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

# Elective 3

Laboratory Activity No. 2

**Image Acquisition and Manipulation**

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*

**31-07-2024**

*Submitted to:*

**Engr. Maria Rizette H. Sayo**

## I. Objectives

This laboratory activity aims to implement the principles and techniques of image acquisition, representation, color models through MATLAB/Octave and open CV using Python

1. Acquire the image.
2. Acquire image representation.
3. Acquire image color models.
4. Modify image representation.
5. Flip Image.

## II. Methods

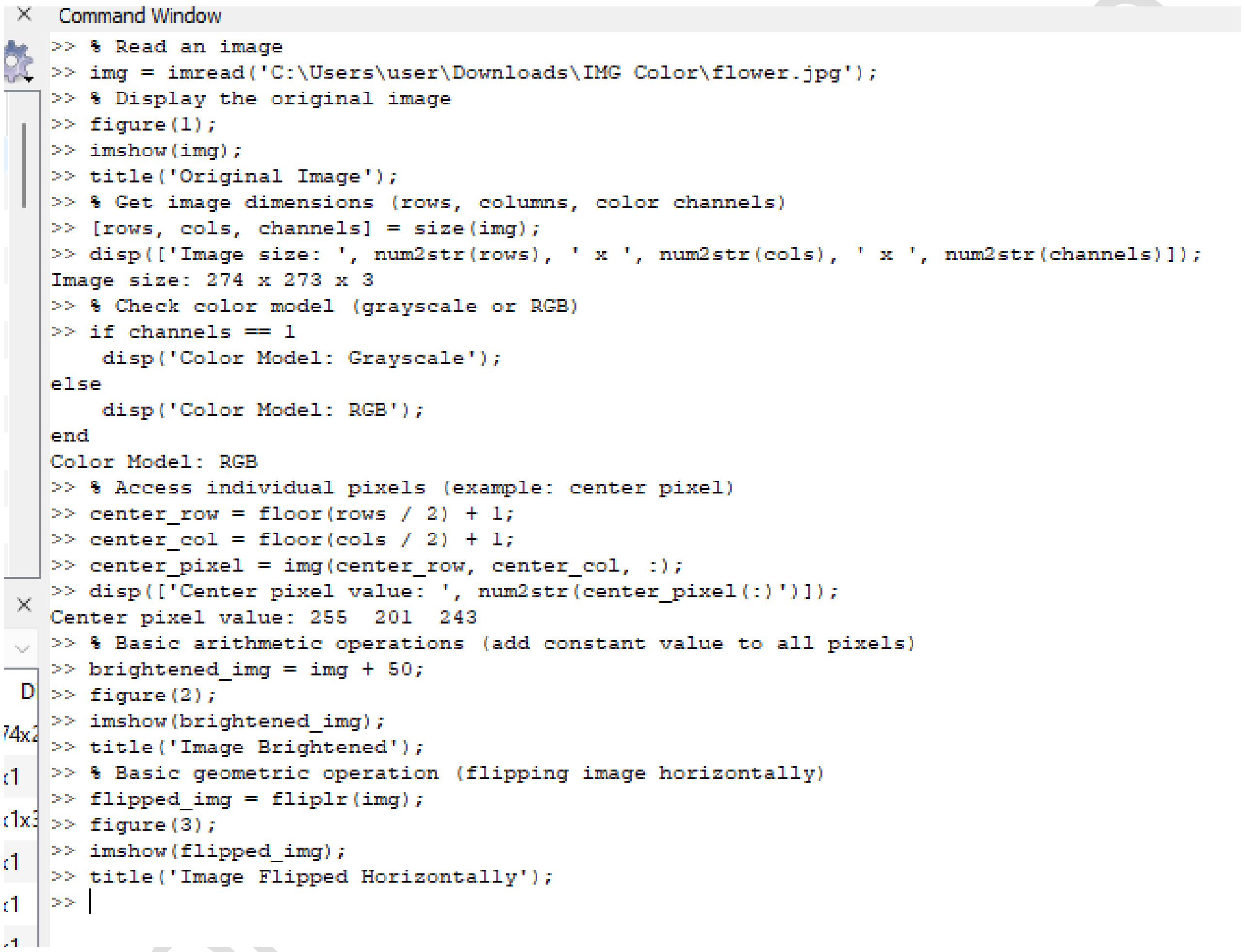
A. Perform a task given in the presentation

- Copy and paste your MATLAB code

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

% Display the image
figure(1);
imshow(img); title('Original Image');

% Get image dimensions (rows, columns, color channels)
[rows, cols, channels] = size(img);
disp(['Image size: ', num2str(rows), ' x ', num2str(cols), ' x ',num2str(channels)]);

% Check color model (grayscale or RGB)
if channels == 1
   disp('Color Model: Grayscale');else
   disp('Color Model: RGB');end

% Access individual pixels (example: center pixel)
center_row = floor(rows/2) + 1;
center_col = floor(cols/2) + 1;
center_pixel = img(center_row, center_col, :); disp(['Center pixel value: ',
num2str(center_pixel)]);

% Basic arithmetic operations (add constant value to all pixels)b
rightened_img = img + 50;
figure (2);
imshow(brightened_img); title ('Image Brightened');
% Basic geometric operation (flipping image horizontally)

flipped_img = fliplr(img);
```

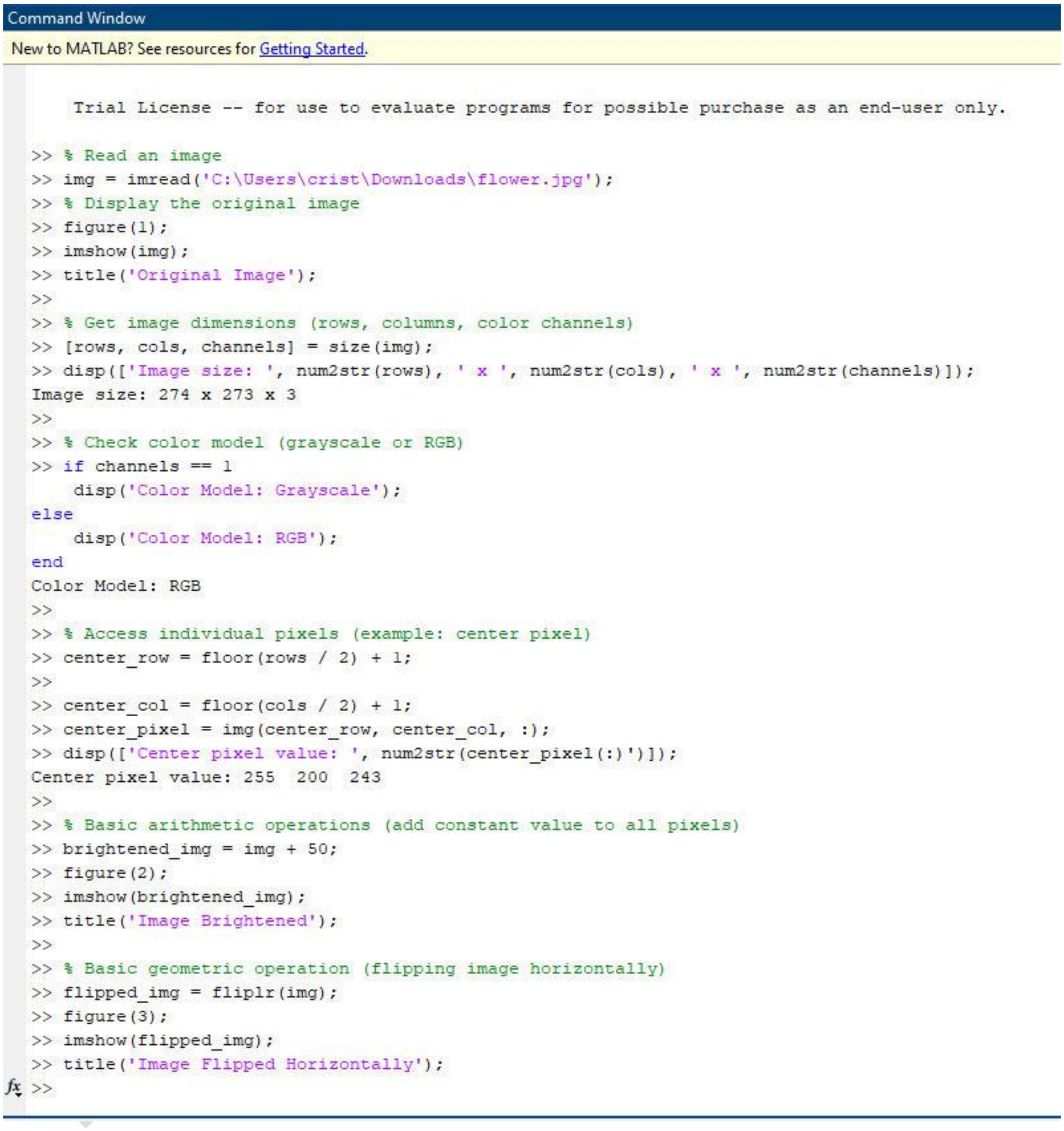
```
 figure(3);
imshow(flipped_img); title('Image Flipped Horizontally');
```

Octave Code



```
>> % Read an image
>> img = imread('C:\Users\user\Downloads\IMG Color\flower.jpg');
>> % Display the original image
>> figure(1);
>> imshow(img);
>> title('Original Image');
>> % Get image dimensions (rows, columns, color channels)
>> [rows, cols, channels] = size(img);
>> disp(['Image size: ', num2str(rows), ' x ', num2str(cols), ' x ', num2str(channels)]);
Image size: 274 x 273 x 3
>> % Check color model (grayscale or RGB)
>> if channels == 1
    disp('Color Model: Grayscale');
else
    disp('Color Model: RGB');
end
Color Model: RGB
>> % Access individual pixels (example: center pixel)
>> center_row = floor(rows / 2) + 1;
>> center_col = floor(cols / 2) + 1;
>> center_pixel = img(center_row, center_col, :);
>> disp(['Center pixel value: ', num2str(center_pixel(:)')]);
Center pixel value: 255  201  243
>> % Basic arithmetic operations (add constant value to all pixels)
>> brightened_img = img + 50;
>> figure(2);
>> imshow(brightened_img);
>> title('Image Brightened');
>> % Basic geometric operation (flipping image horizontally)
>> flipped_img = fliplr(img);
>> figure(3);
>> imshow(flipped_img);
>> title('Image Flipped Horizontally');
>> 
```

MATLAB Code



```
  Trial License -- for use to evaluate programs for possible purchase as an end-user only.

>> % Read an image
>> img = imread('C:\Users\crist\Downloads\flower.jpg');
>> % Display the original image
>> figure(1);
>> imshow(img);
>> title('Original Image');
>>
>> % Get image dimensions (rows, columns, color channels)
>> [rows, cols, channels] = size(img);
>> disp(['Image size: ', num2str(rows), ' x ', num2str(cols), ' x ', num2str(channels)]);
Image size: 274 x 273 x 3
>>
>> % Check color model (grayscale or RGB)
>> if channels == 1
    disp('Color Model: Grayscale');
else
    disp('Color Model: RGB');
end
Color Model: RGB
>>
>> % Access individual pixels (example: center pixel)
>> center_row = floor(rows / 2) + 1;
>>
>> center_col = floor(cols / 2) + 1;
>> center_pixel = img(center_row, center_col, :);
>> disp(['Center pixel value: ', num2str(center_pixel(:)')]);
Center pixel value: 255  200  243
>>
>> % Basic arithmetic operations (add constant value to all pixels)
>> brightened_img = img + 50;
>> figure(2);
>> imshow(brightened_img);
>> title('Image Brightened');
>>
>> % Basic geometric operation (flipping image horizontally)
>> flipped_img = fliplr(img);
>> figure(3);
>> imshow(flipped_img);
>> title('Image Flipped Horizontally');
fx >>
```

B. Supplementary Activity

Python Code

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Read an image
img = cv2.imread('E:\\PLM CET SUBJECTS\\Digital Image Processing\\flower.jpg')

# Convert the image from BGR (OpenCV default) to RGB
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Display the image
plt.figure(1)
plt.imshow(img_rgb)
plt.title('Original Image')
plt.show()

# Get image dimensions (rows, columns, color channels)
rows, cols, channels = img.shape
print(f'Image size: {rows} x {cols} x {channels}')

# Check color model (grayscale or RGB)
if channels == 1:
    print('Color Model: Grayscale')
else:
    print('Color Model: RGB')

# Access individual pixels (example: center pixel)
center_row = rows // 2
center_col = cols // 2
center_pixel = img[center_row, center_col, :]
print(f'Center pixel value: {center_pixel}')

# Basic arithmetic operations (add constant value to all pixels)
brightened_img = cv2.add(img, 50)
brightened_img_rgb = cv2.cvtColor(brightened_img, cv2.COLOR_BGR2RGB)
plt.figure(2)
plt.imshow(brightened_img_rgb)
plt.title('Image Brightened')
plt.show()

# Basic geometric operation (flipping image horizontally)
flipped_img = cv2.flip(img, 1)
flipped_img_rgb = cv2.cvtColor(flipped_img, cv2.COLOR_BGR2RGB)
plt.figure(3)
plt.imshow(flipped_img_rgb)
plt.title('Image Flipped Horizontally')
plt.show()
```

## III. Results

Image Attribute and Color Model

OCTAVE

Image size: 274 x 273 x 3

Color Model: RGB

Center pixel value: 255  200  243

MATLAB

Image size: 274 x 273 x 3

Color Model: RGB

Center pixel value: 255 200 243

PYTHON

Image size: 274 x 273 x 3

Color Model: RGB

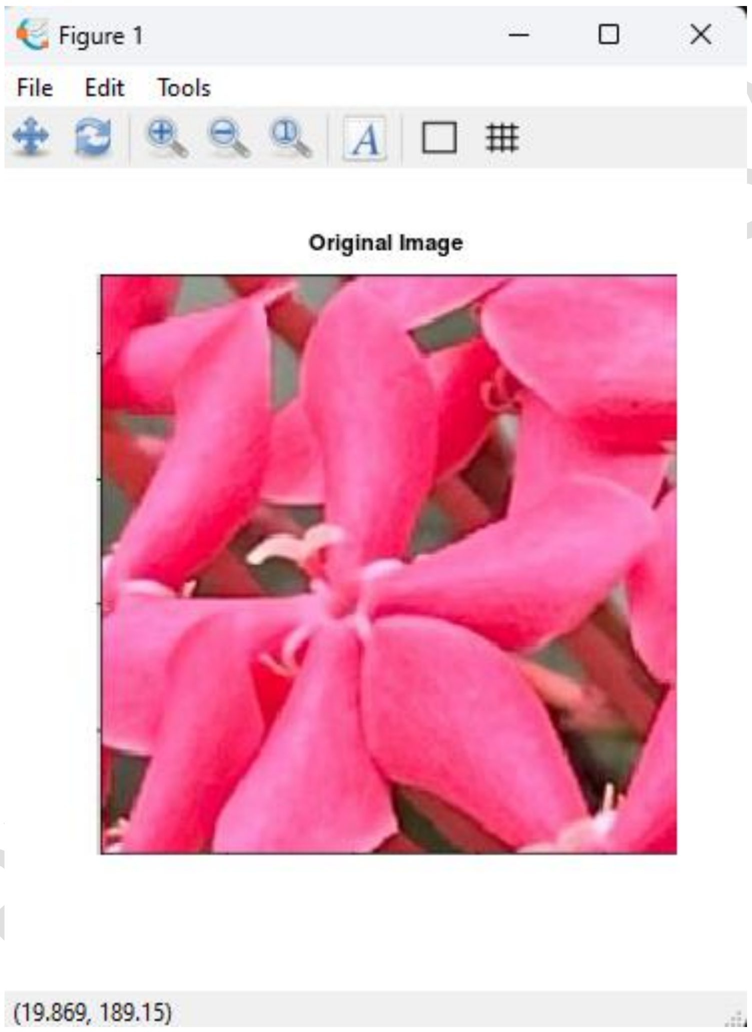
Center pixel value: 243 200 255

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

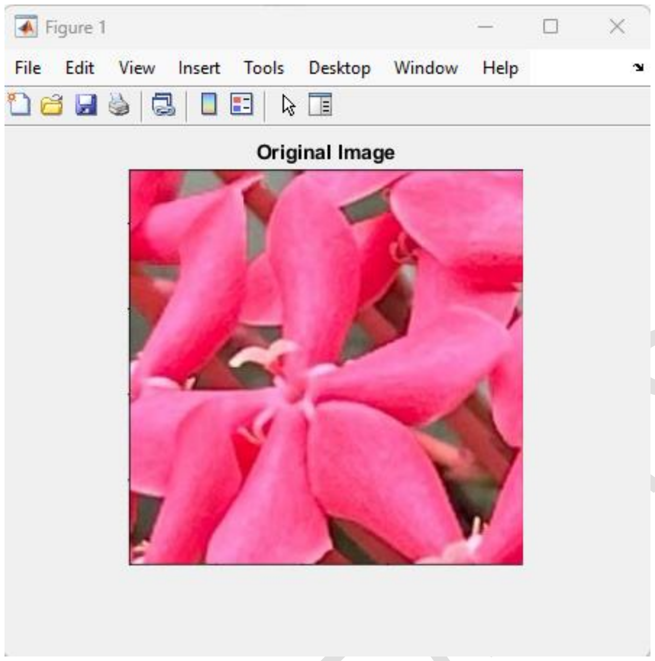
Steps:
1. Copy/crop and paste your results. Label each output (Figure1, Figure2, Figure3)

picture file: flower.jpg

OCTAVE



MATLAB



PYTHON

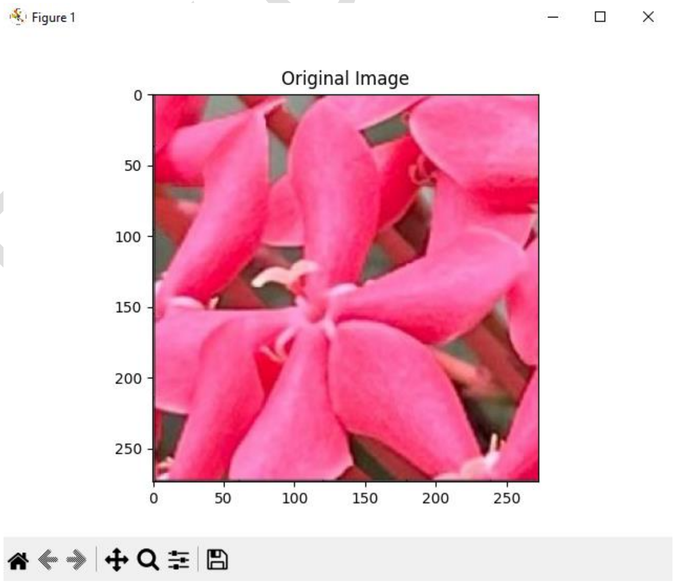
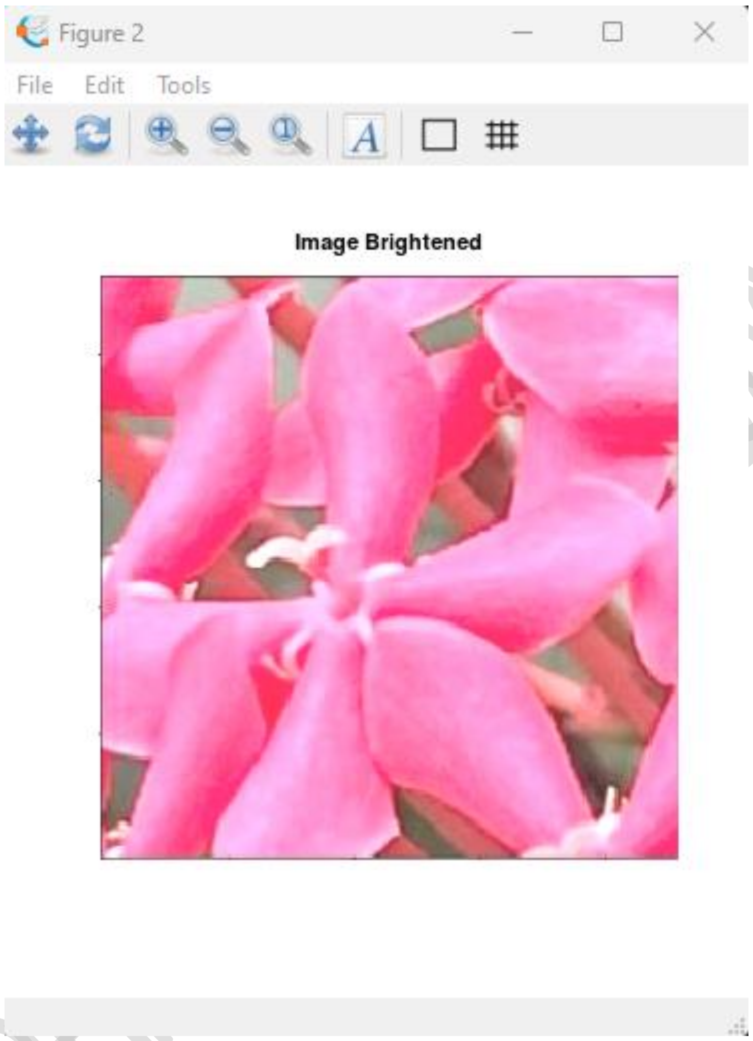
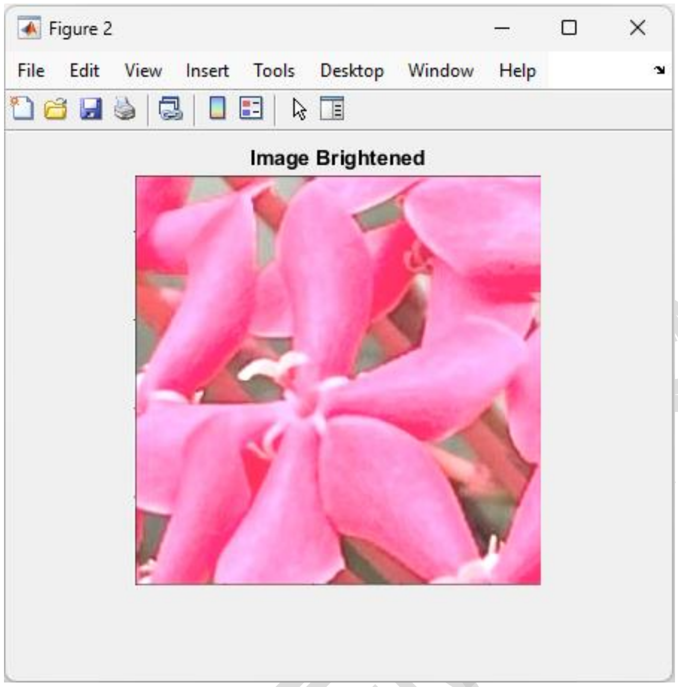


Figure 1: Acquire an Image of a Flower

OCTAVE



MATLAB



PYTHON

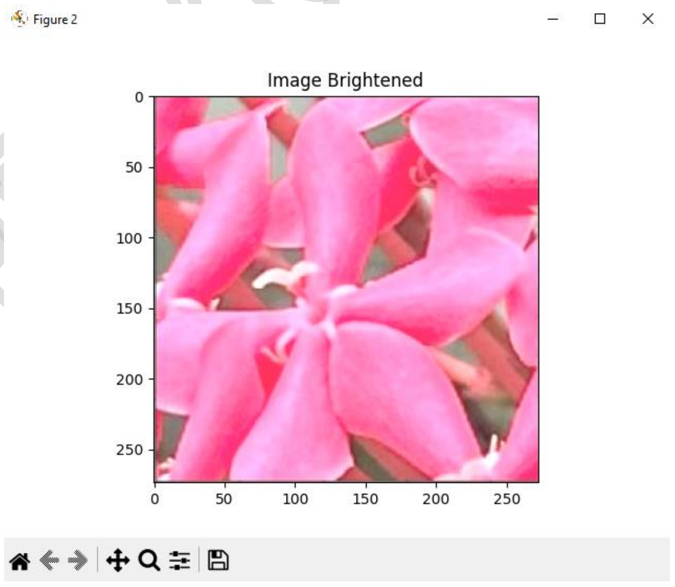


Figure 2: Image brightened

OCTAVE

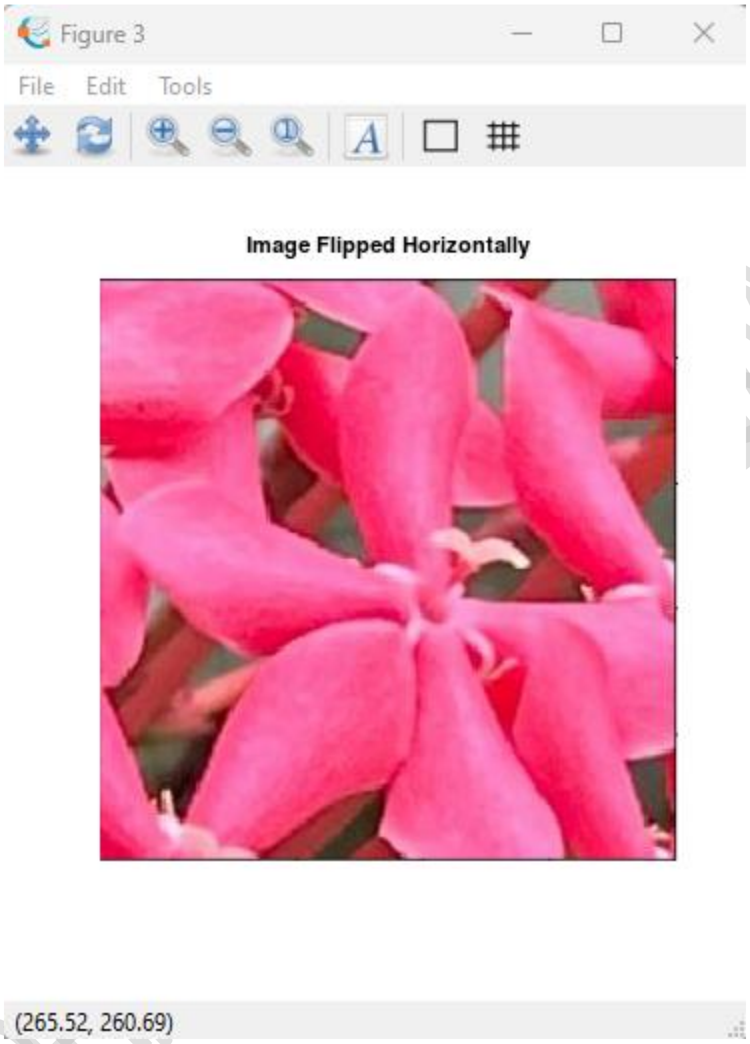


Figure 3
File Edit Tools
Image Flipped Horizontally
(265.52, 260.69)

MATLAB

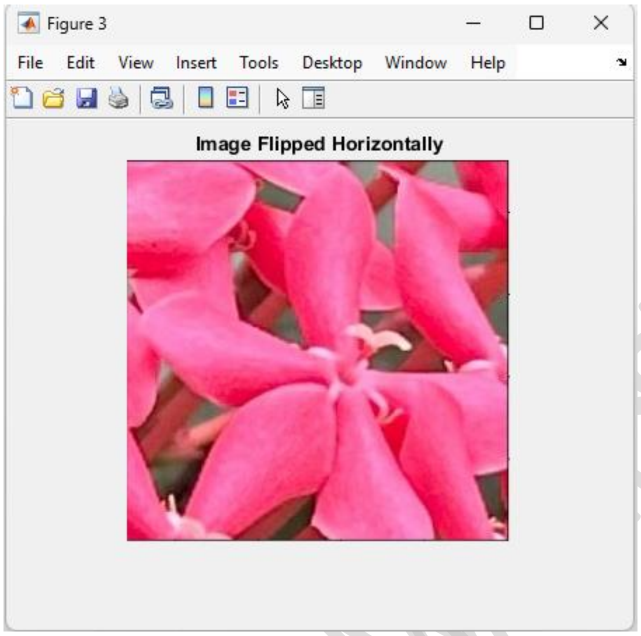


Figure 3
File Edit View Insert Tools Desktop Window Help
Image Flipped Horizontally

PYTHON

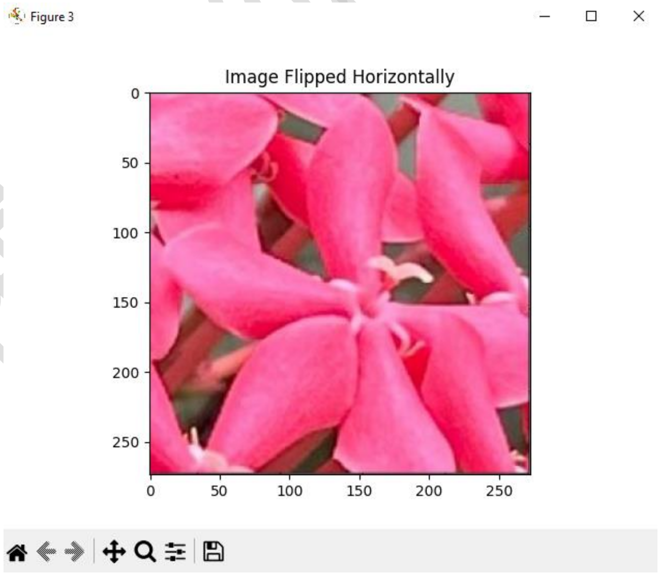


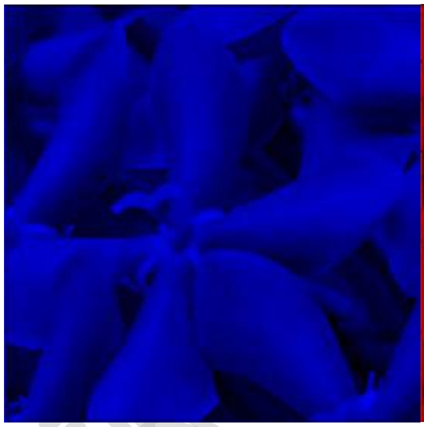
Figure 3: Image flipped horizontally

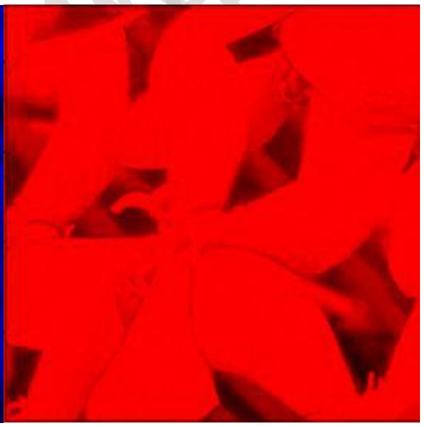
These codes perform the following:

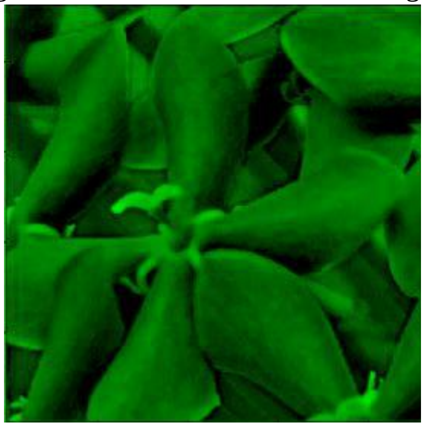
1. Reads an image using imread.
2. Displays the image using imshow.
3. Gets the image dimensions (rows, columns, color channels) using size and displays them.
4. Checks the color model (grayscale or RGB) based on the number of channels.
5. Accesses the value of a specific pixel (center pixel in this case).Performs a basic arithmetic operation (adding a constant value to all pixels) to brighten theimage.
6. Performs a basic geometric operation (flipping the image horizontally) using fliplr.

Parameter Modification

- Try displaying individual color channels for RGB images (e.g., imshow(img(:,:,1)) for red channel).


img_blue


img_red


img_green

- Experiment with different arithmetic operations (subtraction, multiplication).

brightened_img

darkened_image

- Explore other geometric operations like image rotation (imrotate).

flipped_img

2. Visualize the results, analyze and interpret:

Visualizing the results of the image manipulation algorithms reveals significant changes in the image attributes. The brightening algorithm increased the pixel intensity, resulting in a visibly lighter image, while the horizontal flip mirrored the image, providing a reversed perspective. Both transformations were effectively demonstrated through clear and labeled figures.

Analyzing the brightened image, the algorithm successfully enhanced the overall luminance without introducing noise or distortion, indicating its robustness in preserving image quality. This transformation can be particularly useful in enhancing underexposed images, making the details more discernible. Similarly, the horizontal flip effectively changed the image orientation, maintaining the original image resolution and detail, showcasing the reliability of the flipping algorithm.

Interpreting the results, the applied algorithms demonstrated their effectiveness in achieving the desired outcomes of brightening and flipping the image. The brightening algorithm was efficient in improving image visibility, while the horizontal flip provided a straightforward method for altering image orientation. These transformations are valuable tools in various image processing applications, highlighting the practical benefits of the implemented techniques.

## IV. Conclusion

In conclusion, this laboratory activity successfully demonstrated the fundamental techniques of image acquisition, representation, and manipulation using MATLAB/Octave and Python. Through the various tasks, we were able to read and display images, determine image dimensions and color models, and perform basic arithmetic and geometric operations on the images. The brightening and horizontal flipping operations illustrated how simple manipulations could alter the visual properties of an image. The consistency of results across different programming environments highlights the robustness of the techniques used. This exercise not only reinforced our understanding of image processing concepts but also provided practical experience in using popular tools for digital image manipulation.

## References

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