OpenCV with Python

OpenCV is an open source computer vision and machine learning software library. The official library has more than 2500 optimized algorithms, which includes a comprehensive set of classic algorithms as well as state of the art computer vision and machine learning algorithms.

Some popular algorithms in OpenCV can be used :

- To detect and recognize faces.
- To identify objects.
- To track moving objects.
- To stitch images together and lot more.

OpenCV software library is fully supported on Linux, Windows, Android and MacOS. It provides interface in multiple languages including Python, C, C++, Java and MATLAB.

OpenCV-Python is the Python API for OpenCV. It combines the best qualities of OpenCV and power of python scripting. Due to python's full compatibility with machine learning libraries and libraries like Numpy, Scipy, Matplotlib, OpenCV-Python is one of the most popular interface to be used for image processing.

Installation and Setup of OpenCV-Python

On Ubuntu

- Refresh the apt database.
  `# apt-get update`
- Install the package.
  `# apt-get install python3-opencv`
- (Optional) Install related python libraries using PIP.
  `$ pip3 install numpy matplotlib scipy`

OpenCV-Python Programming

1. Getting Started with Images

Things to do :-

- Read an Image from file. (imread)
- Display an Image in an OpenCV Window. (imshow)
- Write an image to a file. (imwrite)

Source Code :

```
import cv2 as cv
import sys

img = cv.imread("samples/starry_night.jpg")
if img is None:
        sys.exit("Could not read the image.")

cv.imshow("Display window", img)
k = cv.waitKey(0)

if k == ord("s"):
        cv.imwrite("processed/starry_night.png", img)
```

Code Analysis :-

```
img = cv.imread("samples/starry_night.jpg")
```

Using `imread(param1, param2)` function, we can read the image file. The First parameter accepts the File path (Absolute or Relative), Second parameter is optional and specifies the format in which we want the image.

RGB 8-bit (Default)

Grayscale

Possible values for Second Parameter :-

- IMREAD_COLOR : loads the image in the BGR 8-bit format. (Default param)
- IMREAD_UNCHANGED : loads the image as it is.
- IMREAD_GRAYSCALE : loads the image as an intensity one.

Above properties can be accessed using cv.* prefix.

```
cv.imshow("Display window", img)
k = cv.waitKey(0)
```

`imshow(param1, param2)` function is used to display the image GUI. The first parameter accepts the title of the window and the second argument is what cv image object to show.

`waitKey(param1)` function is used to tell how long (measured in milliseconds) should it display the image window. A value of 0 means forever.

```
if k == ord("s"):
        cv.imwrite("processed/starry_night.png", img)
```

`imwrite(param1, param2)` function is used to save/write the image file. The first parameter accepts the file name to save with and the second argument is what cv image object to save.

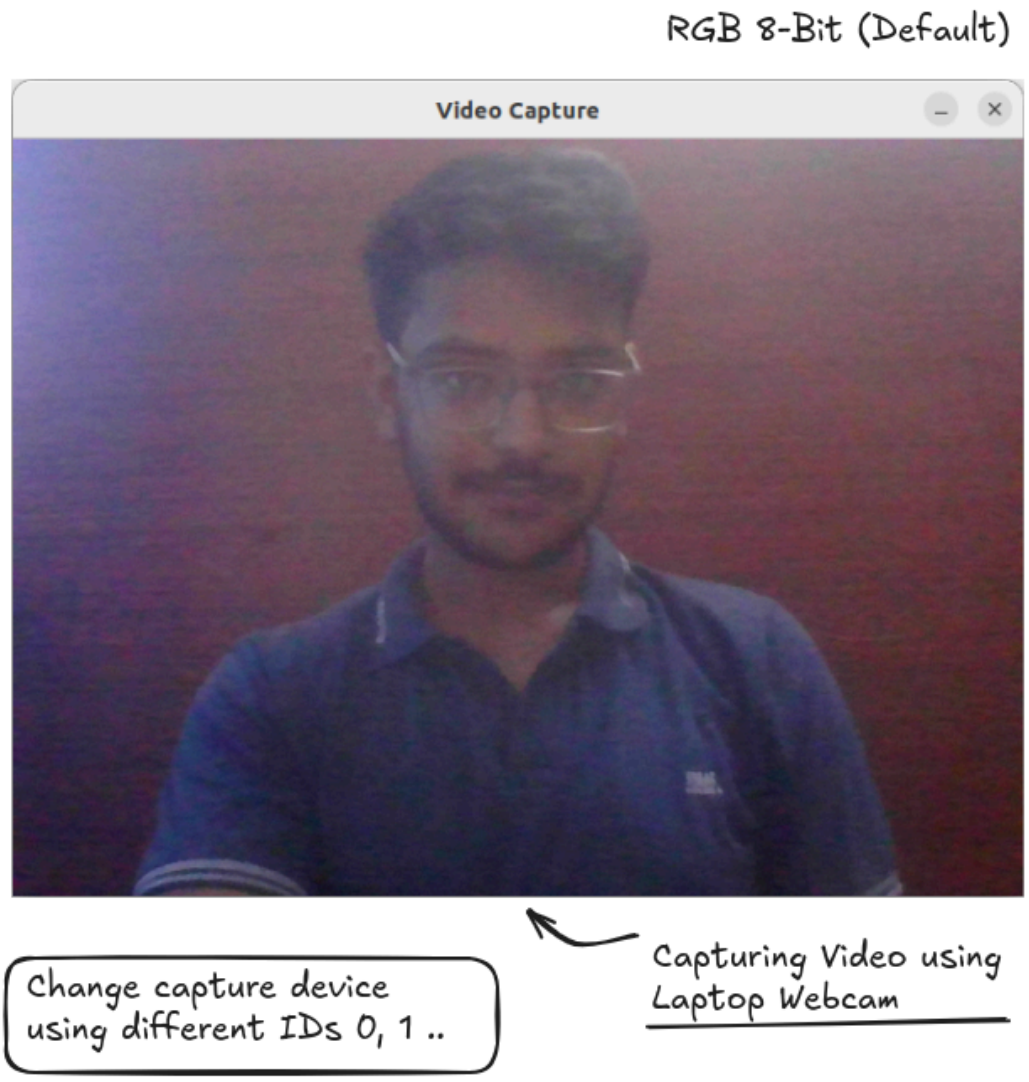
2. Getting started with Videos

Things to do :-

- To read, display and save Video.
- To capture video from a camera.

Capture video from a camera

To capture a video from a camera and convert it to grayscale.



Source Code :-

```
import numpy as np
import cv2 as cv

cap = cv.VideoCapture(0)
if not cap.isOpened() :
    print("Unable to open camera!")
    exit()

while True:
    ret, frame = cap.read()
```

```
    if not ret:
        print("Can't receive frame. Exiting ...")
        break

    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    cv.imshow('Video Capture', gray)
    if cv.waitKey(1) == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

Code Analysis :-

```
cap = cv.VideoCapture(0)
if not cap.isOpened() :
        print("unable to open camera!")
        exit()
```

`videoCapture(camera_ID)` function is used to select a camera hardware to capture video from. camera_ID can be 0, 1, 2 ... etc.

`cap.isOpened()` returns boolean if the requested camera access is granted or not. If not granted, we can use `cap.open()` to manually invoke it.

```
ret, frame = cap.read()

if not ret:
        print("Can't receive frame. Exiting ....")
        break
```

`ret` boolean shows the status of frame, whether it is received successfully or not.
`frame` contains the actual captured image data.

```
gray = cv.cvtColor(frame, cv.CLOR_BGR2GRAY)
```

`cvtColor(param1, param2)` function to transform color space of the captured frame. In the example above, I am converting it to GrayScale.

```
cap.release()
cv.destroyAllWindows()
```

To release and close all the opened camera hardware and quit the program gracefully.

`cap.get(propID)` method can be used to access some of the features(width, height) of the video, where propID is a number from 0 to 18.

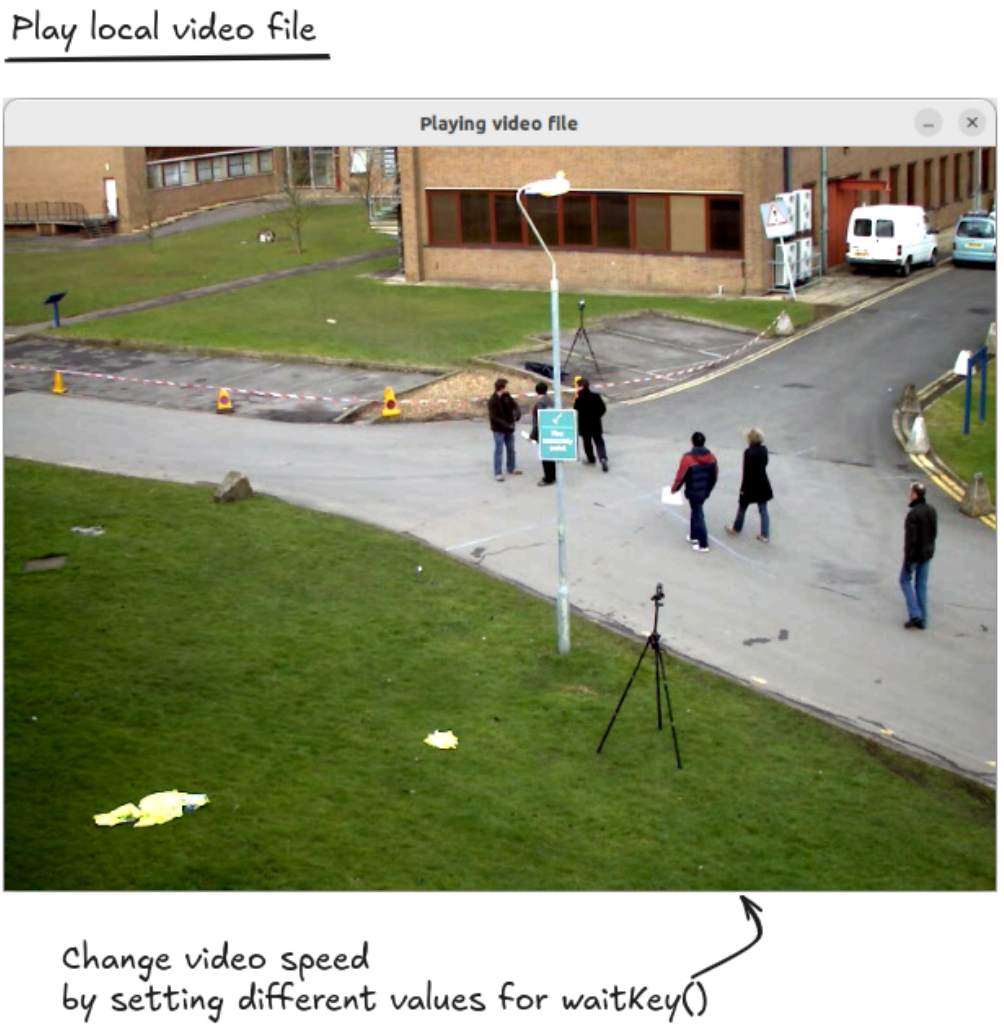
Some of the commonly used propIDs are :-

- CAP_PROP_FRAME_WIDTH
- CAP_PROP_FRAME_HEIGHT

`cap.set(propID, value)` method can be used to modify these capture's propID. For example to set a custom resolution.

Playing Video from file

To play a video file stored locally on the device.



Source Code :-

```
import numpy as np
import cv2 as cv
cap = cv.VideoCapture("samples/vtest.avi")
```

```
while cap.isOpened():
        ret, frame = cap.read()

        if not ret:
                print("Can't receive frame! Exiting ....")
                break

        cv.imshow("Playing video file", frame)
        if cv.waitKey(10) == ord('q'):
                break

cap.release()
cv.destroyAllWindows()
```

Code Analysis :-

```
cv.VideoCapture("samples/vtest.avi")
```

If we want to play a locally stored video file, we can directly pass it the absolute/relative path.

```
if cv.waitKey(25) == ord('q'):
        break
```

renders each frame with a time gap of 25 ms, most closest to real-life scenarios. We can increase the speed of video by reducing it value or increase the value to make it even more slower.

> Make sure that the latest version of either FFMPEG or Gstreamer is installed on your system.