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:

Lumane, Kyla L.

SAT 7AM – 4PM / CPE 0332.1-1

Date Submitted

27-07-2024

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('E:\PLM CET SUBJECTS\Digital Image
Processing\flower.jpg');
% Rotate by 45 degrees
rotated_img = imrotate(img, 45);

% 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 45°'); 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.
- **CODE:**

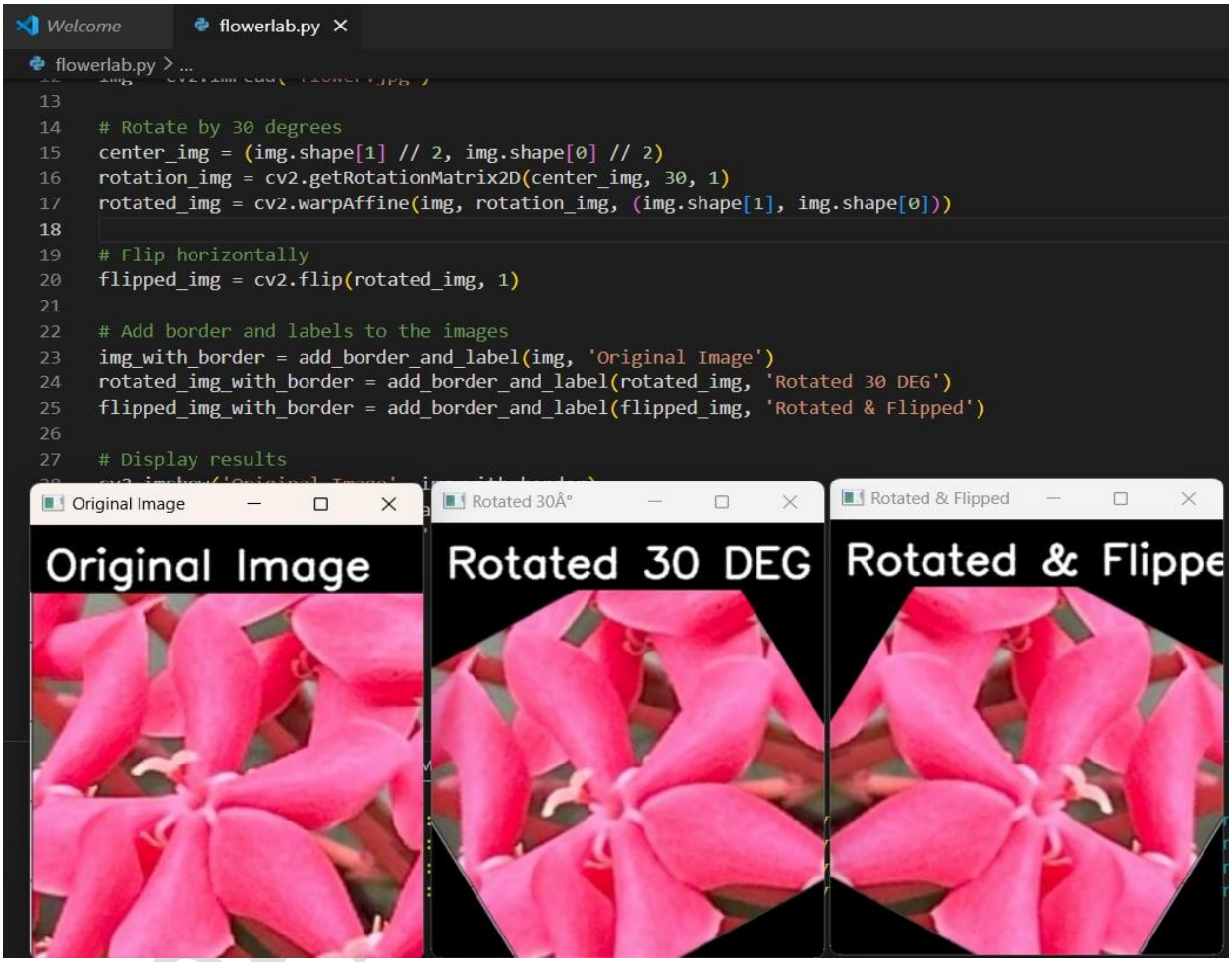
```
- import cv2
-
- # Function to add border and label to the image
- def add_border_and_label(img, label, border_size=50):
-     # Add border to the image
-     bordered_img = cv2.copyMakeBorder(img, border_size, 0, 0, 0,
  cv2.BORDER_CONSTANT, value=(0, 0, 0))
-     # Add label to the border
-     cv2.putText(bordered_img, label, (10, border_size - 10),
  cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)
-     return bordered_img
-
- # Read the image
- img = cv2.imread("flower.jpg")
-
- # Rotate by 30 degrees
- center_img = (img.shape[1] // 2, img.shape[0] // 2)
- rotation_img = cv2.getRotationMatrix2D(center_img, 30, 1)
- rotated_img = cv2.warpAffine(img, rotation_img, (img.shape[1],
  img.shape[0]))
-
- # Flip horizontally
- flipped_img = cv2.flip(rotated_img, 1)
-
- # Add border and labels to the images
- img_with_border = add_border_and_label(img, 'Original Image')
- rotated_img_with_border = add_border_and_label(rotated_img, 'Rotated
  30 DEG')
- flipped_img_with_border = add_border_and_label(flipped_img, 'Rotated
  & Flipped')
- # Display results
- cv2.imshow('Original Image', img_with_border)
- cv2.imshow('Rotated 30°', rotated_img_with_border)
- cv2.imshow('Rotated & Flipped', flipped_img_with_border)
- cv2.waitKey(0)
- cv2.destroyAllWindows()
```

Using VSCode python and OpenCV Libraries installed.

C. Results

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

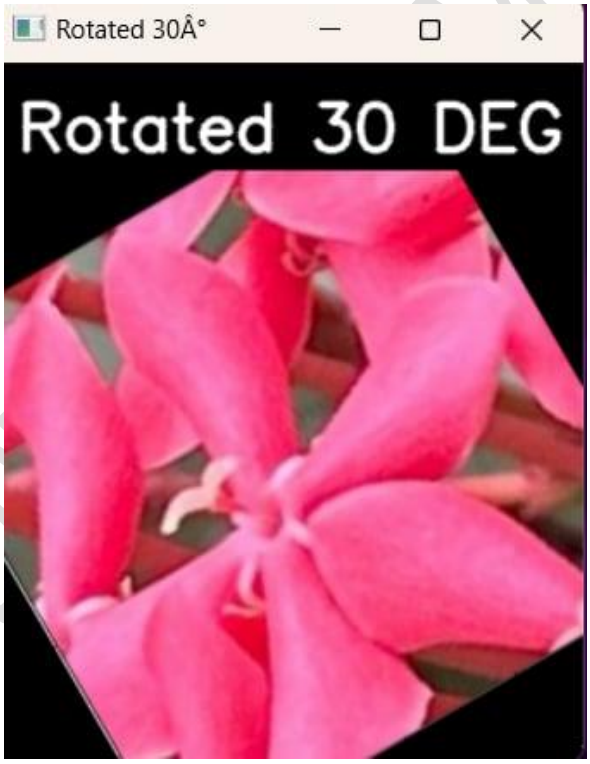
OUTPUTS:



Using VSCode python and OpenCV Libraries installed.

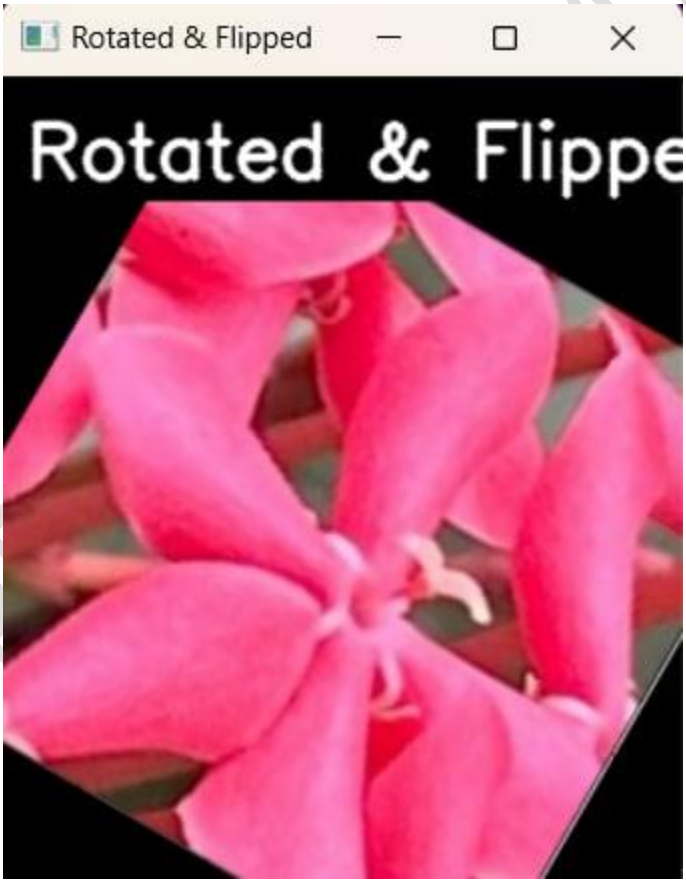
Figure 1: Acquire an Image of a Flower

Figure 2: Rotate by 30 degrees (sample)



OUTPUT

Figure 3: Flip horizontally (sample)



OUTPUT

2. Visualize the results, analyze and interpret:

Rotated Image (30 Degrees):

Effect: The image is rotated by 30 degrees around its center, changing the orientation of the flower. This rotation results in the introduction of black corners in the image where the original content is no longer present. Additionally, a black border with the label "Rotated 30 DEG" is added to the top.
Effectiveness: The rotation effectively reorients the image by the specified angle, demonstrating the capability of image transformation using OpenCV. The added border and label make it clear that this is the rotated version of the original image. The black corners are a common artifact in image rotation, indicating that the image was rotated without resizing or cropping to fit the original dimensions.

Rotated and Flipped Image:

Effect: The already rotated image is flipped horizontally, resulting in a mirror image of the rotated flower. The combined transformations significantly change the image's perspective. A black border with the label "Rotated & Flipped" is added to the top.
Effectiveness: The flipping operation is effective in mirroring the rotated image, providing a new visual perspective. The combined rotation and flipping achieve the desired transformation, showcasing the flexibility of OpenCV for complex image manipulations. The border and label help in identifying this version as both rotated and flipped, making it easy to understand the sequence of transformations applied.

Analysis of the Effects
Border and Labeling:

Effect: The addition of borders and labels does not modify the actual image content but provides a clear visual distinction and identification for each version of the image.
Effectiveness: This approach is highly effective for presentations and comparisons, allowing viewers to easily understand the modifications applied to each image.

Rotation:

Effect: Rotation changes the orientation of the image, introducing black corners due to the bounding box of the image.
Effectiveness: The rotation is effective in reorienting the image by the desired angle. The black corners are an expected artifact and can be addressed if needed by additional cropping or resizing.

Flipping:

Effect: Flipping mirrors the image along the vertical axis, providing a different view of the rotated image.
Effectiveness: The flipping operation is straightforward and effectively mirrors the image, adding to the versatility of the applied transformations.
IV. Conclusion

The applied image processing algorithm, consisting of rotation and horizontal flipping, effectively altered the orientation and perspective of the flower image. While the rotation introduced black corners, this is a common artifact and does not detract significantly from the main content of the image. The combined operations showcase the flexibility and utility of OpenCV for basic image editing tasks, achieving the desired transformations efficiently

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>