**1 Introduction**

Attendance is prime important for both the teacher and student of an educational organization. So it is very important to keep record of the attendance. The problem arises when we think about the traditional process of taking attendance in class room. Calling name or roll number of the student for attendance is not only a problem of time consumption but also it needs energy. So an automatic attendance system can solve all above problems.

There are some automatic attendances making system which are currently used by much institution. One of such system is biometric technique. Although it is automatic and a step ahead of traditional method it fails to meet the time constraint. The student has to wait in queue for giving attendance, which is time taking.

This project introduces an involuntary attendance marking system, devoid of any kind of interference with the normal teaching procedure. The system can be also implemented during exam sessions or in other teaching activities where attendance is highly essential. This system eliminates classical student identification such as calling name of the student, or checking respective identification cards of the student, which can not only interfere with the ongoing teaching process, but also can be stressful for students during examination sessions.

Project overview:

In this project, first to check whether the camera is working properly or not . If the camera is working properly then real time datasets will be capture along with id and name of the particular person once images are captured the images will store in database(local system). And then we will train the images based on the algorithm once images get trained . After images got trained the system we will be validate based on the real time faces .if the person face is matched with database images then attendance will be recorded in excel sheet (as of now) and the excel sheet will be send to the authorized person.

Capturing the datasets by using web camera.by name and id of the particular person

13

Images will be trained based on the algorithm

Validation

If image is not found in the database capture the image Inform to the admin

If image is found generate mail to the person

1.2 Hardware specification

1.2.1 Camera:

Which is used for the capturing image,Where the size of captured image is maa vary with type of the image captures if it is colour image the size of the image is more for the color image and the size of the image is less for gray scale image

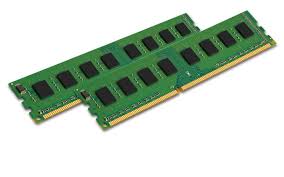
1.2.2 Intel processor:

 Basic needed to run the program which is used to load the program from the secondary memory to processor to run the program.

1.2.3 GPU(Graphical Processor Unit):

To increase the efficiency of the system and also to increase the accuracy of the system

1.2.4RAM

At least 8GB RAM can (run **face recognition** smoothly)

1.3 Software requirements:

* Python 3.6 version
* appdirs==1.4.3
* attrs==19.1.0
* black==19.3b0
* Click==7.0
* cycler==0.10.0
* demjson==2.2.4
* kiwisolver==1.1.0
* matplotlib==3.1.1
* nose==1.3.7
* numpy==1.17.0
* opencv-contrib-python==4.1.0.25
* pandas==0.25.1
* Pillow==6.1.0
* pyparsing==2.4.2
* python-dateutil==2.8.0
* pytz==2019.2
* six==1.12.0
* toml==0.10.0
* virtualenv==16.7.4
* xlrd==1.2.0
* xmltodict==0.12.0
* yagmail==0.11.220
* yapf==0.28.0

Face Detection:

Introduction:

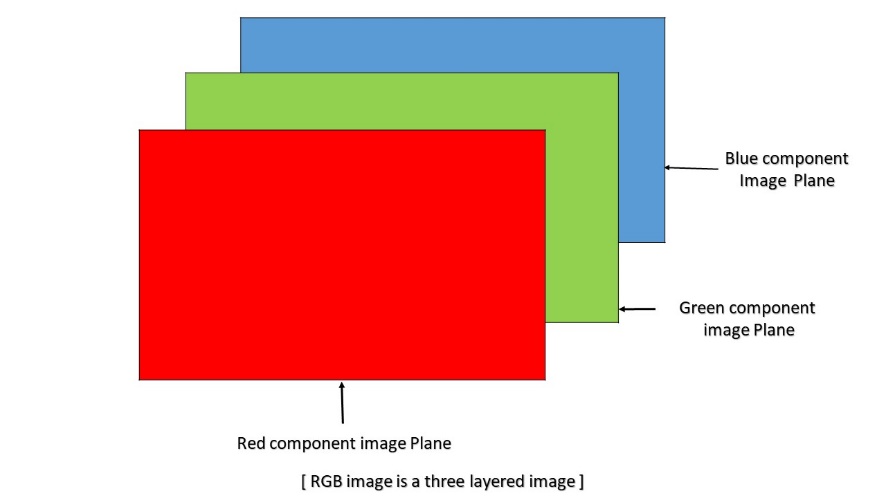
Face detection is defined as finding the position of the face of an individual. In other word it can be defined as locating the face region in an image. After detecting the face of human its facial features is extracted and has wide range of application like facial expression recognition, face recognition, observation systems, human PC interface and so forth…Detecting face in an image of single person is easy but when we consider a group image of an image containing multiple faces, the task becomes difficult.

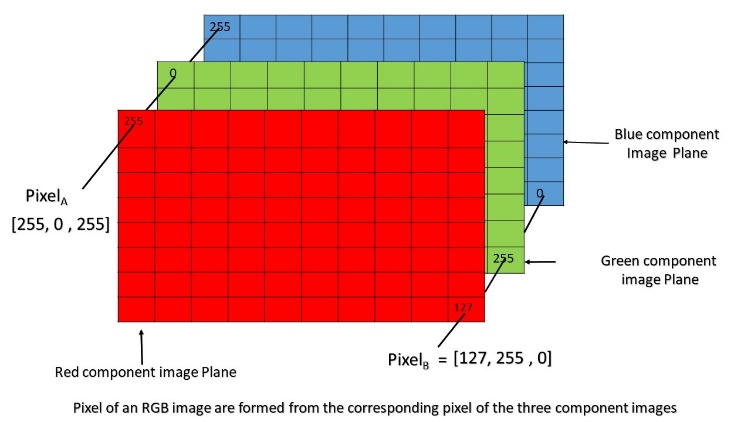
For the application of face recognition, detection of face is very important and the first step. After detecting face the face recognition algorithm can only be functional. Face detection itself involves some complexities for example surroundings, postures, enlightenment etc. There are some existing methodologies for detection of face. Some of them are skin colour based, characteristic or feature based (feature like mouth, nose and eyes) and neural network based. Among the above techniques, the feature based procedure is well thought-out as simplest one. The approach premeditated and applied in this thesis is the feature based face detection method. The algorithm is pretty dynamic as numerous people face can be detected at one time from an image containing many people. In this project GRAY scale image s model is used to detect the face of human being.

RGB Image:

An RGB image can be viewed as three different images(a red scale image, a green scale image and a blue scale image) stacked on top of each other, and when fed into the red, green and blue inputs of a colour monitor, it produces a colour image on the screen.

An RGB image is sometimes referred to as a true colour image as the precision with which a real-life image can be replicated has led to the nickname “true colour image.”



An RGB image is basically a M\*N\*3 array of colour pixel, where each colour pixel is associated with three values which correspond to red, blue and green colour component of RGB image at a specified spatial location.  
So, the colour of any pixel is determined by the combination of the red, green, and blue intensities stored in each colour plane at the pixel’s locationHere each colourplane is a M\*N array.

As can be seen in the above image, Pixel(A) has value (255, 0, 255) and is determined by the combination of intensities stored in the red colour plane, green colour plane and blue colour plane respectively.

Similarly, pixel(B) has value (127, 255, 0) and is determined in the same manner as pixel(A).

**Colour planes of RGB image:**  
Consider an RGB image array ‘I’ then,  
I(:, :, 1) represents the Red colour plane of the RGB image  
I(:, :, 2) represents the Green colour plane of the RGB image  
I(:, :, 3) represents the Blue colour plane of the RGB image

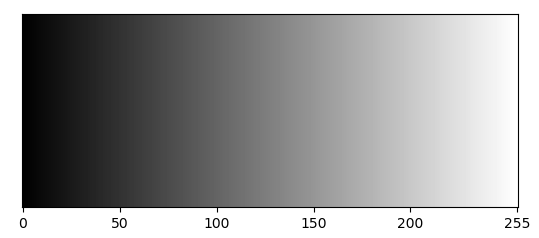
**RGB image array range:**  
if an RGB image is of class ‘double’ then each colour component is a value between 0 and 1.  
Similarly, if an RGB image is of class ‘uint8’, the range of values that each colour component can have is [0 – 255] and [0 – 65535 ] if the RGB image is of class ‘uint16’.

**Bit depth:**  
The number of bits used to store a pixel value of component image determines the bit depth of an RGB image. For example, if each colour component image is an 8-bit image, the RGB image will be said to have 24 bit deep.

**Possible number of colours in RGB image:**  
Let an RGB image is of class ‘uint8’, i.e the range of values a colour component plane can have is [0 – 255 ] ( a total of 256 shades of that colour).  
So, each individual colour plane of An RGB image is capable of showing 256 shade of that colour.  
So total number of combination of colour that can be represented in an RGB image is 256 X 256 X 256 = 16777216, approximately 16 million

Gray Scale Image:

Grayscale is a range of monochromatic shades from black to white. Therefore, a grayscale image contains only shades of gray and no color.

While digital images can be saved as grayscale (or black and white) images, even color images contain grayscale information. This is because each pixel  has a luminance value, regardless of its color. Luminance can also be described as brightness or intensity, which can be measured on a scale from black (zero intensity) to white (full intensity). Most image file formats support a minimum of 8-bit grayscale, which provides 2^8 or 256 levels of luminance per pixel. Some formats support 16-bit grayscale, which provides 2^16 or 65,536 levels of luminance.

Many image editing programs allow you to convert a color image to black and white, or grayscale. This process removes all color information, leaving only the luminance of each pixel. Since digital images are displayed using a combination of red, green, and blue (RGB) colors, each pixel has three separate luminance values. Therefore, these three values must be combined into a single value when removing color from an image. There are several ways to do this. One option is to average all luminance values for each pixel. Another method involves keeping only the luminance values from the red, green, or blue channel. Some programs provide other custom grayscale conversion algotithms that allow you to generate a black and white image with the appearance you prefer.

1.1.1 Checking for the camera

First step in the project and main step in the project , where in this step we will check for the requirements need for the camera for this project ,I am considering the primary camera of the laptop where all the operations done through with this camera.

To specify the primary camera to open in python we will first import the cv2 module.

Cv2 Is the module which contains the all the functions to play with the images for particular task. As in this project we are not bothering about all the function in that module but as of now to capture the image from the primary camera we will use VedioCapture() function which takes single argument , the argument tells which camera we are choosing as of now we are using our web cam which is primary camera we will specify it as 0.

cap=cv2.VedioCapture(0)

If there is any secondary camera, we will specify the argument value as 1.

1.2.2 Capturing the face data

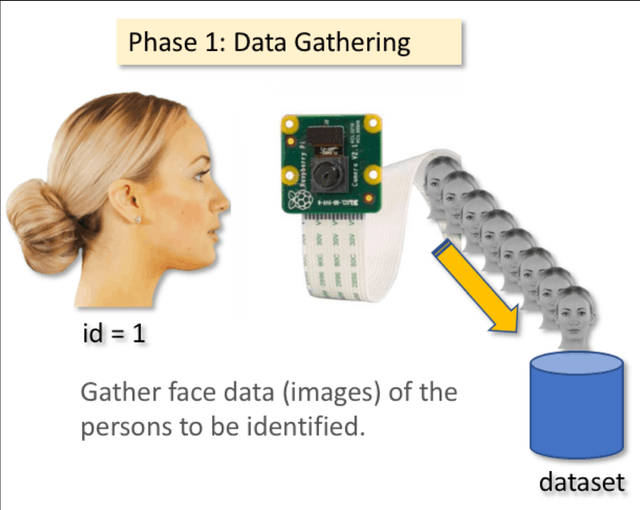
Second step of the project to collect the data(facial).From the first step we done with the setup of the camera.

In these section we will be see how actually we are going to capture the facial data .To capture the face data we want label for each face to differentiate between two faces , for this user has to enter Name and Id of particular student and name should be in alphabet to check whether the enter name is alphabet we will use isalpha() function and Id should be numeric if any one the entry is mismatched then it will throw a message to the user

If the both labels name and Id are matched with requirements then the camera starts capturing the images into data base(local system). Here threshold count of the image id 60+1(base image).this images are used for the train the model.

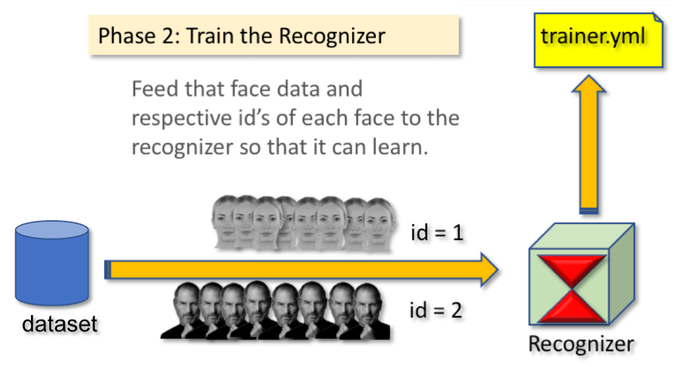
As the count of image is more than 60 then the model will become more accurate but if the number of faces is more, then the training of image will take more time but the model will be accurate.

As here we want only facial data we will use face-harcascade classifier.



1.2.3 Training the captured Data

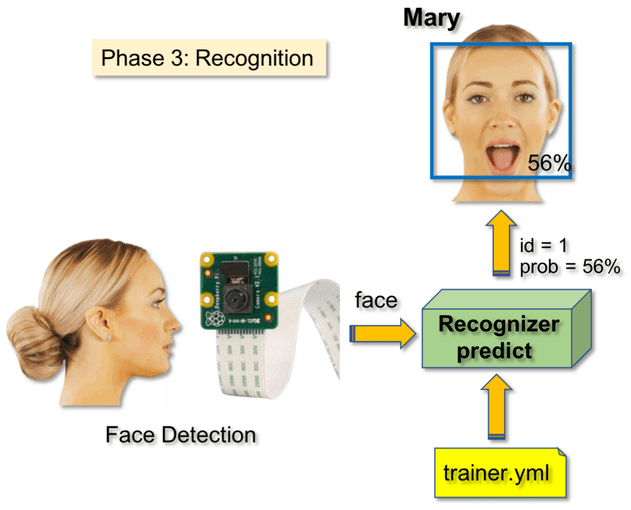
This is the phase where images are convert image data into numpy array 1-d which contains 1 row and around 1634 columns.

The numpy array contains values which represent the image. The

Trained images information will be store in trainer.yml

yml(File created in the YAML (YAML Ain't Markup Language)format, a human-readable data format used for data serialization;allows data to be written and read independent of any particularlanguage; can be incorporated into many different programminglanguages using supporting YAML libraries, including C/C++,Ruby, Python, Java, Perl, C#, PHP, and others.)

1.2.4 Validation phase

 In this phase we are trying to validate how the model is working from the real world data.

Algorithm:

Haracascade Classifier:

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle



LBPH:

LBPH:

Human beings perform face recognition automatically every day and practically with no effort.

Although it sounds like a very simple task for us, it has proven to be a complex task for a computer, as it has many variables that can impair the accuracy of the methods, for example: illumination variation, low resolution, occlusion, amongst other.

In computer science, face recognition is basically the task of recognizing a person based on its facial image. It has become very popular in the last two decades, mainly because of the new methods developed and the high quality of the current videos/cameras.

Note that face recognition is different of face detection:

* Face Detection: it has the objective of finding the faces (location and size) in an image and probably extract them to be used by the face recognition algorithm.
* Face Recognition: with the facial images already extracted, cropped, resized and usually converted to grayscale, the face recognition algorithm is responsible for finding characteristics which best describe the image.

The face recognition systems can operate basically in two modes:

* Verification or authentication of a facial image: it basically compares the input facial image with the facial image related to the user which is requiring the authentication. It is basically a 1x1 comparison.
* Identification or facial recognition: it basically compares the input facial image with all facial images from a dataset with the aim to find the user that matches that face. It is basically a 1xN comparison.

There are different types of face recognition algorithms, for example:

* Eigenfaces (1991)
* Local Binary Patterns Histograms (LBPH) (1996)
* Fisherfaces (1997)
* Scale Invariant Feature Transform (SIFT) (1999)
* Speed Up Robust Features (SURF) (2006)

Each method has a different approach to extract the image information and perform the matching with the input image. However, the methods Eigenfaces and Fisherfaces have a similar approach as well as the SIFT and SURF methods.

Today we gonna talk about one of the oldest (not the oldest one) and more popular face recognition algorithms: Local Binary Patterns Histograms (LBPH).

Objective

The objective of this post is to explain the LBPH as simple as possible, showing the method step-by-step.

As it is one of the easier face recognition algorithms I think everyone can understand it without major difficulties.

LBPH(local binary pattern histogram) algorithm which is simple yet very efficient texture operator which lables the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as binary number.

It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.Using the LBP combined with histograms we can represent the face images with a simple data vector.

Now that we know a little more about face recognition and the LBPH, let’s go further and see the steps of the algorithm:

1.Parameters: the LBPH uses 4 parameters:

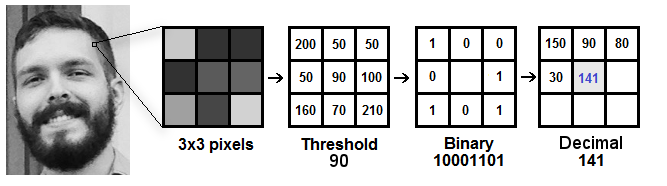
* Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
* Neighbors: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
* Grid X: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
* Grid Y: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

Don’t worry about the parameters right now, you will understand them after reading the next steps.

2. Training the Algorithm: First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let’s see the LBPH computational steps.

3. Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

The image below shows this procedure:



Based on the image above, let’s break it into several small steps so we can understand it easily:

* Suppose we have a facial image in grayscale.
* We can get part of this image as a window of 3x3 pixels.
* It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
* Then, we need to take the central value of the matrix to be used as the threshold.
* This value will be used to define the new values from the 8 neighbors.
* For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
* Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
* Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
* At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.
* Note: The LBP procedure was expanded to use a different number of radius and neighbors, it is called Circular LBP.

It can be done by using bilinear interpolation. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

4. Extracting the Histograms: Now, using the image generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids, as can be seen in the following image:

Based on the image above, we can extract the histogram of each region as follows:

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.

The LBPH algorithm is pretty much it.

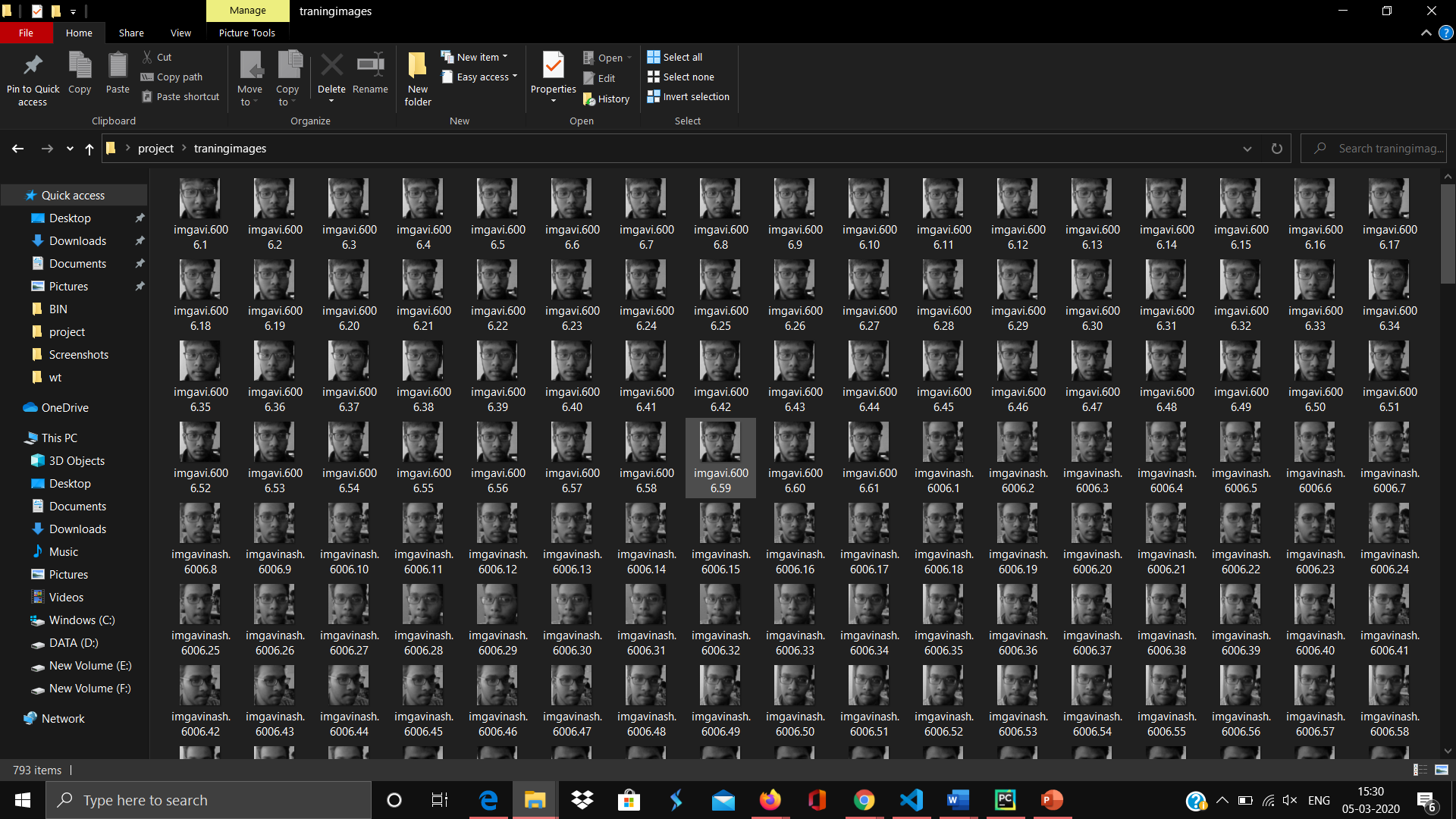
5. Performing the face recognition: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

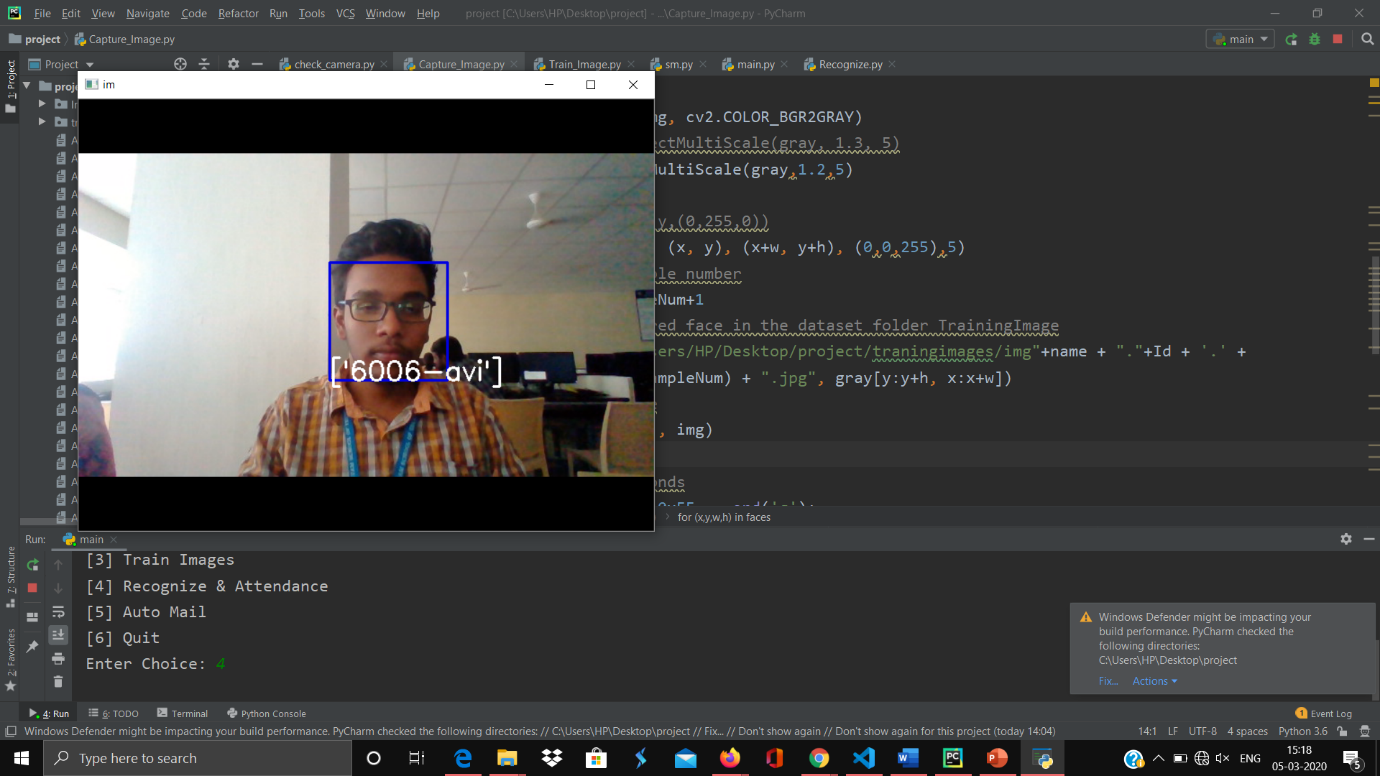
So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.

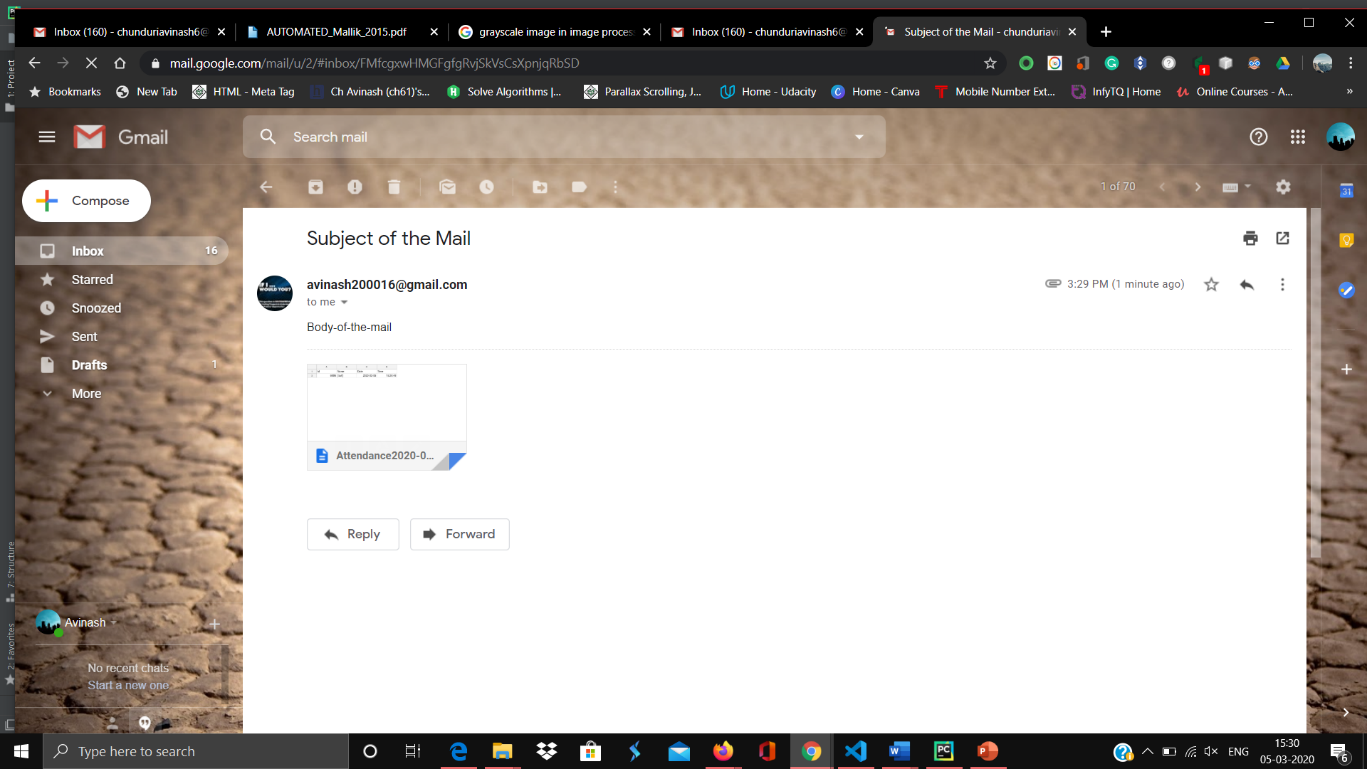
We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

So 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. Note: don’t be fooled about the ‘confidence’ name, as lower confidences are better because it means the distance between the two histograms is closer.

We can then use a threshold and the ‘confidence’ to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

Conclusion:





Future Enhancements:

1. In this system we are using database as local system but the in future we can use cloud to store the image database for processing the images
2. In this system we are just generating the mail of the excel to the particular user but in future we can modify the system in such a way that it will automatically update the particular portal.
3. And to increase the efficiency of the system we can built a neural network to increase the accuracy.

References:

1. <http://ethesis.nitrkl.ac.in/7301/1/AUTOMATED_Mallik_2015.pdf> report related to Face recognition .
2. <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>
3. <https://www.geeksforgeeks.org/matlab-rgb-image-representation/>
4. <https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml>
5. <https://en.wikipedia.org/wiki/Grayscale>