Project Report on

“Facial Recognition through a camera”

Submitted for the award of the Bachelor of Technology degree

by

Abhishek Kumar (Reg. No: 16030141CSE003)
Christopher Morgan Fernandes (Reg. No: 16030141CSE026)
Dinesh Chand Gangwar (Reg. No: 16030141CSE030)
Faizan Khan (Reg. No: 16030141CSE056)

Under the guidance of

Dr.Shekhar R. (Internal Guide)
Department, Alliance Unvieristy

ALLIANCE
UNIVERSITY

Department of Computer Science and Engineering
Alliance College of Engineering and Design
Alliance University
Bangalore

June 2020

ALLIANCE COLLEGE OF ENGINEERING & DESIGN
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Chikkahagade Cross
Chandapura - Anekal Main Road, Anekal
Bangalore – 562106

CERTIFICATE

This is to certify that the report entitled, "Facial Recognition through a camera" is a bonafide record of work done during the academic year 2019-2020 by Abhishek Kumar (Reg.No: 16030141CSE003), Christopher Morgan Fernandes (Reg.No:16030141CSE026), Dinesh Chand Gargwar (Reg.No:16030141CSE030), Faizan Khan(Reg.No:16030141CSE056) in partial fulfilment of the requirements for the award of Bachelor degree in Computer Science and Engineering, Faculty of Engineering, Alliance University.

Internal Guide

Head of the Department

ACKNOWLEDGEMENT

First and foremost, I praise and thank ALMIGTHY GOD whose blessings have bestowed in me the will power and confidence to carry out my Internship.

I would like to place my heart-felt thanks and gratitude to **Dr.Reeba Korah,** the Dean, ACED, and our Head of the Department **Dr.Shekhar R**, for their constant guidance and encouragement in completing this project work.

I feel it a pleasure to be indebted to my Project Guide **Dr.Shekhar R,** for his invaluable support, advice, and encouragement.

I also thank all the staff members of the Department for extending their helping hands to make this Project a success.

I would also like to thank all my friends and my parents who have prayed and helped me during the Project work.

SYNOPSIS

Facial recognition is the process of identifying or verifying the identity of a person using their face. It captures, analyses and compares histogram patterns based on the person's facial features.

1. The **face detection** process is an essential step as it detects and locates human faces in images and videos.
2. The **face capture** process transforms analogue information (a face) into a set of digital information (data) based on the person's facial features.
3. Using the LBPH pattern of a histogram are able to predict the person face through webcam using the LBPH. Face Recognizer to train the model.
4. The **face match** process verifies if two faces belong to the same person or not and able to show the name above the face detected box.

Today it's considered to be the most natural of all biometric measurements.

What are the technologies that work with facial recognition?

First, we require a dataset using Python Open-cv application that can uniquely capture the faces of a person by detecting using Voila jones Algorithm from features extraction using Eyes, Nose, Lips facial Landmark.

After capturing the face dataset train the face pattern using LBHP Face recognizer so we can able to get the histogram of all face edges so can able to compare with the other person faces.

Python is also common amongst the research community.

We are using PyCharm platform for the Development to include the parallel programming process.

You could use python linked with a good camera like the raspberry pi. Coding can be done in visual studio or any other IDE (integrated development environment).

There are some other light tools too but are designed for limited purposes.

LIST OF FIGURES

CONTENT

1. INTRODUCTION

A facial recognition system is a task of identifying an already detected object as known as an unknown face. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analyzing patterns based on the person's facial textures and shape. Facial recognition a term widely establishing its name in the technical advancement whether it is mobile phones or other smart device or computers, facial recognition is now integrated among everything which has a human interactive interface .today we are all surrounded by technology no matter how small or remote location a piece of tech is always around us and facial recognition is easily be used with any sort of tech for example mobiles phones are now using face lock to make it more secure.

Big or small organisations all are shifting to facial recognition for storing their information securely it allows only limited access to the data and provides high security.

What is Face Detection?

The definition of face detection refers to computer technology that is able to identify the presence of people's faces within digital images. In order to work, face detection applications use machine learning and formulas known as algorithms to detecting human faces within larger images. These larger images might contain numerous objects that aren't faces such as landscapes, buildings and other parts of humans (e.g. legs, shoulders and arms).

Face detection is a broader term than face recognition. Face detection just means that a system is able to identify that there is a human face present in an image or video. It has several applications, only one of which is facial recognition. Face detection can also be used to auto focus cameras. And it can be used to count how many people have entered a particular area. It can even be used for marketing purposes.

For example, advertisements can be displayed the moment a face is recognized.

How Face Detection Works?

While the process is somewhat complex, face detection algorithms often begin by searching for human eyes. Eyes constitute what is known as a valley region and are one of the easiest features to detect. Once eyes are detected, the

algorithm might then attempt to detect facial regions including eyebrows, the mouth, nose, nostrils and the iris. Once the algorithm surmises that it has detected a facial region, it can then apply additional tests to validate whether it has, in fact, detected a face.

What is Face Detection and Face Recognition?

One of the most important applications of face detection, however, is facial recognition. Face recognition describes a biometric technology that goes way beyond recognizing when a human face is present. It actually attempts to establish whose face it is. The process works using a computer application that captures a digital image of an individual's face (sometimes taken from a video frame) and compares it to images in a database of stored records. While facial recognition isn't 100% accurate, it can very accurately determine when there is a strong chance that an person's face matches someone in the database.

There are lots of applications of face recognition. Face recognition is already being used to unlock phones and specific applications. Face recognition is also used for biometric surveillance. Banks, retail stores, stadiums, airports and other facilities use facial recognition to reduce crime and prevent violence.

So in short, while all facial recognition systems use face detection, not all face detection systems have a facial recognition component.

1.1 DIFFERENT APPROACHES OF FACE RECOGNITION:

There are two predominant approaches to the face recognition problem: Geometric (feature-based) and photometric (view based). As researcher interest in face recognition continued, many different algorithms were developed, three of which have been well studied in face recognition literature.

Recognition algorithms can be divided into two main approaches:

1. **Geometric**: Is based on a geometrical relationship between facial landmarks, or in other words the spatial configuration of facial features. That means that the main geometrical features of the face such as the eyes, nose and mouth are first located and then faces are classified based on various geometrical distances and angles between features.

2. **Photometric stereo**: Used to recover the shape of an object from several images taken under different lighting conditions. The shape of the

recovered object is defined by a gradient map, which is made up of an array of surface normals (Zhao and Chellappa, 2006) (Figure 2)

Popular recognition algorithms include:

Local Binary Patterns Histogram (LBPH) – Training

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number.

Using the LBP combined with histograms we can represent the face images with a simple data vector.

As LBP is a visual descriptor it can also be used for face recognition task.

Viola-Jones Algorithm – Recognition

The Viola-Jones algorithm is a widely used mechanism for object detection. The main property of this algorithm is that training is slow, but detection is fast. This algorithm uses Haar basis feature filters, so it does not use multiplications.
The efficiency of the Viola-Jones algorithm can be significantly increased by first generating the integral image.
The integral image allows integrals for the Haar extractors to be calculated by adding only four numbers.
Detection happens inside a detection window. A minimum and maximum window size is chosen, and for each size a sliding step size is chosen.

Each face recognition filter (from the set of *N* filters) contains a set of cascade-connected classifiers. Each classifier looks at a rectangular subset of the detection window and determines if it looks like a face. If it does, the next classifier is applied. If all classifiers give a positive answer, the filter gives a positive answer and the face is recognized. Otherwise the next filter in the set of *N* filters is run.

Each classifier is composed of Haar feature extractors (weak classifiers). Each Haar feature is the weighted sum of 2-D integrals of small rectangular areas attached to each other. The weights may take values. Haar features relative to the enclosing detection window. Gray areas have a positive weight and white areas have a negative weight. Haar feature extractors are scaled with respect to the detection window size.

The cascade architecture is very efficient because the classifiers with the fewest features are placed at the beginning of the cascade, minimizing the total required computation. The most popular algorithm for features training is AdaBoost.

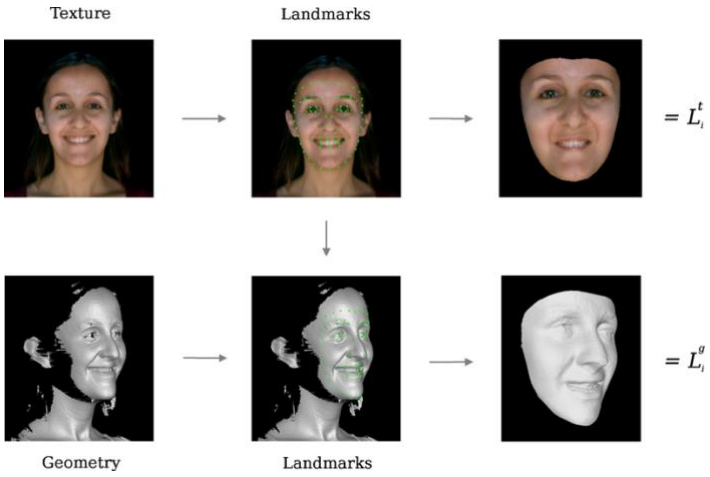

Fig1.1 Geometric image landmarks

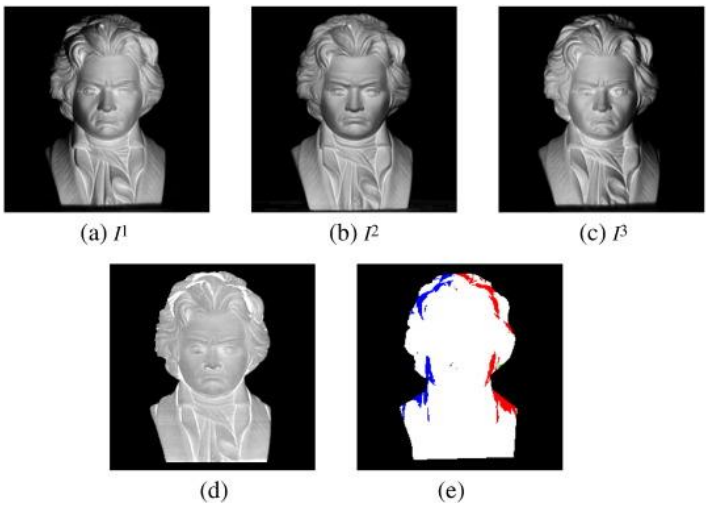

Fig1.2 Photometric stereo image

1.2 FACE DETECTION:

Face detection involves separating image windows into two classes,

One containing faces (training the background (clutter). It is difficult because although commonalities exist between faces, they can vary considerably in terms of age, skin colour and facial expression. The problem is further complicated by differing lighting conditions, image qualities and geometries, as well as the possibility of partial occlusion and disguise. An ideal face detector would, therefore, be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).

The face detection system can be divided into the following steps:-

- **Pre-Processing:** To reduce the variability in the faces, the images are processed before they are fed into the network. All positive examples that are the face images are obtained by cropping images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.

- **Classification:** Neural networks are implemented to classify the images as faces or nonfaces by training on these examples. We use both our implementation of the neural network and the Mat lab neural network toolbox for this task. Different network configurations are experimented with to optimize the results.

- **Localization:** The trained neural network is then used to search for faces in an image and if present localize them in a bounding box. Various Feature of Face on which the work has done on:- Position Scale Orientation Illumination

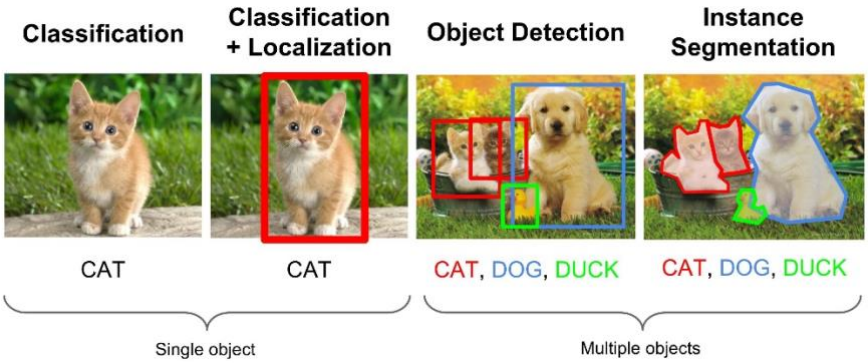

Fig 1.3 Facial detection algorithm

1.3 Survey of Literature

Face detection is a computer technology that determines the location and size of a human face in an arbitrary (digital) image. The facial features are detected and any other objects like trees, buildings and bodies etc are ignored from the digital image. It can be regarded as a _specific' case of object-class detection, where the task is finding the location and sizes of all objects in an image that belong to a given class. Face detection can be regarded as a more _general' case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). There are two types of approaches to detect facial part in the given image i.e. feature base and image-based approach. The feature-based approach tries to extract features of the image and match it against the knowledge of the face features. While the image base approach tries to get the best match between training and testing images.

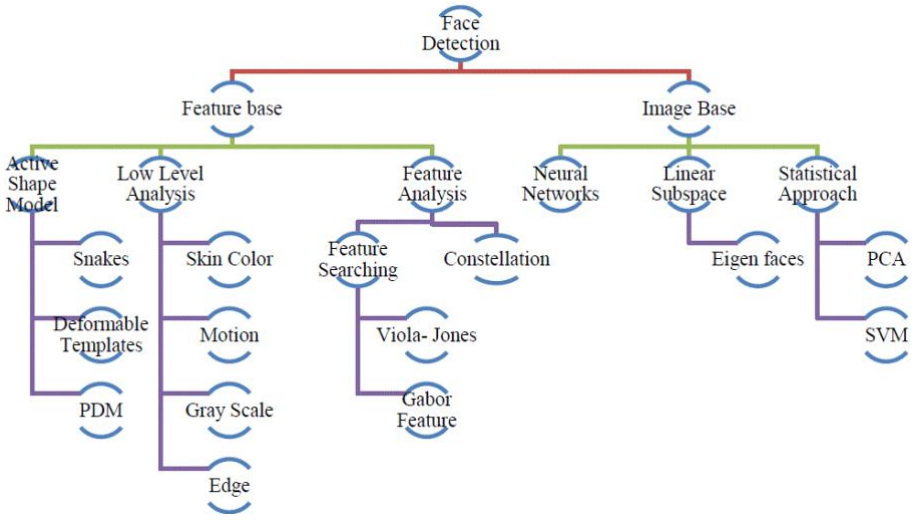

Fig1.4 Detection methods

1.4 Feature-based approach

Active Shape Model Active shape models focus on complex non-rigid features like actual physical and higher-level appearance of features Means that Active Shape Models (ASMs) are aimed at automatically locating landmark points that define the shape of any statistically modelled:

- Facial model from a training set containing images with manually annotated landmarks. ASMs is classified into three groups i.e. snakes, PDM, Deformable templates

- Snakes: The first type uses a generic active contour called snakes, first introduced by Kass et al. in 1987 Snakes are used to identify head boundaries [8,9,10,11,12]. To achieve the task, a snake is first initialized at the proximity around a head boundary. It then locks onto nearby edges and subsequently assumes the shape of the head. The evolution of a snake is achieved by minimizing an energy function, Esnake (analogy with physical systems), denoted asEsnake = Einternal + EExternal WhereEinternal and EExternal are internal and external energy functions. Internal energy is the part that depends on the intrinsic properties of the snake and defines its natural evolution. The typical natural evolution in snakes is shrinking or expanding. The external energy counteracts the

internal energy and enables the contours to deviate from the natural evolution and eventually assume the shape of nearby features—the head boundary at a state of equilibria. Two main consideration for forming snakes i.e. selection of energy terms and energy minimization. Elastic energy is used commonly as internal energy. Internal energy varies with the distance between control points on the snake, through which we get contour an elastic-band characteristic that causes it to shrink or expand. On the other side external energy relay on image features. Energy minimization process is done by optimization techniques such as the steepest gradient descent. Which needs the highest computations. Huang and Chen and Lam and Yan both employ fast iteration methods by greedy algorithms. Snakes have some demerits like contour often becomes trapped onto false image features and another one is that snakes are not suitable in extracting non-convex features.

1.4.1 Deformable Templates:

Deformable templates were then introduced by Yuille et al. to take into account the a priori of facial features and to better the performance of snakes. Locating a facial feature boundary is not an easy task because the local evidence of facial edges is difficult to organize into a sensible global entity using generic contours. The low brightness contrast around some of these features also makes the edge detection process. Yuille et al. took the concept of snakes a step further by incorporating global information of the eye to improve the reliability of the extraction process. Deformable templates approaches are developed to solve this problem. Deformation is based on the local valley, edge, peak, and brightness. Other than face boundary, salient feature (eyes, nose, mouth and eyebrows) extraction is a great challenge of face recognition.

E = Ev + Ee + Ep + Ei + E internal ;

where Ev , Ee , Ep , Ei , E internal are external energy due to valley, edges, peak and image brightness and internal energy

1.4.2 PDM (Point Distribution Model):

Independently of computerized image analysis, and before ASMs were developed, researchers developed statistical models of shape. The idea is that once you represent shapes as vectors, you can apply standard statistical methods to them just like any other multivariate object. These models learn allowable constellations of shape points from training examples and use principal components to build what is called a Point Distribution Model. These have been used in diverse ways, for example for categorizing Iron Age broaches. Ideal Point Distribution Models can only deform in ways that are characteristic of the object. Cootes and his colleagues were seeking models which do exactly that so if a beard, say, covers the chin, the shape model can override the image" to approximate the position of the chin under the beard. It was therefore natural (but perhaps only in retrospect) to adopt Point Distribution Models. This synthesis of ideas from image processing and statistical shape modelling led to the Active Shape Model. The first parametric statistical shape model for image analysis based on principal components of inter-landmark distances was presented by Cootes and Taylor in. On this approach, Cootes, Taylor, and their colleagues, then released a series of papers that cumulated in what we call the classical Active Shape Model.

1.4.3 Low-level analysis

Based on low-level visual features like colour, intensity, edges, motion etc. Skin Color BaseColor is the avital feature of human faces. Using skin-colour as a feature for tracking a face has several advantages. Colour processing is much faster than processing other facial features. Under certain lighting conditions, colour is orientation invariant. This property makes motion estimation much easier because only a translation model is needed for motion estimation. Tracking human faces using colour as a feature has several problems like the colour representation of a face obtained by a camera is influenced by many factors (ambient light, object movement, etc

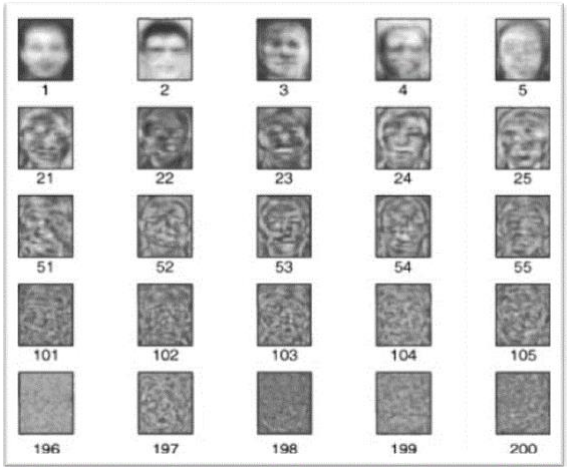

Fig1.5 Factors on low-level visual features

Majorly three different face detection algorithms are available based on RGB, YCbCr, and HIS colour space models. In the implementation of the algorithms, there are three main steps viz.

(1) Classify the skin region in the colour space,

(2) Apply threshold to mask the skin region and

(3) Draw the bounding box to extract the face image.

Crowley and Coutaz suggested the simplest skin colour algorithms for detecting skin pixels. The perceived human colour varies as a function of the relative direction to the illumination.

The pixels for skin region can be detected using a normalized colour histogram and can be normalized for changes in intensity on dividing by luminance. Converted an [R, G, B] vector is converted into an [r, g] vector of normalized colour which provides a fast means of skin detection. This algorithm fails when there is some more skin region like legs, arms, etc. Cahi and Ngan [27] suggested skin colour classification algorithm with YCbCr colour space. The research found that pixels belonging to skin region having similar Cb and Cr values. So that the thresholds be chosen as [Cr1, Cr2] and [Cb1, Cb2], a pixel is classified to have skin tone if the values [Cr, Cb] fall within the thresholds. The skin colour distribution gives the face portion in the colour image. This algorithm is also having the constraint that the image should be having only face as the skin region. Kjeldson and Kender defined a colour predicate HSV colour space to separate skin region from the background. Skin colour classification in HSI colour space is the same as YCbCr

colour space but here the responsible values are hue (H) and saturation (S). Similar to above the threshold be chosen as [H1, S1] and [H2, S2], and a pixel is classified to have skin tone if the values [H, S] fall within the threshold and this distribution gives the localized face image. Similar to the above two algorithms this algorithm is also having the same constraint.

1.5 Motion base

When the use of video sequence is available, motion information can be used to locate moving objects. Moving like face and body parts can be extracted by simply thresholding accumulated frame differences. Besides face regions, facial features can be located by frame differences.

1.5.1 Gray Scale Base

Grey information within a face can also be treat as important features. Facial features such as eyebrows, pupils, and lips appear generally darker than their surrounding facial regions. Various recent feature extraction algorithms search for local grey minima within segmented facial regions. In these algorithms, the input images are first enhanced by contrast-stretching and grey-scale morphological routines to improve the quality of local dark patches and thereby make detection easier. The extraction of dark patches is achieved by low-level grey-scale thresholding. Based method and consist of three levels. Yang and huang presented new approach i.e. face grayscale behaviour in pyramid (mosaic) images. This system utilizes a hierarchical Face location consist of three levels. Higher two-level based on mosaic images at a different resolution. In the lower level, the edge detection method is proposed. Moreover, these algorithms give a fine response in the complex background where the size of the face is unknown Edge Base:

Face detection based on edges was introduced by Sakai et al. This work was based on analyzing line drawings of the faces from photographs, aiming to locate facial features. Then later Craw et al. proposed a hierarchical framework based on Sakai work to trace a human head outline. Then after remarkable works were carried out by many researchers in this specific area. The method suggested by Anila and Devarajan was very simple and fast. The proposed framework which consists of three steps i.e. initially the images are enhanced by applying the median filter for noise removal and histogram equalization for contrast adjustment. In the second step, the edge image is constructed from the enhanced image by applying a Sobel

operator. Then a novel edge tracking algorithm is applied to extract the sub-windows from the enhanced image based on edges. Further, they used the Backpropagation Neural Network (BPN) algorithm to classify the sub-window as either face or non-face.

1.6 Feature Analysis

These algorithms aim to find structural features that exist even when the pose, viewpoint, or lighting conditions vary and then use these to locate faces. These methods are designed mainly for face localization

1.6.1 Feature Searching

- **Viola-Jones Method:**

  Paul Viola and Michael Jones presented an approach for object detection which minimizes computation time while achieving high detection accuracy. Paul Viola and Michael Jones [39] proposed a fast and robust method for face detection which is 15 times quicker than any technique at the time of release with 95% accuracy at around 17 fps. The technique relies on the use of simple Haar-like features that are evaluated quickly through the use of new image representation. Based on the concept of an ―Integral Image‖ it generates a large set of features and uses the boosting algorithm AdaBoost to reduce the overcomplete set and the introduction of a degenerative tree of the boosted classifiers provides for robust and fast interferences. The detector is applied in a scanning fashion and used on grey-scale images, the scanned window that is applied can also be scaled, as well as the features evaluated.

- **Gabor Feature Method:**
- Sharif et al proposed an Elastic Bunch Graph Map (EBGM) algorithm that successfully implements face detection using Gabor filters. The proposed system applies 40 different Gabor filters on an image. As are a result of which 40 images with different angles and orientation nare received. Next, maximum intensity points in each filtered image are calculated and mark them as fiducial points. The system reduces these points following the

distance between them. The next step is calculating the distances between the reduced points

- using the distance formula. At last, the distances are compared with the database. If the match occurs, it means that the faces in the image are detected. Equation of Gabor filter [40] is shown below`

$$\psi_{u,v}(z) = \frac{\| k_{u,v} \|^2}{\sigma^2} e^{\left(\frac{\|k_{u,v}\|^2 \|z\|^2}{2\sigma^2} \right)} \left[e^{i\vec{k}_{u,v} z} - e^{-\frac{\sigma^2}{2}} \right]$$

Where

$\phi_u = \frac{u\pi}{8}$, $\phi_u \in [0, \pi)$ gives the frequency,

Fig 1.6 Equation of Gabor Filter

- **Constellation method:**

  All methods discussed so far are able to track faces but still some issue like locating faces of various poses in the complex background is truly difficult. To reduce this difficulty investigator, form a group of facial features in face-like constellations using more robust modelling approaches such as statistical analysis. Various types of face constellations have been proposed by Burl et al. They establish the use of statistical shape theory on the features detected from a multiscale Gaussian derivative filter. Huang et al. also apply a Gaussian filter for pre-processing in a framework based on image feature analysis. Image Base Approach.

1.7 Advatages and Disadvantages of PCA

The advantages of PCA are listed below:

1) Lack of redundancy of data given the orthogonal components.

2) Reduced the complexity in face images grouping with the use of PCA

3) Smaller database representation since only the trainee images are stored in the form of their projections on a reduced basis.

4) Noise reduction since the maximum variation basis is chosen and so the small variations in the back ground are ignored automatically.

5)2DPCA over 1DPCA is that the feature vector is now two-dimensional so the problem of dimensionality is greatly reduced.

2. RESEARCH AND ANALYSIS

Before starting this project we have conducted detailed research, by referring to several recent research articles related to this domain. The research details are summarized for the implementation of it in our project. A thorough analysis of these research details has helped us to develop the methodology scientifically.

2.1 Our Approach

The proposed method has several steps and the final purpose is to extract eight features for each face type. These features will be used later in the classification stage. The features (distances) extracted are described in Facial expression classification: An approach based on the fusion of facial deformations using the transferable belief model.

The five types of distances:

1. Eye-opening distance between upper and lower eyelids
2. Distance between the interior corner of the eye and the interior corner of the eyebrow/
3. Mouth opening width, the distance between the left and right mouth corners
4. Mouth opening height, the distance between the upper and lower lip
5. Distance between a corner of the mouth and the corresponding external eye corner.

   To extract these distances the following sequence of steps were followed, steps which will be detailed later in this article:

- Extract the face using Haar classifiers from OpenCV library
- Rotate the face so the line that connects the eyes is always horizontal
- For each eye, identify the exact eye contour and approximate this contour using Bezier curves.
- Extract three features (distances) for each eye and the two features related to the mouth.

2.2 Implementing Face Recognition in 2 minutes

Did you find yourself surprised when you found out Facebook automatically tags your friends in your pictures? It was then we realized that machines have

become much smarter nowadays. Our face conveys a lot of information including our emotional state. It has also shown its application in the domain of security where facial recognition helps in the identification of any human.

In human beings, it is the temporal lobe of the brain which is responsible for recognition of faces. In machine learning, the machine is fed a lot of images of face which the machine trains. When a test image is given, the model tries to match it with existing images.

Following are the use cases where Face Recognition is used:

- **Fraud Detection for Passports and Visas** — The Australian passport office uses automatic face recognition software and according to reports, this system is more efficient than a human detecting fraud.

- **Identification of Criminals** — The law enforcement agencies have started implementing facial recognition system to improve the quality of investigation.

- **Track attendance** — Few organizations use a facial recognition system to track attendance of their employees.

There have been many implementations in this domain especially DeepFace by Facebook and FaceNet by Google.

But for a beginner who is trying to implement it for some project or want to explore facial recognition need not learn the advance implementation. Here is my attempt to implement face recognition in a simple way using the local binary pattern.

2.2.1 Local Binary Pattern (LBP)

Local Binary Patterns Histogram algorithm was proposed in 2006. It is based on the local binary operator. It is widely used in facial recognition due to its computational simplicity and discriminative power.

Why Local Binary Pattern?

- LBPH Method is one of the best performing texture descriptors.
- The LBP operator is robust against monotonic greyscale transformations.
- In LBPH each image is analysed independently, while the eigenfaces and fisher face method looks at the dataset as a whole.
- LBPH method will probably work better than fisher faces in different environments and light conditions. It also depends on our training and testing data sets.
- LBPH can recognise both side and front faces.

Since this blog focuses on the implementation of LBP, we will move straight to the implementation.

We will first import the libraries required for our code;

```
import cv2
import os
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
```

cv2 — LBPH algorithm is a part of opencv

os — For specifying directory path

matplotlib — For visualizing images

numpy — For passing labels in train function

%matplotlib inline — Visualizing the plots in jupyter

Detect Face:

```
def detect_faces_predict(img):
    gray_image = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    lbp_cascade_face = cv2.CascadeClassifier('lbpcascade_frontalface.xml')

    faces = lbp_cascade_face.detectMultiScale(gray_image, scaleFactor = 1.2, minNeighbors = 5)
    if(len(faces)==0):
        return None,None
    (x,y,w,h) = faces[0]

    return gray_image[y:y+w, x:x+h] , faces[0]
```

Time to break down the function

In OpenCV, face detection is implemented on gray images. We need to convert our coloured image into gray image first

```
gray_image = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

The brain behind LBP classifier is the knowledge file. Since we are detecting face, the frontal face knowledge file is required.

Link to the xml file: https://github.com/opencv/opencv/blob/master/data/lbpcascades/lbpcascade_frontalface.xml

```
lbp_cascade_face = cv2.CascadeClassifier('lbpcascade_frontalface.xml')
```

We use the *detectMultiScale* method to detect faces in the gray image. This returns coordinates of the detected face.

```
faces = lbp_cascade_face.detectMultiScale(gray_image,
   scaleFactor = 1.2, minNeighbors = 5)
```

If no face is found, we return None, else we return the coordinates of the face detected.

```
if(len(faces)==0):
      return None,None
   (x,y,w,h) = faces[0]
```

To check if the function is working properly, we will pass a image and then plot the detected face.

```
detect_image = sample(lbp_cascade_face, test_image)
plt.imshow(convertToRGB(detect_image))
```

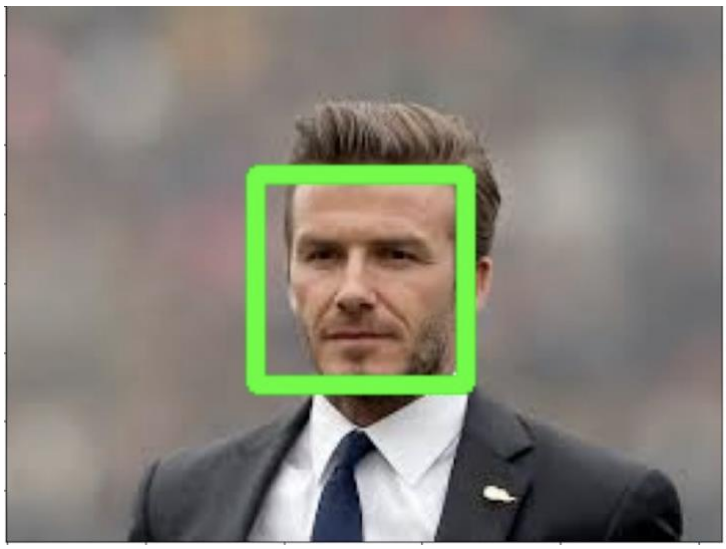

Fig2.1 David Beckham's face plotted

Looks like it is working properly, Let's move on to data preparation step

Prepare the data:

Before writing the code, let me explain how our directory looks

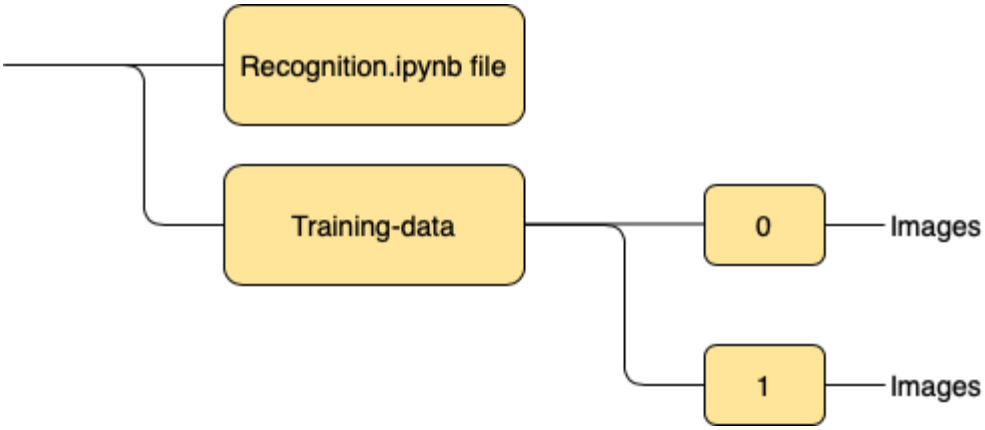

Fig2.2 Directory for preparing the data

‘Training-data’ folder contains 2 folders 0 and 1

Note : The folder name will act as a label for our model which is the reason why we are naming it 0 and 1

0 folder contains Barack Obama pictures whereas 1 folder contains David Beckham pictures.

```
def prepare_data(data_path):
    faces = []
    labels = []
    dir_path = os.listdir(data_path)
    for dir_name in dir_path:
        label = int(dir_name)
        sub_dir_path = data_path + "/" + dir_name
        sub_dir = os.listdir(sub_dir_path)
        for image_dir_path in sub_dir:
            image_path = sub_dir_path + "/" + image_dir_path
            img = cv2.imread(image_path)
            face , rect = detect_faces(img)
            #check if face is present
            if face is not None:
                #append the faces and label
                faces.append(face)
                labels.append(label)
    return faces , labelsfaces , labels = prepare_data('training-data')
```

Let’s break down the code:

We first initialize the faces and labels to null.

```
faces = []
labels = []
```

We get the required directory path using os

```
dir_path = os.listdir(data_path)
```

Since we have two sub folders, we have to loop in the directory

```
for dir_name in dir_path:
```

We now find the sub directory path using os

```
sub_dir_path = data_path + "/" + dir_name
sub_dir = os.listdir(sub_dir_path)
```

In order to read every image inside the folders, we enter the loop inside each sub directory. We then give each image it's path and then read it.

```
for image_dir_path in sub_dir:
    image_path = sub_dir_path + "/" + image_dir_path
    img = cv2.imread(image_path)
```

We will now get our detected face by passing image in *detect_faces function*

```
face , rect = detect_faces_predict(img)
```

Check if face is present then return face and the label

```
if face is not None:
    faces.append(face)
    labels.append(label)
```

Before defining our model, let's prepare our data

```
faces , labels = prepare_data('training-data')
```

Face Recognizer Model:

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
```

The *LBPHFaceRecognizer_create* function helps in creating face recognizer model from LBPH

It is time to train our model on our data

```
recognizer.train(faces , np.array(labels))
```

Let's give our label appropriate names. Since 0 folder contains Barack Obama pictures and 1 folder contains David Beckham pictures.

```
name = [ 'Barack Obama' , 'David Beckham']
```

Face Prediction:

We have reached the final step of our code

```
def predict_face(test_image):
    img = test_image.copy()
    face , rect = detect_faces(img)
    label = recognizer.predict(face)
    final_name = name[label[0]]
    (x,y,w,h) = rect    final_image = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    cv2.rectangle(final_image,(x,y),(x+w,y+h),(0,255,0),2)
    cv2.putText(final_image,final_name,(x,y-
    5),cv2.FONT_HERSHEY_PLAIN,1.2,(0,255,0),2)

    return final_image
```

Let's break down the function

We first create a copy of our test_image and pass that image in our *detect_faces function to get image and coordinates of the rectangle*

```
img = test_image.copy()
face , rect = detect_faces(img)
```

We used the *predict* function of our recognizer to predict the face and return the predicted label. We then use our *name* list to get appropriate names.

```
label = recognizer.predict(face)
final_name = name[label[0]]
```

cv2.rectangle function helps in plotting rectangle around the detected face and

cv2.putText function helps in writing down the label name above the rectangle

```
final_image = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
cv2.rectangle(final_image,(x,y),(x+w,y+h),(0,255,0),2)
cv2.putText(final_image,final_name,(x,y-
   5),cv2.FONT_HERSHEY_PLAIN,1.2,(0,255,0),2)
```

Time to test the model using a test image

```
test_image = cv2.imread('test.jpg')
final_image = predict_face(test_image)
plt.figure(figsize=(10,10))
plt.imshow(final_image)
```

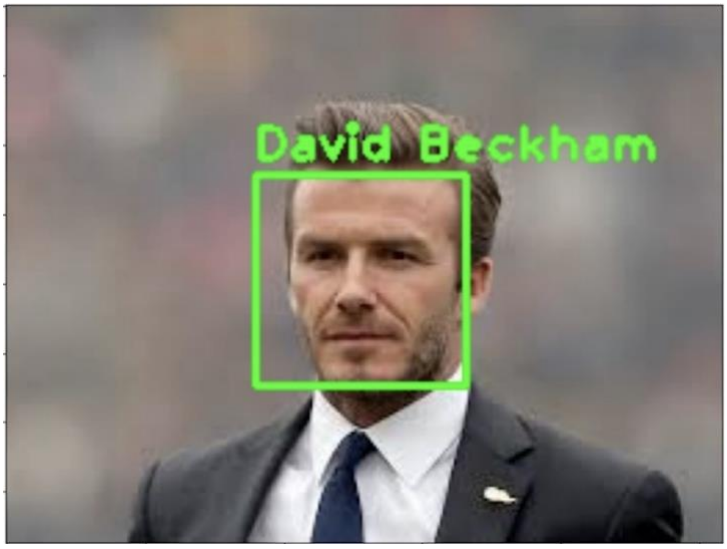

Fig2.3 David Beckham recognition successful

We have successfully implemented Face Recognition using LBPH algorithm.

2.2.2 Understanding LBPH Algorithm

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.

- Neighbours: 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.

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 parameter's radius and neighbours.

The image below shows this procedure:

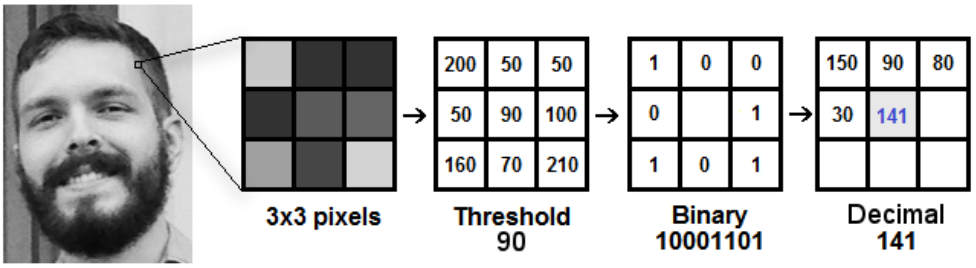

Fig2.4 LBP image procedure

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 neighbours.
- For each neighbour 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 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 neighbours, it is called Circular LBP.

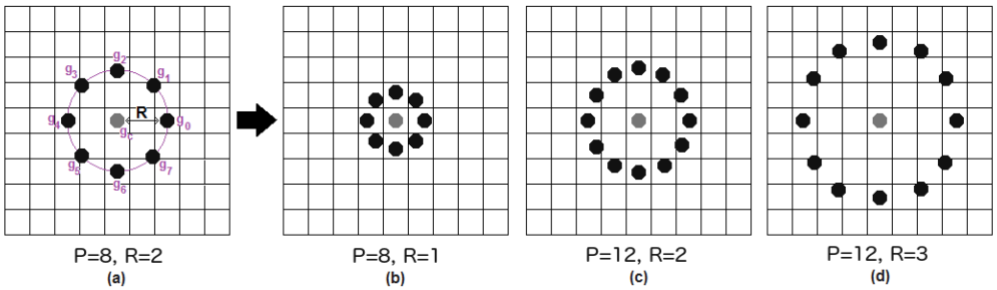

Fig2.4 Circular LBP

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:

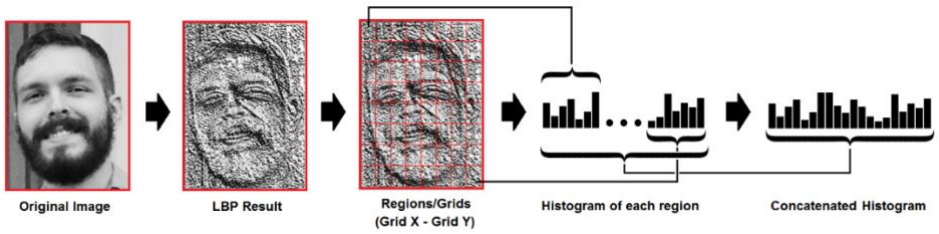

Fig2.5 Extracting Histogram

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 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:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

Fig2.6 Euclidean distance 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.

Conclusions

- LBPH is one of the easiest face recognition algorithms.
- It can represent local features in the images.
- It is possible to get great results (mainly in a controlled environment).
- It is robust against monotonic greyscale transformations.
- It is provided by the OpenCV library.

2.3 Functional Requirements

- The system should be developed using python programming on PyCharm framework on Windows Operating System.
- Once the user can upload this data on the .csv file format
- The user can covert .csv file format into ARTF file format (Attribute relation file format) CSV-(Comma separated value)
- This system must have a processing module which has to clear any unnecessary data.
- User is also able to divide the processed data into a training dataset and test the dataset in the data preparation module

2.4 Non-Functional Requirements

- It should be supported with good lighting and conditions which means face should be illuminated all the time.
- It should have the user interface for easy operation.
- Aesthetically, well built

3. SYSTEM DESIGN

3.1 Introduction

Face Recognition is a computer application that is capable of detecting, tracking, identifying or verifying human faces from an image or video captured using a digital camera. Although a lot of progress has been made in the domain of face detection and recognition for security, identification and attendance purpose, still issues are hindering the progress to reach or surpass human-level accuracy. These issues are variations in human facial appearance such as; varying lighting condition, noise in face images, scale, pose etc. This research paper presents a new method using Local Binary Pattern (LBP) algorithm combined with advanced image processing techniques such as Contrast Adjustment, Bilateral Filter, Histogram Equalization and Image Blending to address some of the issues hampering face recognition accuracy to improve the LBP codes, thus improve the accuracy of the overall face recognition system. Our experiment results show that our method is very accurate, reliable and robust for a face recognition system that can be practically implemented in a real-life environment as an automatic attendance management system.

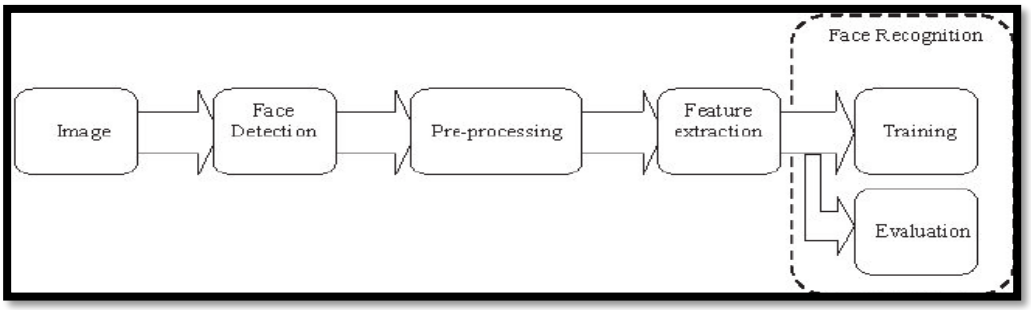

Fig.3.1 System architecture

It is important to complete all tasks and meet deadlines. Many project management tools are available to help project managers manage their tasks and schedule and one of them is the flowchart.

A flowchart is one of the seven basic quality tools used in project management and it displays the actions that are necessary to meet the goals of a particular task in the most practical sequence. Also called as process maps, this type of tool displays a series of steps with branching possibilities that depict one or more inputs and transforms them into outputs. The advantage of flowcharts is that they show the activities involved in a project including the decision points, parallel paths, branching loops as well as the overall sequence of processing through mapping the operational details within the horizontal value chain. Moreover, this particular tool is very used in estimating and understanding the cost of quality for a particular process. This is done by using the branching logic of the workflow and estimating the expected monetary returns

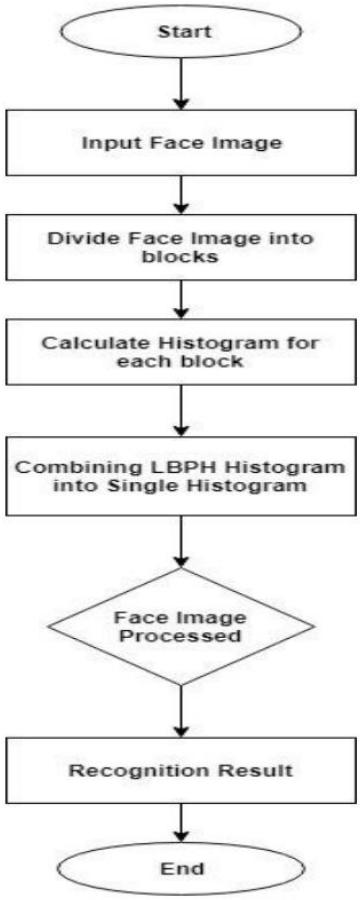

Fig.3.2 Flow diagram

3.2 Use case diagrams :

A use case is a set of scenarios that describing an interaction between a source and a destination. A use case diagram displays the relationship between actors and use cases. The two main components of a use case diagram are use cases and actors. shows the use case diagram.

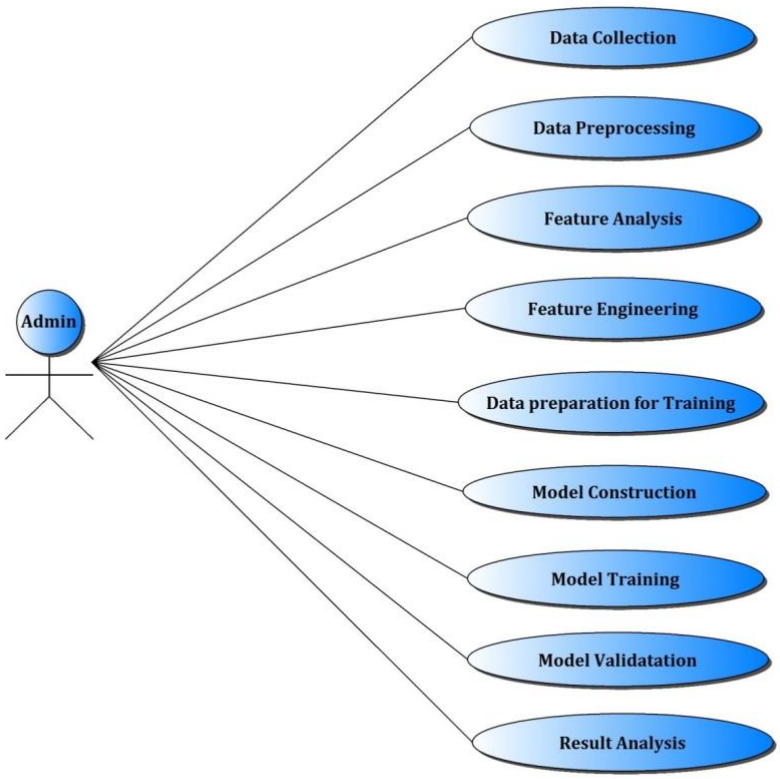

Fig.3.3 Use case diagram

3.3 Data flow diagrams :

0-level DFD:

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

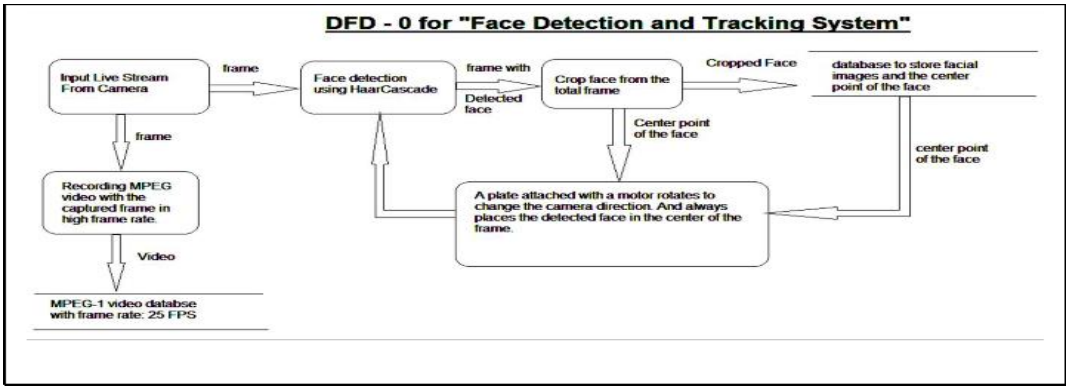

Fig.3.4 Data flow diagram – Level 0

1-level DFD: In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes.in this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.

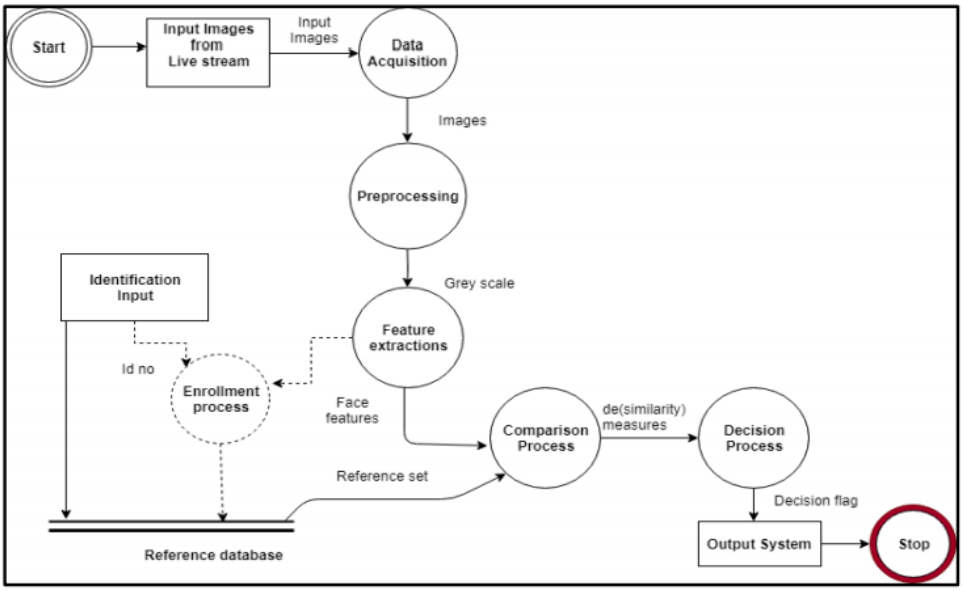

Fig.3.4 Data flow diagram – Level 1

2-level DFD:

2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system's functioning.

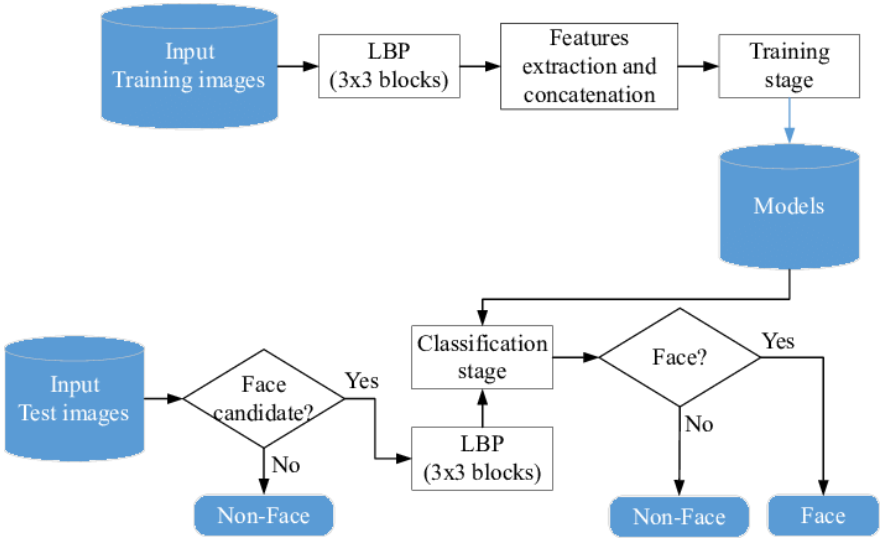

Fig.3.6 System Implementation

3.4 Python Introduction

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive** − you can sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Features:

Python's features include – All these.

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

3.5 Python Libraries

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now.

There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

The second library we use is Numpy

Numpy - NumPy is a python library used for working with arrays. It also has functions for working in the 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 an array, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

3.6 OS Module in Python

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.

3.7 Why python emerging as a leader:

There's a battle out there happening in the minds of aspiring data scientists to choose the best data science tool. Though there are quite some data science tools

that provide the much-needed option, the close combat narrows down between two popular languages Python and R.

Between the two, Python is emerging as the popular language used more in data science applications.

Take the case of the tech giant Google that has created the deep learning framework called tensor flow – Python is the primary language used for creating this framework. Its footprint has continued to increase in the environment promoted by Netflix. Production engineers at Face book and Khan Academy have for long been using it as a prominent language in their environment.

Python has other advantages that speed up it's an upward swing to the top of data science tools. It integrates well with the most cloud as well as platform-as-a-service providers. In supporting multiprocessing for parallel computing, it brings the distinct advantage of ensuring large-scale performance in data science and machine learning. Python can also be extended with modules written in C/C++.

3.7.1 Where Python becomes the perfect fit:

There are tailor-made situations where it is the best data science tool for the job. It is perfect when data analysis tasks involve integration with web apps or when there is a need to incorporate statistical code into the production database. The full-fledged programming nature of Python makes it a perfect fit for implementing algorithms.Its packages rooted for specific data science jobs. Packages like Numpy, SciPy, and pandas produce good results for data analysis jobs. While there is a need for graphics, Python's matplotlib emerges as a good package, and for machine learning tasks, sci-kit-learn becomes the ideal alternate.

3.7.2 Why is Python preferred over other data science tools?

It is 'Pythonic' when the code is written in a fluent and natural style. Apart from that, it is also known for other features that have captured the imaginations of the data science community.

Easy to learn:

The most alluring factor of Python is that anyone aspiring to learn this language can learn it easily and quickly. When compared to other data science

languages like R, Python promotes a shorter learning curve and scores over others by promoting an easy-to-understand syntax.

Scalability:

When compared to other languages like R, Python has established a lead by emerging as a scalable language, and it is faster than other languages like Matlab and Stata. Python's scalability lies in the flexibility that it gives to solve problems, as in the case of YouTube that migrated to Python. Python has come good for different usages in different industries and rapid development of applications of all kinds.

Choice of data science libraries:

The significant factor giving the push for Python is the variety of data science/data analytics libraries made available for the aspirants. Pandas, StatsModels, NumPy, SciPy, and Scikit-Learn, are some of the libraries well known in the data science community. Python does not stop with that as libraries have been growing over time. What you thought was a constraint a year ago would be addressed well by Python with a robust solution addressing problems of a specific nature.

Python community:

One of the reasons for the phenomenal rise of Python is attributed to its ecosystem. As Python extends its reach to the data science community, more and more volunteers are creating data science libraries. This, in turn, has led the way for creating the most modern tools and processing in Python.

The widespread and involved community promotes easy access for aspirants who want to find solutions to their coding problems. Whatever queries you need, it is a click or a Google search away. Enthusiasts can also find access to professionals on Code mentor and Stack Overflow to find the right answers for their queries.

Graphics and visualization:

Python comes with varied visualization options. Matplotlib provides the solid foundation around which other libraries like Seaborn, pandas plotting, and ggplot have been built. The visualization packages help you get a good sense of data, create charts, graphical plot and create web-ready interactive plots.

3.7.3 Why Choose Python?

If you're going to write programs, there are dozens of commonly used languages to choose from. Why choose Python? Here are some of the features that make Python an appealing choice.

Python is Popular

Python has been growing in popularity over the last few years. The 2018 Stack Overflow Developer Survey ranked Python as the 7th most popular and the number one most wanted technology of the year. World-class software development countries around the globe use Python every single day.

According to research by Dice Python is also one of the hottest skills to have and the most popular programming language in the world based on the. The popularity of Programming Language Index

Due to the popularity and widespread use of Python as a programming language, Python developers are sought after and paid well. If you'd like to dig deeper into Python salary statistics and job opportunities, you can do so here.

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to

a human user. The expediency of coding in an interpreted language is typically worth it for most applications.

Python is Free

The Python interpreter is developed under an OSI-approved open-source license, making it free to install, use, and distribute, even for commercial purposes. A version of the interpreter is available for virtually any platform there is, including all flavours of Unix, Windows, macOS, smartphones and tablets, and probably anything else you ever heard of. A version even exists for the half dozen people remaining who use OS/2.

Python is Portable

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.)

Python is Simple

As programming languages go, Python is relatively uncluttered, and the developers have deliberately kept it that way. A rough estimate of the complexity of a language can be gleaned from the number of keywords or reserved words in the language. These are words that are reserved for special meaning by the compiler or interpreter because they designate specific built-in functionality of the language. Python 3 has 33 keywords, and Python 2 has 31. By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter examples probably vary somewhat by implementation or dialect. Python code has a simple and clean structure that is easy to learn and easy to read. As you will see, the language definition enforces code structure that is easy to read.

But It’s Not That Simple

For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.

Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

3.7.4 Is Python 'the' tool for machine learning?

When it comes to data science, machine learning is one of the significant elements used to maximize value from data. With Python as the data science tool, exploring the basics of machine learning becomes easy and effective. In a nutshell, machine learning is more about statistics, mathematical optimization, and probability. It has become the most preferred machine learning tool in the way it allows aspirants to 'do the math' easily.

Name any math function, and you have a Python package meeting the requirement. There is Numpy for numerical linear algebra, CVXOPT for convex optimization, Scipy for general scientific computing, SymPy for symbolic algebra, PYMC3, and Statsmodel for statistical modelling.

With the grip on the basics of machine learning algorithm including logistic regression and linear regression, it makes it easy to implement machine learning systems for predictions by way of its sci-kit-learn library. It's easy to customize for neural networks and deep learning with libraries including Keras, Theano, and TensorFlow.

Data science landscape is changing rapidly, and tools used for extracting value from data science have also grown in numbers. The two most popular languages that fight for the top spot are R and Python. Both are revered by enthusiasts, and both come with their strengths and weaknesses.

4. Test Result and Experiment

4.1 Getting Images through the Webcam

1. We**,** Will, open the camera through Python-OpenCV program to get the images from video streaming.
2. From here we will get the image of the person to continue the further process regarding the Face detection part.
3. Saving around 30 image for the person to do desired folder so we can to identify the person id as for the names we entered.

```
import os
import cv2
def createFolder(directory):
    try:
        if not os.path.exists(directory):
            os.makedirs(directory)
    except OSError:
        print('Error: Creating directory. ' + directory)
newname=input("enter folder name")
bb="trainingImages/"
folder_name=bb+newname
createFolder(folder_name)
folderpath = os.path.abspath(folder_name)
a = folderpath
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
face_id = input("Enter the object name")
direct = face_id
path = os.path.join(a,direct)
cap=cv2.VideoCapture(0)
count = 0
while True:
    ret,test_img=cap.read()
    if not ret :
```

```
        continue
    cv2.imwrite(path+"%d.jpg" % count, test_img)     # save frame as JPG file
    count += 1
    resized_img = cv2.resize(test_img, (1000, 700))
    cv2.imshow('face detection Tutorial ',resized_img)
    k = cv2.waitKey(100) & 0xff
    if k == 27:
        break
    elif count >= 30:
        break
cap.release()
cv2.destroyAllWindows
```

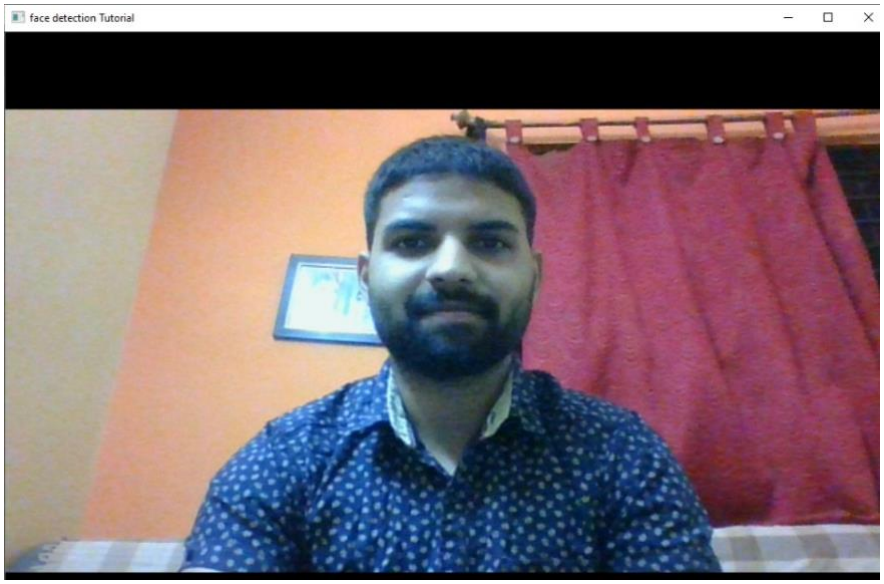

Fig 4.1 An image of Dinesh

Attached above is a image of Dinesh in good lighting condtion.

4.2 Resizing the image to identify the face of the trained data.

1. Getting the image from the saved location resize the image to the face part for putting that part in the Training model.
2. Make the resolution to 100 * 100 pixels for better detection from cleaning apart.

```
import cv2
```

```
import os
import numpy as np
count=0
for path, subdirnames, filenames in os.walk("C:/Users/Dinesh/PycharmProjects/FaceRecognition-master/trainingImages"):
    for filename in filenames:
      if filename.startswith("."):
        print("Skipping File:",filename)#Skipping files that startwith .
        continue
      img_path=os.path.join(path, filename)#fetching image path
      print("img_path",img_path)
      id=os.path.basename(path)#fetching subdirectory names
      img = cv2.imread(img_path)
      if img is None:
        print("Image not loaded properly")
        continue
      resized_image = cv2.resize(img, (100, 100))
      new_path="C:/Users/Dinesh/PycharmProjects/FaceRecognition-master/resizedTrainingImages"+"/"+str(id)
      print("desired path is",os.path.join(new_path, "frame%d.jpg" % count))#write all images to resizedTrainingImages/id directory
      cv2.imwrite(os.path.join(new_path, "frame%d.jpg" % count),resized_image)
      count += 1
```

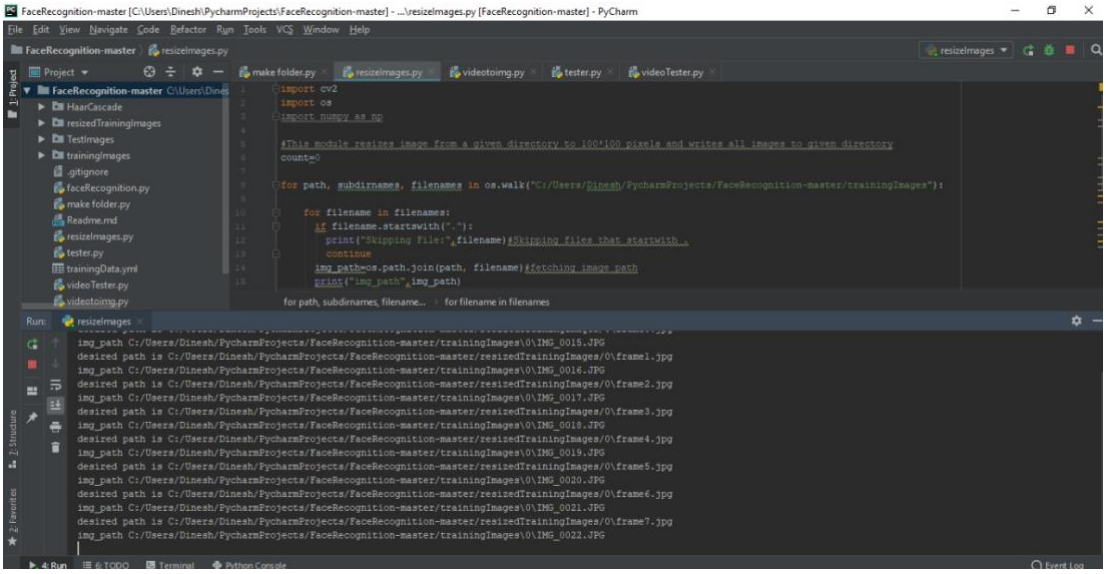

Fig 4.2 Code for 100 x 100 pixels

4.3 Training the Image Data to the LBPH Face Recognizer model for the name display to recognizer part.

1. Putting the collected images to the LBPH Face Recognizer.
2. Using median pixel value as the threshold, it compares a pixel to its 8 closest pixels using this function.
3. If the value of neighbour is greater than or equal to the central value it is set as 1 otherwise it is set as 0.
4. Thus, we obtain a total of 8 binary values from the 8 neighbours.
5. After combining these values, we get an 8-bit binary number which is translated to decimal number for our convenience.
6. This decimal number is called the pixel LBP value and its range is 0-255.
7. Later it was noted that a fixed neighbourhood fails to encode details varying in scale. The algorithm was improved to use the different number of radius and neighbours, now it was known as circular LBP.
8. The idea here is to align an arbitrary number of neighbours on a circle with a variable radius. This way the following neighbourhoods are captured.
9. After the generation of LBP value histogram of the region is created by counting the number of similar LBP values in the region.

10. After the creation of histogram for each region, all the histograms are merged to form a single histogram, and this is known as a feature vector of the image.

```
import cv2
import os
import numpy as np
import faceRecognition as fr
#This module takes images  stored in diskand performs face recognition
test_img=cv2.imread('C:/Users/Dinesh/PycharmProjects/FaceRecognition-master/TestImages/dinesh9.jpg')#test_img path
faces_detected,gray_img=fr.faceDetection(test_img)
print("faces_detected:",faces_detected)
face_recognizer=cv2.face.LBPHFaceRecognizer_create()
face_recognizer.read('trainingData.yml')#use this to load training data for subsequent runs
name={0:"dinesh",1:"Kangana",2 :"priyanka",3:"dhruv",5:"dd"}#creating dictionary containing names for each label
for face in faces_detected:
   (x,y,w,h)=face
   roi_gray=gray_img[y:y+h,x:x+h]
   label,confidence=face_recognizer.predict(roi_gray)#predicting the label of given image
   print("confidence:",confidence)
   print("label:",label)
   fr.draw_rect(test_img,face)
   predicted_name=name[label]
   if(confidence>37):#If confidence more than 37 then don't print predicted face text on screen
      continue
   fr.put_text(test_img,predicted_name,x,y)
resized_img=cv2.resize(test_img,(1000,1000))
cv2.imshow("face dtecetion tutorial",resized_img)
cv2.waitKey(0)#Waits indefinitely until a key is pressed
```

cv2.destroyAllWindows

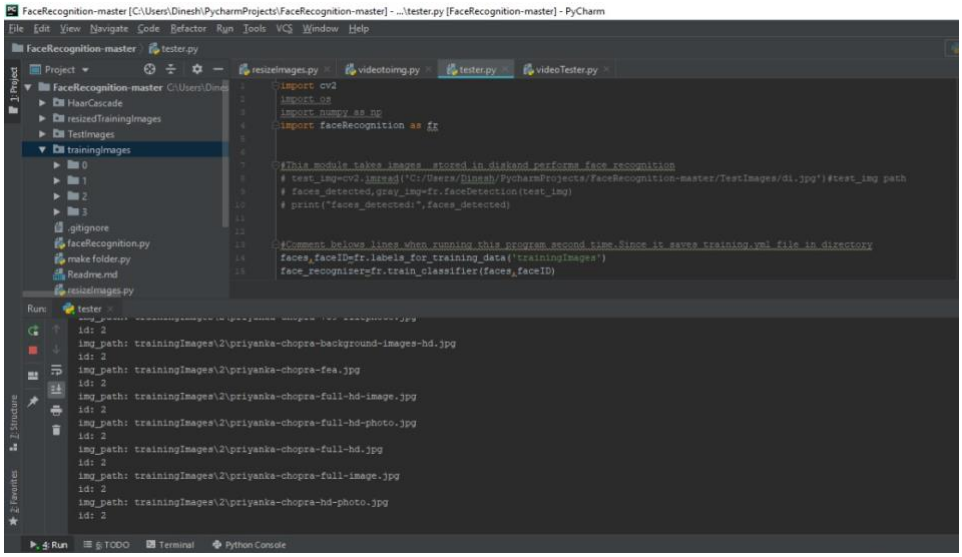

Fig 4.3 Code for training the images with stored images

Face Recognition Through the Image.

1. Getting the image from the Dataset.
2. Using the Viola-Jones algorithm to detect the Eyes, Nose, Lips and Other features from the image with the help of Front Facial XML file for the better understanding to types of faces.
3. Convert into Greyscale image for ROI (Region Of Interest) for getting data for face and save to the .jpg file.
4. Saving the image to the Desired Id with person name for putting the image for the training the Classifier.
5. After all, work done trying to display the name according to the id where we saved the of the person along with the image Id.'
6. Using the put text function from Open-cv library.

```
import cv2
import os
import numpy as np
#This module contains all common functions that are called in tester.py file
#Given an image below function returns rectangle for face detected alongwith
gray scale image
def faceDetection(test_img):
    gray_img=cv2.cvtColor(test_img,cv2.COLOR_BGR2GRAY)#convert
color image to grayscale
```

```
face_haar_cascade=cv2.CascadeClassifier('C:/Users/Dinesh/PycharmProjects/FaceRecognition-master/HaarCascade/haarcascade_frontalface_default.xml')#Load haar classifier
faces=face_haar_cascade.detectMultiScale(gray_img,scaleFactor=1.32,minNeighbors=5)#detectMultiScale returns rectangles

    return faces,gray_img

#Given a directory below function returns part of gray_img which is face alongwith its label/ID
def labels_for_training_data(directory):
    faces=[]
    faceID=[]

    for path,subdirnames,filenames in os.walk(directory):
        for filename in filenames:
            if filename.startswith("."):
                print("Skipping system file")#Skipping files that startwith .
                continue

            id=os.path.basename(path)#fetching subdirectory names
            img_path=os.path.join(path,filename)#fetching image path
            print("img_path:",img_path)
            print("id:",id)
            test_img=cv2.imread(img_path)#loading each image one by one
            if test_img is None:
                print("Image not loaded properly")
                continue
            faces_rect,gray_img=faceDetection(test_img)#Calling faceDetection function to return faces detected in particular image
            if len(faces_rect)!=1:
                continue #Since we are assuming only single person images are being fed to classifier
```

```
        (x,y,w,h)=faces_rect[0]
        roi_gray=gray_img[y:y+w,x:x+h]#cropping region of interest i.e. face area from grayscale image
        faces.append(roi_gray)
        faceID.append(int(id))
    return faces,faceID


#Below function trains haar classifier and takes faces,faceID returned by previous function as its arguments
def train_classifier(faces,faceID):
    face_recognizer=cv2.face.LBPHFaceRecognizer_create()
    face_recognizer.train(faces,np.array(faceID))
    return face_recognizer

#Below function draws bounding boxes around detected face in image
def draw_rect(test_img,face):
    (x,y,w,h)=face
    cv2.rectangle(test_img,(x,y),(x+w,y+h),(255,0,0),thickness=5)

#Below function writes name of person for detected label
def put_text(test_img,text,x,y):

cv2.putText(test_img,text,(x,y),cv2.FONT_HERSHEY_DUPLEX,2,(255,0,0),4)
```

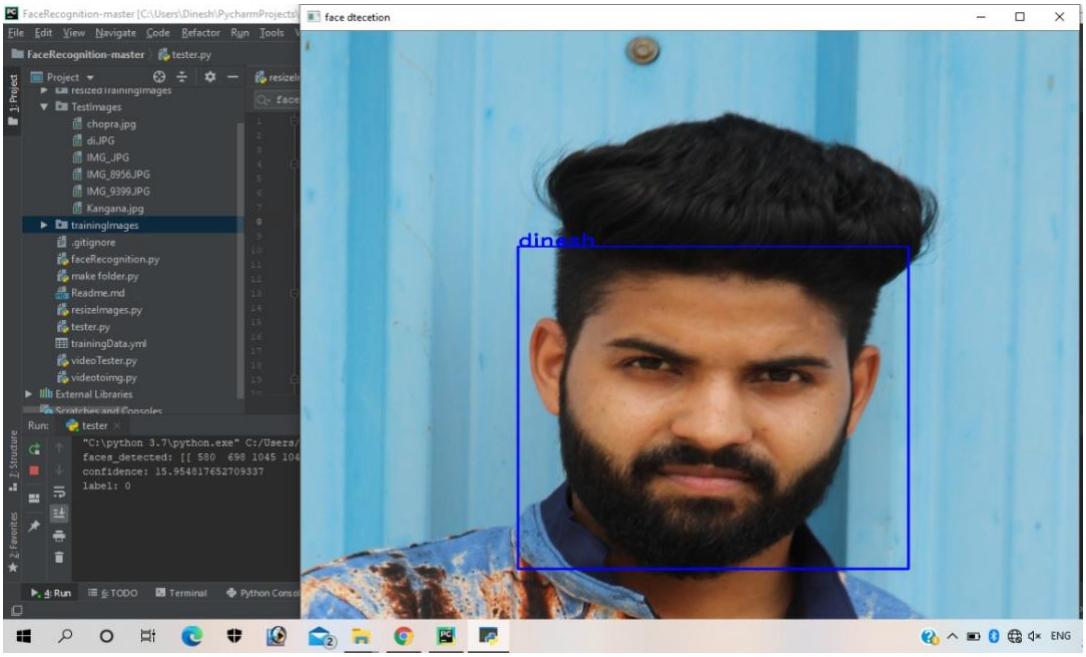

Fig 4.4 Face Detection

We give the image through a source it shows an image of Dinesh with output confidence =15 and label=0 and level 0 that belongs to dinesh, hence it detect Dinesh.

For label one we train all the images in clean shave but gives the input picture with beard so its shows confidence =15

Face Recognition through the Webcam

1. Open Webcam gets the image.
2. Train the recognizer.
3. Put the id according to the person name for real-time Recognition.
4. Put text on above on box of name of person.

```
import os
import cv2
import numpy as np
import faceRecognition as fr
#This module captures images via webcam and performs face recognition
face_recognizer = cv2.face.LBPHFaceRecognizer_create()
face_recognizer.read('C:/Users/Dinesh/PycharmProjects/FaceRecognition-master/trainingData.yml')#Load saved training data
name = {0 : "dinesh",1:"Amitabh Bachchan",2 :"Mr. Kalam ",3:"dhruv"}
cap=cv2.VideoCapture(0)
while True:
```

```
    ret,test_img=cap.read()# captures frame and returns boolean value and
captured                                                              image
    faces_detected,gray_img=fr.faceDetection(test_img)

    for                  (x,y,w,h)                in                faces_detected:
      cv2.rectangle(test_img,(x,y),(x+w,y+h),(255,0,0),thickness=7)

    resized_img          =          cv2.resize(test_img,          (1000,          700))
    cv2.imshow('face          recognition          by          video          ',resized_img)
    cv2.waitKey(10)

    for                  face                in                faces_detected:
        (x,y,w,h)=face
        roi_gray=gray_img[y:y+w,                                              x:x+h]
        label,confidence=face_recognizer.predict(roi_gray)#predicting the label
of                                    given                                image
        print("confidence:",confidence)
        print("label:",label)
        fr.draw_rect(test_img,face)
        predicted_name=name[label]
        if confidence < 100:#If confidence less than 37 then don't print predicted
face                        text                        on                        screen
          fr.put_text(test_img,predicted_name,x,y)


    resized_img          =          cv2.resize(test_img,          (1000,          700))
    cv2.imshow('face          detection          by          video          ',resized_img)
    if    cv2.waitKey(10)    ==    ord('q'):#wait    until    'q'    key    is    pressed
        break

cap.release()
cv2.destroyAllWindows
```

Sample 1: -

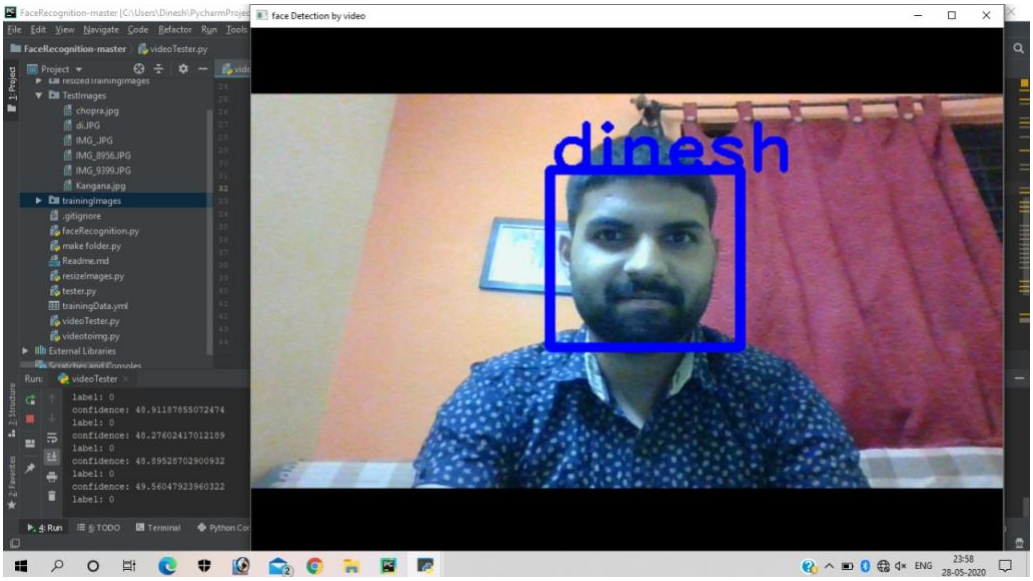

Fig 4.5 Face Detection by video

The confidence value is an average of 49 because of low light quality.

Sample: - 2

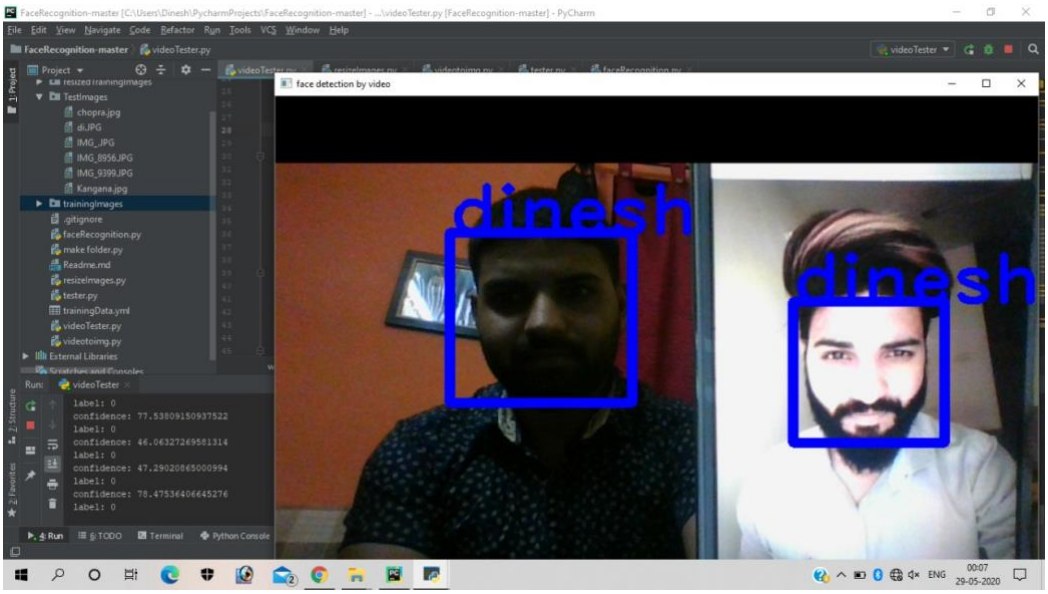

Fig 4.6 Another example of Face detection with a video

We have a common label = 0 which belong to Dinesh so its displays his name for both the image but we get two different confidence value one is an average of 46 which belongs to first image.

The second image we get confidence= 76 Its has more confidence value because this image is not coming directly to the camera it's shows with the help of a smartphone hence it reduces The pixel quality and we get a higher confidence value.

Sample: - 3

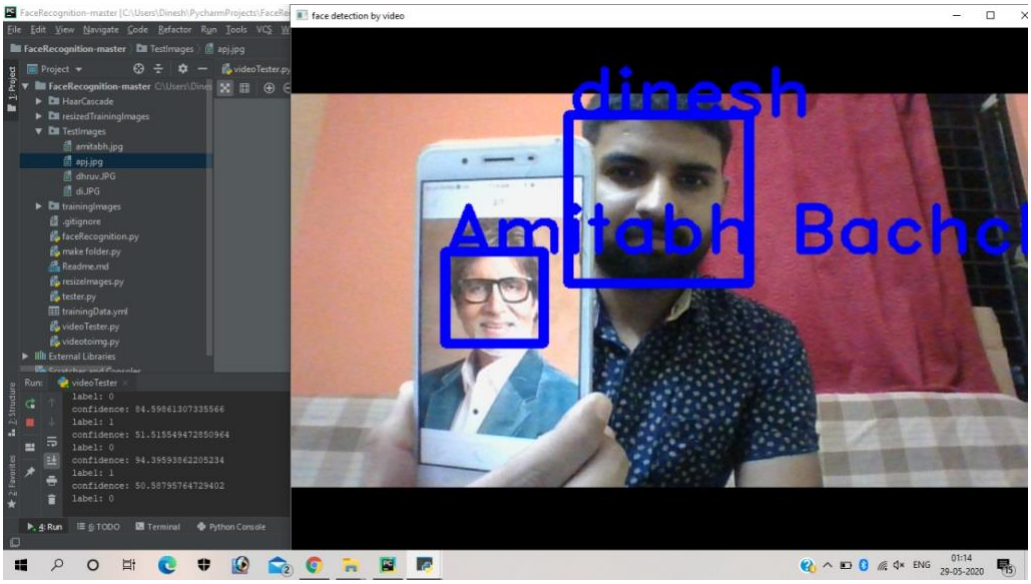

Fig 4.7 Face detection of two individuals

Confidence value of Dinesh is 46 and confidence value of Amitabh Bachan is 9 that is very high due to less pixels captured from the phone.

Sample: - 4

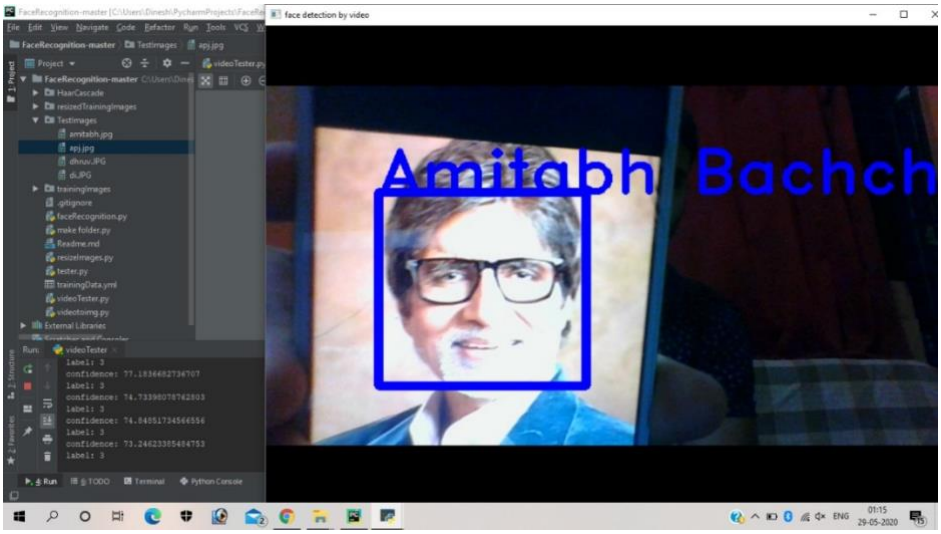

Fig 4.8 Face detection of Amitabh Bachan

We get confidence as 74 for amithab by showing the image through mobile in this sample.
In this image we display only one person so our system focus on object and obtains an almost accurate result.

5. CONCLUSION AND SCOPE FOR FURTHER WORK

5.1 What have we done?

Face recognition is an emerging technology that can provide many benefits. Face recognition can save resources and time, and even generate new income streams, for companies that implement it right. We use two algorithms Voila-Jones algorithm and LBPH algorithm with the help of python libraries.

5.2 Advantage and Disadvantages of the facial recognition system

1. The Improvement of Security Level

As we said in the first paragraph, a face biometric system greatly improves your security measures. All corporation's premises would be protected since you'll be able to track both the employees and any visitors that come into the area. Anyone who doesn't have access or permission to be there will be captured by the recognition system that alerts you instantly about the trespassing.

As an example, let's take a 24/7 drugstore. Any owner prefers to keep their money and clients safe, avoiding unpleasant troubles with difficult visitors. When you have an FRT in place, you'd be instantly alerted as soon as the wanted or suspicious character arrives. Which leads to a significantly reduces expenses one usually spends on security staff.

2. Easy Integration Process

Most of the time, facial recognition tools work pretty flawlessly with the existing security software that companies have installed. And they're also easy to program for interaction with a company's computer system.

Why is it great for business? Well, you won't need to spend additional money and time on redeveloping your software to make it suitable for FRT integration. Everything will be already adaptable.

3. High Accuracy Rates

These days, the success level of face tracking technology became higher than ever before. Thanks to the assistance of 3D facial recognition technologies and infrared cameras the process of identification happens to be incredibly accurate and showing great results. It’s possible but difficult to fool such a system, so you can be sure that an FR digital security software will successfully track every aspect of attendances to provide a better level of protection for your facilities.

Accuracy ensures that there won’t be any misunderstandings and uncool awkwardness that comes from bad face recognition software. With high levels of accuracy, you'd sure that the right person will be recognized at the right time.

4. Full Automation

Instead of manual recognition, which is done by security guards or the official representatives outside of the company's premises, the facial recognition tech automates the identification process and ensures its flawlessness every time without any haltings. You won’t even need an employee to monitor the cameras 24/7.

Automation means convenience and reduces the expenses too. Therefore, any entrepreneur would be fond of the fact that image identification systems are fully automated.

5. Forget the Time Fraud

One of the big benefits that facial recognition technology companies offer is the time attendance tracking that allows excluding the time fraud among the workers. No more buddy favours from securities for staff members, since everyone now has to pass a face-scanning device to check-in for work. And the paid hours begin from this moment till the same check-out procedure. And the process will be fast because employees don't have to prove their identities or clock in with their plastic cards.

Businessmen must trust their workers but keep an eye on them just in case. Unfortunately, time fraud is one of the most common violations of the work ethics, but the facial identification tech will spare you a headache regarding this matter.

Disadvantages of Facial Recognition system

1. Processing & Storing

Storages are like gold in the digital world since you have to save huge amounts of data for future usage. Even though you get HD-video in a pretty low resolution, it still requires significant space. Just as the high-quality image visuals. There is no need to process every video's frame – it's an enormous waste of resources. That's why most of the time only a fraction (around 10 – 25%) is being put through an FRT.

Professional agencies use whole clusters of computers to minimize total processing time. But every added computer means considerable data transfer via network, which can be influenced by input-output limitations that lower a processing speed.

2. Image Size & Quality

Facial recognition is a super-advanced software that requires HQ digital cameras for algorithms to operate accurately. A face-detection system captures a face in the photo or screen-shot from a video, then the relative size of that face image will be compared with the size of the enrolled one. So, the photo's quality here affects the whole face recognition process, how well it would be done. Imagine, the already small size picture is coupled with a distance that was between a target and a CCTV… What proportions will the detected face have? No more than 100×200 pixels.

Pretty hard to get a clear identity in such a case. What's more, scanning a photo for varying face sizes is a processor-intensive task. Most systems allow

identification of a face-size range to eliminate false recognition and speed up image processing. But the initial investment in such face-tracking software is not a cheap one, however, it will pay off in no time.

3. Surveillance Angle

The identification process is also under great pressure of the surveillance angle that was responsible for the target's face capturing. To enrol a face through the recognition software, the multiple angles are being used – profile, frontal, 45-degree, etc. But to generate a clear template for the face, you'll need nothing less than a frontal view. The higher resolution photo has and the more direct its angle is (goes for both enrolled and compared images) the more accurate resulting matches would be.

Then, there are also troubles with such things as facial hair or sunglasses. One can still fool the FRT with a suddenly appeared or removed beard, same goes for obscuring face's parts with glasses or masks. To avoid such failures, the databases must be regularly updated with the most up-to-date images.

5.3 Conclusion

We are looking to get the ID or name with the help of a face. Once the image is being processed with the help of software and its function. It will detect the features of the face and try to match the pattern from the database.

If an anonymous person comes in front of the camera, you will receive a notification alert that the face does not recognise.

Privacy matters hence privacy refers to any rights you have to control your personal information and how it's used — and that can include your faceprint:

- **Security**.
- **Prevalence**.
- **Ownership**.
- **Safety**.
- **Mistaken identity**.

- **Basic freedoms**.

The retrieval of images containing human faces requires detection of human faces in such images. We implemented a new method that segments skin regions out and locate faces using template matching in order to detect frontal human faces. We used 30 images to test the performance of this implementation and we got 76% of accuracy.

The misses usually included regions with a similar skin likelihood values and regions that certainly were skin regions but corresponded to other parts of the body such as legs and arms. In other cases, misses were found due the constrain we set of having one or more holes in a skin region to be in considered for the processing described in the previous sections.

Our current implementation is limited to the detection of frontal human faces. A possible and interesting extension would be to expand the template matching process to include sided-view faces as well.

5.4 Scope for the future

The use of spherical canonical images allows us to perform matching in the spherical harmonic transform domain, which does not require preliminary alignment of the images. The errors introduced by embedding into an expressional space with some predefined geometry are avoided. In this facial expression recognition setup, end-to-end processing comprises the face surface acquisition and reconstruction, smoothening, subsampling to approximately 2500 points. Facial surface cropping measurement of large positions of distances between all the points using a parallelized parametric version is utilized. The general experimental evaluation of the face expressional system guarantees better face recognition rates. Having examined techniques to cope with expression variation, in future it may be investigated in more depth about the face classification problem and optimal fusion of colour and depth information. Further study can be laid down in the direction of an allele of gene matching to the geometric factors of the facial expressions. The genetic property evolution framework for a facial expressional system can be studied to suit the requirement of different security models such as criminal detection, governmental confidential security breaches etc.

REFERENCES

- Ahonen, Timo, Abdenour Hadid, and Matti Pietikainen. "**Face description with local binary patterns: Application to face recognition**." *IEEE transactions on pattern analysis and machine intelligence* 28.12 (2006): 2037–2041.

- Ojala, Timo, Matti Pietikainen, and Topi Maenpaa. "**Multiresolution gray-scale and rotation invariant texture classification with local binary patterns**." *IEEE Transactions on pattern analysis and machine intelligence* 24.7 (2002): 971–987.

- Ahonen, Timo, Abdenour Hadid, and Matti Pietikäinen. "**Face recognition with local binary patterns**." *Computer vision-eccv 2004* (2004): 469–481.

- *Y.M. Lui, D.S. Bolme, P.J. Phillips, J.R. Beveridge and B.A. Draper*. **Preliminary studies on the Good, the Bad, and the Ugly face recognition challenge problem**. Computer Vision and Pattern Recognition Workshops (CVPRW), pages 9-16. 2012.

- *G. Heusch, Y. Rodriguez, and S. Marcel*. **Local Binary Patterns as an Image Preprocessing for Face Authentication**. In IEEE International Conference on Automatic Face and Gesture Recognition (AFGR), 2006.

- *X. Tan and B. Triggs*. **Enhanced local texture feature sets for face recognition under difficult lighting conditions**. IEEE Transactions on Image Processing, 19(6):1635-1650, 2010.

- *B. Moghaddam, W. Wahid and A. Pentland*. **Beyond eigenfaces: probabilistic matching for face recognition**. IEEE International Conference on Automatic Face and Gesture Recognition, pages 30-35. 1998.

- *W. Zhao, A. Krishnaswamy, R. Chellappa, D. Swets and J. Weng*. **Discriminant analysis of principal components for face recognition**, pages 73-85. Springer Verlag Berlin, 1998.

- *M. Turk and A. Pentland*. **Eigenfaces for recognition**. Journal of Cognitive Neuroscience, 3(1):71-86, 1991.

- **Face Recognition: The Problem of Compensating for Changes in Illumination Direction**
  IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):721-732.

- Allinson, N.M. and Ellis, A.W. (1992).
  Face recognition: Combining cognitive psychology and image engineering
  Electronics & Communication Engineering Journal, 4:291-300.

- Baron, R.J. (1979).
  A bibliography on face recognition
  The SISTM Quaterly Incorporating the Brain Theory Newsletter, II(3):27-36.

- Belhumeur, P.N., Hespanha, J.P., and Kriegman, D.J. (July 1997).
  Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection
  IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):711-720.
- Bruce, V., Burton, A.M., and Craw, I.G. (1992).
  Modelling face recognition
  Philosophical Transactions of the Royal Society of London, Series B, 335:121-128.

- Brunelli, R., Falavigna, D., Poggio, T., and Stringa, L. (1995).
  Automatic Person Recognition by Using Acoustic and Geometric Features
  Machine Vision and Applications, 8(5):317-325.

- Chellappa, R., Wilson, C.L., and Sirohey, S. (May 1995).
  Human and Machine Recognition of Faces: A Survey
  Proceedings of the IEEE, 83(5):705-740.

- Gerl, S. and Levi, P. (1997).
  3-D human face recognition by self-organizing matching approach
  Pattern Recognition and Image Analysis, 7(1):38-46.

- Goldstein, A.J., Harmon, L.D., and Lesk, A.B. (1971).
  Identification of human faces
  Proceedings of the IEEE, 59(5):748-760.

- Harmon, L.D., Khan, M.K., Lash, R., and Ramig, P.F. (1981).
  Machine identification of human faces
  Pattern Recognition, 13(2):97-110.

APPENDIX

Hardware Requirements

- System Processor : Core i3 / i5/i7
- Hard Disk : 500 GB.
- Ram : 4 GB.

✓ *Any desktop / Laptop system with above configuration or higher level.*

Software Requirements

- **Operating system** : Windows 8 / 10
- **Programming Language** : Python
- **Framework** : Py Charm
- **IDE** : IDE Charm
- **DL Libraries** : Numpy, Open CV

PROFORMA FOR APPROVAL OF PROJECT REPORT

Name of the student :

Name of the Department : Computer Science and Engineering

Branch : CSE

Year : 2018-2019

Date of Submission :

Batch Number : 2015-2019

Items	Yes	No
Whether all pages are as per instruction?		
Whether the contents of report are in correct order as given in the manual?		
Whether the margin specification and spacing for the text correctly followed as per instruction in the manual?		
Whether typed in black?		
Whether numbers given for figures,tables & other pages serially and in correct position?		
Whether tables numbered properly with captions given as top of tables?		
Whether figures & diagram numbered properly with caption given below?		
Whether binding is OK?		
Whether references are marked in text within square brackets?		
Whether CD containing executable s/w, required libraries & source code?		
Comment: **Rating: /10**		

Internal Guide **Head of the Department**