but might introduce some artificial sharpening artifacts.
- Noise Addition and Removal: Gaussian noise is artificially added to the grayscale image, simulating imperfections that might occur during image capture. A median filter is then used to remove this noise. Median filters effectively remove impulsive noise but can slightly blur sharp edges.
- Deblurring: Finally, an attempt is made to reverse the motion blur using deconvolution. This process aims to recover the original sharp image, but its effectiveness depends on the accuracy of the estimated blur parameters and the amount of noise present.

Parameter Modification

<You can modify it to explore other functionalities>

```
% Gaussian filtering
h_gaussian = fspecial('gaussian', [5, 5], 10); % Original [5,5], 1
img_gaussian_filtered = imfilter(img_gray, h_gaussian);

% Display the Gaussian filtered
image figure;
imshow(img_gaussian_filtered);
title('Filtered Image with Experimented Value (Gaussian)');

% Histogram (Gaussian Filtered)
figure;
imhist(img_gaussian_filtered);
title('Histogram of the Experimented Value (Gaussian Filtered)');


% Add Gaussian noise
img_noisy_exp1 = imnoise(img_gray, 'gaussian', 0.5);
img_noisy_exp2 = imnoise(img_gray, 'gaussian', 0.1);

% Display the noisy
figure;
imshow(img_noisy_exp1);
title('Noisy Using Experimented Value (Gaussian is 0.5)');

figure;
imshow(img_noisy_exp2);
title('Noisy Using Experimented Value (Gaussian is 0.1)');

% Display the histogram for Noisy
figure;
imhist(img_noisy_exp1);
title('Histogram of Noisy Image Experimented Value 1');

figure;
imhist(img_noisy_exp2);
title('Histogram of Noisy Image Experimented Value 2');
```

PYTHON:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_filter, median_filter
from skimage import restoration

# Read the image
```

```
img = cv2.imread('flower.png')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Display the original image
plt.figure()
plt.imshow(img_rgb)
plt.title('Original Image')
plt.axis('off')
plt.show()

# 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.title('Grayscale')
plt.axis('off')
plt.show()

# Add blur to the image
len = 21
theta = 11
psf = np.zeros((len, len))
psf[len//2, :] = 1
psf = cv2.warpAffine(psf, cv2.getRotationMatrix2D((len/2, len/2), theta, 1.0),
(len, len))
psf = psf / psf.sum()
img_blur = cv2.filter2D(img_gray, -1, psf)

# Show the blurred image
plt.figure()
plt.imshow(img_blur, cmap='gray')
plt.title('Motion Blurred Image')
plt.axis('off')
plt.show()

# Gaussian filtering with different parameters
h_gaussian = gaussian_filter(img_gray, sigma=10)
img_gaussian_filtered = cv2.filter2D(img_gray, -1, h_gaussian)

# Display the Gaussian filtered image
plt.figure()
plt.imshow(img_gaussian_filtered, cmap='gray')
plt.title('Filtered Image with Experimented Value (Gaussian)')
plt.axis('off')
plt.show()

# Display the histogram of the Gaussian filtered image
```

```
plt.figure()
plt.hist(img_gaussian_filtered.ravel(), bins=256, fc='k', ec='k')
plt.title('Histogram of the Experimented Value (Gaussian Filtered)')
plt.show()

# Add Gaussian noise with different values
img_noisy_exp1 = img_gray + np.random.normal(0, 0.5 * 255,
img_gray.shape).astype(np.uint8)
img_noisy_exp2 = img_gray + np.random.normal(0, 0.1 * 255,
img_gray.shape).astype(np.uint8)
img_noisy_exp1 = np.clip(img_noisy_exp1, 0, 255).astype(np.uint8)
img_noisy_exp2 = np.clip(img_noisy_exp2, 0, 255).astype(np.uint8)

# Display the noisy images
plt.figure()
plt.imshow(img_noisy_exp1, cmap='gray')
plt.title('Noisy Using Experimented Value (Gaussian is 0.5)')
plt.axis('off')
plt.show()

plt.figure()
plt.imshow(img_noisy_exp2, cmap='gray')
plt.title('Noisy Using Experimented Value (Gaussian is 0.1)')
plt.axis('off')
plt.show()

# Display the histograms for the noisy images
plt.figure()
plt.hist(img_noisy_exp1.ravel(), bins=256, fc='k', ec='k')
plt.title('Histogram of Noisy Image Experimented Value 1')
plt.show()

plt.figure()
plt.hist(img_noisy_exp2.ravel(), bins=256, fc='k', ec='k')
plt.title('Histogram of Noisy Image Experimented Value 2')
plt.show()
```

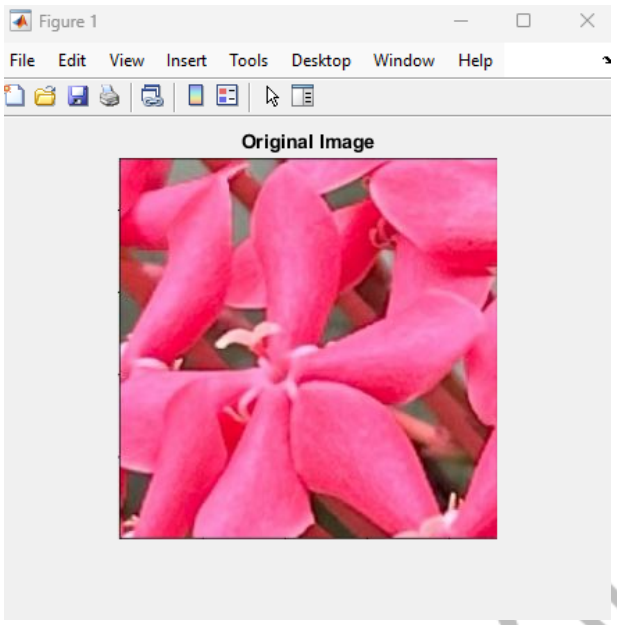


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image

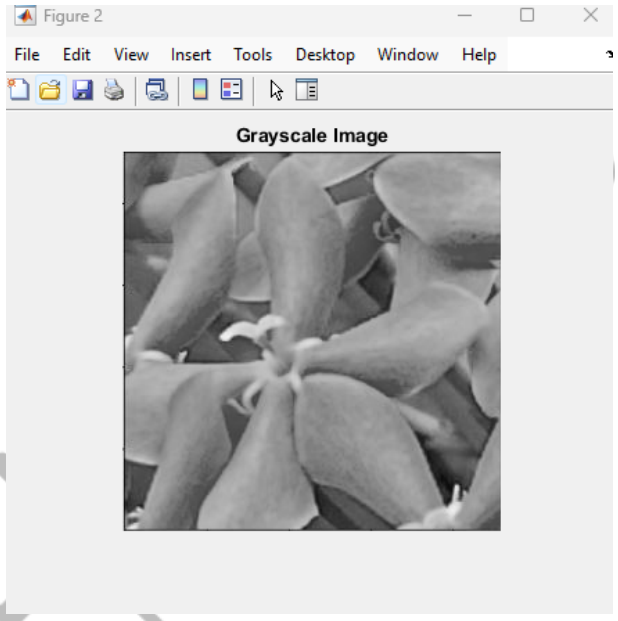


Figure 2
File Edit View Insert Tools Desktop Window Help
Grayscale Image

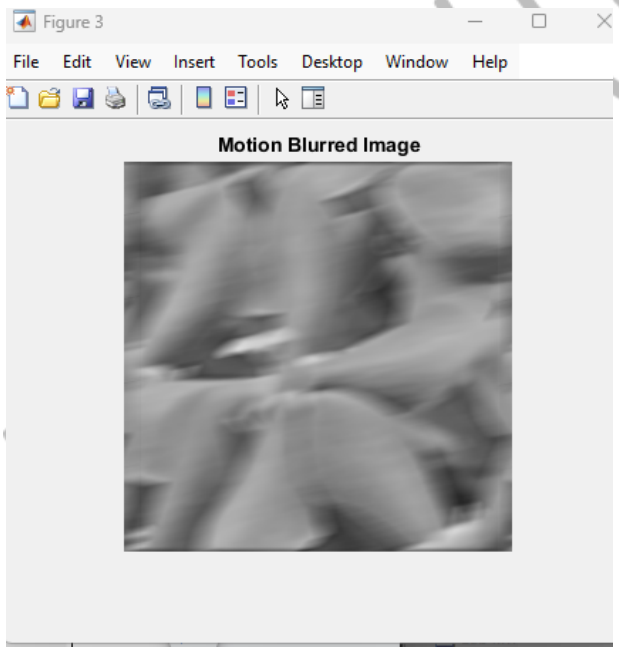


Figure 3
File Edit View Insert Tools Desktop Window Help
Motion Blurred Image

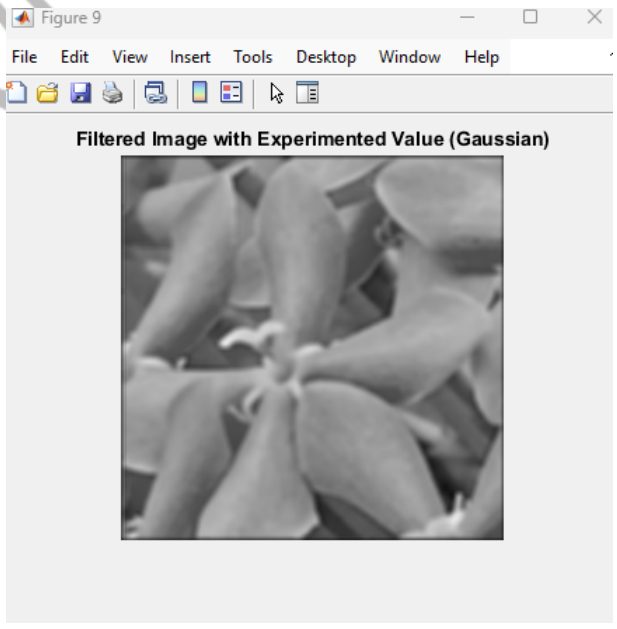


Figure 9
File Edit View Insert Tools Desktop Window Help
Filtered Image with Experimented Value (Gaussian)

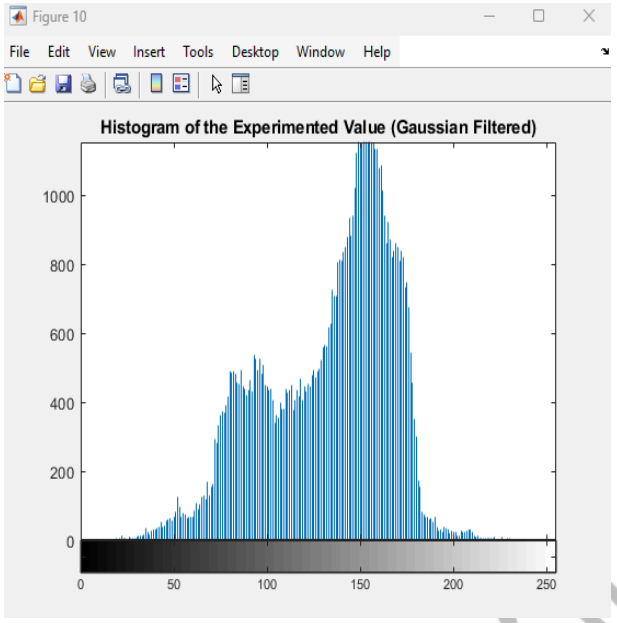


Figure 10
File Edit View Insert Tools Desktop Window Help
Histogram of the Experimented Value (Gaussian Filtered)
1000
800
600
400
200
0
0 50 100 150 200 250

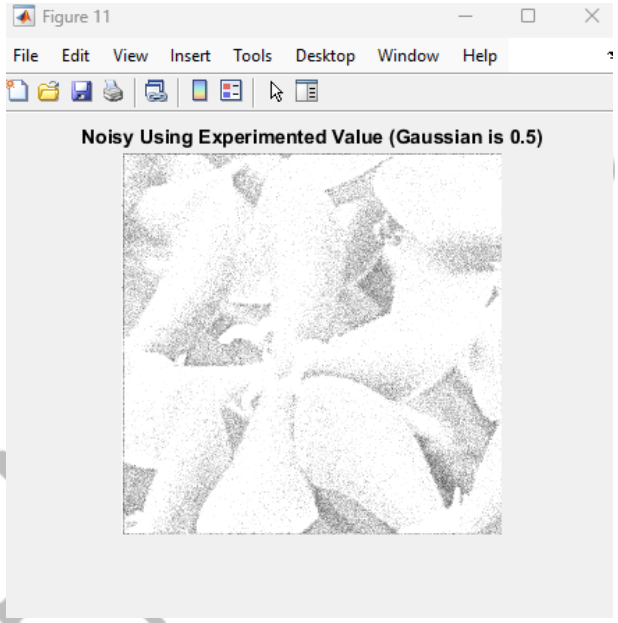


Figure 11
File Edit View Insert Tools Desktop Window Help
Noisy Using Experimented Value (Gaussian is 0.5)

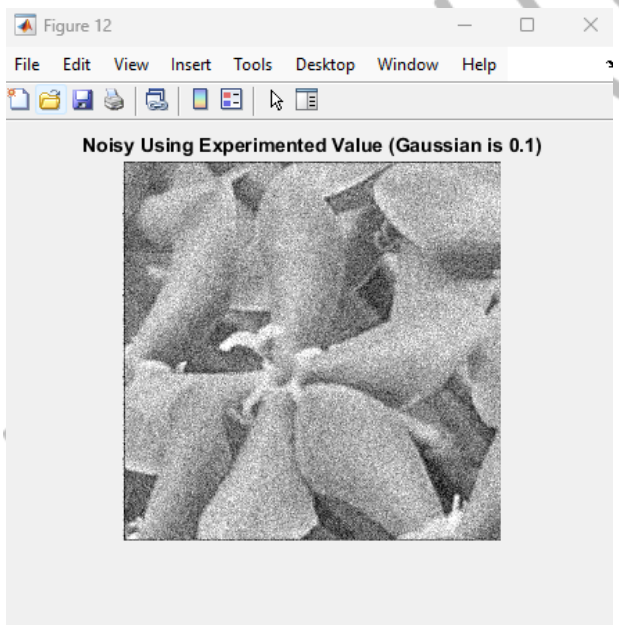


Figure 12
File Edit View Insert Tools Desktop Window Help
Noisy Using Experimented Value (Gaussian is 0.1)

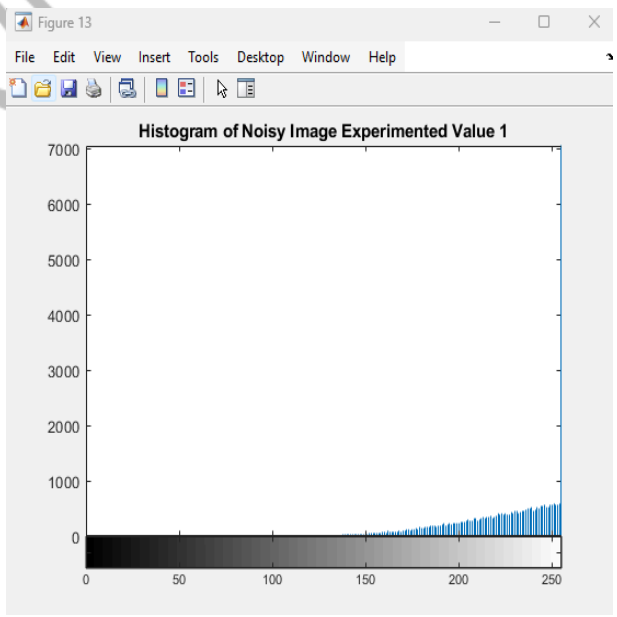


Figure 13
File Edit View Insert Tools Desktop Window Help
Histogram of Noisy Image Experimented Value 1
7000
6000
5000
4000
3000
2000
1000
0
0 50 100 150 200 250

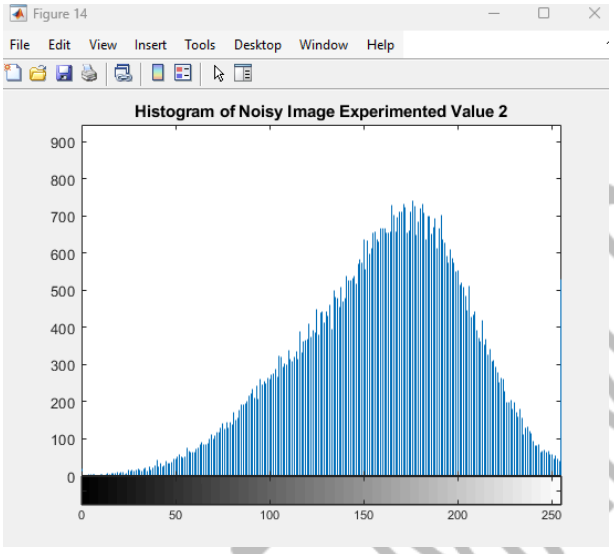


Figure 14
File Edit View Insert Tools Desktop Window Help
Histogram of Noisy Image Experimented Value 2
900
800
700
600
500
400
300
200
100
0
0
50
100
150
200
250

MATLAB

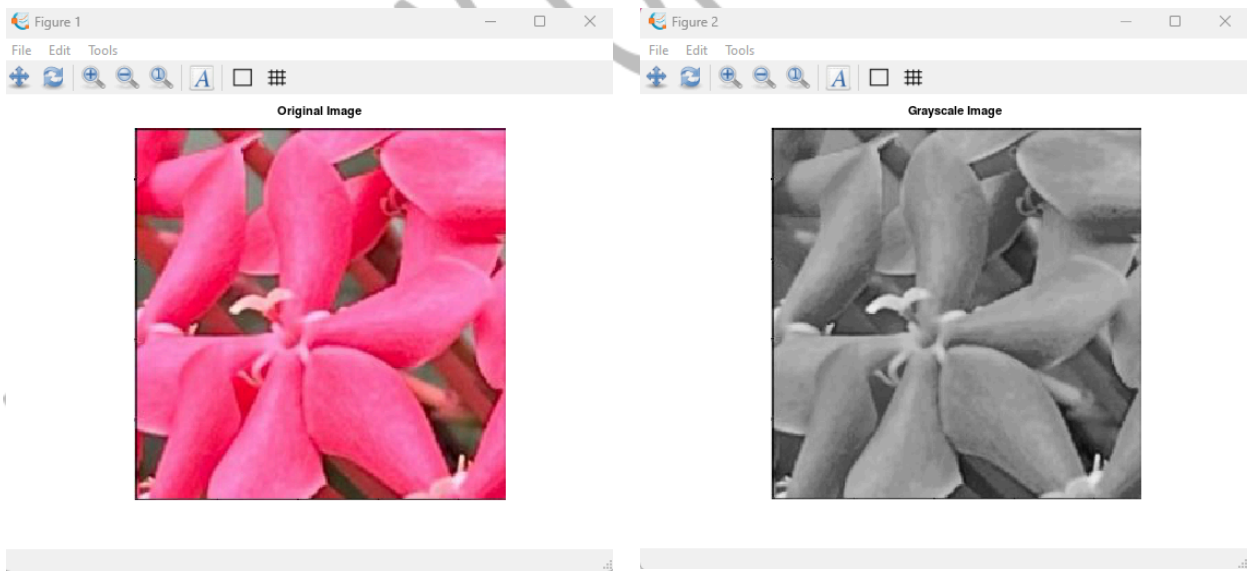


Figure 1
File Edit Tools
Original Image
Figure 2
File Edit Tools
Grayscale Image

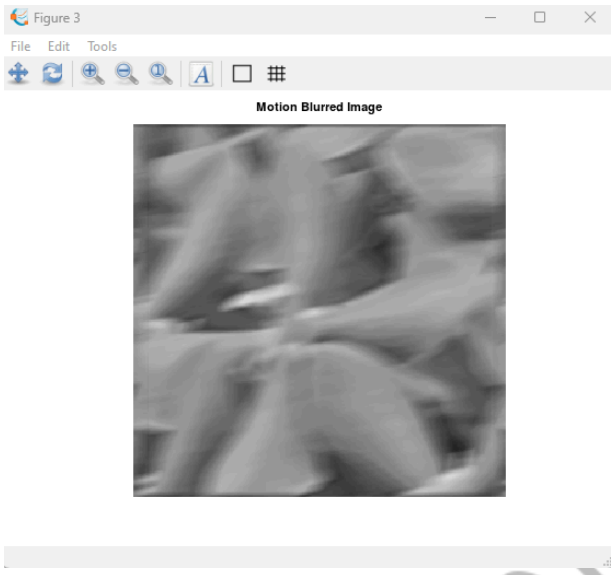


Figure 3
File Edit Tools
Motion Blurred Image

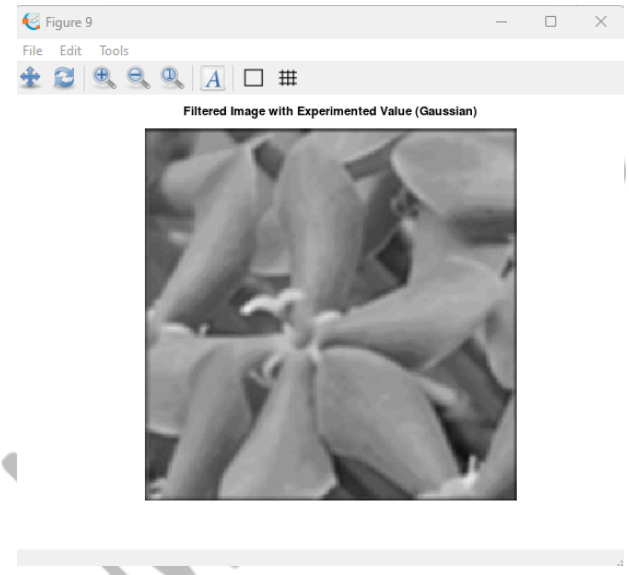


Figure 9
File Edit Tools
Filtered Image with Experimented Value (Gaussian)

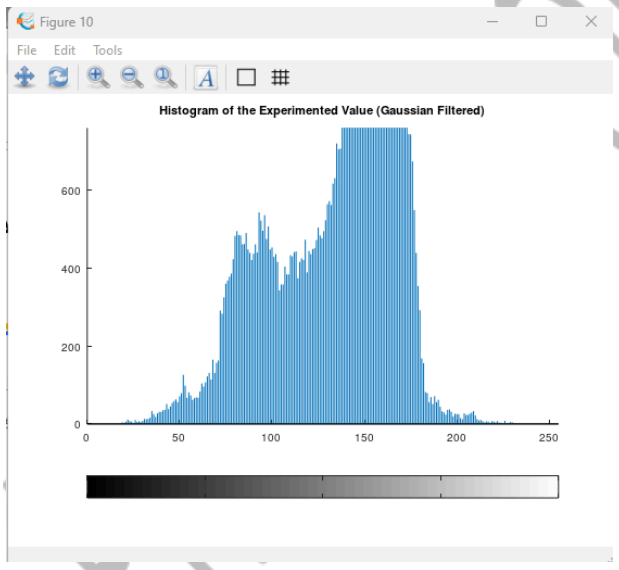


Figure 10
File Edit Tools
Histogram of the Experimented Value (Gaussian Filtered)
600
400
200
0
0
50
100
150
200
250

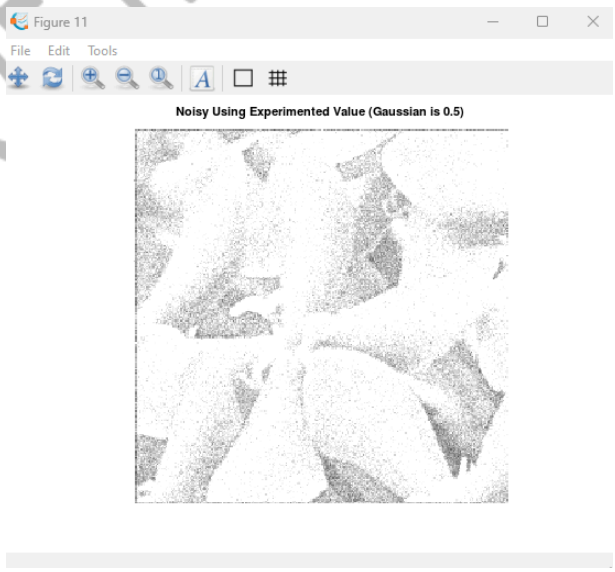


Figure 11
File Edit Tools
Noisy Using Experimented Value (Gaussian is 0.5)

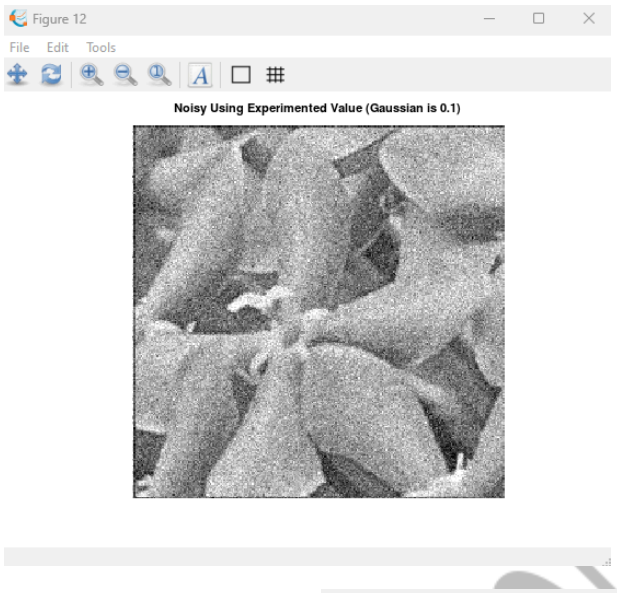


Figure 12
File Edit Tools
Noisy Using Experimented Value (Gaussian is 0.1)

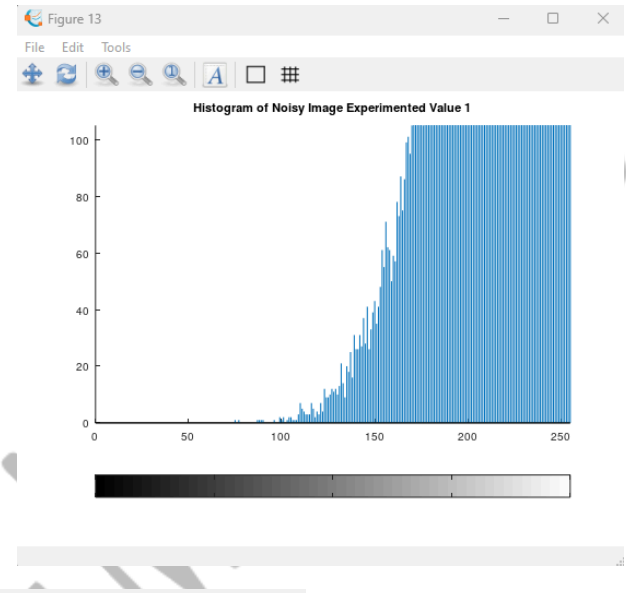


Figure 13
File Edit Tools
Histogram of Noisy Image Experimented Value 1
100
80
60
40
20
0
0
50
100
150
200
250

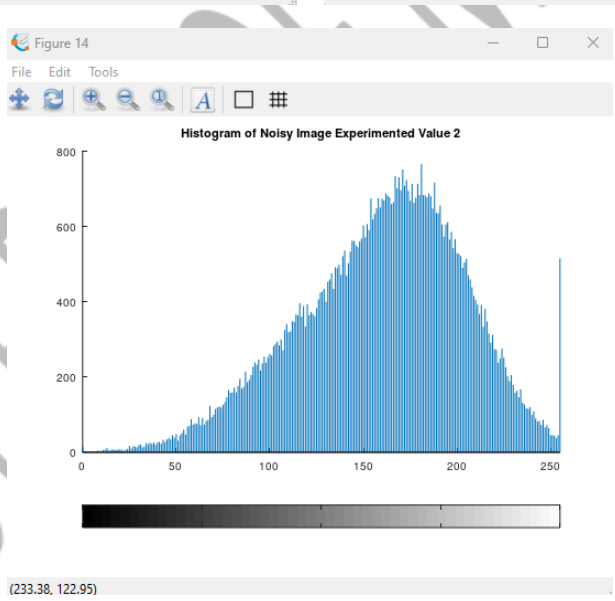


Figure 14
File Edit Tools
Histogram of Noisy Image Experimented Value 2
800
600
400
200
0
0
50
100
150
200
250
(233.38, 122.95)

OCTAVE

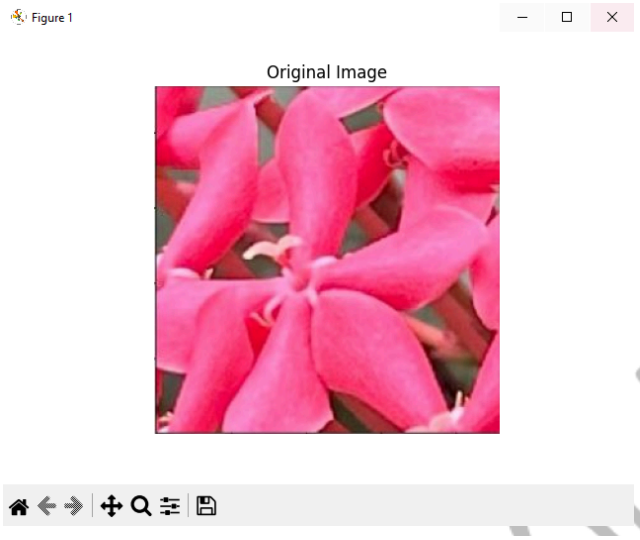


Figure 1
Original Image

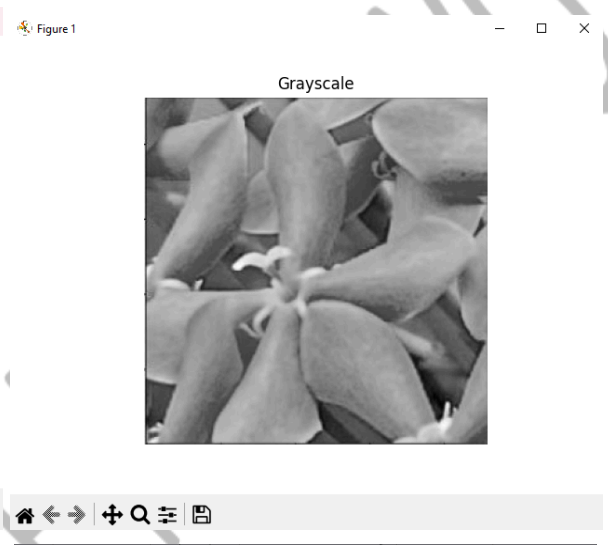


Figure 1
Grayscale

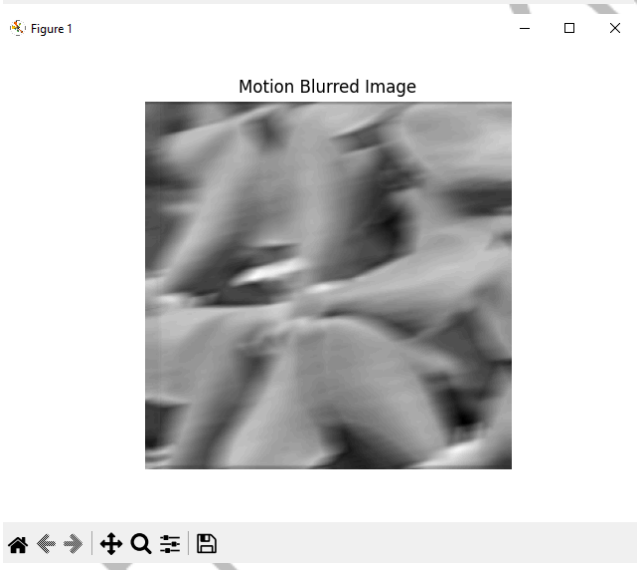


Figure 1
Motion Blurred Image

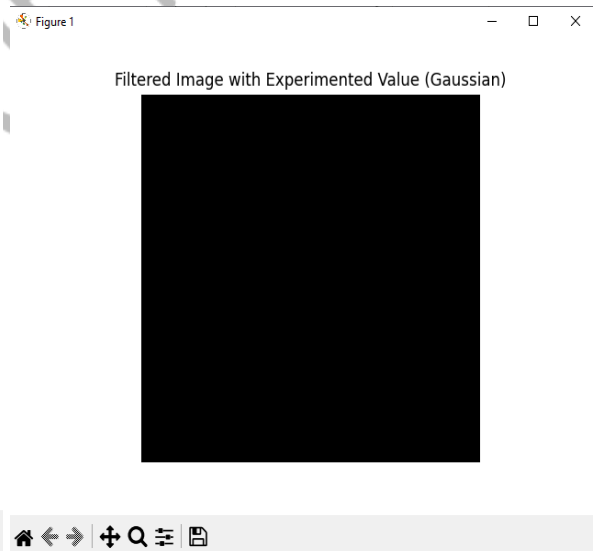
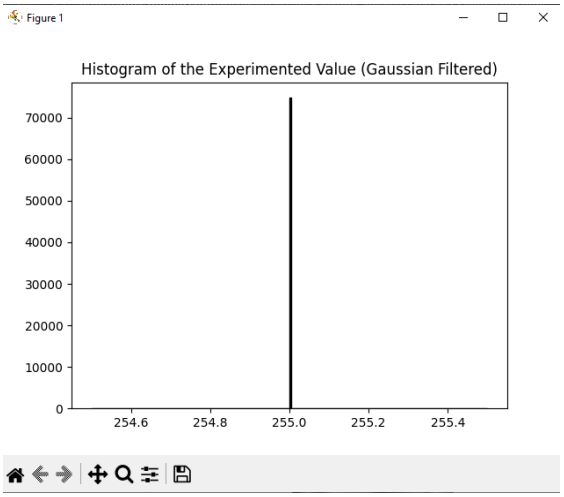
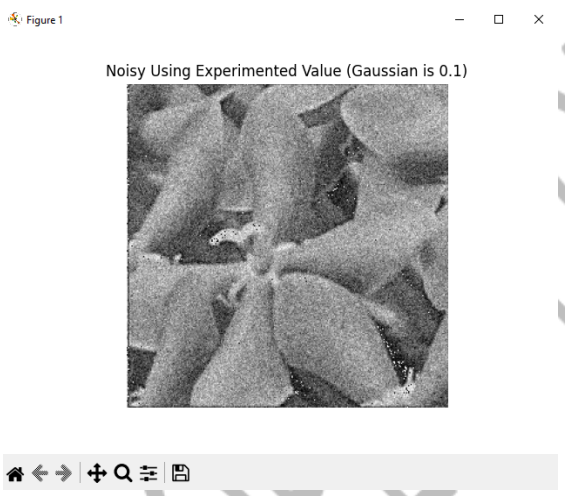
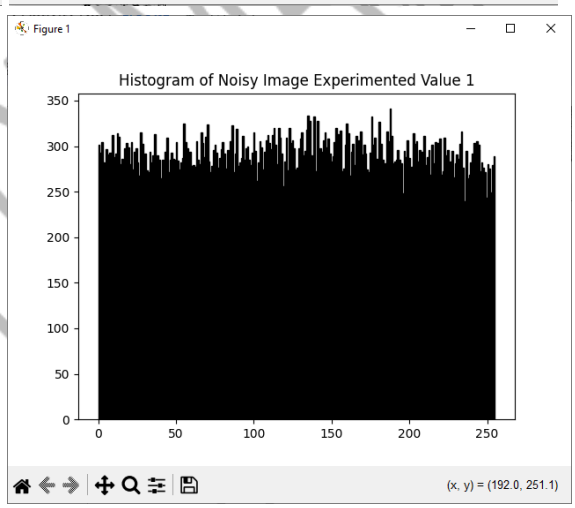
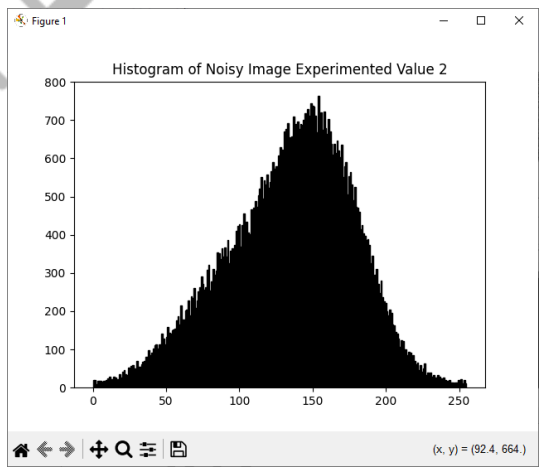


Figure 1
Filtered Image with Experimented Value (Gaussian)









PYTHON

Figure 9: Parameters Modification

2. Visualize the results, analyze and interpret:

The image restoration process involved converting the original image to grayscale to simplify processing, applying a motion blur filter to simulate camera movement, and using Gaussian filtering to reduce noise, though this also blurred details. Unsharp masking was then used to enhance edges and restore some clarity. Gaussian noise was added to the grayscale image to simulate real-world imperfections, and a median filter effectively removed this noise while preserving edges. Finally, Wiener deconvolution was applied to address motion blur, with its effectiveness depending on accurate blur parameter estimation. Collectively, these techniques demonstrated a robust approach to improving image quality by reducing noise and restoring sharpness.

IV. Conclusion

The laboratory activity effectively demonstrated various image processing techniques using different algorithms. Techniques such as grayscale conversion, motion blur simulation, Gaussian filtering, unsharp masking for sharpening, Gaussian noise addition and removal using a median filter, and deblurring using Wiener deconvolution were applied to analyze their impact on image quality. The motion blur and Gaussian filtering helped simulate and reduce noise, while the unsharp masking enhanced edges. Adding Gaussian noise and subsequently removing it with a median filter demonstrated noise reduction's effectiveness despite some edge blurring. Finally, the deblurring technique illustrated an attempt to recover the original image quality, contingent on accurate blur parameter estimation. Overall, the applied algorithms showcased the significant transformations and improvements possible in digital image processing.

References

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