OpenCV, which stands for Open Source Computer Vision is a library of programming functions which deals with computer vision. The library is cross-platform and free for use under the open-source BSD license and was originally developed by Intel. It supports the deep learning frameworks TensorFlow, Torch/PyTorch, and Caffe.

This article is a quick programming introduction to face detection, which basically is a classification which classifies between a face or non-face image. For this we will be using Haar Classifier, which is a machine learning based approach, an algorithm created by Paul Viola and Michael Jones; which are trained from many many positive images (with faces) and negatives images (without faces).

**Dependencies**

For this program, we will need a webcam-enabled system with Python 3.x and OpenCV 3.2.0 installed on it.

To install OpenCV with terminal use

sudo apt-get install python-opencv

To install this package with conda run

conda install -c conda-forge opencv

**Program**

This is an OpenCV program to detect face in real time:

import cv2

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

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

cap = cv2.VideoCapture(0)

while 1:

ret, img = cap.read()

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

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

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

cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)

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

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

eyes = eye\_cascade.detectMultiScale(roi\_gray)

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

cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,127,255),2)

cv2.imshow('img',img)

k = cv2.waitKey(30) & 0xff

if k == 27:

break

cap.release()

cv2.destroyAllWindows()

**Explanation**

As usual, we will start with importing the required libraries which here is OpenCV and will be imported as cv2.

We will then define our classifiers for face and eyes which are prebuilt in OpenCV as  
haarcascade\_frontalface\_default.xml and haarcascade\_eye.xml, respectively.

**NOTE**  
In some setup,

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

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

may not work, so instead of just file name, the whole path for the files will needed, like:

face\_cascade = cv2.CascadeClassifier('anaconda3/lib/python3.6/site-packages/cv2/data/haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier('anaconda3/lib/python3.6/site-packages/cv2/data/haarcascade\_eye.xml')

Now cap will capture frames from a camera. Now for each frame, we have to classify the face so it is inside while loop. We first read the frame, then convert to a grayscale of each frame and then detects faces of different sizes in the input image. Now our work will be to draw rectangles on the classified face images and to classify the eyes inside each of those rectangles, so after making rectangles we have added the eye classifier and make rectangles around eyes also. And finally, Display.

eyes = eye\_cascade.detectMultiScale(roi\_gray)

This all will be repeated until we press the escape button, so the final step is to break the loop when Esc is pressed.

**Summary**

We made a real model which classifies the faces and eyes from the webcam.