**EXPLAINATION OF FACE DETECTION PROGRAM**

**OpenCV provides a facility to capture and record videos :**

cap = cv2.VideoCapture(0)

**VideoCapture()** takes parameter as the id of the camera through which the object has to capture. Here 0 stands for first camera or primary/main camera, if your system has a lot of cameras, you can specify their number as 1, 2, 3, etc, depends on which the number of cameras you want to turn on.

Now, as the cap variable is storing all the frames that have been captured, it is important to set an infinite loop and wait for the keypress by the user to break the loop or close the camera. Also, it’s important to read the frame and showcase it using imshow()

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

cv2.imshow('Frame', frame)

if(cv2.waitKey(1) == ord('v')):

break

cap.release()

cap.destroyAllWindows()

**cap.read()** returns the frame captured and flag that specifies everything is working error-free.

The camera must be released so that other applications can use it if it’s required, hence **cap.release()**

To break the loop, i.e. to close the window/camera, it’s important to specify an instruction that will make the user escape from the window.

Hence waitKey() is compared with the ord() of the desired characteristics that must be press to escape. In this case, the way to escape is by pressing the key ‘v’.

Note:  **ord(char)** takes a char as a parameter and returns its ASCII value.

Output :



*Video Capture*

Now,

For detection of objects from the frame, it must be converted to grayscale as many operations in OpenCV are done through grayscale.

The method used for conversion is

cv2.cvtColor()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

**cvtColor()** takes two compulsory parameters:

1. source and

2. color space conversion code -> specifies from which color system to which the source has to convert.

Now, if we display the gray instead of the frame, we can see the gray camera mode.

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

cv2.imshow('Frame', frame)

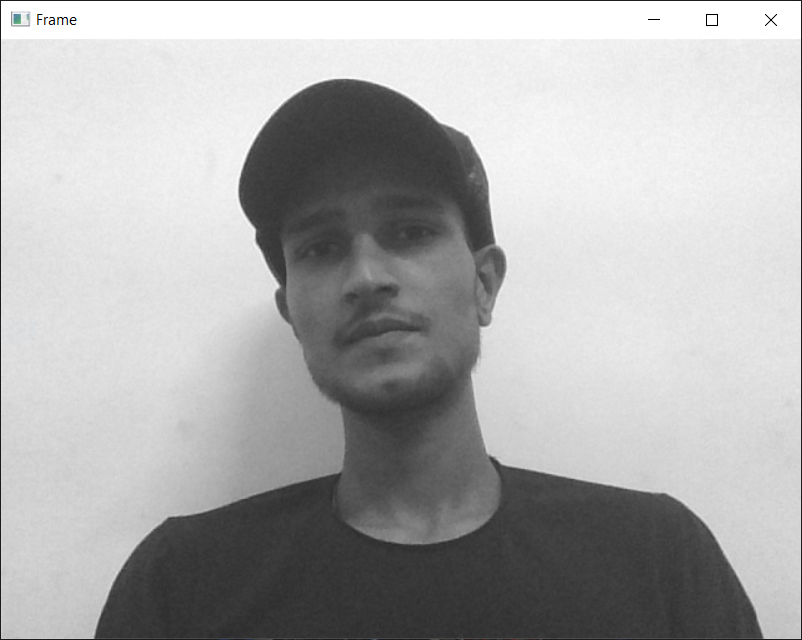
if(cv2.waitKey(1) == ord('v')):

break

cap.release()

cap.destroyAllWindows()

Output :



Video Capture in grayscale

After face detection, the program must specify a rectangle or circle or eclipse any shape you desire to differ face from other objects.

Let’s see how to draw shapes with openCV in video capture:

img = cv2.rectangle(img, (300, 300), (200, 200), (134, 56, 132), 6)

cv2.rectangle() draws rectangle over the image/ video capture i.e. over the source and takes parameters as

1. source

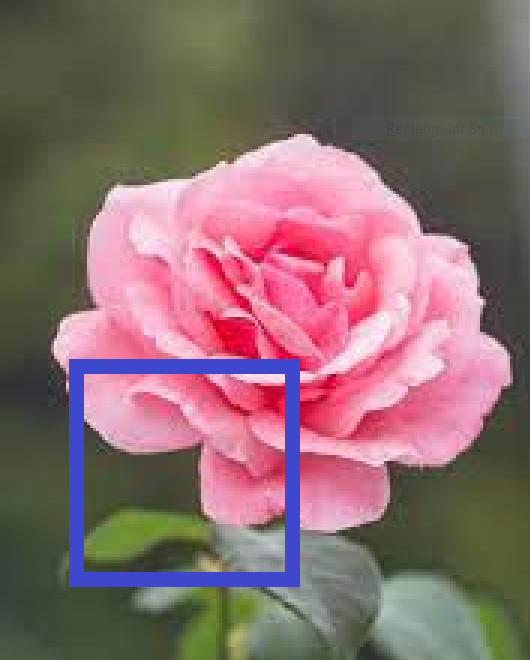
2. starting point co-ordinates

3. ending point co-ordinates

4. color value (RGB)

5. thickness in px (-1 to fill)

**Output:**



You can see the rectangle over the image. Similarly, you can draw lines, eclipse, circles and can put the text as well!

For object detection, OpenCV has various classifiers. The object passed through the corresponding classifiers and returns the detected result if any. OpenCV has a ‘Haar Cascade Classifier’ which classifies eyes, face, upper body, etc

This classifier has corresponding XML files that must be passed through CascadeClassifier() as:

#path for face cascade, path where it is stored specific classifier

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

How will we detect face then? With the help of the above Haar Cascade path and using detectMultiScale() we can store coordinates of face/faces in a tuple.

faces = face\_cascade.detectMultiScale(gray, 1.3,5)

detectMultiScale()  takes 3 parameters:

1. image/source

2. scale factor -> specifies how much the image size is reduced at each image scale.

3. min Neighbors ->Parameter specifying how many neighbors each candidate rectangle should have to retain it.

Let’s dive into the main program by assembling these little functions that we learned:

import cv2

import numpy as np

cap=cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

while True:

ret,frame = cap.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3,5)

for (x,y,w,h) in faces:

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),5)

cv2.imshow('frame',frame)

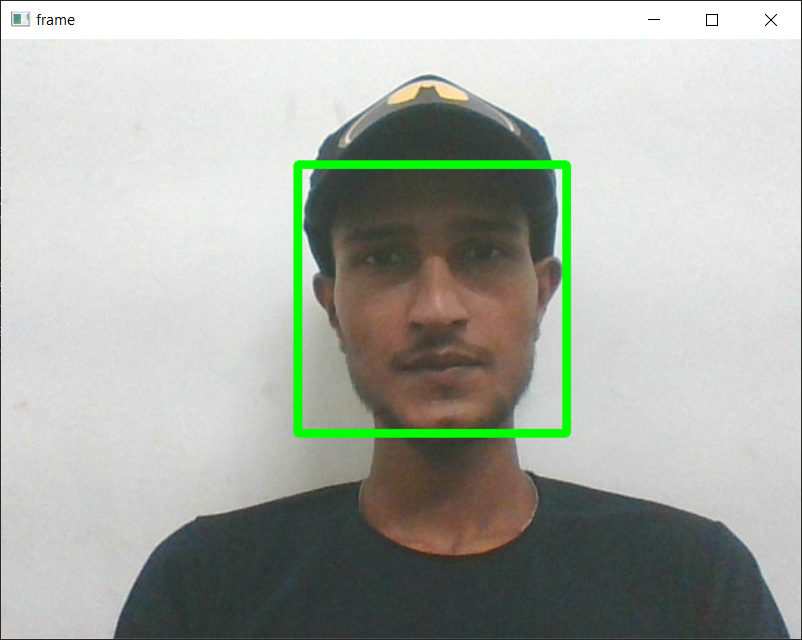
if(cv2.waitKey(1)==ord('v')):

break

cap.release()

cv2.destroyAllWindows()

**Output:**



Face Detection

That’s it!

Now it can also detect eyes just include Haar cascade eyes classifier as:

import numpy as np

import cv2

cap=cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_eye.xml')

while True:

ret,frame = cap.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3,5)

for (x,y,w,h) in faces:

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),5)

roi\_gray = gray[y:y+w,x:x+w]

roi\_color = frame[y:y+h,x:x+w]

eyes= eye\_cascade.detectMultiScale(roi\_gray,1.3,5)

for (ex,ey,ew,eh) in eyes:

cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(255,0,0),5)

cv2.imshow('frame',frame)

if(cv2.waitKey(1)==ord('v')):

break

cap.release()

cv2.destroyAllWindows()

To detect eyes sharp and clear, what has been done here is, region\_of\_interest, ROI variable has made so that only the face region is passed through eye\_cascade.detectMultiScale and then the rectangles for eyes have drawn on the actual frame image.

Let’s detect eyes and faces from an mp4 video. Include mp4 instead of camera id.

Just add mp4/img file path inside VideoCapture() and that’s it.

cap = cv2.VideoCapture('faceprojectvideo.mp4')

Output :



Here, the green rectangle denotes the faces and the blue one denotes the eyes.

That’s the end for Face and Eye Detection using OpenCV – Python!

You can really play around with this imaging array and video files using NumPy functions. Do try!

For more information about OpenCV- Python and its functions, you can refer to: <https://docs.opencv.org/master/d6/d00/tutorial_py_root.html>

For HaarCascade Github: <https://github.com/opencv/opencv/tree/master/data/haarcascades>

Thank you for climbing here!