**Project name :** **Face Detection with C++ using openCV**

**Group Name: 0000**

**Group Member ID: 50, 59, 87**

In this section we propose an open source method to efficiently detect and extract faces from an image given using OpenCV, the most popular library for computer vision. Originally written in C/C++. For face detection, the algorithm starts at the top left of a picture and moves down across small blocks of data, looking at each block, constantly asking, “Is this a face? . . . Is this a face? . . . Is this a face?” Since there are 6,000 or more tests per block, you might have millions of calculations to do, which will grind your computer to a halt. To get around this, OpenCV uses cascades. Like a series of waterfalls, the OpenCV cascade breaks the problem of detecting faces into multiple stages. For each block, it does a very rough and quick test. If that passes, it does a slightly more detailed test, and so on. The algorithm may have 30-50 of these stages or cascades, and it will only detect a face if all stages pass. The advantage is that the majority of the pictures will return negative during the first few stages, which means the algorithm won’t waste time testing all 6,000 features on it. Instead of taking hours, face detection can now be done in real time. Though the theory may sound complicated, in practice it is quite easy. The cascades themselves are just a bunch of XML files that contain OpenCV data used to detect objects. You initialize your code with the cascade you want, and then it does the work for you.

## Face Detection

Face detection has gained a lot of attention due to its real-time applications. A lot of research has been done and still going on for improved and fast implementation of the face detection algorithm. Why is face detection difficult for a machine? Face detection is not as easy as it seems due to lots of variations of image appearance, such as pose variation (front, non-front), occlusion, image orientation, illumination changes and facial expression.

OpenCV contains many pre-trained classifiers for face, eyes, smile etc. The XML files of pre-trained classifiers are stored in opencv/data/. For face detection specifically, there are two pre-trained classifiers:

1. Haar Cascade Classifier
2. LBP Cascade Classifier

We used Haar Cascade Classifier for detecting face.

### **Haar Cascade Classifier**

It is a machine learning based approach where a cascade function is trained from a lot of positive (images with face) and negative images (images without face). The algorithm is proposed by Paul Viola and Michael Jones.

The algorithm has four stages:

1. **Haar Feature Selection:** Haar features are calculated in the subsections of the input image. The difference between the sum of pixel intensities of adjacent rectangular regions is calculated to differentiate the subsections of the image. A large number of haar-like features are required for getting facial features.
2. **Creating an Integral Image:** Too much computation will be done when operations are performed on all pixels, so an integral image is used that reduce the computation to only four pixels. This makes the algorithm quite fast.
3. **Adaboost:** All the computed features are not relevant for the classification purpose. Adaboost is used to classify the relevant features.
4. **Cascading Classifiers:** Now we can use the relevant features to classify a face from a non-face but algorithm provides another improvement using the concept of cascades of classifiers. Every region of the image is not a facial region so it is not useful to apply all the features on all the regions of the image. Instead of using all the features at a time, group the features into different stages of the classifier.Apply each stage one-by-one to find a facial region. If on any stage the classifier fails, that region will be discarded from further iterations. Only the facial region will pass all the stages of the classifier.