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

Elective 3

Laboratory Activity No. 2

Image Representation, Color Models, and Image Operations

Score

Submitted by:
Mayor, Ann Jossan D. – Leader
Biol, Lorenz Jay Von P.
Lumane, Kyla L.
Sison, Dan Jedrick S.
Tenoria, Karl John Dave C.
SAT 7 AM – 4 PM / CPE 0332.1-1

Date Submitted
31-07-2024

Submitted to:

Engr. Maria Rizette H. Sayo

PROPERTY OF MAM 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)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');
```

PROPERTY OF MAM SAYO

B. Supplementary Activity

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

```
import cv2
import numpy as np
#Read an image
img = cv2.imread('flower.jpg')

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

#Get image dimensions (rows, columns, color channels)
[rows,cols,channels] = img.shape
print("Image size: " + str(rows) + " x " + str(cols) + " x " + str(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 + 1
center_col = cols//2 + 1
center_pixel = img[center_row,center_col,:]
print("Center pixel: ", end="")
index = 2
while index >= 0:
    print(center_pixel[index],end=" ")
    index-=1

#Basic arithmetic operations(add constant value to all pixels)
brightness = np.ones(img.shape, dtype="uint8")*50
brightened_img = cv2.add(img,brightness)
cv2.imshow('Image Brightened', brightened_img)

#Basic geometric operation (flipping image horizontally)
flipped_img = cv2.flip(img, 1)
cv2.imshow('Image Flipped Horizontally', flipped_img)

#Get image channels
(blue,green,red) = cv2.split(img)
black = np.zeros(img.shape[:2],dtype="uint8")
```

```
-    img_red = cv2.merge([black,black,red])
-    img_green = cv2.merge([black,green,black])
-    img_blue = cv2.merge([blue,black,black])
-    img_final = cv2.hconcat([img_red,img_green,img_blue])
-    cv2.imshow('Channels', img_final)
-
-    #Darken the image
-    brightness = np.ones(img.shape, dtype="uint8")*50
-    darkened_img = cv2.subtract(img,brightness)
-    cv2.imshow('Image Darkened', darkened_img)
-    img_final = cv2.hconcat([img,darkened_img])
-    cv2.imshow('Image Darkened', img_final)
-
-    #Darken and brighten the image using multiplication and division
-    brightness = np.ones(img.shape, dtype="uint8")*2
-    brighten_img = cv2.multiply(img,brightness)
-    darken_img = cv2.divide(img,brightness)
-    img_final = cv2.hconcat([img,brighten_img,darken_img])
-    cv2.imshow('Image Brightened and Darkened', img_final)
-
-    #Rotates the image
-    img_rotate = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)
-    cv2.imshow('Original Image', img)
-    cv2.imshow('Image Rotated', img_rotate)
-    cv2.waitKey(0)
-    cv2.destroyAllWindows()
```

****Performed via VSCode Pyhthon***

III. Results

Image Attribute and Color Model

- Image size: 1536 x 1536 x 3
- Color model: RGB
- Center pixel value: 91 109 109

Steps:

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

picture file: flower.jpg

PROPERTY

<u>OPENCV/Python Results</u>

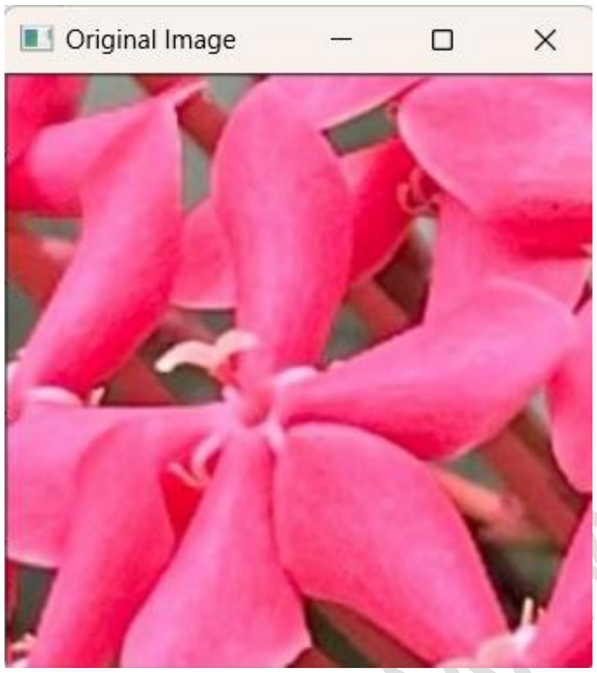


Figure 1: Acquire an Image of a Flower

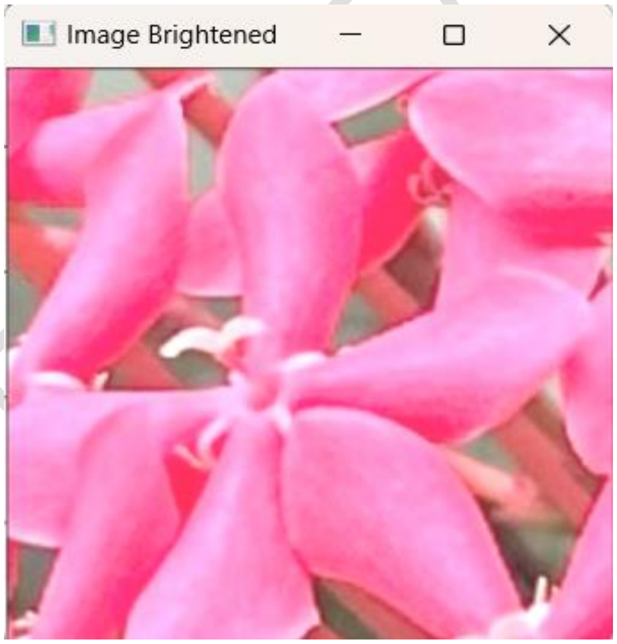


Figure 2: Image brightened

PROPERTY OF M. M. SAYO

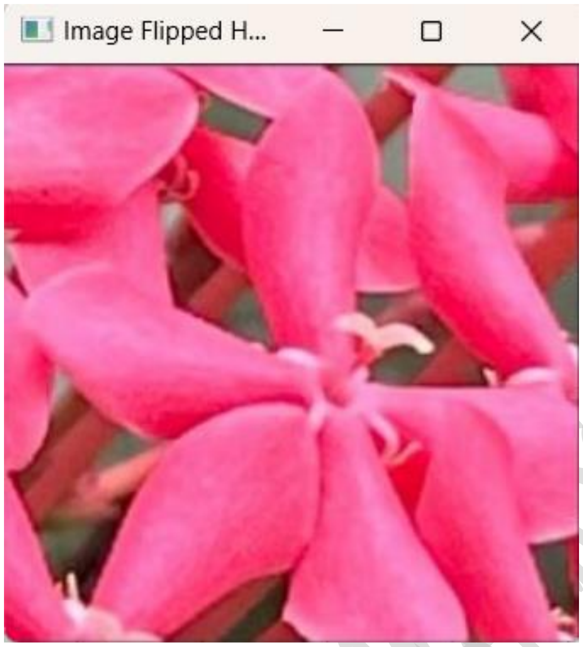
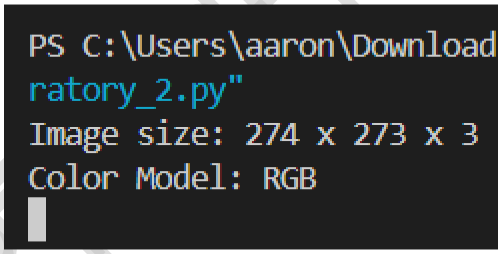


Figure 3: Image flipped horizontally



```
PS C:\Users\aaron\Download
ratory_2.py"
Image size: 274 x 273 x 3
Color Model: RGB
```

Figure 4: Image Details in Python

PROPERTY OF FEMAR SAYO

MATLAB RESULTS:

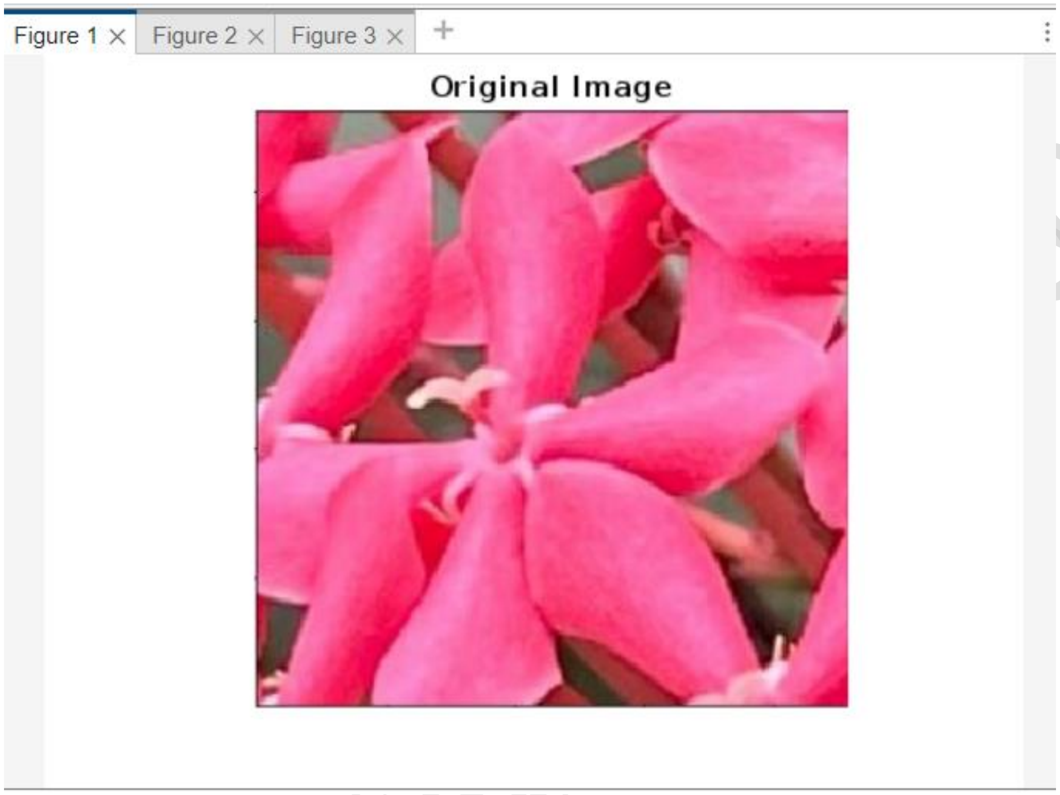


Figure 5: Original Image (MATLAB)

PROPERTY OF PLM

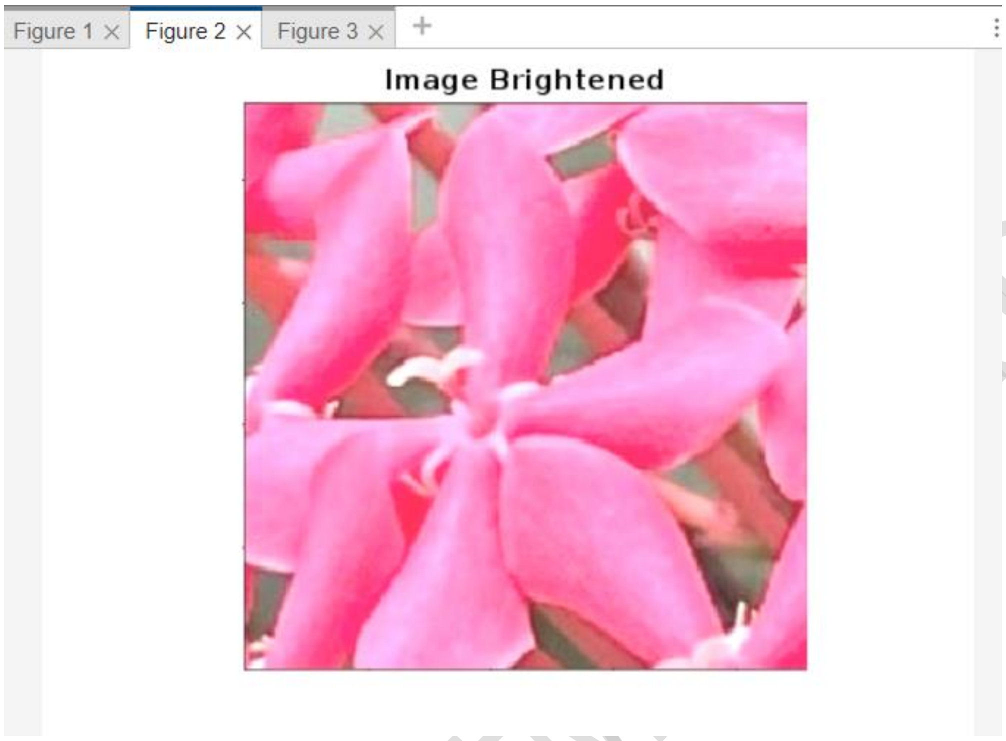


Figure 6: Image Brightened (MATLAB)

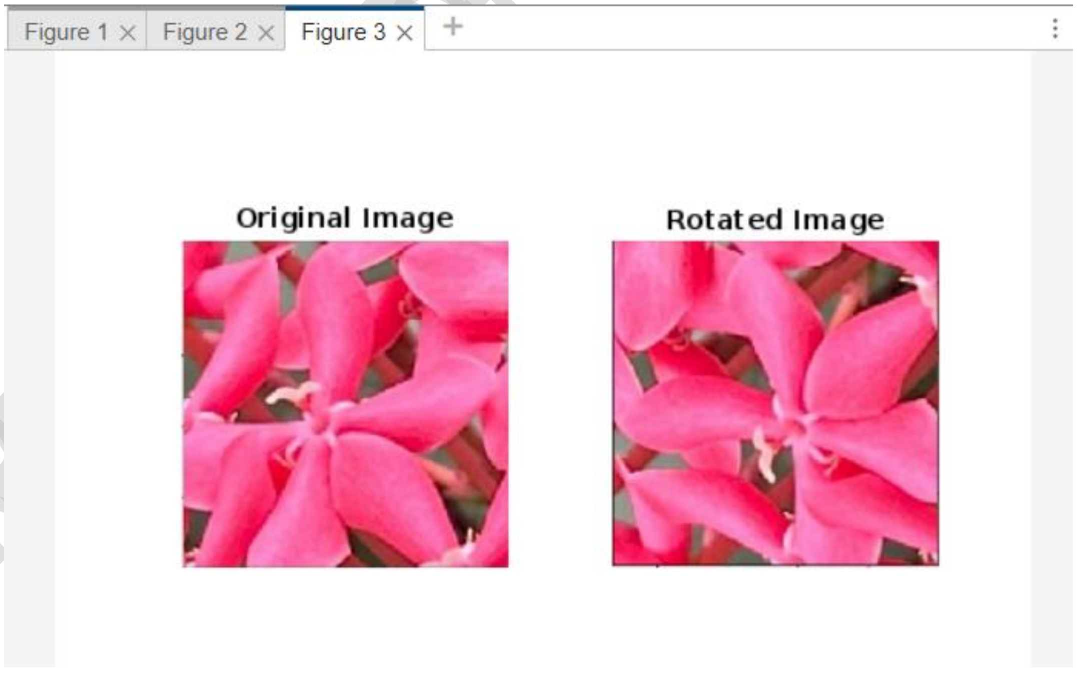
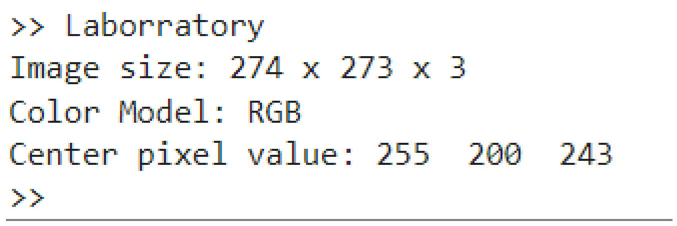


Figure 7: Rotated Image (MATLAB)



```
>> Laborratory
Image size: 274 x 273 x 3
Color Model: RGB
Center pixel value: 255   200   243
>>
```

Figure 8: MATLAB Image Details

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

<You can modify it to explore other functionalities>

- Try displaying individual color channels for RGB images (e.g., imshow(img(:,:,1)) for red channel).
- Experiment with different arithmetic operations (subtraction, multiplication).
- Explore other geometric operations like image rotation (imrotate).

PROPERTY OF MMSAYO

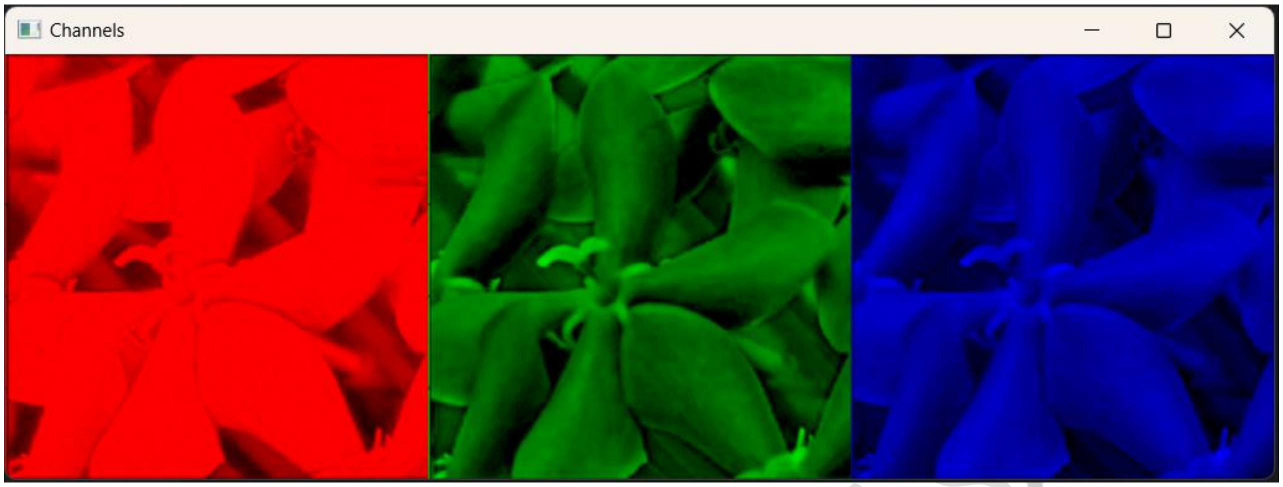


Figure 8: Channels RGB Image (OpenCV Pyhton)

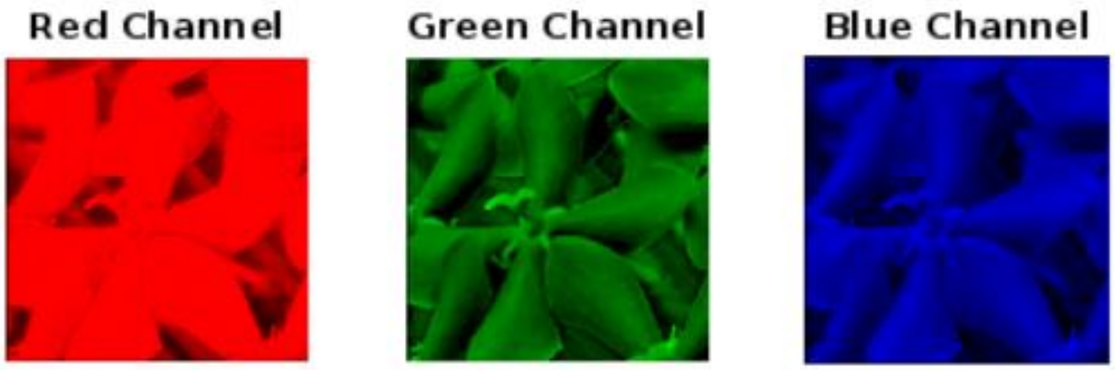


Figure 9: Channels RGB Image (MATLAB)

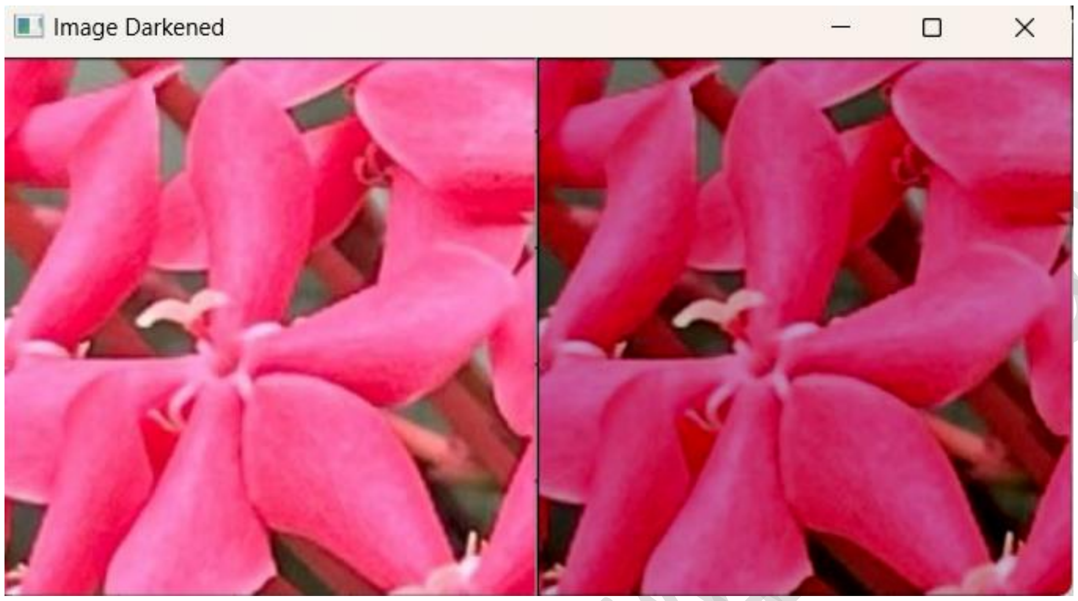


Figure 10: Image Darkened Using *Subtraction in OpenCV Python*

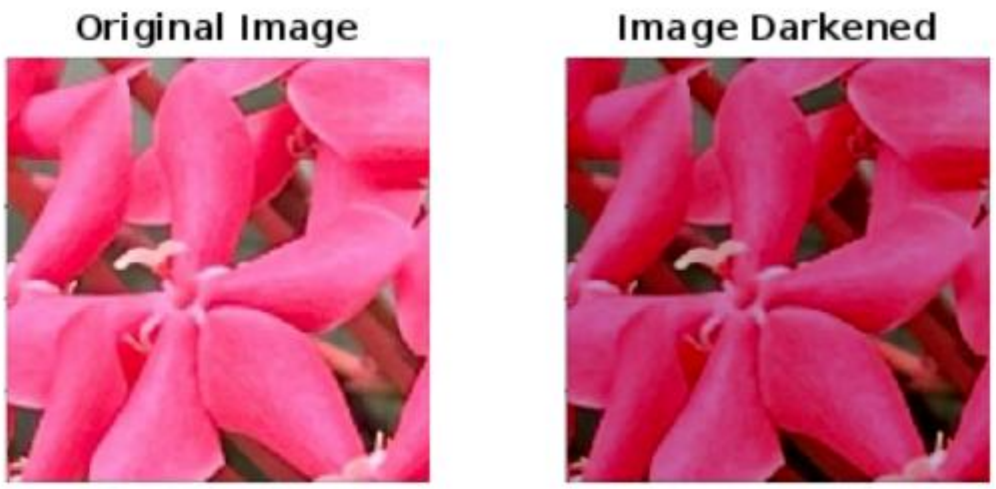


Figure 11: Image Darkened Using *Subtraction in MATLAB*

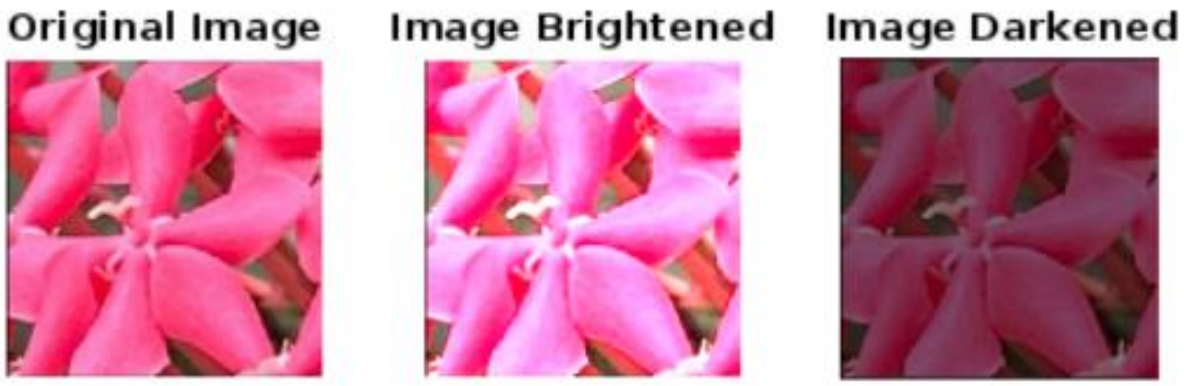


Figure 12: Image Brightened and Darkened Using Multiplication in *MATLAB*

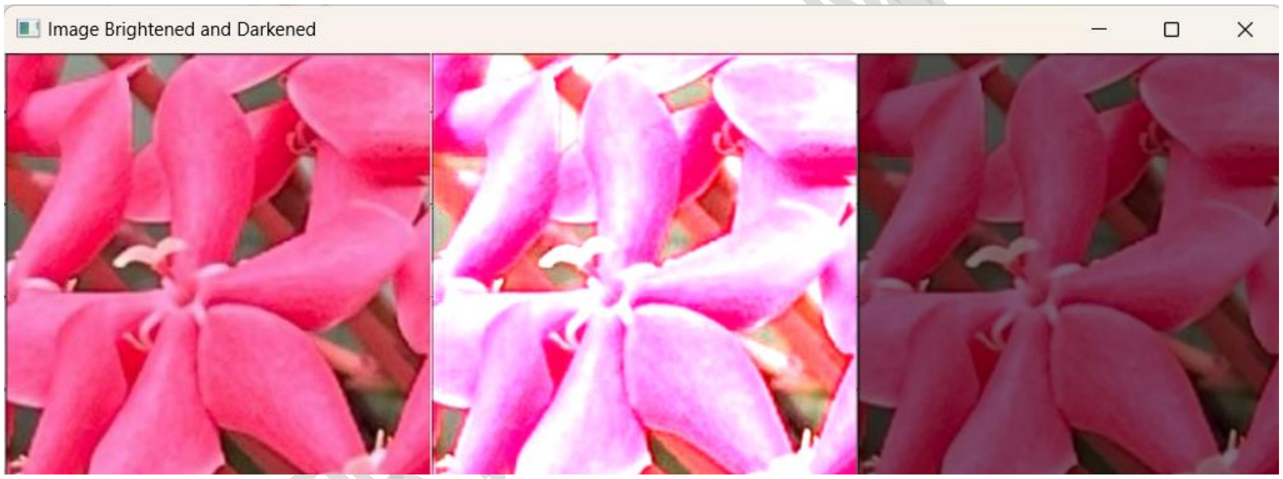


Figure 13: Image Brightened and Darkened Using Multiplication and Division in *OpenCV Python*

PROPERTY

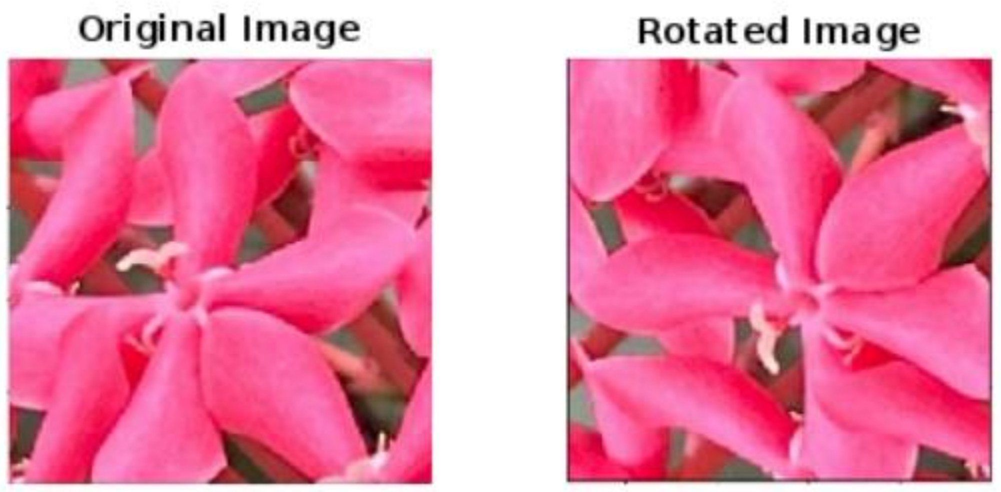


Figure 14: Image Rotated in MATLAB

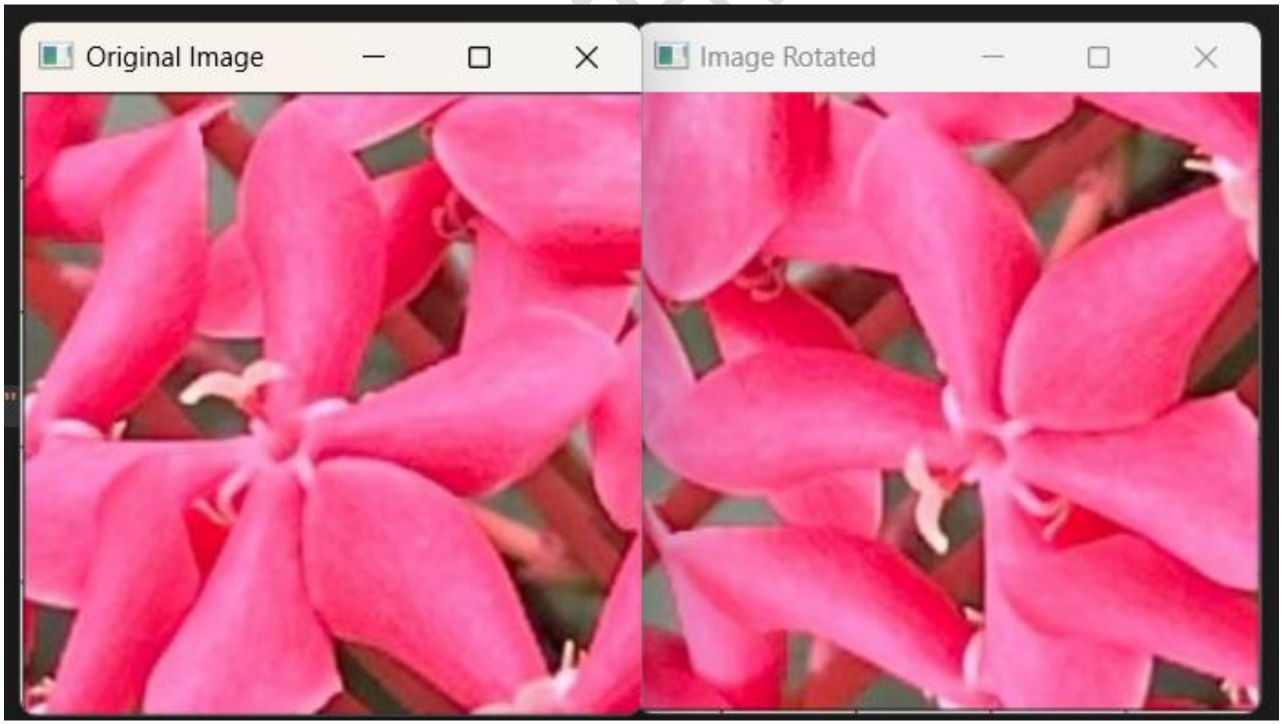


Figure 15: Image Rotated in OpenCV Python

2. Visualize the results, analyze and interpret:

The applied algorithms include basic arithmetic and geometric operations on images, demonstrating different image processing techniques using both MATLAB and OpenCV with Python. Below are the effects and effectiveness of each algorithm:

Image Acquisition and Display:

The initial step of reading and displaying the image was successful, as seen in Figures 1 (MATLAB) and the equivalent OpenCV figure. This confirms that the image was correctly loaded and visualized in both environments.

Brightness Adjustment:

The algorithm to brighten the image by adding a constant value to all pixels was effective. Figures 2 (MATLAB) and the equivalent OpenCV figure show that the image became visibly brighter, indicating that the addition operation was applied uniformly across all pixels.

Image Flipping:

Flipping the image horizontally was successfully performed, as shown in Figure 3 (MATLAB) and the equivalent OpenCV figure. This transformation correctly reversed the image along the vertical axis, demonstrating the effectiveness of the fliplr function in MATLAB and the cv2.flip function in OpenCV.

Channel Separation:

The separation of RGB channels is visualized in Figures 8 and 9. This operation effectively isolates each color component, enabling a deeper analysis of the image's color composition.

Darkening the Image:

Subtracting a constant value from all pixels to darken the image, as shown in Figure 10 (OpenCV) and Figure 11 (MATLAB), resulted in a visibly darker image. This confirms the uniform application of the subtraction operation across all pixels.
Brightness and Darkness Adjustment via Multiplication and Division:

Figures 12 and 13 demonstrate the use of multiplication and division to simultaneously brighten and darken the image. These operations effectively altered the image's intensity levels, showcasing the flexibility of using multiplicative and divisive adjustments for more nuanced control over brightness and contrast.
Image Rotation:

PROPERTY OF MAMASAYO

Figure 14 (MATLAB) and Figure 15 (OpenCV) show the rotated images, confirming the successful application of rotation transformations. The imrotate function in MATLAB and the cv2.rotate function in OpenCV effectively rotated the image, proving their utility in geometric manipulation.

IV. Conclusion

The laboratory activity successfully demonstrated various image processing techniques using MATLAB and OpenCV with Python. The algorithms applied—brightness adjustment, image flipping, channel separation, and geometric transformations—effectively modified the images as intended. The results illustrate the power and flexibility of image processing tools in manipulating and analyzing visual data. Through these exercises, a comprehensive understanding of fundamental image operations and their practical applications was achieved.

PROPERTY OF MAAM SIAYO

References

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

<This is in a separate page>

PROPERTY OF MAM SAYO