Lab1

IMAGE PROCESSING

Image processing is a crucial field in computer science and technology that involves manipulating, analyzing, and enhancing digital images. Python, a powerful and versatile programming language, offers a wide range of libraries and frameworks specifically designed for image processing tasks. In this presentation, we will explore the key Python libraries used for image processing and learn how they can be leveraged to perform various image manipulation and analysis tasks.

OVERVIEW OF PYTHON FOR IMAGE PROCESSING:

NumPy:

Discover how NumPy provides a foundation for numerical computing and array manipulation.

OpenCV:

Explore the OpenCV library and its extensive set of functions and algorithms for image processing.

PIL/Pillow:

Discover the Python Imaging Library (PIL) and its successor, Pillow, for image loading,saving, and manipulation.

OPENCV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

OpenCV images are in BGR color format.

PILLOW (PIL)

The Pillow library contains all the basic image processing functionality. You can do image resizing, rotation and transformation.

Pillow module allows you to pull some statistics data out of image using histogram method, which later can be used for statistical analysis and automatic contrast enhancement.

Pillow images are in RGB color format.

THE DIFFERENCE BETWEEN RGB AND BGR

RGB is commonly used in image editing and display applications, where the order is assumed as red, green, and blue. On the other hand

BGR is often used in image processing applications, and the order is assumed blue, green, and red.

Today's methods:

- For importing:
  - Import cv2, numpy as np, matplotlib.pyplot
  - Import os
- For reading and showing images and images features:
  - Cv2.imread()
  - Cv2.imshow()
  - .shape
  - .shape[::]
  - plt.imshow()
- For changing the image color and image directory:
  - Cv2.cvtColor(image,cv2,COLOR_BGR2RGB)
  - Cv2.imwrite()

In [1]:
```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\rom\anaconda3\lib\si
te-packages (4.9.0.80)
Requirement already satisfied: numpy>=1.19.3 in c:\users\rom\anaconda3\lib\si
te-packages (from opencv-python) (1.26.2)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\rom\anaconda3\lib\site-
packages (3.4.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\rom\anaconda3\li
b\site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\rom\anaconda3
\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\rom\anaconda3\lib\si
te-packages (from matplotlib) (8.4.0)
Requirement already satisfied: cycler>=0.10 in c:\users\rom\anaconda3\lib\sit
e-packages (from matplotlib) (0.10.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\rom\anaconda3\lib
\site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: numpy>=1.16 in c:\users\rom\anaconda3\lib\site
-packages (from matplotlib) (1.26.2)
Requirement already satisfied: six in c:\users\rom\anaconda3\lib\site-package
s (from cycler>=0.10->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\rom\anaconda3\lib\site-packa
ges (1.26.2)
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
#Importing libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
#reading image from disk using imread() function
image = cv2.imread('Pic.jpeg')

#1 is a flag(flag should be 1/0/-1) 1 colored photo, 0 gray photo while and bl
#image2 = cv2.imread('Pic.jpeg',1)
```

In [3]:
```
#Creating GUI image to display image on screen
#imshow(title of image/window, image)
cv2.imshow('Personal picture',image)

#الصورة يتعمللها شو لعدد الثواني اللي اديته، هو 0 يبقى تفضل ظاهرة
#cv2.waintKey methon to hold the window on screen. Parameter for holding the s
cv2.waitKey(0)

#Removing/deleting created GUI window from screen and memory
#cv2.destroyAllWindows()
```

Out[3]:
```
-1
```

In [4]:
```
#get the image shape, height and width and shape channel 3=RGB
image.shape
```

Out[4]:
```
(600, 600, 3)
```

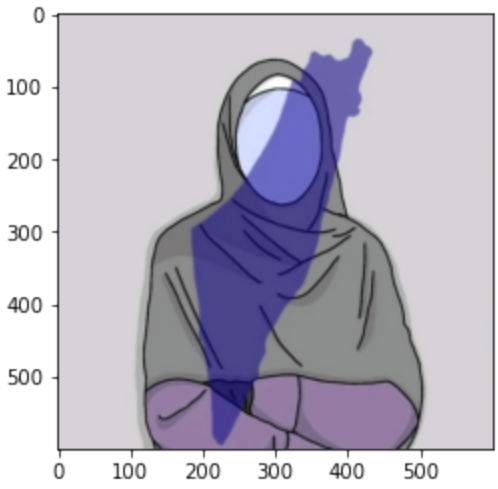
In [9]:
```
#Extracting height and width of an image
h, w = image.shape[:2]
print("Height=", h , "and Weight=",w)
```

```
Height= 600 and Weight= 600
```

In [10]:
```
#Show image using plt.imshow()
#using plt prints the image in BGR so we change the type after it

plt.imshow(image)
```

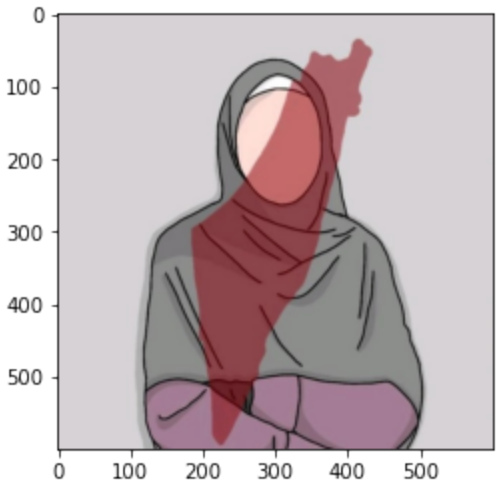
Out[10]:
```
<matplotlib.image.AxesImage at 0x1ed13054160>
```



In [11]:
```
#Convert the BGR color to RGB color format

RGB_image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
plt.imshow(RGB_image)
```

Out[11]: <matplotlib.image.AxesImage at 0x1ed1537dee0>



In [22]:
```
#To change the image place in computer
import os

#Get image path
image_path = 'Pictures/picSarah.jpeg'

#The place you want to make the image in
image_directory = 'D:\E-JUST'

#Read the image using imread()
image_change_place = cv2.imread(image_path)

#Change the current directory
os.chdir(image_directory)

#New image name
newImgName = 'SavedImage.jpg'

#save the new image in the new place
cv2.imwrite(newImgName,image)

print('Successfully saved')
```

```
Successfully saved
```