**WHAT IS FACE DETECTION:**

Face detection is a computer vision problem that involves finding faces in photos. It is a trivial problem for humans to solve and has been solved reasonably well by classical feature-based techniques, such as the cascade classifier. More recently deep learning methods have achieved state-of-the-art results on standard benchmark face detection datasets. One example is the Multi-task Cascade Convolutional Neural Network, or MTCNN for short.

**Face Detection Methods**

There are two main approaches for Face Detection:

1. Feature Base Approach
2. Image Base Approach

**Feature Base Approach**

Objects are usually recognized by their unique features. There are many features in a human face, which can be recognized between a face and many other objects. It locates faces by extracting structural features like eyes, nose, mouth etc. and then uses them to detect a face. Typically, some sort of statistical classifier qualified then helpful to separate between facial and non-facial regions. In addition, human faces have particular textures which can be used to differentiate between a face and other objects. Moreover, the edge of features can help to detect the objects from the face. In the coming section, we will implement a feature-based approach by using OpenCV.

**Image Base Approach**

In general, Image-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and non-face images. The learned characteristics are in the form of distribution models or discriminant functions that is consequently used for face detection. In this method, we use different algorithms such as Neural-networks, HMM, SVM, AdaBoost learning. In the coming section, we will see how we can detect faces with MTCNN or Multi-Task Cascaded Convolutional Neural Network, which is an Image-based approach of face detection

## ****Face detection algorithm****

One of the popular algorithms that use a feature-based approach is the Viola-Jones algorithm and here I am briefly going to discuss it. If you want to know about it in detail, I would suggest going through this article, [Face Detection using Viola Jones Algorithm](https://www.mygreatlearning.com/blog/viola-jones-algorithm).

**Viola-Jones** algorithm is named after two computer vision researchers who proposed the method in 2001, Paul **Viola** and Michael **Jones** in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features”. Despite being an outdated framework, Viola-Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection. This algorithm is painfully slow to train but can detect faces in real-time with impressive speed.

Given an image(this algorithm works on grayscale image), the algorithm looks at many smaller subregions and tries to find a face by looking for specific features in each subregion. It needs to check many different positions and scales because an image can contain many faces of various sizes. Viola and Jones used Haar-like features to detect faces in this algorithm.

## ****Face Detection using Python****

we can detect faces by using an Image-based approach. MTCNN or Multi-Task Cascaded Convolutional Neural Network is unquestionably one of the most popular and most accurate face detection tools that work this principle. As such, it is based on a deep learning architecture, it specifically consists of 3 neural networks (P-Net, R-Net, and O-Net) connected in a cascade.

So, let’s see how we can use this algorithm in Python to detect faces in real-time. First, you need to install MTCNN library which contains a trained model that can detect faces.

pip install mtcnn

Now let us see how to use MTCNN:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | from mtcnn import MTCNN  import cv2  detector = MTCNN()  #Load a videopip TensorFlow  video\_capture = cv2.VideoCapture(0)    while (True):      ret, frame = video\_capture.read()      frame = cv2.resize(frame, (600, 400))      boxes = detector.detect\_faces(frame)      if boxes:            box = boxes[0]['box']          conf = boxes[0]['confidence']          x, y, w, h = box[0], box[1], box[2], box[3]            if conf > 0.5:              cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 255, 255), 1)        cv2.imshow("Frame", frame)      if cv2.waitKey(25) &amp; 0xFF == ord('q'):          break    video\_capture.release()  cv2.destroyAllWindows() |

## ****Face Detection using OpenCV****

In this section, we are going to use OpenCV to do real-time face detection from a live stream via our webcam.

As you know videos are basically made up of frames, which are still images. We perform the face detection for each frame in a video. So when it comes to detecting a face in still image and detecting a face in a real-time video stream, there is not much difference between them.

We will be using Haar Cascade algorithm, also known as Voila-Jones algorithm to detect faces. It is basically a machine learning object detection algorithm which is used to identify objects in an image or video. In OpenCV, we have several trained  Haar Cascade models which are saved as XML files. Instead of creating and training the model from scratch, we use this file. We are going to use “haarcascade\_frontalface\_alt2.xml” file in this project. Now let us start coding this up.

import cv2

import os

cascPath = os.path.dirname(

    cv2.\_\_file\_\_) + "/data/haarcascade\_frontalface\_alt2.xml"

faceCascade = cv2.CascadeClassifier(cascPath)

video\_capture = cv2.VideoCapture(0)

while True:

    # Capture frame-by-frame

    ret, frame = video\_capture.read()

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

    faces = faceCascade.detectMultiScale(gray,

                                         scaleFactor=1.1,

                                         minNeighbors=5,

                                         minSize=(60, 60),

                                         flags=cv2.CASCADE\_SCALE\_IMAGE)

    for (x,y,w,h) in faces:

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

        # Display the resulting frame

    cv2.imshow('Video', frame)

    if cv2.waitKey(1) &amp; 0xFF == ord('q'):

        break

video\_capture.release()

cv2.destroyAllWindows()

OUTPUT: <https://youtu.be/MRtODg95es4>

# **How to Run Face Detector in Real-Time (Webcam):-**

Requirement for Running the code- Python, OpenCV, Webcam, Numpy.

**#import libraries**import cv2  
import numpy as np**#import classifier for face and eye detection**  
face\_classifier = cv2.CascadeClassifier(‘Haarcascades/haarcascade\_frontalface\_default.xml’)**# Import Classifier for Face and Eye Detection**face\_classifier = cv2.CascadeClassifier(‘Haarcascades/haarcascade\_frontalface\_default.xml’)  
eye\_classifier = cv2.CascadeClassifier (‘Haarcascades/haarcascade\_eye.xml’)  
def face\_detector (img, size=0.5):**# Convert Image to Grayscale**gray = cv2.cvtColor (img, cv2.COLOR\_BGR2GRAY)  
faces = face\_classifier.detectMultiScale (gray, 1.3, 5)  
If faces is ():  
return img**# Given coordinates to detect face and eyes location from ROI**for (x, y, w, h) in faces  
x = x — 100  
w = w + 100  
y = y — 100  
h = h + 100  
cv2.rectangle (img, (x, y), (x+w, y+h), (255, 0, 0), 2)  
roi\_gray = gray[y: y+h, x: x+w]  
roi\_color = img[y: y+h, x: x+w]  
eyes = eye\_classifier.detectMultiScale (roi\_gray)  
for (ex, ey, ew, eh) in eyes:  
cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,0,255),2)  
roi\_color = cv2.flip (roi\_color, 1)  
return roi\_color**# Webcam setup for Face Detection**cap = cv2.VideoCapture (0)  
while True:  
ret, frame = cap.read ()  
cv2.imshow (‘Our Face Extractor’, face\_detector (frame))  
if cv2.waitKey (1) == 13: #13 is the Enter Key  
break**# When everything done, release the capture**cap.release ()  
cv2.destroyAllWindows ()