**Elective 3**

Laboratory Activity No. 4

**Image Restoration**

|  |
| --- |
|  |

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*

**09-08-2024**

*Submitted to:*

**Engr. Maria Rizette H. Sayo**

1. 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).
4. 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");

1. Supplementary Activity

* Write a Python program that will implement the output in Method A.
* import cv2  
  import numpy as np  
  import matplotlib.pyplot as plt  
  import scipy.ndimage as sc  
  from skimage import restoration,io  
  #Read the image  
  img = cv2.imread('flower.jpg')  
  #Display the original image  
  cv2.imshow('Original Image', img)  
  #Convert to graysacle 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', img\_gray)  
  #Add blur to the image  
  kernel\_motion\_blur = np.zeros((21, 21))  
  kernel\_motion\_blur[10, :] = np.ones(21)  
  kernel\_motion\_blur = kernel\_motion\_blur / 21  
  img\_blur = cv2.filter2D(img\_gray, -1, kernel\_motion\_blur)  
  #Show the image  
  cv2.imshow('Motion Blurred Image', img\_blur)  
  #Gaussian filtering  
  img\_gaussian\_filtered = sc.gaussian\_filter(img\_blur, sigma=3)  
  #Display the Gaussian filtered image  
  cv2.imshow('Filtered Image (Gaussian)', img\_gaussian\_filtered)  
  #Sharpening using unsharp masking  
  kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])  
  img\_sharpened = cv2.filter2D(img\_blur, -1, kernel)  
  #Display the sharpened image  
  cv2.imshow('Sharpened Image', img\_sharpened)  
  #Add Gaussian noise and remove it using median filter  
  noise = np.random.normal(25, 15, img\_gray.shape).astype(np.uint8)  
  img\_noisy = cv2.add(img\_gray, noise)  
  img\_noisy\_removed = cv2.medianBlur(img\_noisy, 5)  
  #Display the noise image  
  cv2.imshow('Noisy', img\_noisy)  
  #Display the noise-removed images  
  cv2.imshow('Noise Removed', img\_noisy\_removed)  
  #Deblurring  
  img2 = io.imread('flower.jpg', as\_gray=True)  
  kernel\_motion\_blur = np.zeros((21, 21))  
  kernel\_motion\_blur[10, :] = np.ones(21)  
  kernel\_motion\_blur = kernel\_motion\_blur / 21  
  img\_blur2 = cv2.filter2D(img2, -1, kernel\_motion\_blur)  
  img\_deblurred = restoration.wiener(img\_blur2,np.ones((1,1)),1)  
  cv2.imshow('Deblurred Image', img\_deblurred)  
  #Parameter Modification  
  #Gaussian Filtering  
  img\_gaussian\_filtered2 = sc.gaussian\_filter(img\_blur, sigma=5)  
  cv2.imshow('Filtered Image with Experimented Value (Gaussian)', img\_gaussian\_filtered2)  
  #Histogram (Gaussian Filtered)  
  hist1 = cv2.calcHist([img\_gaussian\_filtered2], [0], None, [256], [0, 256])  
  plt.figure(1)  
  plt.plot(hist1)  
  plt.title('Histogram of the Experimented Value (Gaussian Filtered)')  
  #Add Gaussian noise  
  noise1 = np.random.normal(100, 15, img\_gray.shape).astype(np.uint8)  
  noise2 = np.random.normal(25, 15, img\_gray.shape).astype(np.uint8)  
  img\_noisy\_exp1 = cv2.add(img\_gray, noise1)  
  img\_noisy\_exp2 = cv2.add(img\_gray, noise2)  
  #Display the noisy  
  cv2.imshow('Noisy Using Experimented Value (Gaussian is 0.5)', img\_noisy\_exp1)  
  cv2.imshow('Noisy Using Experimented Value (Gaussian is 0.1)', img\_noisy\_exp2)  
  #Display the histogram for Noisy  
  hist2 = cv2.calcHist([img\_noisy\_exp1], [0], None, [256], [0, 256])  
  hist3 = cv2.calcHist([img\_noisy\_exp2], [0], None, [256], [0, 256])  
  plt.figure(2)  
  plt.plot(hist2)  
  plt.title('Histogram of Noisy Image Experimented Value 1')  
  plt.figure(3)  
  plt.plot(hist3)  
  plt.title('Histogram of Noisy Image Experimented Value 2')  
  plt.show()  
  cv2.waitKey(0)  
  cv2.destroyAllWindows()

1. Results

Steps:

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

**MATLAB Results**

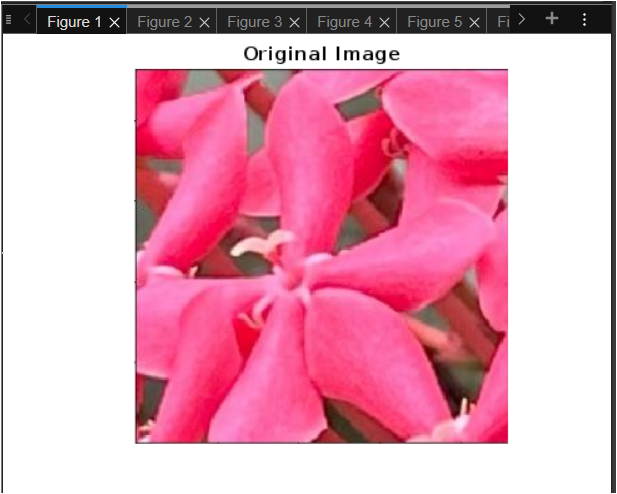


Figure 1a. Acquire an Image of a Flower

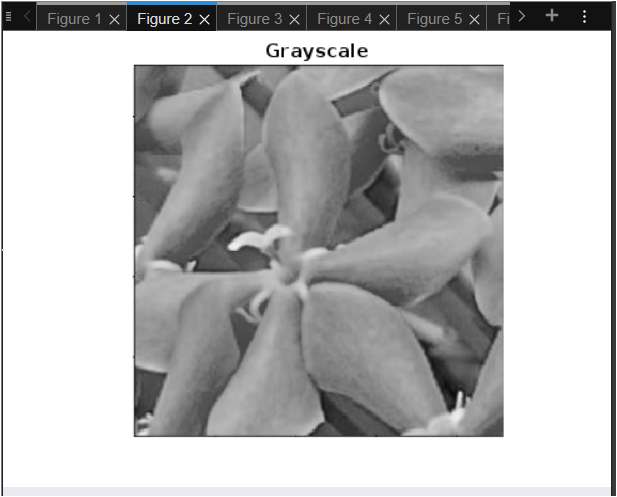


Figure 2a. Original and Grayscale Image



Figure 4a. Gaussian-filtered Image



Figure 3a. Motion Blurred Image



Figure 8a. Deblurred Image

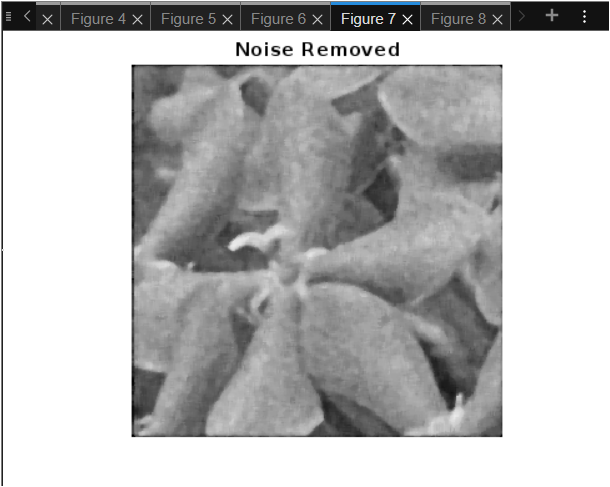


Figure 7a. Added Gaussian Noise and Removed Image

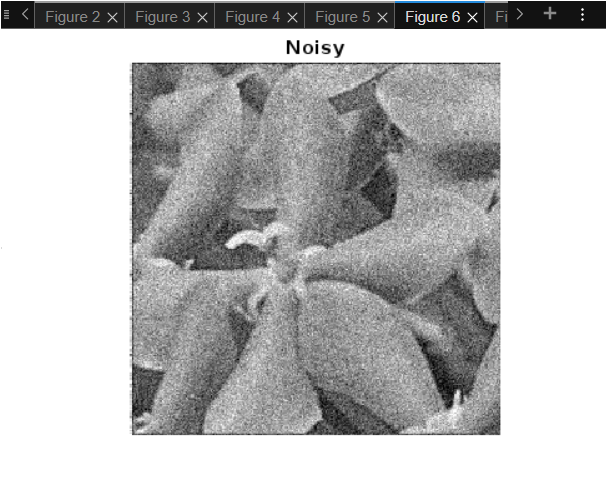


Figure 6a. Gaussian-Noise Image



Figure 5a. Sharpen Image

**Octave Results**

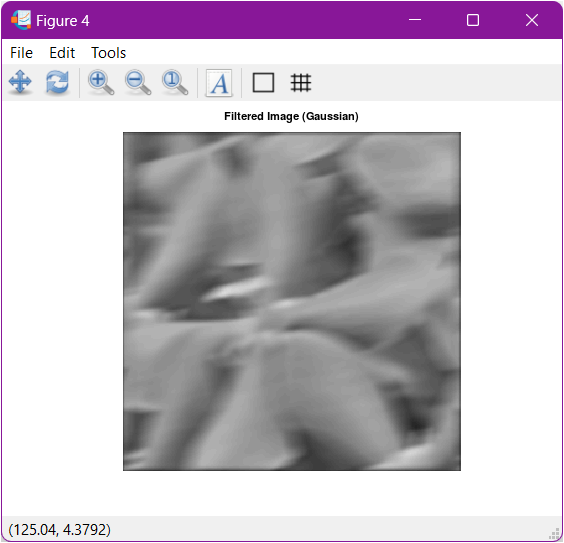


Figure 4b. Gaussian-filtered Image

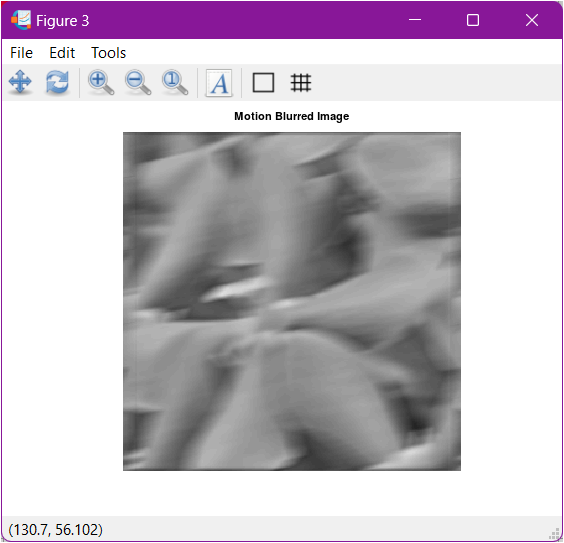


Figure 3b. Motion Blurred Image

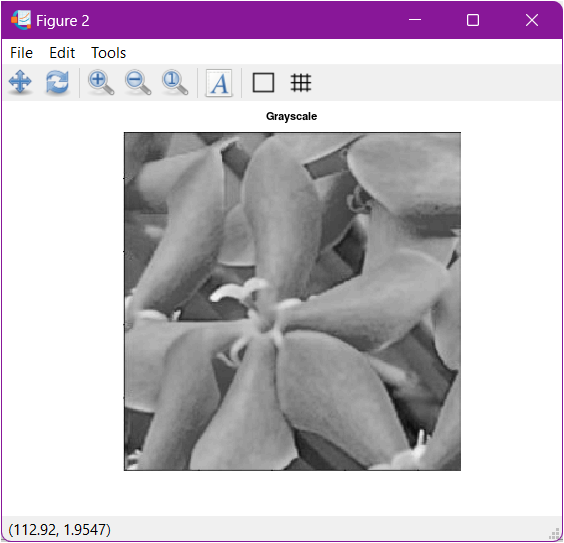


Figure 2b. Original and Grayscale Image

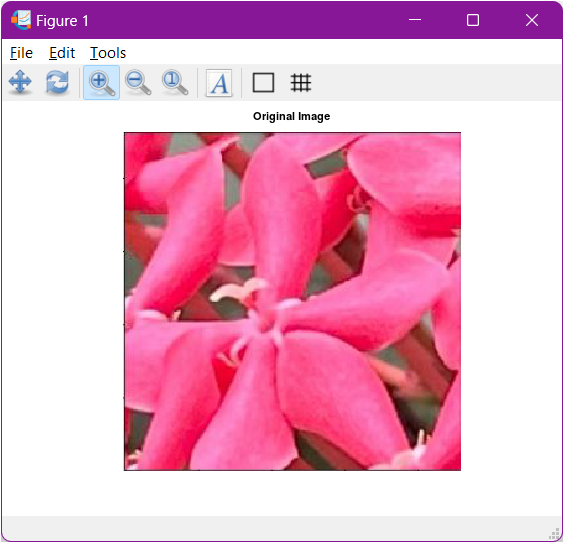


Figure 1b. Acquire an Image of a Flower

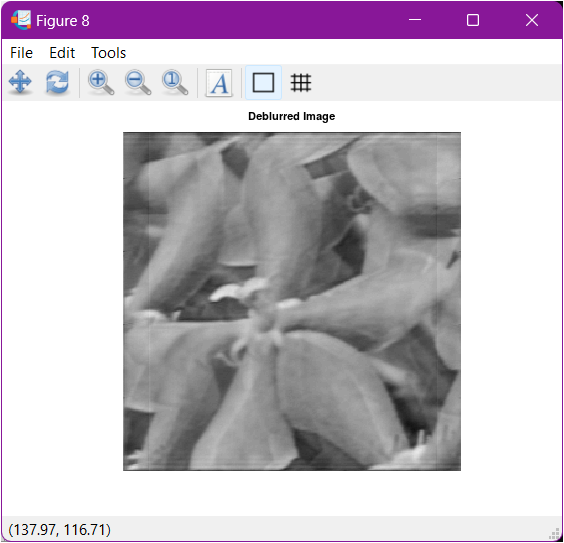


Figure 8b. Deblurred Image

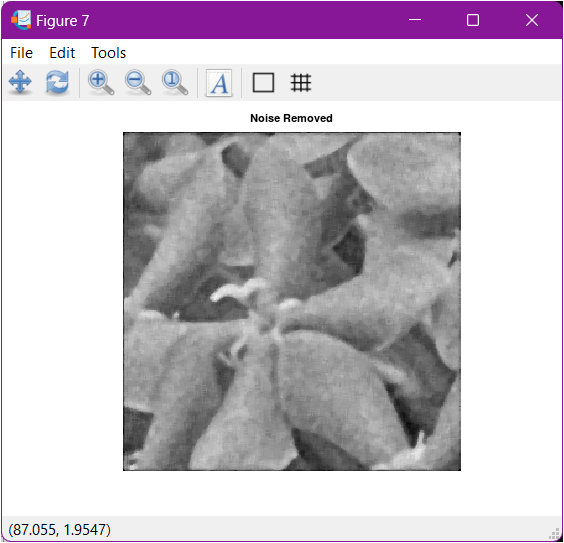


Figure 7b. Added Gaussian Noise and Removed Image

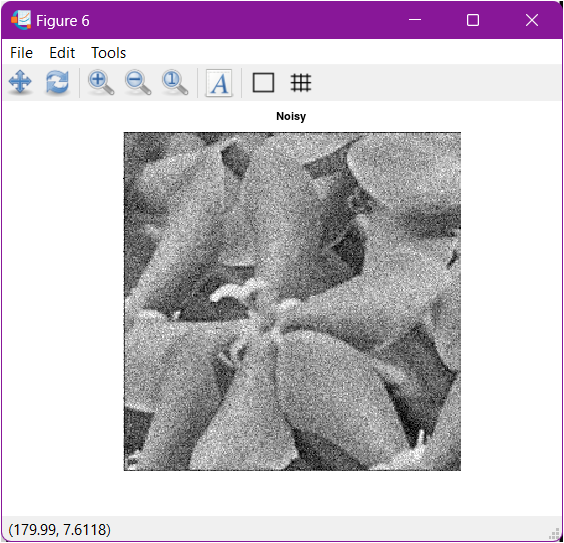


Figure 6b. Gaussian Noise Image

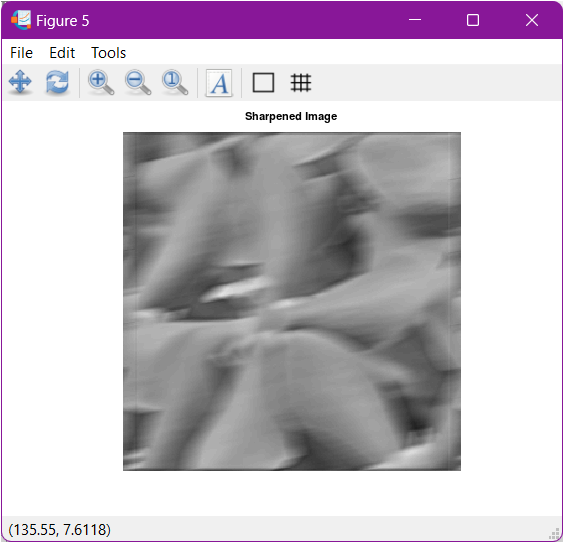


Figure 5b. Sharpen Image

**Python Results**

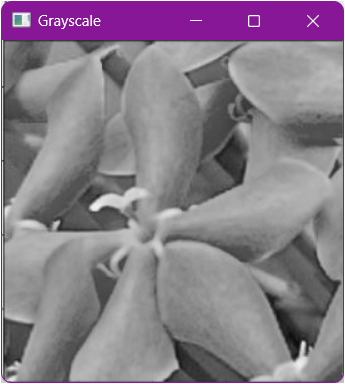


Figure 2c. Original and Grayscale Image

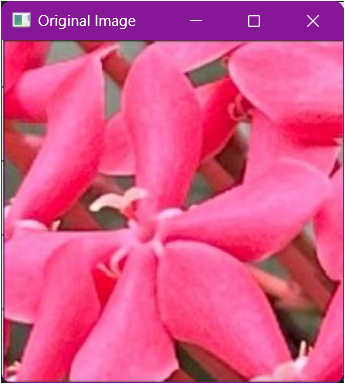


Figure 1c. Acquire an Image of a Flower



Figure 4c. Gaussian-filtered Image

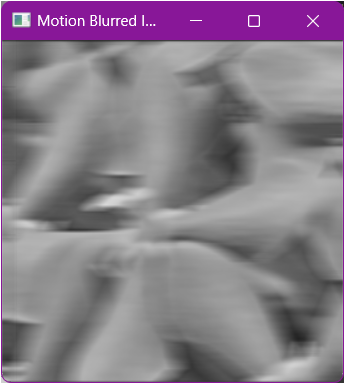


Figure 3c. Motion Blurred Image

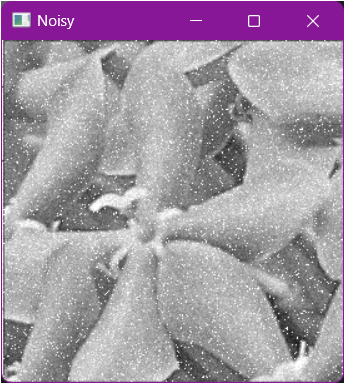


Figure 6c. Gaussian Noise Image



Figure 5c. Sharpen Image

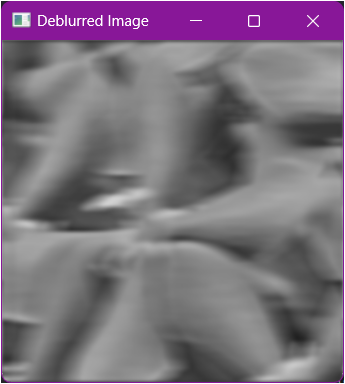


Figure 8c. Deblurred Image

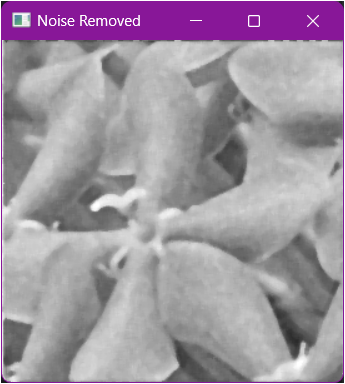


Figure 7c. Added Gaussian Noise and Removed Image

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');

**MATLAB Results**

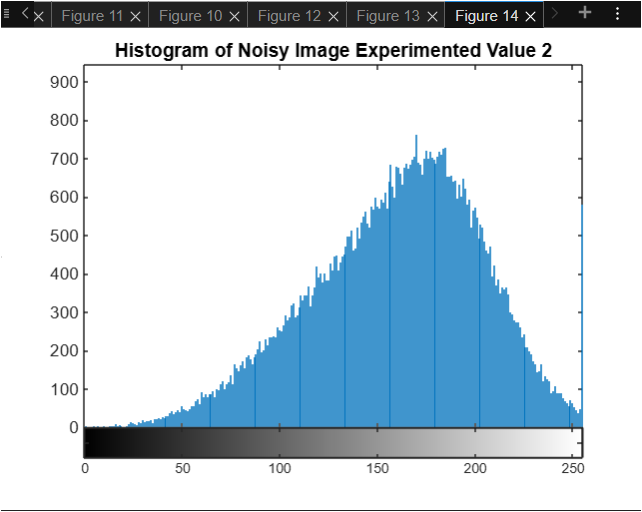
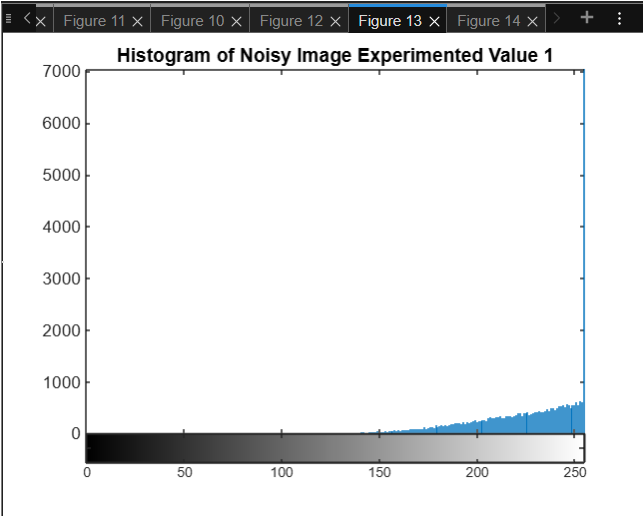
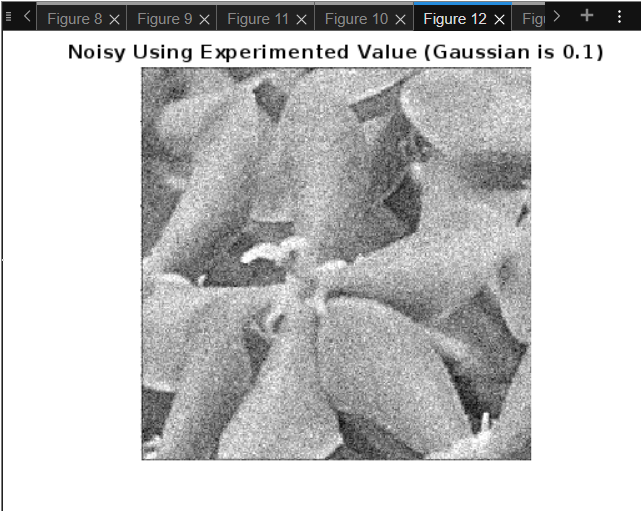
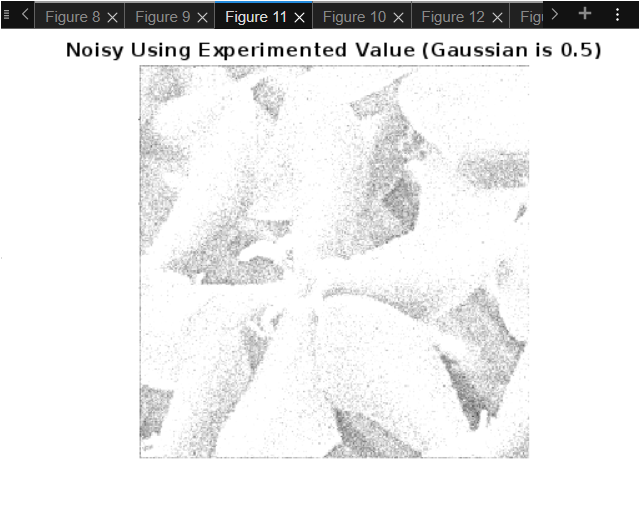
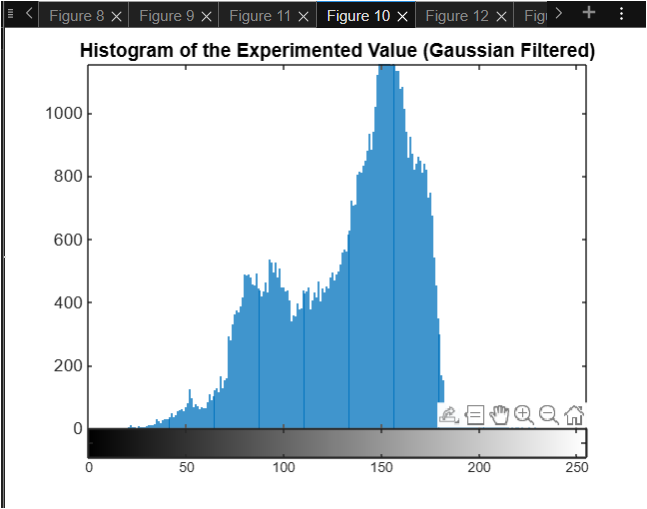
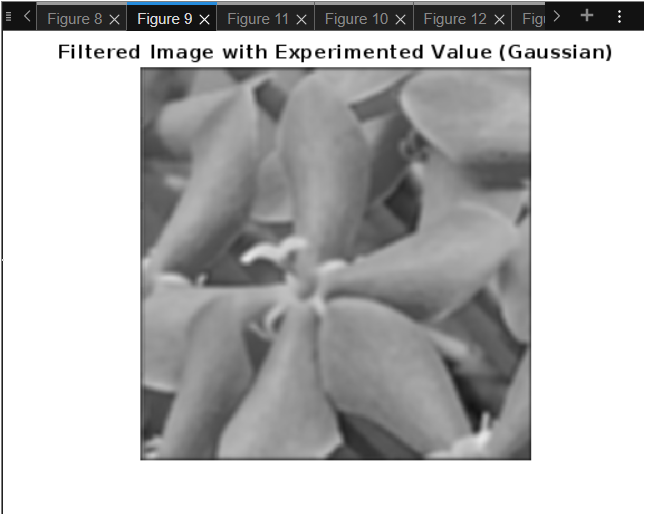


Figure 9a. Parameters Modification

**Octave Results**

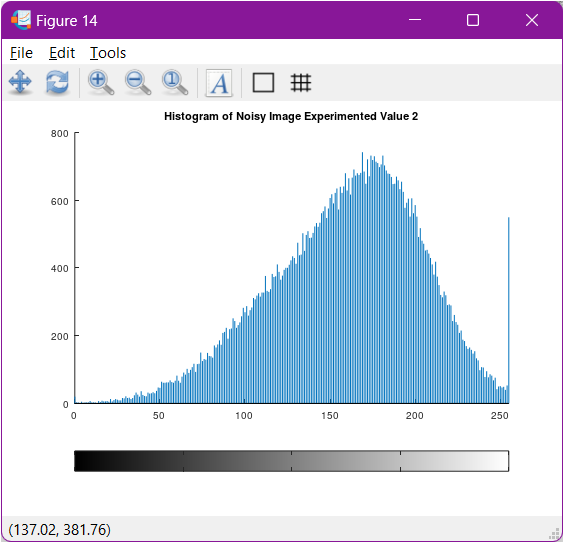
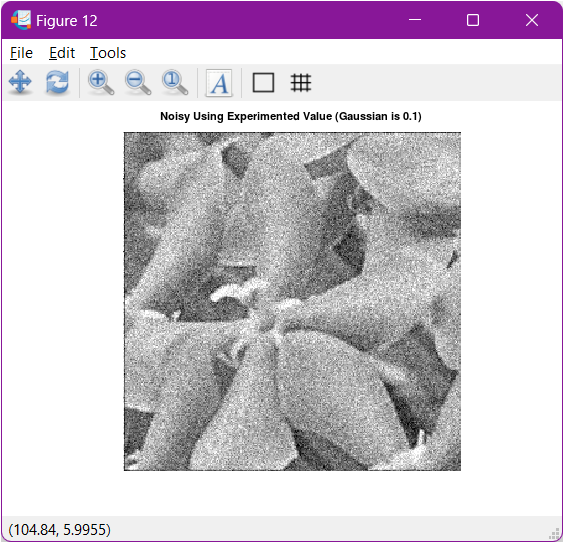
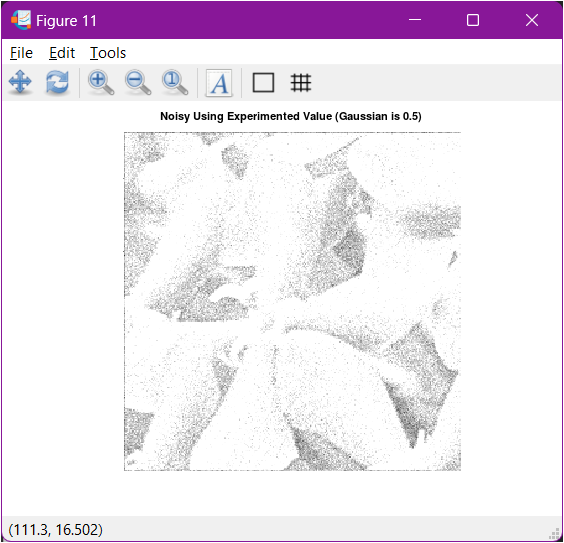
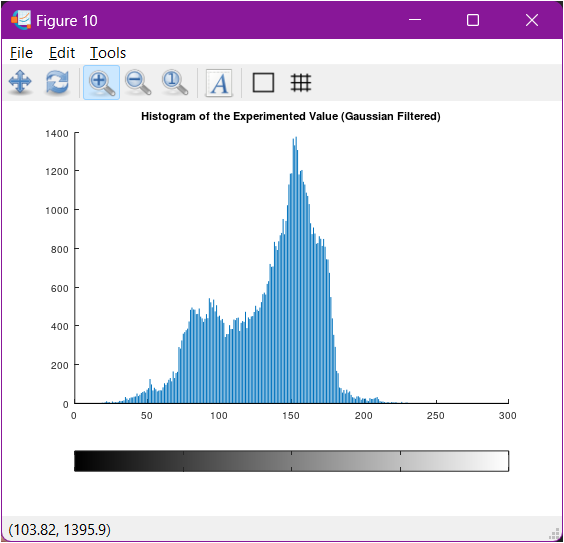
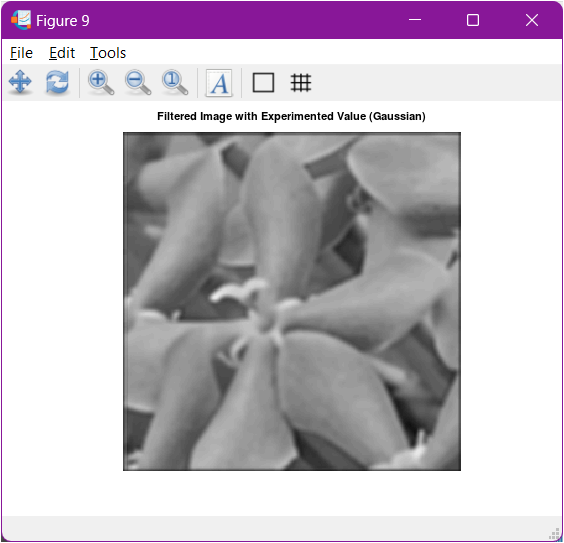


Figure 9b. Parameter Modification

**Python Results**

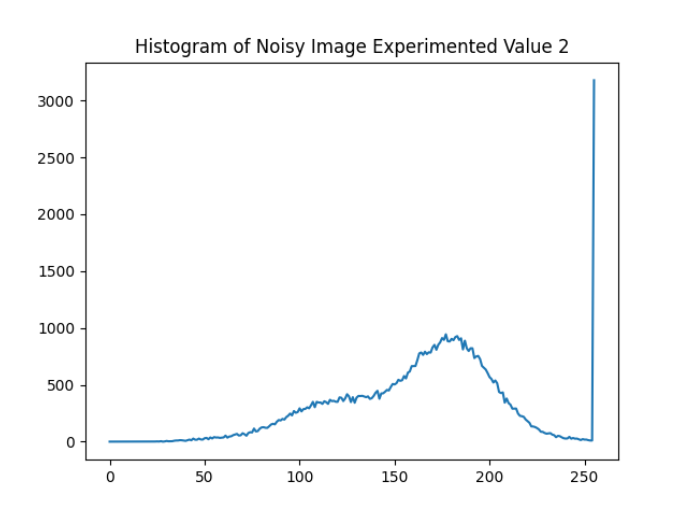
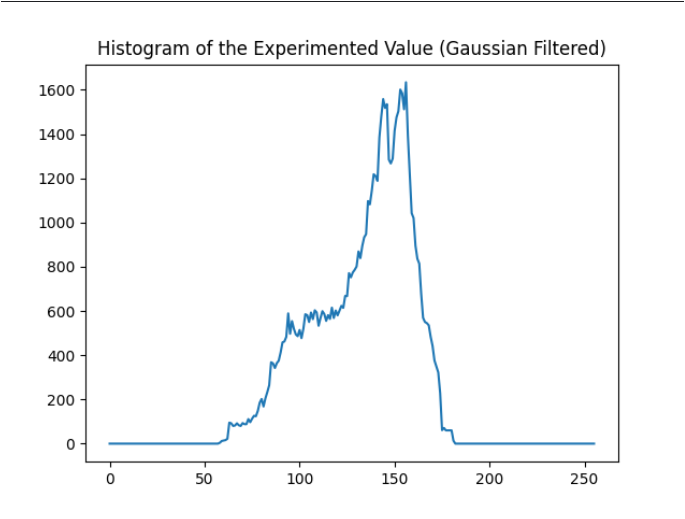
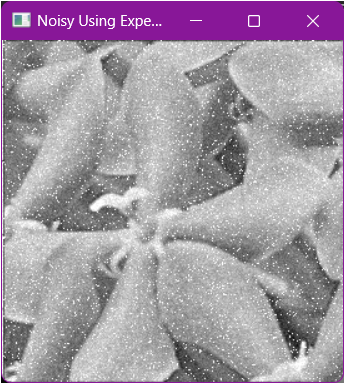
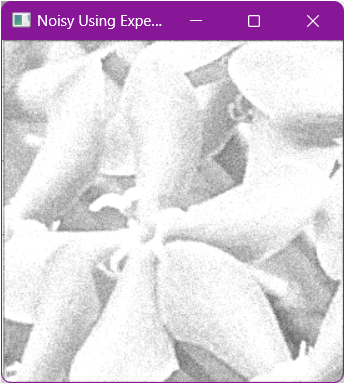
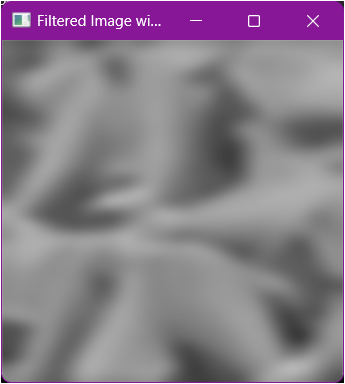


Figure 9c. Parameter Modifications

1. Visualize the results, analyze and interpret:

Gaussian filtering and deblurring are common image processing techniques implemented in MATLAB, Octave, and Python. The outputs from these processes can vary slightly depending on the platform, but the underlying principles are consistent. Gaussian filtering, which is used to blur an image by applying a Gaussian function to reduce noise and detail, can be performed in MATLAB using the imfilter function. In Octave, it is achieved using the imfilter function with a Gaussian kernel generated by fspecial, similar to MATLAB. In Python, similar results can be obtained using either OpenCV's cv2.GaussianBlur or Scipy's scipy.ndimage.gaussian\_filter. Although MATLAB and Octave generally produce similar outputs due to Octave's goal of replicating MATLAB's functionality, Python might yield slightly different results because of differences in kernel implementation and interpolation methods.

Deblurring, which involves reversing the effects of blurring, is more complex and typically utilizes techniques like Wiener deconvolution. In MATLAB, this can be done using the deconvwnr function, while in Octave, similar functionality is available but may require external packages like the "image" package. In Python, deblurring can be performed using libraries such as Scipy or skimage, with the wiener function or restoration.wiener function being commonly used. The outputs from deblurring operations tend to vary more between platforms due to the complexity of the process and differences in the implementation of deconvolution methods.

IV. Conclusion

In conclusion, while Gaussian filtering generally produces consistent results across MATLAB, Octave, and Python, with only subtle differences, deblurring outputs can differ more significantly depending on the platform and the specific method used. The choice of platform often depends on the specific application, the availability of functions, and personal preference for syntax and environment.

**References**

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

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

Simsangcheol. (2023, January 25). *OpenCV – Histogram of Grayscale Image.* [**https://medium.com/@sim30217/opencv-histogram-of-grayscale-image-8de86fb248e1**](https://medium.com/@sim30217/opencv-histogram-of-grayscale-image-8de86fb248e1)