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

Elective 3

Laboratory Activity No. 1

Image Acquisition and Manipulation

Score

Submitted by:

Alambra, Joseph Nathaniel

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

Date Submitted

24-07-2024

Submitted to:

Engr. Maria Rizette H. Sayo

I. Objectives

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

- Acquire the image.
- Rotate the image by 30 degrees.
- Flip the image horizontally.

II. Methods

A. Perform a task given in the presentation

- Copy and paste your MATLAB code

```
% Read the image
img = imread("C:/Users/Elitebook 840 G7/Documents/3rd Year - 3rd
Sem/Elective (Laboratory)/flower.jpg");
% Rotate by 45 degrees
rotated_img = imrotate(img, 30);

% Flip horizontally
flipped_img = fliplr(rotated_img);

% Display results
figure(1);
plot(1,1);
imshow(img);
title('Original Image');
figure(2);
plot(1,1);
imshow(rotated_img);
title('Rotated 30°'); figure(3); plot(1,1);
imshow(flipped_img); title('Rotated & Flipped');
```

B. Supplementary Activity

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

Source Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read the image
img = cv2.imread("C:/Users/Elitebook 840 G7/Documents/3rd Year - 3rd
Sem/Elective (Laboratory)/flower.jpg")

# Rotate by 30 degrees
(h, w) = img.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, 30, 1.0)
rotated_img = cv2.warpAffine(img, M, (w, h))

# Flip horizontally
flipped_img = cv2.flip(rotated_img, 1)

# Convert BGR to RGB for displaying using matplotlib
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rotated_img_rgb = cv2.cvtColor(rotated_img, cv2.COLOR_BGR2RGB)
flipped_img_rgb = cv2.cvtColor(flipped_img, cv2.COLOR_BGR2RGB)

# Display results
plt.figure(1)
plt.imshow(img_rgb)
plt.title('Original Image')

plt.figure(2)
plt.imshow(rotated_img_rgb)
plt.title('Rotated 30°')

plt.figure(3)
plt.imshow(flipped_img_rgb)
plt.title('Rotated & Flipped')

plt.show()
```

C. Results

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

TASK A

picture file: flower.jpg

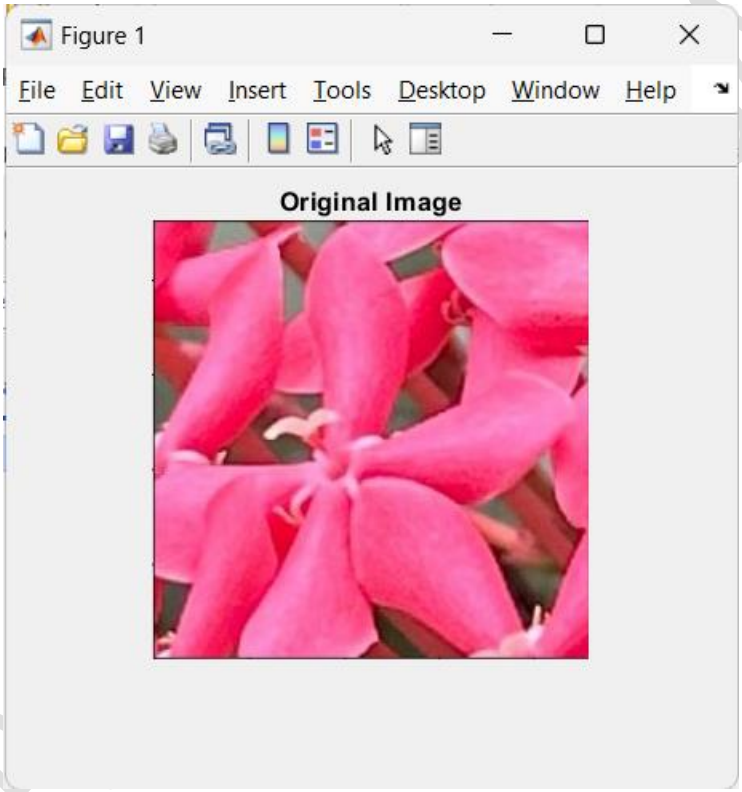

Figure 1: Original Image of a Flower

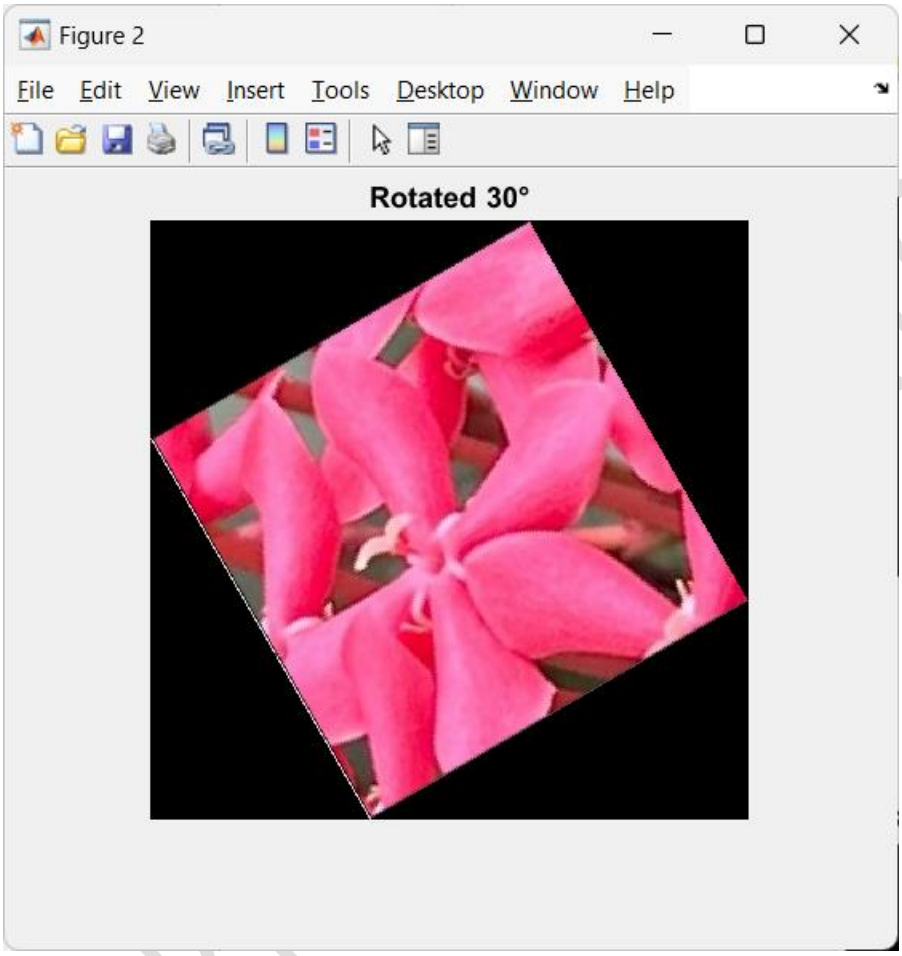

Figure 2: Flower Rotated by 30 Degrees

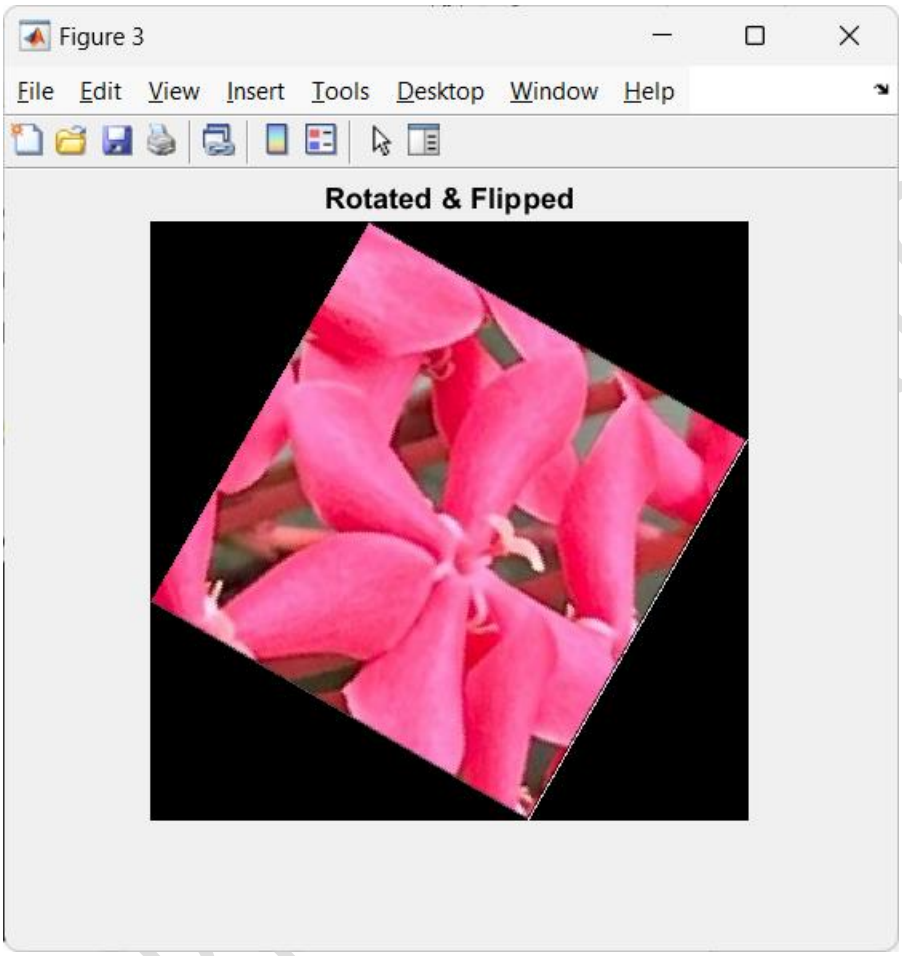

Figure 3: Flower Flip Horizontally

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

TASK B

picture file: flower.jpg

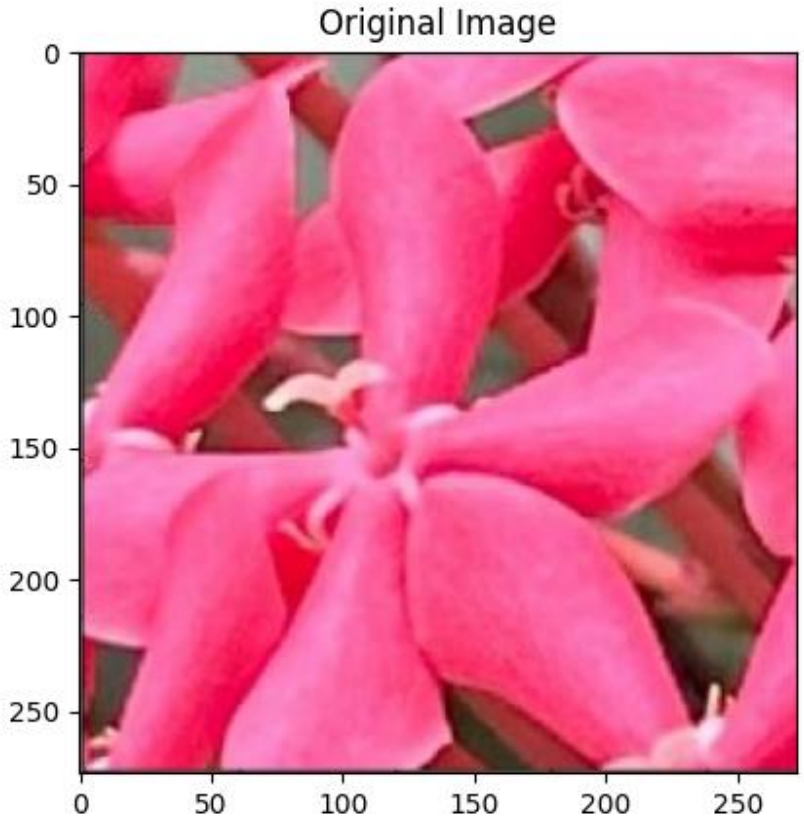

Figure 4: Original Image of Flower

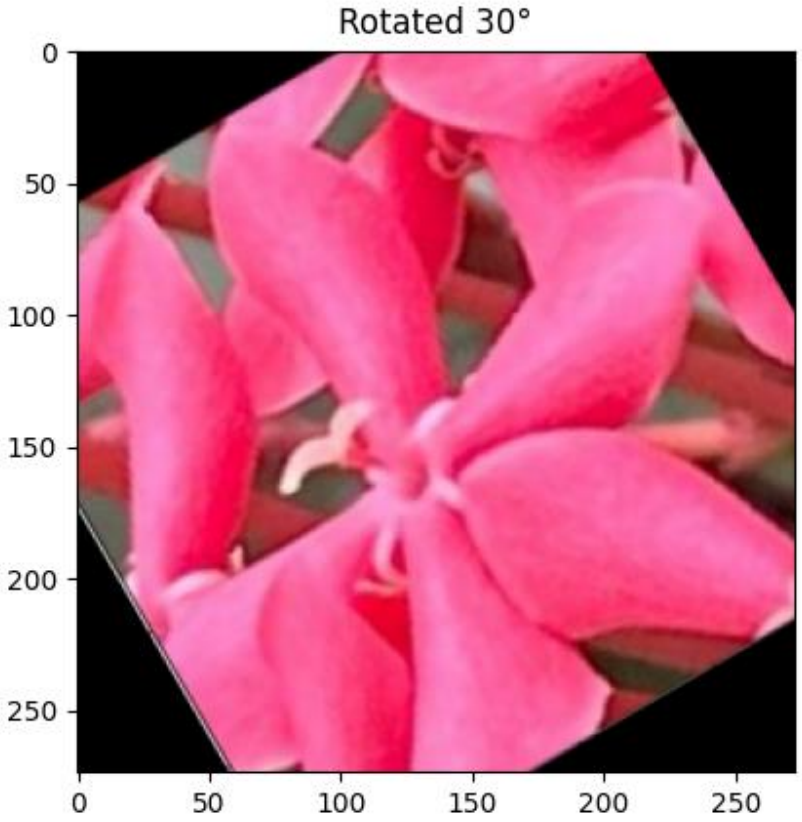

Figure 5: Flower Rotated by 30 Degrees

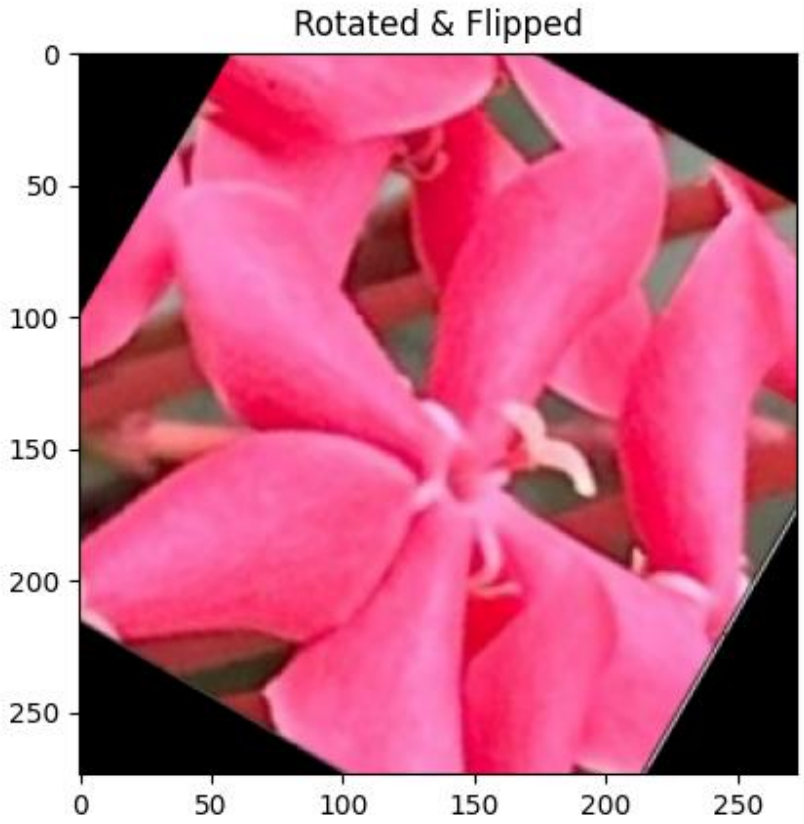

Figure 6: Flower Flip Horizontally

2. Visualize the results, analyze and interpret:

< Discuss the effects of the applied algorithm on the image and its effectiveness in achieving the desired outcome. Handwritten>

TASK A

In TASK A, the image is first rotated by 30 degrees using the `imrotate` function and then flipped horizontally using the `fliplr` function. The original image, rotated image, and rotated & flipped image are displayed sequentially. The rotation by 30 degrees alters the orientation of the image, making it tilted to the right. The subsequent horizontal flip inverts the image along the vertical axis, effectively mirroring the rotated image. The effectiveness of this approach lies in its simplicity and ease of implementation in Online MATLAB.

TASK B

In TASK B, the image is manipulated using PyCharm. The image is read using `cv2.imread` and then rotated by 30 degrees using the `cv2.getRotationMatrix2D` and `cv2.warpAffine` functions. The horizontal flip is achieved using `cv2.flip`. The images are converted from BGR to RGB format for correct color representation in Matplotlib. The original, rotated, and rotated & flipped images are displayed using Matplotlib. The algorithm applied in PyCharm is effective in achieving the desired outcome as it provides more control over the rotation center and allows for more extensive image processing capabilities.

Effectiveness Comparison

Both TASK A and TASK B successfully achieve the desired outcome of rotating and flipping the image. However, TASK B offers more precision and flexibility in image manipulation due to the comprehensive capabilities of PyCharm. Additionally, the use of Matplotlib provides superior visualization options. TASK A, while effective, is simpler and more straightforward, making it suitable for quick image manipulations in Online MATLAB.

To sum it up, the applied algorithms in both tasks are effective in achieving the desired image transformations, with TASK B providing additional advantages in terms of control and visualization.

IV. Conclusion

In this laboratory activity, I successfully acquired and manipulated images using both Online MATLAB and OpenCV in Python Using PyCharm. The initial step involved reading the image file, which I seamlessly executed using the `imread` function in MATLAB and `cv2.imread` in PyCharm. This step laid the groundwork for subsequent image processing tasks, demonstrating the ease of image acquisition in both environments.

My next objective was to rotate the image by 30 degrees. In MATLAB, I achieved this using the `imrotate` function, while in PyCharm, I used the `cv2.getRotationMatrix2D` and `cv2.warpAffine` functions. Both methods effectively altered the orientation of the image, providing clear visual confirmation of the rotation. The precise control over the rotation center in PyCharm added an extra layer of flexibility, highlighting its advanced capabilities in image processing.

Finally, I flipped the images horizontally to complete the transformation. MATLAB's `fliplr` function and PyCharm's `cv2.flip` function were used to mirror the images along the vertical axis. Both approaches accomplished the task efficiently, with the results displayed using MATLAB's `imshow` and Matplotlib's `imshow` functions respectively. I met this objective with equal effectiveness in both environments, showcasing the robustness of the applied algorithms in achieving the desired image transformations.

References

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

[2] Simplilearn. (2024, August 5). What is image processing : overview, applications, benefits, and more. Simplilearn.com. https://www.simplilearn.com/image-processing-article

[3] Joshi, A. (2022, June 15). Image manipulation: What’s okay and not okay and who addresses it? Editage Insights. https://www.editage.com/insights/image-manipulation-whats-okay-and-not-okay-and-who-addresses-it/

[4] Moore, R. (1981). Acquisition and manipulation of diagnostic images. Computers in Biology and Medicine, 11(4), 173–179. https://doi.org/10.1016/s0010-4825(81)80019-4