# Laboratory Exercise 1
Introduction to OpenCV

Submitted by:

## Group 1

| SO | PI | Category | Exceptional 4 | Acceptable 3 | Marginal 2 | Unacceptable 1 | Score |
|---|---|---|---|---|---|---|---|
| b | 1 | **Compliance** 30% | All procedures were followed, the output is as expected, additional related functionalities were augmented. | All procedures were followed and the output is as expected. | All procedures were followed but the output is not as expected. | Did not follow the set procedures | |
| b | 1 | **Analysis** 20% | Data interpretation is professionally written with appropriate and clear illustrations. | Data is clearly and correctly explained with proper illustrations. | Data is not clearly explained, has minor flaws, or no illustrations | There is no data explanation about the data or results. | |
| b | 1 | **Validity** 20% | The implementation uses the concepts and principles of the experiment as well as advanced topics. | The implementation uses the concepts and principles of the theory discussed for the experiment. | Implementation did not clearly express the use of theory discussed for the experiment. | There is no implementation. | |
| b | 1 | **Interpretation** 20% | The conclusion is professionally written and points the theories in the experiment and its implications in engineering. | The conclusion points to the main ideas and applications of the theory in the experiment. | The conclusion does not point out the main ideas and applications of the theory in the experiment. | There is no conclusion. | |
| | | **Format and Clarity** 10% | Follows the prescribed format, observes proper and technical grammar, and observes proper citation and referencing according to IEEE journal standards. | Follows the prescribed format, observes proper and technical grammar, and observes proper IEEE citation and referencing. | Did not follow the prescribed format, has poor grammar, or incorrect citations and references scheme. | Did not follow the prescribed format, has poor grammar, and has no citations and references. | |
| | | **TOTAL SCORE** | | | | | |

| Group Members | | | |
|---|---|---|---|
| **STUDENT NUMBER** | **NAME** | **CONTRIBUTION** | **SCORE** |
| 201113488 | CRUZ, SUZETTE L. | | |
| 201912207 | LABAO, GIO VINCENT P. | | |
| 202020002 | SALVADOR, DANIEL D. | | |
| 202020001 | SALVADOR, GABRIEL D. | | |

Submitted to:
Engr. Dexter James L. Cuaresma

Date:
02/12/2024

## OBJECTIVES

- To be familiarized with python and anaconda using PyCharm environment.
- To create a simple python program to interface camera in PyCharm using OpenCV.

## DISCUSSION

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code. (OpenCV, 2020)

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers. (OpenCV, 2020)

## MATERIALS

**Software:**

- PyCharm
- OpenCv
- Anaconda
- Python

## PROCEDURES

1. Install the following Software Applications (Latest Version)
    a. Pycharm
    b. Python
    c. Anaconda
2. Use python and Open Library to capture image and video.

    Python provides various libraries for image and video processing. One of them is OpenCV. OpenCV is a vast library that helps in providing various functions for image and video operations. With OpenCV, we can capture a video from the camera. It lets you create a video capture object which is helpful to capture videos through webcam and then you may perform desired operations on that video.

    OpenCV (Open Source Computer Vision) library. Following types of files are supported in OpenCV library:

    - Windows bitmaps – *.bmp, *.dib
    - JPEG files – *.jpeg, *.jpg
    - Portable Network Graphics – *.png
    - WebP – *.webp
    - Sun rasters – *.sr, *.ras
    - TIFF files – *.tiff, *.tif

- Raster and Vector geospatial data supported by GDAL

The steps to read and display an image in OpenCV are:

1. Read an image using imread() function.

2. Create a GUI window and display image using imshow() function.

3. Use function waitkey(0) to hold the image window on the screen by the specified number of seconds, o means till the user closes it, it will hold GUI window on the screen.

4. Delete image window from the memory after displaying using destroyAllWindows() function.

Let's start reading an image. using cv2.

To read the images cv2.imread() method is used. This method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

Example 1

```
# Python code to read image
import cv2
img = cv2.imread("image file name", cv2.IMREAD_COLOR)

# Creating GUI window to display an image on screen
cv2.imshow("image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Example 2

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

img = cv2.imread("image file name ")
# Displaying image using plt.imshow() method
plt.imshow(img)

# hold the window
plt.waitforbuttonpress()
plt.close('all')
```

Example 3

```
import cv2
# path
path = r'image filename'

# Using cv2.imread() method
# Using 0 to read image in grayscale mode
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)

# Displaying the image
cv2.imshow('image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Example 4

```
import cv2

# define a video capture object
vid = cv2.VideoCapture(0)
while (True):

    ret, frame = vid.read()
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

vid.release()
cv2.destroyAllWindows()
```

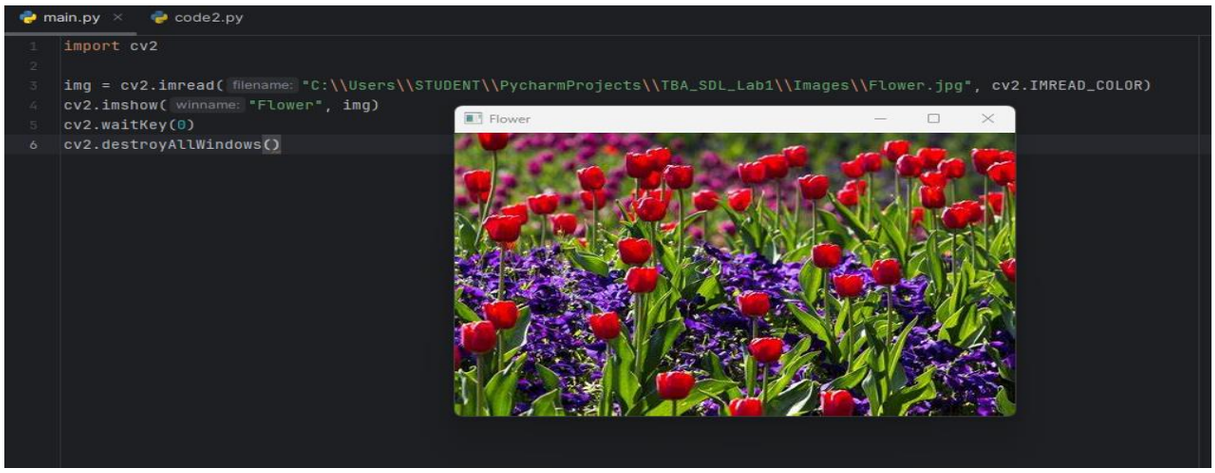
3. Take a screen shot of your design and **describe each image**.

4. Save your works as Lab1_GroupNo. as a PDF file. And upload it to the submission file.

## RESULTS AND DISCUSSION

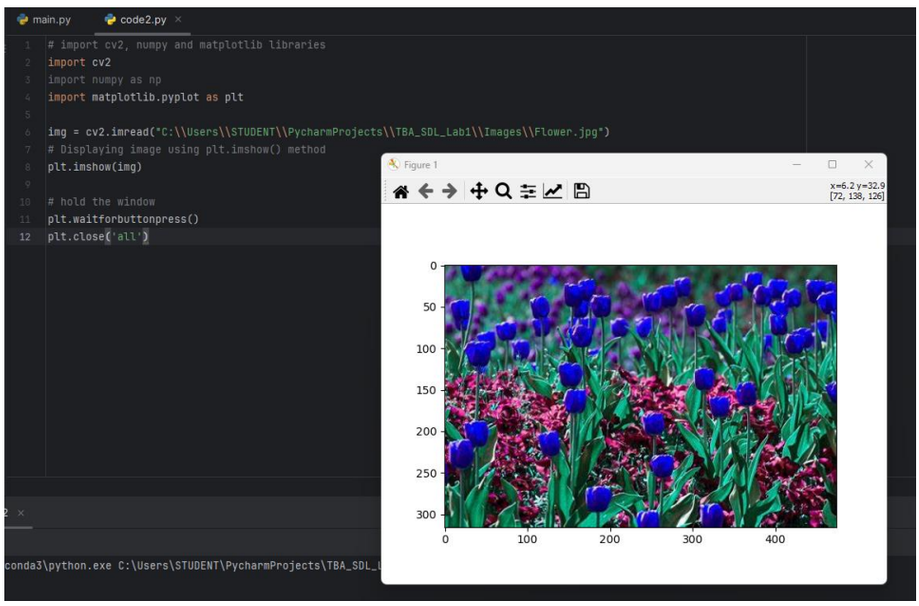
❖ OpenCV:

a. Example 1



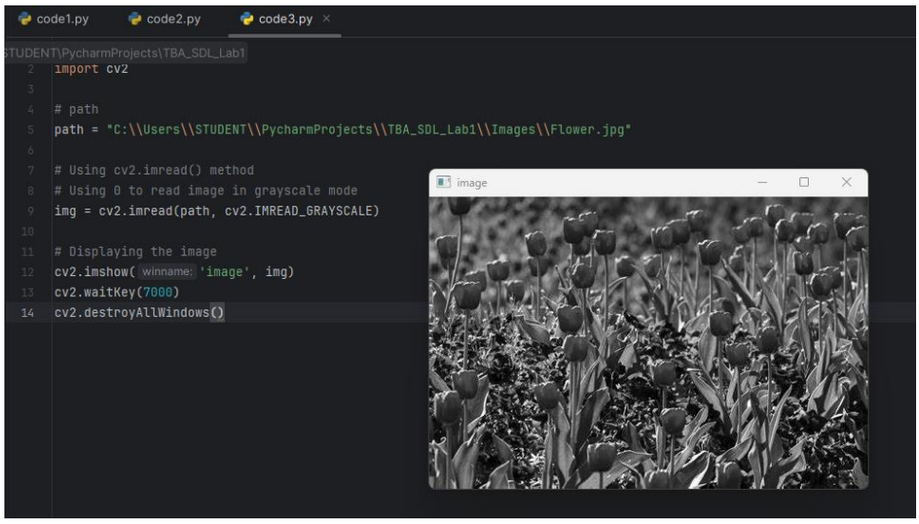

In the first activity, students use PyCharm to read an image file, a fundamental step in image processing. They begin by importing necessary libraries, such as OpenCV, which provides functions for image editing. The function cv2.imread() is commonly used for this purpose. It's essential to note that when specifying the file path, students must use double backslashes (\\) to define the absolute path on Windows systems.

b. Example 2



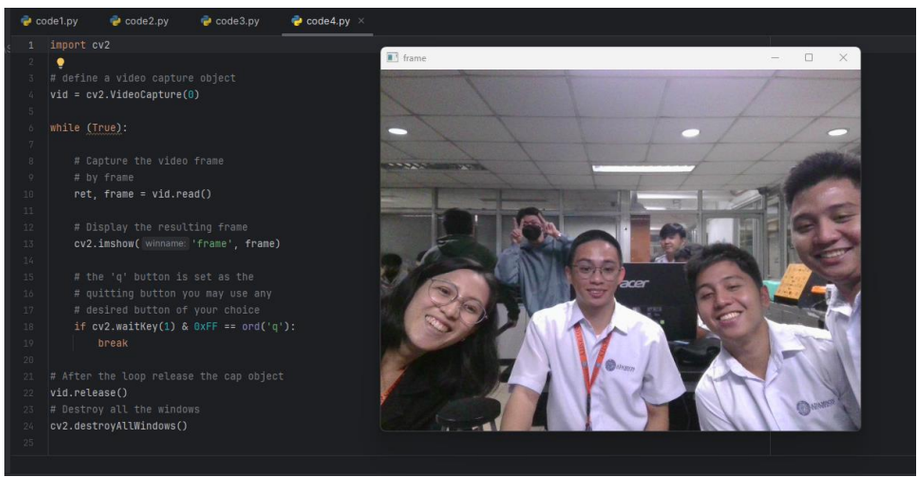

Students learned in creating a window using PyCharm and displaying the loaded image within this window. This task involves utilizing the imshow() function from the OpenCV library. The students used a red flower image and in this step after inserting the image, it changes the color from red to blue flower. In the third step, students created image editing further by converting the image to HSV. This function puts the image in a plot creating an x and y axis outside of the image.

c. Example 3




In this example, the image is displayed in grayscale. It is almost the same on example one, the only difference being the colors are removed from the image since the color space of the image is in grayscale. Understanding the concept of timing is crucial in image processing. Students explore the waitKey() function, which controls the duration the image remains displayed in the window, measured in milliseconds. For instance, setting waitKey(7000) would cause the image to close after 7 seconds.

d. Example 4




Lastly, students learned video processing by using the VideoCapture class from OpenCV to capture video streams. They can display video frames within a window using cv2.imshow(), enabling real-time visualization of the captured video. This activity broadens students' understanding beyond static images to dynamic video streams, opening up possibilities for various applications in computer vision.

## CONCLUSION

In conclusion, these activities serve as a comprehensive introduction to image processing using PyCharm and OpenCV. Beginning with the essential task of loading image

files, students familiarize themselves with library imports and fundamental functions like `cv2.imread()`. Attention to detail, such as the requirement for double backslashes in file paths on Windows systems, ensures a solid understanding of file management principles.

Advancing through tasks like creating GUI windows and manipulating image colors, students gain practical experience in visualizing and editing images. The exploration of timing mechanisms through the `waitKey()` function further enhances their comprehension, enabling precise control over image display durations. Finally, the introduction to video processing broadens students' horizons, empowering them to capture and visualize dynamic video streams.