# SMART ATTENDANCE SYSTEM

M PRAVEEN KUMAR(21ECB0B31)

## LIBRARIES USED:

- Open cv
- NumPy
- Pillow(PIL)
- OS

## MODELS USED:

- Haar Cascade classifier
- LBPH Algorithm

# OPEN CV

## Computer Vision

Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps.

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

# What is NumPy?

NumPy is a Python library used for working with arrays.
It also has functions for working in domain of linear algebra, fourier transform, and matrices.
NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
NumPy stands for Numerical Python.

# Why Use NumPy?

- In Python we have lists that serve the purpose of arrays, but they are slow to process.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
- The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

## Pillow(PIL):

- The Python Imaging Library adds image processing capabilities to your Python interpreter.
- This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.
- The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.
- Used for Opening, rotating and displaying an image.
- The image object we received using **Image. open()** can be used to resize, crop, draw or other image manipulation method calls on this Image object.

## OS library:

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

# FACE RECOGNITION

Facial recognition is a way of identifying or confirming an individual's identity using their face. Facial recognition systems can be used to identify people in photos, videos, or in real-time.

Facial recognition is a category of biometric security. Other forms of biometric software include voice recognition, fingerprint recognition, and eye retina or iris recognition. The technology is mostly used for security and law enforcement, though there is increasing interest in other areas of use.

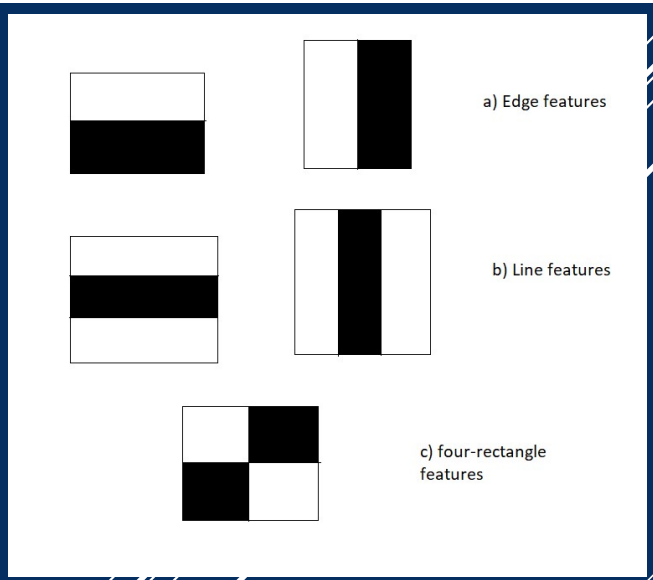
In the past few years, face recognition owned significant consideration and appreciated as one of the most promising applications in the field of image analysis. The method of face detection in pictures is complicated because of variability present across human faces such as pose, expression, position and orientation, skin colour, the presence of glasses or facial hair, differences in camera gain, lighting conditions, and image resolution.

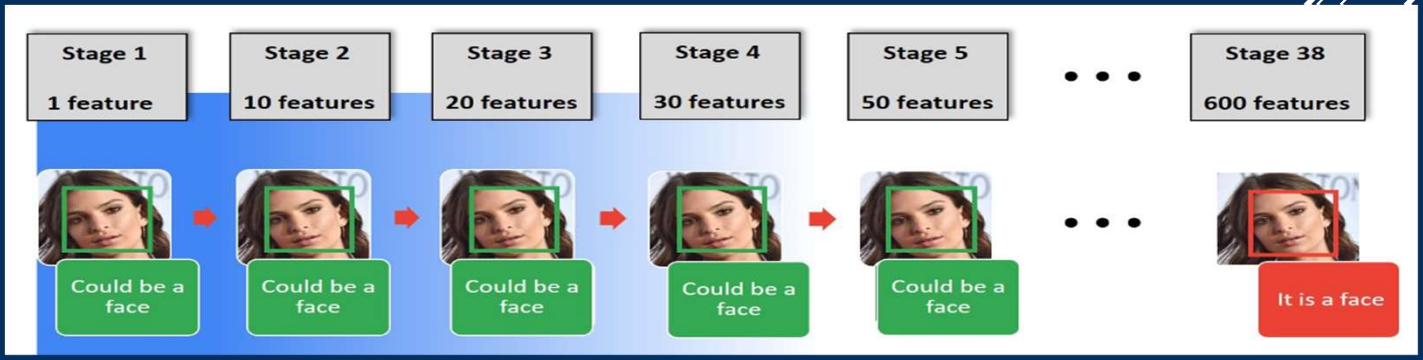
# HAAR CASCADE CLASSIFIER

It is an Face Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features” . The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them.

It works in four stages:

- Haar-feature selection: A Haar-like feature consists of dark regions and light regions. It produces a single value by taking the difference of the sum of the intensities of the dark regions and the sum of the intensities of light regions. It is done to extract useful elements necessary for identifying an object. The features proposed by viola and jones are 180000.



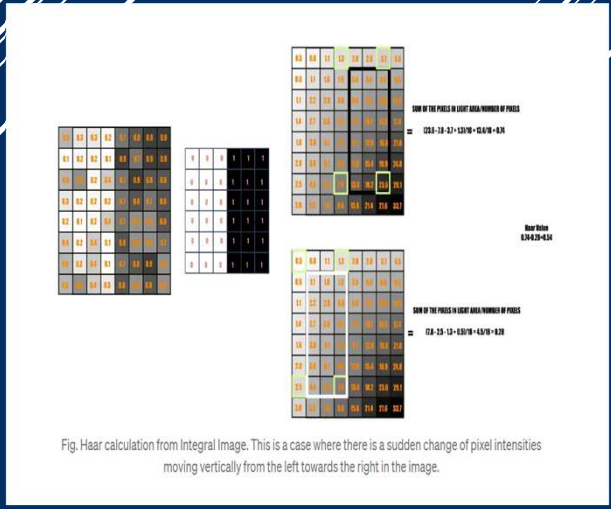



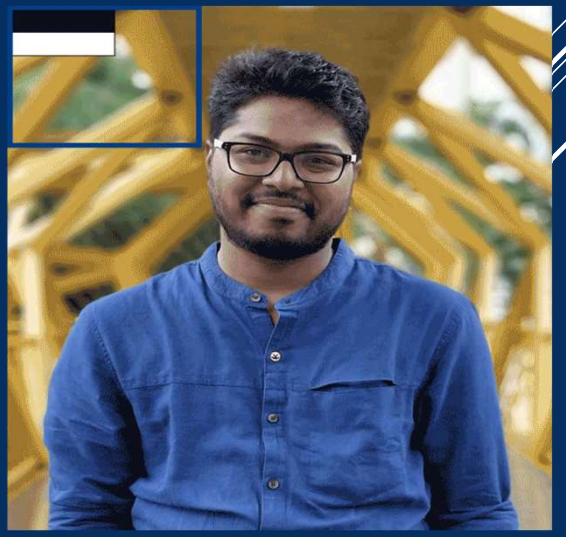
Fig. Haar calculation from Integral Image. This is a case where there is a sudden change of pixel intensities moving vertically from the left towards the right in the image.

•**Creation of Integral Images**: A given pixel in the integral image is the sum of all the pixels on the left and all the pixels above it. Since the process of extracting Haar-like features involves calculating the difference of dark and light rectangular regions, the introduction of Integral Images reduces the time needed to complete this task significantly.

•**AdaBoost Training**: This algorithm selects the best features from all features. It combines multiple “weak classifiers” (best features) into one “strong classifier”. The generated “strong classifier” is basically the linear combination of all “weak classifiers” .It uses boosting technique to classify whether it is a face or not and it is used in ensemble learning.

•**Cascade Classifier**: It is a method for combining increasingly more complex classifiers like AdaBoost in a cascade which allows negative input (non-face) to be quickly discarded while spending more computation on promising or positive face-like regions. It significantly reduces the computation time and makes the process more efficient.

OpenCV comes with lots of pre-trained classifiers. Those XML files can be loaded by cascade Classifier method of the cv2 module. Here we are going to use haarcascade_frontalface_default.xml for detecting faces.
Take each 24x24 window. Apply 6000 features to it.
We are able to achieve 20 fps on jetsan nano.



# CAPTURING ATTENDANCE USING FACE RECOGNITION:

In many of the educational institutions, managing attendance of students/candidates is tedious, as there would be large number of students in the class and keeping track of all is onerous. There are situations where student act as proxies for their friends even though they are not present. This system, which is based on face detection and recognition algorithms, spontaneously detects the student when he enters the class room and marks the attendance by recognizing him. The database is then modified or updated automatically. This reduces time and effort of manually updating the attendance. This system also provides authentication using recognition of face of the admin or teacher to unlock as there are chances of trespassing by the third party. So this rises the security of the system and it save the time .

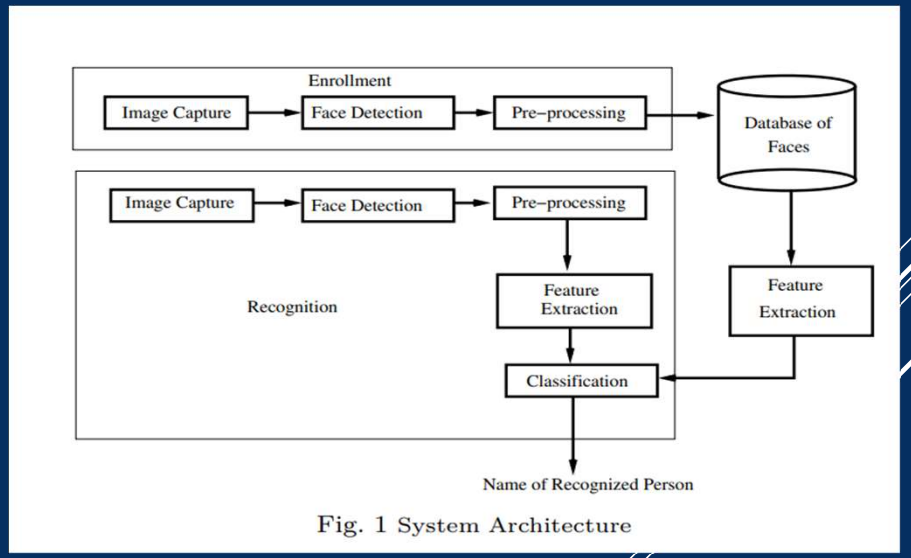
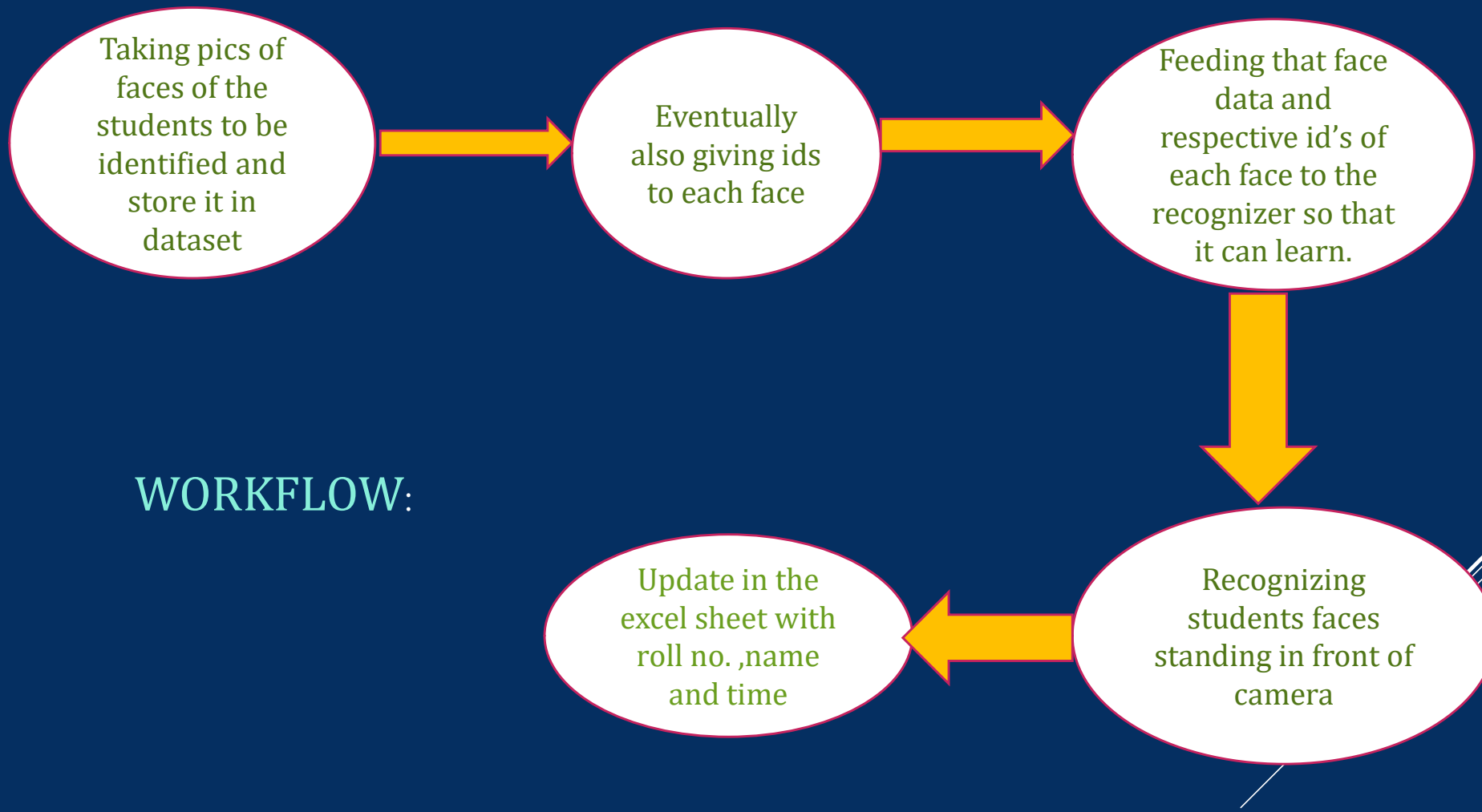


Fig. 1 System Architecture

## WORKFLOW:


Taking pics of faces of the students to be identified and store it in dataset
Eventually also giving ids to each face
Feeding that face data and respective id's of each face to the recognizer so that it can learn.
Recognizing students faces standing in front of camera
Update in the excel sheet with roll no. ,name and time

# DETECT MULTISCALE

We use v2.CascadeClassifier.detectMultiScale() to find faces or eyes, and it is defined like this:

cv2.CascadeClassifier.detectMultiScale( image, scaleFactor, minNeighbors, flags, minSize , maxSize )
Where the parameters are:

- **image** : Containing an image where objects are detected.
- **scalefactor** : Parameter specifying how much the image size is reduced at each image scale.

Since the classifier model was trained with the fixed face size images which can be seen in the xml file. It can detect faces with the same size which was used during the training. So what if our input image has faces smaller or bigger than what was used then. Now, for this reason, we want our image in different scales so that the model can detect a face.

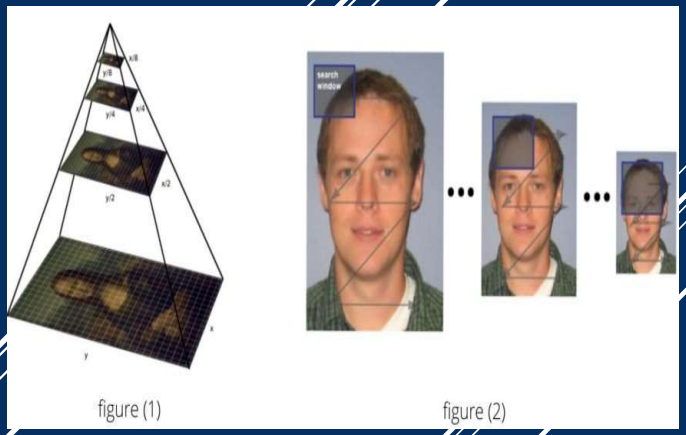



figure (1) figure (2)

Scale factor =1.05 Means we reduce the image size by 5% each time we scale. This would generate more number of layers since the image is downsized each time by only 5%. With this large number of layers, the algorithm works slower. And since it is more precise the algorithm would take some non-face areas into consideration.

***Scale factor =1.8*** This large value would generate less number of layers and accordingly the algorithm works faster and *can miss some faces*

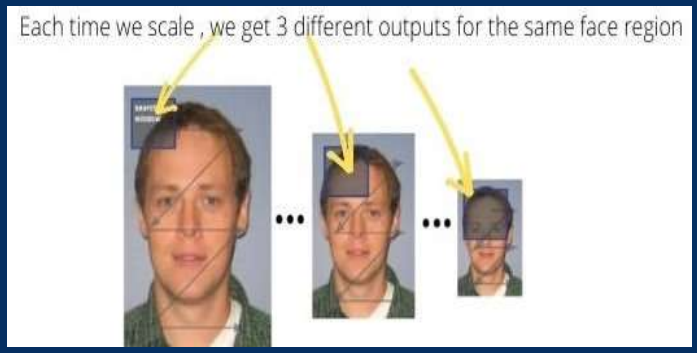
Scale factor = 1.3 is the value which we take into consideration giving the output as

**minneighbors** : Parameter specifying how many neighbors each candidate rectangle should have to retain it. This parameter will affect the quality of the detected faces: higher value results in less detections but with higher quality. We're using 5 in the code.

Since the object detection works in the combination of the *image pyramid (multi-scaling)* and *sliding window,* we get **multiple** true outputs for a single region of the face. These true outputs are the window region which satisfies the Haar features (could be actual face area or a non-face area taken into consideration).

Therefore, to get an actual face rectangle, the number of rectangles for the same face region must be higher than the *minNeighbors*.

minNeighbor = 0. In the figure below some areas have multiple true outputs.




Setting minNeighbor = 3, the area with less than 3 true outputs will not get detected by the classifier. It also includes non faces too.Since minNeighbor is the threshold value for the number of true outputs required to detect a face.

Setting minNeighbor = 5, the area with less than 5 true outputs will not get detected by the classifier. Since minNeighbor is the threshold value for the number of true outputs required to detect a face.

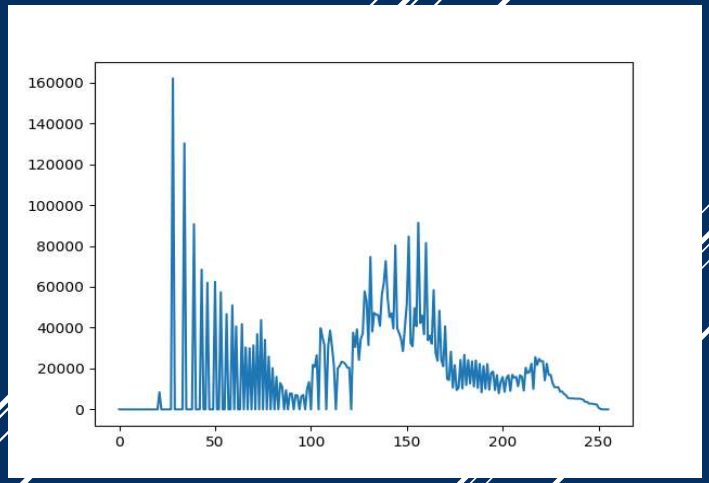
minSize : Minimum possible object size. Objects smaller than that are ignored.

maxSize : Maximum possible object size. Objects larger than that are ignored.

# HISTOGRAM

Histogram is a graphical representation of the intensity distribution of an image. In simple terms, it represents the number of pixels for each intensity value considered.

In the figure, X-axis represents the total scale (black at the left and white at the right), and Y-axis represents the number of pixels in an image. Here, the histogram shows the number of pixels for each brightness level (from black to white), and when there are more pixels, the peak at the certain brightness level is higher .




USES:

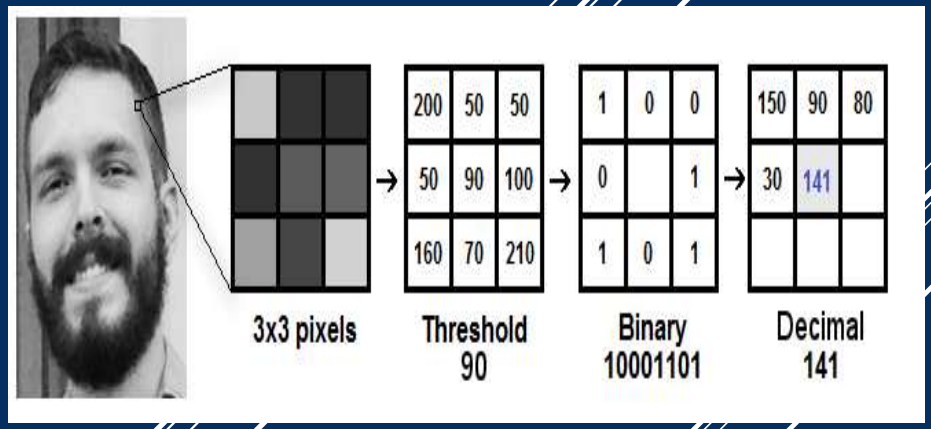
- To Remove noise from the images
- It is used to analyze an image. Properties of an image can be predicted by the detailed study of the histogram.
- It is used for image equalization. Gray level intensities are expanded along the x-axis to produce a high contrast image.
- Image processing
- The brightness of the image can be adjusted by having the details of its histogram.

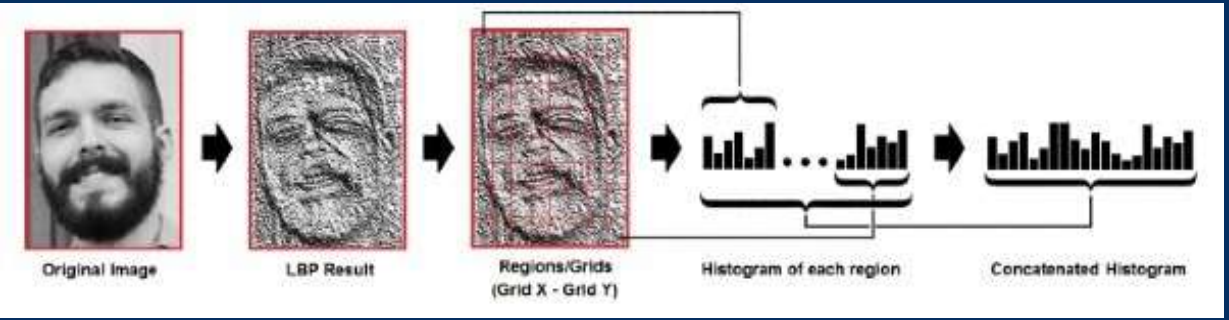
# LBPH(LOCAL BINARY PATTERNS HISTOGRAMS) ALGORITHM

It makes use of a data-set of images of the people to be included in the recognition process. Each image is given a unique ID as either a number or the name of a person so that the algorithm can use this information to identify the image and export the output. The pictures of the same person are always placed under the same ID.

## Computational steps:

1) LBP Procedure: Here, an intermediate image has been created to better represent the original image through a sliding window concept, taking into account two parameters: the neighbor and the radius. New values are created in the form of binary by comparing the 8 neighbor values to the threshold value. For each neighbor value greater than the threshold value, the value is set to 1 and 0 for every neighbor value less than the threshold value. This forms a matrix of binary numbers excluding the threshold. A central value of the matrix is created by the conversion of the binary number to a decimal value which corresponds to the pixels of the original image. For a better representation of the characteristics of the original image.






2) To Extract Histograms: The image obtained in step is divided into multiple grids, with the help of the Grid parameters X and Y. This image is in grayscale, each of the histograms of each of the grids is to represent the intensity of the occurrences of each pixel. Each histogram is then combined to create a new histogram that represents the attributes of the original image.

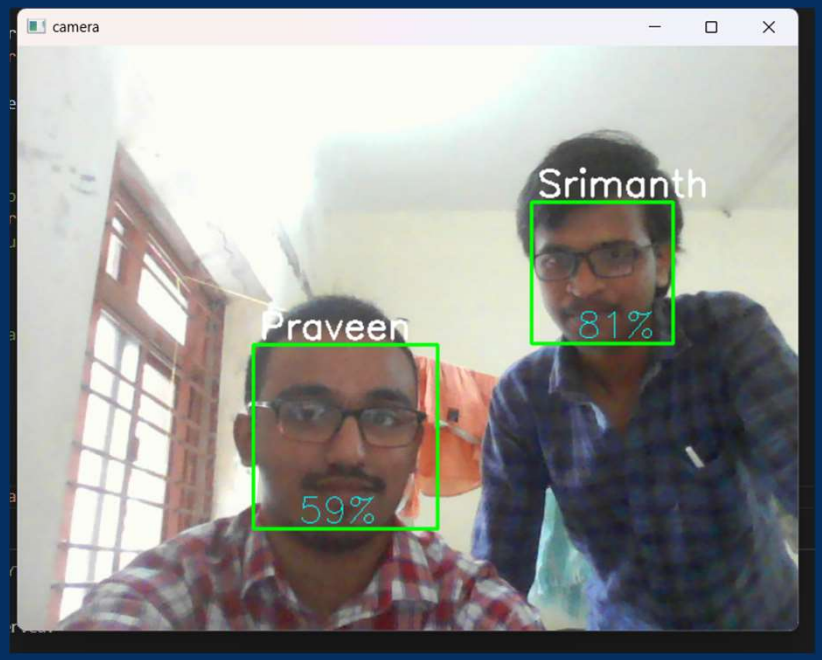
As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.

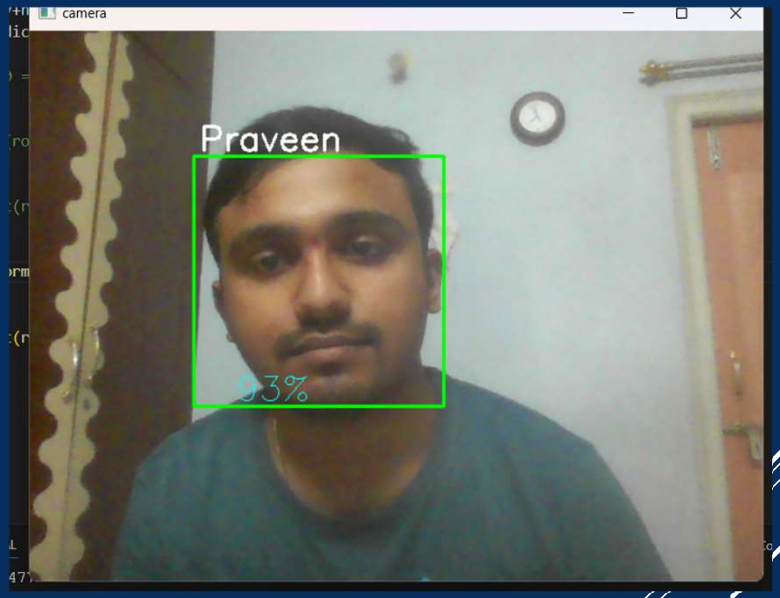
Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16,384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

## •3)Accurate face recognition:

- In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.
- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula: to output the image with the closest histogram matches to an input image.
- The algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a ‘confidence’ measurement.

# RESULTS:




Thank you