**FACE DETECTION USING OPEN CV**

**FACE DETECTION is a Computer technology which determines the size and the location of the faces of human beings in real world or digital pictures. It detects facial features ignoring every other object detection such as trees, pencils, etc . This can even be regarded as a more general case of the localization of the face, where the basic task is of finding the size as well as the location of a known number of faces.**

For building up this project, we need to first perform detection of face and extracting the face embeddings . Here, comes the use of OpenCV , we can also make the use of dlib to detect the facial landmarks in an image such as mouth , jaw , eyes, eyebrows, etc.

**OpenCV-Python is a library of Python bindings to solve major computer vision problems, hence it is abbreviated as “Open Source** Computer Vision Library”.

OpenCV provides a training method (Cascade Classifier) that can be read under .xml file extensions (which is basically an extension for an Extensible Markup Language ) . The Object Dectection using HaarCascade Feature based cascade classifier is very effective object detection method . It is basically a machine learning based approach where cascade function is trained from a lot of positive as well as negative images . Then, further its use comes in detecting of the objects in other images.While training the cascade classifier , the algorithm works on a lot of positive as well as negative images i.e. the images with and without faces respectively , and in the similar way the features are extracted from it.

**First, a**[**cv::CascadeClassifier**](https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html)**is created and the necessary XML file is loaded using the**[**cv::CascadeClassifier::load**](https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html#a1a5884c8cc749422f9eb77c2471958bc)**method. Afterwards, the detection is done using the**[**cv::CascadeClassifier::detectMultiScale**](https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html#aaf8181cb63968136476ec4204ffca498)**method, which returns boundary rectangles for the detected faces or eyes.**

**SOME KEY POINTS TO NOTE BEFORE DIVING DEEP INTO THE PROJECT:**

* The image that is read is first converted into Grayscale image using **[cv2.cvtColor(img, COLOR\_BGR2GRAY) and storing this in a variable].**This is a necessary step because the detection works only upon the grayscale image.(where grayscale image contains only the shades of gray or no color).
* **detectMultiScale function** is being used to detect the faces which have 3 arguments which are namely the image input , the scaleFactor and the third argument will be the **minNeighbours.**

***scaleFactors:*** The parameter that specifies how much the image size should be reduced at each image scale , even by rescaling the input image you can resize a larger face to a smaller one making it detectable by the algorithm. A good possible value that can be considered is 1.05 which means that you use a small step for resizing i.e. reducing the size by 5% .

***minNeighbors:*** The parameter that specifies how many neighbors watch candidate rectangle should have to retain it. This parameter effects the quality of the detected faces.Approximately 3-6 is considered a good value since higher values can result in less dection although with high quality.

Also we can detect faces in videos . Since, videos are also the blend of the frames which are still the images . So,we perform the face detection for each frame in a video . But the difference that the face detection in a vedio will create is the use of an infinite loop through each of the frames in the video .