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:

Group 1
Antoni, Austine C.
Atencia, Dan Eric B.
Bauyon, Jared Randalle B.
Jao, Steven Rey C.
Leonardo, Aibel Renzo T.
Saturday, 7AM-4PM / 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)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');
```

B. Supplementary Activity

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

```python
import cv2

# Read an image
img = cv2.imread('flower.jpg')

# Get image dimensions (rows, columns, color channels)
dim = img.shape

# Access individual pixels (example: center pixel)
centerRow = dim[0] // 2
centerCol = dim[1] // 2

centerPix = img[centerRow, centerCol]

# Declaring a value for brightness
constVal = 50

# An arithmetic function (+) will increase the brightness by 50 to all pixels
brightImg = cv2.add(img, constVal)

# Flip an image horizontally (1)
flippedImg = cv2.flip(img, 1)

# Display dimensions (rows, columns, color channels)
print("\nImage size: ", dim[0], " x ", dim[1], " x ", dim[2])
```

```
# Check and display color model
if dim[2] == 1:
    # If the third index is equal to 1, then, the image is in grayscale
    print("Color Model: Grayscale")
else:
    # Access 3rd index to check if the image is BGR or grayscale
    print("Color Model: BGR")
# Print the center pixel
print("Center pixel: ", str(centerPix))

# Display the image
cv2.imshow('Original Image', img)
cv2.imshow('Image Brightened', brightImg)
cv2.imshow('Image Flipped Horizontally', flippedImg)
cv2.waitKey(0)
```

III. Results

Image Attribute and Color Model

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

a. MATLAB results:

```
>> Lab_Act_2_MATLAB
Image size: 274 x 273 x 3
Color Model: RGB
Center pixel value: 255  200  243
```

Figure 1: MATLAB command window result

Figure 2: Acquire an Image of a Flower

Figure 3: Image brightened

Figure 4: Image flipped horizontally

b. OpenCV results:

```
PS C:\Users\Austine\Desktop\Lab 2> & C:/Python/python.exe "c:/Users/Austine/Desktop/Lab 2/Lab_2_OpenCV.py"

Image size:  274  x  273  x  3
Color Model: BGR
Center pixel:  [243 200 255]
```

Figure 5: OpenCV terminal output

Figure 6: Acquire an Image of a Flower

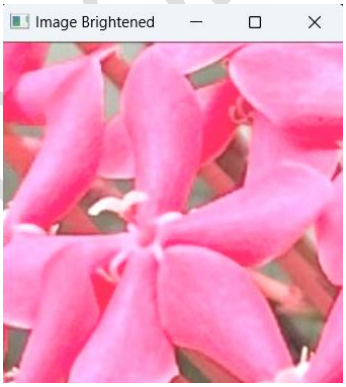

Figure 7: Image brightened

Figure 8: 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 the image.
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).
- Experiment with different arithmetic operations (subtraction, multiplication).
- Explore other geometric operations like image rotation (imrotate).

c. MATLAB (parameters modified):

Code:

```
% Read an image
img = imread('flower.jpg');
figure(1);
imshow(img);
title('Original Image');

% Using different arithmetic operations (-, *)
subImg = img - 50;
mulImg = img * 50;

% Geometric operations (imrotate)
angle = 45;
rotImg = imrotate(img, angle);

% Displaying individual color channels for RGB images
% Red channel
figure(2);
imshow(img(:, :, 1));
title ('Red Channel');

% Green channel
figure(3);
imshow(img(:, :, 2));
title('Green Channel');

% Blue channel
figure(4);
imshow(img(:, :, 3));
title('Blue Channel');

% Display rotated image
figure(5);
imshow(rotImg);
title('Rotated Image');

% Display subtracted image
figure(6);
```

```
imshow(subImg);
title('Subtracted Image');

% Display multiplied image
figure(7);
imshow(mulImg);
title('Multiplied Image');
```

Figure 9: Acquire an Image of a Flower

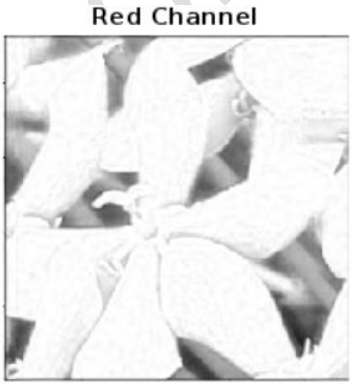

Figure 10: Image utilizing the Red Channel

Figure 11: Image utilizing the Green Channel

Figure 12: Image utilizing the Blue Channel

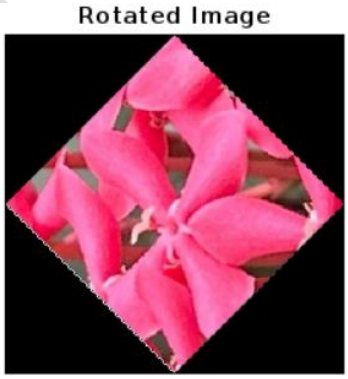

Figure 13: Rotated Image by 45 degrees

Figure 14: Subtracted Image

Figure 15: Multiplied Image

d. OpenCV (modified):

Code:

```
import cv2

# Read the image
img = cv2.imread('flower.jpg')

# Get image dimensions and center
height, width = img.shape[:2]
centerX, centerY = width // 2, height // 2

# Rotate the image by an angle
angle = 45
scale = 0.7 # 0.7 is used to fit the image inside the window
rotMatrix = cv2.getRotationMatrix2D((centerX, centerY), angle, scale)
rotImg = cv2.warpAffine(img, rotMatrix, (width, height))

# Create a darkened image by subtracting a constant value
constVal = 50
darkImg = cv2.subtract(img, constVal)

# Create a multiplied image
multipliedImg = cv2.multiply(img, 50)

# Extract and display color channels (BGR is the default in OpenCV)
redChannel = img[:, :, 2]
greenChannel = img[:, :, 1]
blueChannel = img[:, :, 0]

# Display images
cv2.imshow('Original Image', img)
cv2.imshow('Red Channel Image', redChannel)
cv2.imshow('Green Channel Image', greenChannel)
cv2.imshow('Blue Channel Image', blueChannel)
cv2.imshow('Rotated Image', rotImg)
cv2.imshow('Subtracted Image', darkImg)
cv2.imshow('Multiplied Image', multipliedImg)
```

```
cv2.waitKey(0)
```

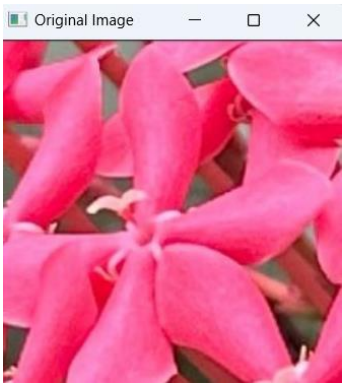

Figure 16: Acquire an Image of a Flower

Figure 17: Image utilizing the Red Channel

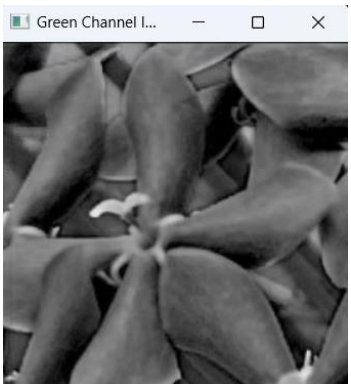

Figure 18: Image utilizing the Green Channel

Figure 19: Image utilizing the Blue Channel

Figure 20: Rotated Image by 45 degrees

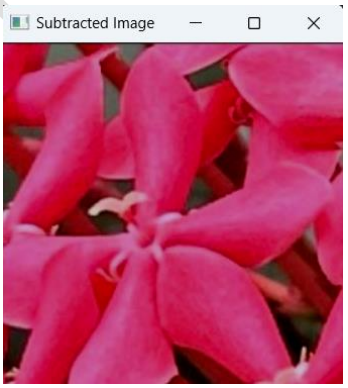

Figure 21: Subtracted Image

Figure 22: Multiplied Image

2. Visualize the results, analyze and interpret:

MATLAB:
The MATLAB Lab Task 1 code starts by reading an image file named 'flower.jpg' into the variable img and displaying it in a figure window called 'Original Image'. It collects and shows the image dimensions (number of rows, columns, and color channels), determining if the image is grayscale or RGB based on the number of channels. If the color channel is equal to 1, then the image is grayscale. Otherwise, if it is 3, then it is RGB. The algorithm then determines the coordinates of the central pixel and extracts its value by dividing the columns and rows by 2, which is displayed in the command window terminal. Following that, the code does a basic arithmetic operation to raise the brightness of the image by adding 50 to each pixel value before displaying the brightened image in a new figure window. Finally, the code uses the fliplr function to flip the image horizontally, which is then displayed in another figure window titled 'Image Flipped Horizontally'.

This MATLAB Lab Task 2 code begins by reading the picture file 'flower.jpg' into the variable img and displaying it under the caption 'Original picture'. Arithmetic operations are then applied to the image: subImg subtracts 50 from each pixel value which darkens the image, and mulImg multiplies each pixel value by 50 which immensely brightens the image. imrotate rotates the image by 45 degrees and stores the result in rotImg. The code additionally separates and displays each color channel of the RGB image: the red channel (img(:,:, 1)) in one picture, the green channel (img(:,:, 2)) in another, and the blue channel (img(:,:, 3)) in a third. Finally, it displays the rotated image, the image with subtracted pixel values, and the image with multiplied pixel values, each in their own figure.

OpenCV:
This OpenCV Lab Task 1 Python code reads and manipulates a picture using the OpenCV package. It begins by importing the cv2 module, which then reads an image file called 'flower.jpg' into the variable img. The shape attribute is used to extract the image's dimensions (rows, columns, and color channels), which are then stored in variable dim. The center pixel's coordinates are computed by half the number of rows and columns, and the pixel value at this center point is retrieved and stored in centerPix. The brightness adjustment is set at a constant value of 50. The image is then brightened by applying this constant value to all pixel values with the cv2.add function, resulting in the new image brightImg. The picture is also flipped horizontally with the cv2.flip function and a flip code of 1, resulting flippedImg. The dimensions and color model of the image are printed on the console. If the image has only one-color channel, it is considered grayscale; otherwise, it is BGR which has 3 color channels. The central pixel value is also printed. Finally, the original image, the brightened image, and the horizontally flipped image are displayed in separate windows using cv2.imshow, and the program closes the windows by waiting for a user input with cv2.waitKey(0).

In Task 2, the image is initially read into the variable img. The image's dimensions (height and width) are retrieved, and the center coordinates are determined. To guarantee that the image fits within the display

window, it is rotated 45 degrees around its center and scaled by 0.7. The rotation is accomplished by using the cv2.getRotationMatrix2D function to obtain the rotation matrix, which is then combined with cv2.warpAffine to produce the rotated image rotImg. To create a darkened version of the picture, the cv2.subtract function subtracts a constant value of 50 from each pixel, resulting in darkImg. Additionally, an image with pixel values multiplied by 50 is created using cv2.multiply and saved in multipliedImg—this will increasingly brighten the image. The code extracts and displays the image's color channels (red, green, and blue) from OpenCV's default BGR format: the blue channel with img[:,:, 0], the green channel with img[:,:, 1], and the red channel is accessible with img[:,:, 2]. Finally, using cv2.imshow, the program displays the original image, the individual color channels, the rotated image, the darkened image, and the multiplied image in separate windows before closing the windows with cv2.waitKey(0).

IV. Conclusion

In this laboratory activity, we learned additional ways to manipulate an image using MATLAB and OpenCV via Python. Some essential image manipulations we performed using MATLAB and OpenCV were increasing the brightness of an image by using the addition operation, accessing each pixel using the image's dimensions, and flipping an image in task 1. As well as displaying an RGB image's individual color channels (red, green, and blue), and using arithmetic and geometric operations to alter an image in task 2. We also learned some additional commands to display information about an image such as its dimensions and color channels, whether it is grayscale or RGB, and its center pixel value. Both MATLAB and OpenCV were straightforward and have had similar outputs when typing a syntax to perform a certain image manipulation.

References

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