

# **Real-Time Face Recognition**

A  
PROJECT REPORT

*Submitted in partial fulfilment of the requirements for the award of the degree*

*of*  
**BACHELOR OF TECHNOLOGY**

**IN**  
**COMPUTER SCIENCE**

*Under the supervision*  
*of*

**Dr. Vivek Sehgal**  
**(Associate Professor)**

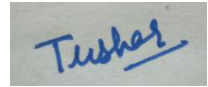
*by*  
**Tushar Gautam (161350)**



**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY**  
**WAKNAGHAT, SOLAN – 173234**  
**HIMACHAL PRADESH, INDIA**  
**June – 2020**

## Candidate's Declaration

I hereby declare that the work presented in this report entitled “**Real-Time Face Recognition**” in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2019 to December 2019 under the supervision of **Dr. Vivek Sehgal**, Associate Professor (Senior Grade), Department of Computer Science & Engineering and Information Technology. The matter embodied in the report has not been submitted for the award of any other degree or diploma.



Tushar Gautam  
(161350)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



Dr. Vivek Sehgal

Associate Professor (Senior Grade)

Department of Computer Science & Engineering and Information Technology

Dated: -

## ACKNOWLEDGEMENT

It is our privilege to express our deep gratitude and regards to our project supervisor **Dr. Vivek Sehgal** for his mentoring, valuable inputs, guidance and constructive criticism throughout the duration of this project. We also express our sincere thanks for encouraging and allowing us to present the project on the topic “**Real-Time Face Recognition**” for the partial fulfilment of the requirements leading to the award of B.Tech. Degree. We would also like to thank Dr Satya Prakash Ghrera, Head of Department (CSE) for providing us a great opportunity to work on such an interesting project. Last but not least we would like to express our sincere gratitude to our family members who stood by us and supported us in every phase of this project and gave us the much required moral support in carrying out this project successfully.

# TABLE OF CONTENTS

STUDENT'S DECLARATION	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv-v
ABBREVIATIONS AND TABLES	vi
LIST OF FIGURES	vii
ABSTRACT	viii
<b>CHAPTER 1: INTRODUCTION</b>	<b>1-5</b>
1.1 Face Detection and Face Recognition	1
1.1.1 Face Detection	1
1.1.2 Face Recognition	3
1.2 Objective	5
1.3 Problem Statement	5
<b>CHAPTER 2: LITERATURE REVIEW</b>	<b>6-15</b>
2.1 Related Work	
<b>CHAPTER 3: SYSTEM DEVELOPMENT</b>	<b>16-26</b>
3.1 Software Requirement Specification	16
3.2 Model Development	16
3.3 Face Detection	16
3.4 Stages of Face Recognition	17
3.4.1 Stage I	18
3.4.2 Stage II	18
3.4.3 Stage III	23
3.5 Libraries Used	24

<b>CHAPTER 4: IMPLEMENTATION AND RESULT</b>	27-33
4.1 Test Cases	27
4.2 Collecting Image Samples	27
4.3 Model Training and Testing	28
4.4 Implementation	29
4.5 Outputs	31
<b>CHAPTER 5: CONCLUSION</b>	34
5.1 Conclusions	34
5.2 Future Scope	34
REFERENCES	35

## ABBREVIATIONS

<b>ML</b>	Machine Learning
<b>LBPH</b>	Local Binary pattern Histogram
<b>AI</b>	Artificial Intelligence
<b>EF</b>	EigenFace
<b>OpenCV</b>	Open Source Computer Vision

## TABLES

<b>Table no.</b>	<b>Description</b>	<b>Page No.</b>
1	Result analysis for paper-1	6
2	Result analysis for paper-1	7
3	Result analysis for paper-7	14
4	Result analysis for paper-7	15

## LIST OF FIGURES

<b>Figure no.</b>	<b>Description</b>	<b>Page No.</b>
1	Haar wavelet and features	2
2	Haar-like features	2
3	Haar-cascade flow chart	3
4	Photometric Stereo	3
5	Geometric	4
6	OpenCV design process	9
7	Modified LBPH flowchart	10
8	MLBPH flowchart	11
9	Algorithm Flowchart	13
10	Model Development	16
11	Flow chart of the face detection application	17
12	Stages of Face Recognition	17
13	LBPH-1	19
14	LBPH-2	20
15	LBPH-3	21
16	Face Recognition Flowchart	24
x	Output	31-33

## ABSTRACT

Face detection and recognition is one of the most studied field in the area of computer vision. As we all know that each and every individual's face is the identity to distinguish them from another individual. Facial detection and recognition describes the way each individual's identification system works with the use of facial characteristics which vary from person to person. Under this report, two major steps involved are: face detection, this process includes the detection of human faces and is a very fast process for human faces as well as the objects that are placed close to the camera lens, secondly the faces are categorized as per the facial attributes as they are different for each individual. Facial recognition is the future of one of the much-studied biometrics technology.

The main motive of our project is to detect faces in a given video/image frame which has to be done using the HAAR-Cascades and then use facial recognition algorithm to validate the detected face. On the other hand, there are various methods in which facial recognition can be done: Eigenface method, LBPH method and Fisherface method. The Eigenface method works on the algorithm called PCA which stands for Principal Component Analysis and this algorithm reduces the dimensions of the face expression to extract the facial characteristics. LBPH method stands for Local Binary Pattern Histogram which is the method we are going to talk about later. In this project, python is required as it consists on all the suitable modules and libraries that we require i.e OpenCV and opencv-contrib-python libraries.

**Keyword:** face detection, LBPH face recognition, OpenCV, python



# Chapter 1

## INTRODUCTION

The following report is on the mini project for face detection and recognition. The purpose of the project was to build a system that not only detects faces but also recognises them, and this was done using face detection and face recognition algorithms using the HAAR-classifier and the LBPH algorithm respectively. This field has been studied upon since the last few years, not only for the security areas but also for various areas such as Attendance systems, validating identity at ATM's, finding criminals and many other areas. Face recognition is one of the fastest method to locate faces and identify multiple faces at the same time. The major concern which arrises here is that face detection and recognition are two separate areas and differ from each other in all the ways. Face detection deals with the detection of the face in the given image frame whereas the face recognition validates that whether the detected face is in the database or not. In this project we are going to work on each one of them, starting from the various face detection techniques to the face recognition techniques. The report covers all the various areas under face detection and recognition. We will also discuss about the pros and cons of the algorithms used.

### 1.1 Face Detection and Face Recognition

#### 1.1.1) Face Detection:

In the past few years, face recognition closely-held important thought and appreciated joined of the foremost promising applications within the field of image analysis. Face detection will take into account a considerable a part of face recognition operations, in keeping with its strength to focus machine resources on the section of a picture holding a face.

HAAR cascade is a mathematical fiction in which square waves are formed at the start and at the last, whose purpose is to recognize pixel values using the square shapped patterns. This method can easily identify edges, circles of different colours using the combination of various waves. In this method, there is a need to visualize the input using HAAR cascades and the scale is adjusted to a lesser value than the given input image. This is then overlaped with the input and

average value of each pixel is adjusted accordingly. In a human face, the facial features differ from one individual to another and therefore face detection is done by validating combinations of various features. An individual classifier is not enough to detect a human face and that is why a combination of HAAR-features is used. Face detection works best when various appropriate classifiers are collectively used.

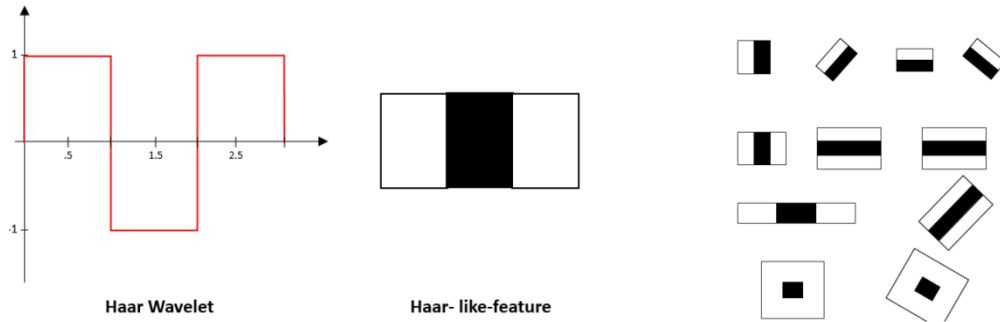


Figure 1: A Haar wavelet and resulting Haar-like features.



Figure 2: Several Haar-like-features matched to the features of authors face.

The above given method is used to detect and combine features like eyes, faces and mouth to achieve better results. Classifiers can be combined to get more precise detection algorithm as given in the image below.

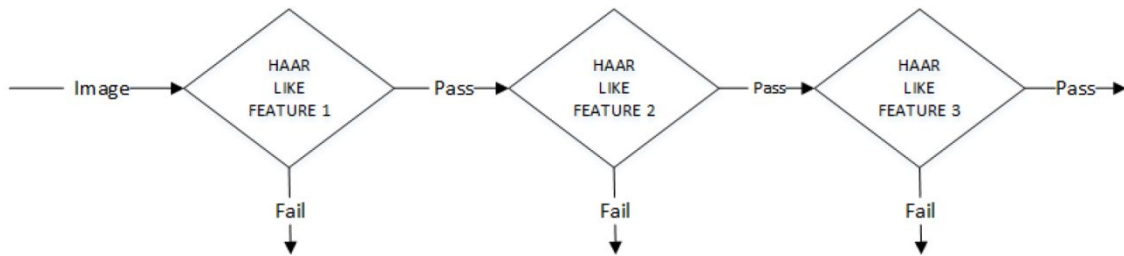


Figure 3: Haar-cascade flow chart

### 1.1.2) Face Recognition:

Face recognition validates that whether the face that is detected using face detection is known or an unknown face by checking the database. It basically includes two major methods:

- a) **Geometric:** The geometric approach deals with the geometric relationship between the various facial features such as mouth, eyes and nose, which are mapped under this method and validated on the basis of the geometrical relationships and there features.
- b) **Photometric stereo:** A collection of various images takes place under different circumstances such as low light to bright light. This resultant is then mapped using an array of gradient vectors.

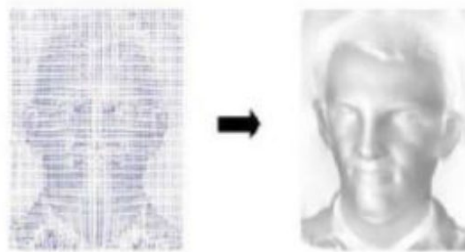


Figure 4: Photometric Stereo

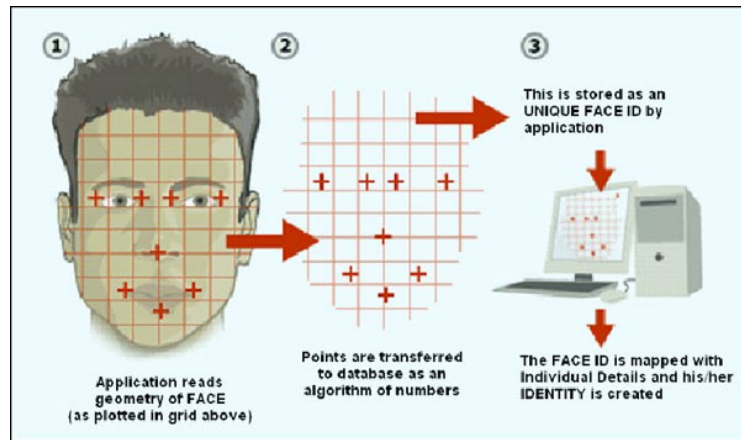


Figure 5: Geometric

Some of the popular face recognition algorithms are:

- PCA
- LBPH
- Fisherface

## **1.2 OBJECTIVE**

Face detection and recognition deals with finding a known face from a given real time video source. A decent amount of study has been done on this topic. The primary objective of this project is to detect faces from a given video source which has to be done using the HAAR-Cascades and then use facial recognition algorithm to validate the detected face. There are various methods in which facial recognition can be done: Eigenface method, LBPH method and Fisherface method.

## **1.3 PROBLEM STATEMENT**

The purpose of the project is to build a system that not only detects faces but also recognises them, and this was done using face detection and face recognition algorithms using the HAAR-classifier and the LBPH algorithm respectively. This field has been studied upon since the last few years, not only for the security purposes but also for various others such as Attendance systems, validate identity at ATM's, find criminals and many other areas. Face recognition is one of the fastest method to locate faces and identify multiple faces at the same time. The major concern which arises here is that face detection and recognition are two separate areas and differ from each other in all the ways.

Therefore this project deals with the use of HAAR-cascades along with face recognition algorithms to detect as well as recognize human face.

## Chapter 2

### LITERATURE REVIEW

#### Paper – 1

A Comparative Study between LBP and Haar-like features for Face Detection Using OpenCV by *Kushsairy Kadir, Mohd Khairi Kamaruddin, Haidawati Nasir, Sairul I Safie, Zulkifli Abdul Kadir Bakti*

#### Main Contribution:-

In this paper, the author uses two algorithms for face detection, feature extraction and LBP algorithm. This paper works on the detection speed and the hit ratio. Microsoft visual c++ was used in the implementation that allows the use of OpenCV module. OpenCV is a feature rich module that reduces the work of mathematical calculations.

#### Algorithm Used:-

- Linear Binary pattern
- HAAR-cascade

#### Result Analysis:-

The results show us that LBP algorithm is the most powerful algorithm, as it is has a high accuracy rate and is capable to use it as a real time application.

TABLE 1  
Haar Result

Algorithm	Dataset	Detected Faces	Hit Rate	Detection Speed (ms)
Haar	Color FERET	976/1127	86.6%	235.4117
	MIT	1670/2000	83.5%	255.5048
	Taarlab	647/759	85.2%	231.5865
<b>Overall</b>		<b>3293/3886</b>	<b>84.7%</b>	<b>241</b>

TABLE 2  
LBP Result

Algorithm	Dataset	Detected Faces	Hit Rate	Detection Speed (ms)
LBP	Color FERET	1004/1127	89%	95.44924
	MIT	1779/2000	89%	101.8864
	Taarlab	674/759	88.8%	104.9335
<b>Overall</b>		<b>3457/3886</b>	<b>89%</b>	<b>101</b>

### **Future Scope:-**

The further work under this includes creation of real-time face detection algorithm using LBP algorithm as it is much more efficient as compared to HAAR-cascade.

### **Paper – 2**

**Face Detection and Recognition Using OpenCV by *Maliha Khan, Sudeshna Chakraborty, Rani Astya, Shaveta Khepra* (ICCCIS/2009)**

### **Main Contribution:-**

In this paper, the author works on the Principal Component Analysis (PCA). The paper revolves around the use of PCA to create an application that not only detects faces but also recognizes them efficiently. This is done with the use of various algorithms such as HAAR-Cascade, Eigenface, Fisherface, and LBPH that can be done with the use of OpenCV module in python.

### **Algorithm Used:-**

- Linear Binary pattern
- HAAR-cascade
- FisherFace
- EigenFace

### **Result Analysis:-**

The results show that the LBPH algorithm works best for Face detection and these face detection and recognition technologies would be used in the next few years. This would make all the gadgets around us smart and help us living a better life.

### **Future Scope:-**

The future scope includes the work on the face projection of humans that would help build a better neural network classifier and enhance the recognition efficiency.

### **Paper – 3**

**HAAR like Feature-Based Car Key Detection Using Cascade Classifier by *Paawan Sharma, Mukul K. Gupta, Amit K. Mondal and Vivek Kaundal* (2016)**

### **Main Contribution:-**

This paper reports powerful continuous usage for specific item discovery in an exceedingly image or succession of images. Vehicle keys were detected using various algorithms. The classifier is formed utilizing the OpenCV module. The system incorporates getting ready and discovery. A wide assortment of article footage are utilised for getting ready reason. The created xml classifier is then tried on distinct check footage.

### **Algorithm Used:-**

- HAAR-cascade



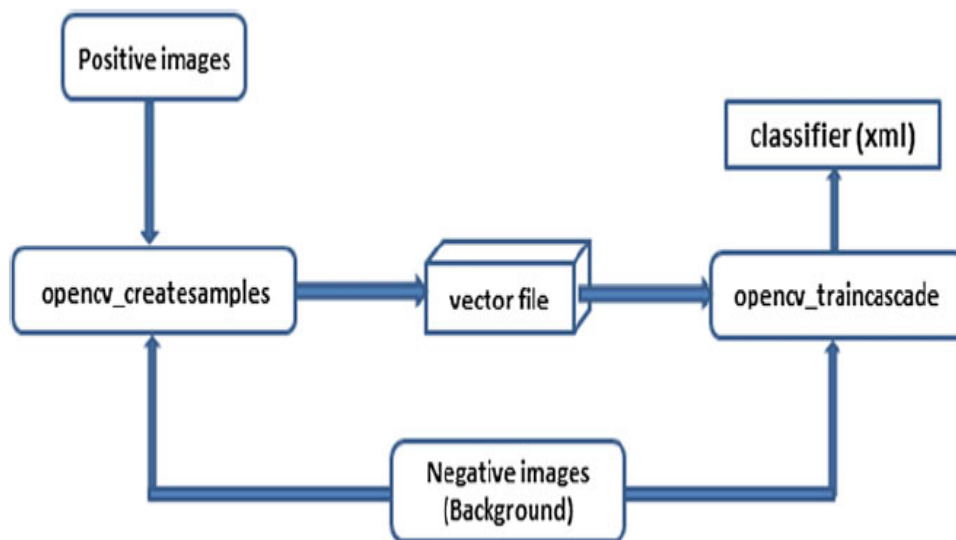


Figure 6: OpenCV design process

### Result Analysis:-

The results show the detection of car keys using the Haar cascade classifier that was created using the cascade trainer, the success rate and false positive detections could have been better by using more positive and negative image samples.

### Future Scope:-

The future scope includes the addition of more positive and negative image samples to get better success rate and reduce the number of false positive detections.

### Paper – 4

The System of Face Detection Based on OpenCV by *Xianghua Fan, Fuyou Zhang, Haixia Wang, Xiao Lu*

### Main Contribution:-

The author discusses about the face recognition technologies for the video surveillance applications. This was done using the modified AdaBoost algorithm which includes two methods

of its own, timer and dual thread. The paper discusses about these two methods and which one is better for a real-time face detection application.

**Algorithm Used:-**

- AdaBoost: Methods of timer and dual-thread

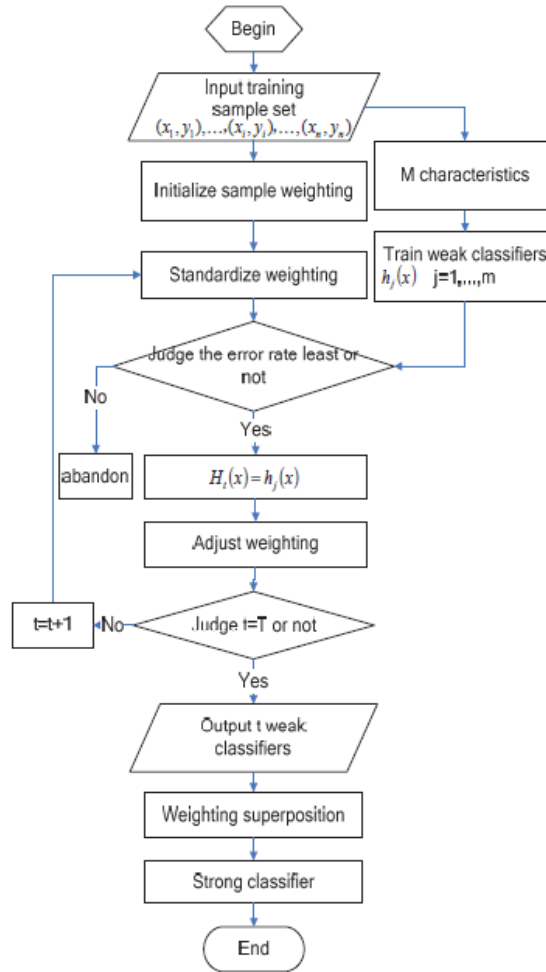


Figure 7: Modified LBPH flowchart

**Result Analysis:-**

Two experiments were done to check which methods is better for face detection. Both methods performed effectively but the dual-thread method was smooth for multiple face detections and certainly worked better to detect faces in different lighting conditions.

## Paper – 5

A Real-time Face Recognition System Based on the Improved LBPH Algorithm by *XueMei Zhao, ChengBing Wei* (IEEE/2017)

### Main Contribution:-

In this paper, the author works on real-time face recognition system using the local binary pattern histogram algorithm. This algorithm was used because it can detect from front face as well as the side face. A modified LBPH algorithm was used for face recognition.

### Algorithm Used:-

- Modified LBPH (MLBPH)

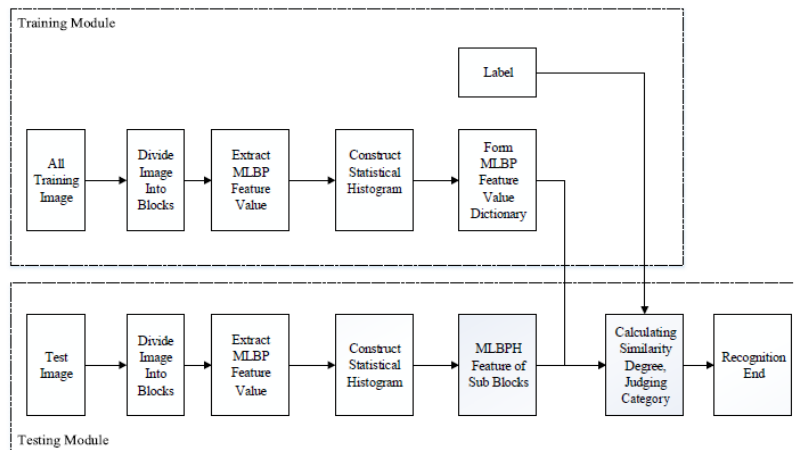


Figure 8: MBPH flowchart

### Result Analysis:-

The results shows that there were some drawbacks of using LBPH algorithm, therefore there is a need of MLBPH algorithm that reduces the illumination and the attitude deflection.

### Future Scope:-

The future scope includes the work on the problem faced, that of lens distortion. If this problem is solved, then the rate of face recognition would increase.

## **Paper – 6**

**Class Attendance Using Face Detection and Recognition with OPENCV by *K. Yamini, S. Mohan Kumar, S. Sonia, P. V. Yugandhar, T. Bharath kumar***

### **Main Contribution:-**

In this paper, the author works on real-time attendance application that works on the domain of face recognition. It manages the way toward taking the participation with use camera and robotizing the participation procedure that will stamp the participation for the understudies in simple and straightforward way without burning through of time and lessen Statistical procedure, the proposed framework utilizes face discovery for distinguishing proof of face from articles and face acknowledgment for coordinating of appearances from put away database pictures (validation) and give participation as indicated by the coordinated face. To achieve this face discovery and acknowledgment, we use viola-Jones calculation (HAAR-Cascade) for face identification and direct double example histograms for face verification utilizing python and bringing in the OPENCV structure to python IDE.

### **Algorithm Used:-**

- HAAR-Cascade
- RFID
- AdaBoost
- HOG: Histogram of Oriented Gradients
- LBP: Local Binary Patterns

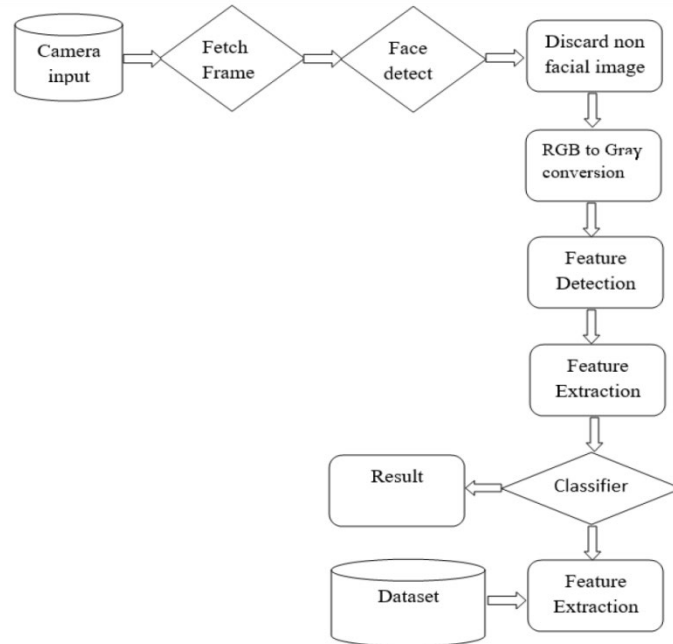


Figure 9: Algorithm Flowchart

### Result Analysis:-

The results shows that a framework was built that uses face recognition algorithms to detect faces and can be replace the fingerprint biometrics.

### Paper – 7

**A Study of LBPH, Eigenface, Fisherface and Haarlike features for Face recognition using OpenCV by A. M. Jagtap, Vrushabh Kangale, and Kushal Unune. (ICISS2019)**

### Main Contribution:-

In this paper, the author works on an application that works on the domain of face recognition and selecting the best algorithm. The author works on different facial acknowledgment algos like Eigenface, LBPH and Fisherface, also prepared the calculations utilizing similar informational index and have a few bits of knowledge, from which we have attempted to recognize which calculation gives us the best outcomes. Unique calculations are looked at and their operations are talked about. Toward the end, even examinations are given. With the goal that it is more clear the contrast between calculations.

### Algorithm Used:-

- HAAR-Cascade
- EigenFace
- FisherFace
- LBPH: Local Binary Patterns Histogram

### Result Analysis:-

The results are given in the below tables:

Table 3

5 Subject						
	Training 10 pics of Subject			Training 40 Pics of Subject		
	Correct	Error	Result	Correct	Error	Result
Eigenface	46 pics	4 pics	92%	50 pics	0 pics	100%
Fisherface	46 pics	4 pics	92%	50 pics	0 pics	100%
LBPH	48 pics	2 pics	96%	50 pics	0 pics	100%

Table 4

10 Subject						
	Training 10 pics of Subject			Training 40 Pics of Subject		
	Correct	Error	Result	Correct	Error	Result
Eigenface	46 pics	4 pics	92%	50 pics	0 pics	100%
Fisherface	46 pics	4 pics	92%	50 pics	0 pics	100%
LBPH	48 pics	2 pics	96%	50 pics	0 pics	100%

**Future Scope:-**

The future scope includes more work on PCA algorithm as they were found to be better than LBPH algorithm. Eigen face and Fisherface algorithms work better when used individually and LBPH algorithm works better along-side HAAR-Cascade classifier.

## Chapter 3

### SYSTEM DEVELOPMENT

#### 3.1) Software and Hardware Requirement Specification:

- Intel Core i5 7<sup>th</sup> generation (or above).
- 6 GB + RAM
- Windows 10(recommended)
- Python version: 3.5+
- Camera source

#### 3.2) Model Development:

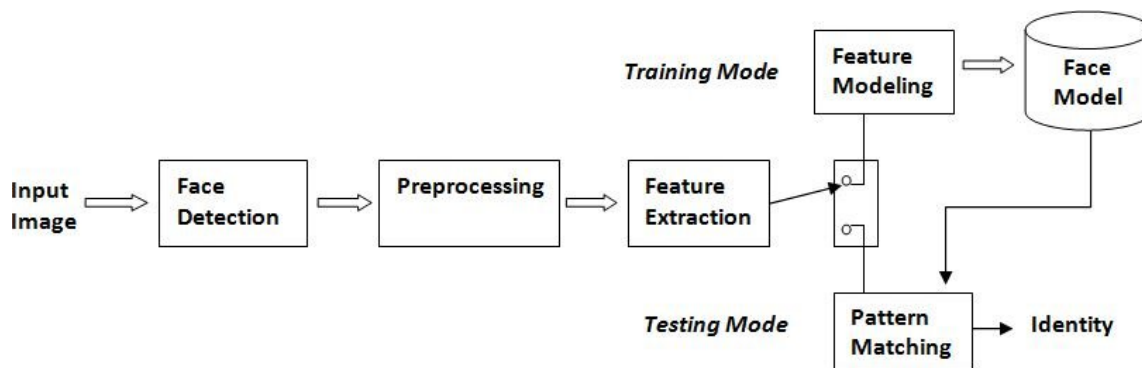


Figure 10: Model Development

The initial step includes the collection of user images using a cascade classifier that detects human faces and thereafter images are stored, then these image samples are then used to train a model which includes the feature modeling and pattern matching algorithms to detect human face using combination of features such as eyes, nose, and mouth.

#### 3.3) Face Detection:

The first stage includes the detection of face using the HAAR-cascade classifier, these detected facial images are cropped and stored in a folder. Training a cascade classifier is the biggest challenge in face detection. In this project a human face cascade classifier was used in detection of these faces. The below image describes the process of face detection:



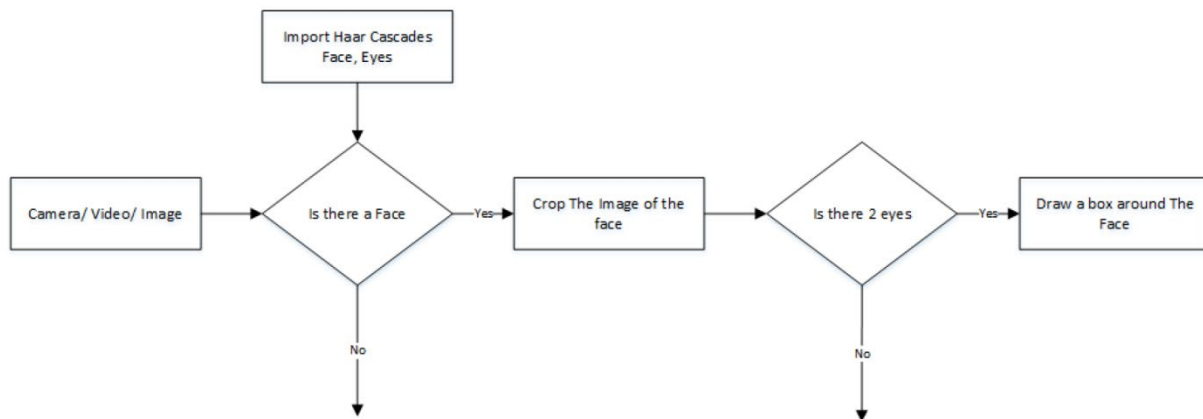


Figure 11: Flow chart of the face detection application

OpenCV is a library in python that was used to detect faces using the cascade classifier class with the use of the `cv2.CascadeClassifier()` function which is to be executed after the XML file is loaded to the application. Video input is necessary and can be done using the function `cv2.VideoCapture()` and to capture the video using a webcam we provide the input as 0 to the video source. Sometimes the face might be too close or sometimes it might be too far from the camera so this needs to be scaled to such a size that various sizes of objects can be matched, this is done using `detectMultiScale()` function.

### 3.4) Stages of Face Recognition:

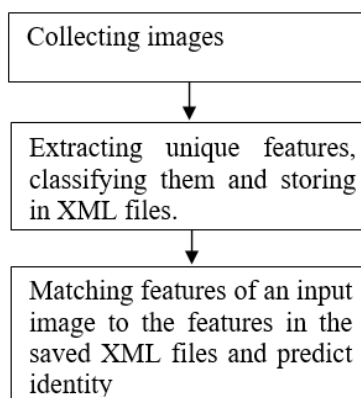


Figure 12: Stages of Face Recognition

Each one of the above mentioned face recognition stages are mentioned below in detail.

## 1) Stage I:

Stage I includes the *collection of image samples*. The collection of images is done using a HAAR-cascade classifier that detects human face and crops out the rest of the captured area. This cropped image is then stored in a database which is used to train a model in later stages. The images are captured only in bright light and ignored if no face is detected. A count of 100 is set such that 100 image samples are stored for each user. The geometrical location of eyes are validated after the face detection is completed. The captured image is cropped and stored in the corresponding database. HAAR-cascade classifier is one of the most efficient way to collect image samples till date.

## 2) Stage II:

This is the major step that includes Training the classifiers. The training of model is done using the OpenCV module in python (cv2) which includes several FacialRecognizer algos such as `cv2.face.createEigenFaceRecognizer()`, `cv2.face.createFisherfaceRecognizer()`, and `cv2.face.createLBPHFaceRecognizer()`. The previously stored images are imported to the python file which are then converted to grayscale images to reduced time complexity of the code. The face recognizer class helps in most of it, each of the recognizer has its own unique approach from feature extraction to validation of the input given to it. Somehow EigenFace and FisherFace have similar approach. The various recognizers are described below:

- a) **LBPH:** It stands for Local Binary Patterns Histograms which was created in the year 1996. It uses the algorithm Local Binary Pattern, known as LBP found in 1994 which labels the pixels into an array of the input given, the adjacent values of each pixel are thresholded giving the output as binary number. It is a powerful algorithm that has the ability to extract even the smallest of the features. It uses data vectors that are created by combining the HOG (Histogram of oriented gradients) to get more accurate results. This enhances the face recognition. The step by step implementation is given below:
  - 1) *Parameters:* This algorithm requires 4 parameters which are: radius, neighbours, gridx and gridy. Each one of the parameters has its own use. Radius value describes the local binary pattern along the center that is set to 1 by default. The neighbours of each pixels are used to form the LBPs. This value is 8 by default as each pixel has at most 8 neighbouring pixels that help creating a circular LBP. The gridx and the gridy

values represent the number of spaces in the x-axis and the y-axis respectively. They have a default value set to 8.

2) *Model Training*: This step includes the training of the algorithm which is done with the use of the previously collected image samples for each user. If there are multiple user image samples then we need to label each one of the image for this algorithm such that it identifies each user uniquely. Further we discuss the real algorithm functionality.

3) *LBP Algorithm*: The LBP algorithm needs to be applied to the model trained, hence the first step includes the formation of a transitional image. The features in an image are the key to face recognition and this transitional image contains the features extracted for each user which is used to describe the original image. The below image describes it step-by-step:

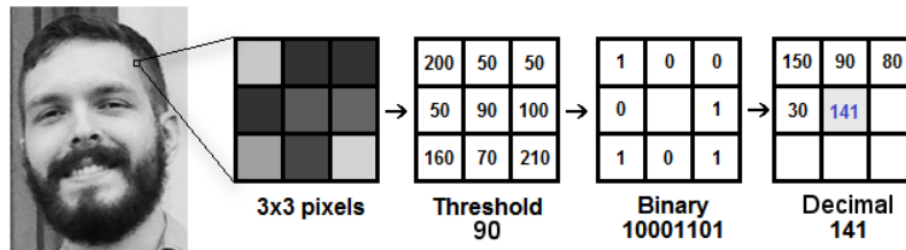


Figure 13-Grayscale Image

In the image given above, it needs to be converted to gray scale due to the processing speed. If a given image is in any other format (RGB, PGM, etc) then the processing speed would be much slower as compared to the gray scale images. We know that each pixel is surrounded by 8 other pixels, so after adding them up, we get a matrix of size 3x3. Each pixel has a value in range 0 to 255. This value signifies the intensity of each pixel. A threshold value needs to be set and it has to be taken as the center of the matrix, as given in the image above 90 is the intensity of the center pixel so the threshold is set to 90 for the given 3x3 matrix. After the threshold value is fixed, we need to create a binary matrix that satisfies the following equation:

$$f(x,y) = \begin{cases} 0, & \text{pixelintensity}(x,y) < \text{threshold} \\ 1, & \text{otherwise} \end{cases} \quad 3.1$$

In the given equation, the pixel value ie  $f(x, y)$  will be set to 0 if the intensity(x, y) less than the threshold value else it will be set to 1. After we get the binary matrix we need to create a decimal matrix for each pixel and ignoring the center pixel. This process includes the concatenation of the binary values in line-by-line format which creates a new binary matrix. Even though there are many ways to get to the new binary matrix, but the output would always be same irrespective of the way you do it (clockwise/ anticlockwise). The next step includes the formation of decimal matrix using the new binary matrix, and this value is set to the central pixel value. This helps us to get a better quality image than that of the original image. This in the most reliable algorithm in terms of speed and accuracy, and is also one of the most efficient algorithm. The various samples are given below by assuming values of radius and neighbours:

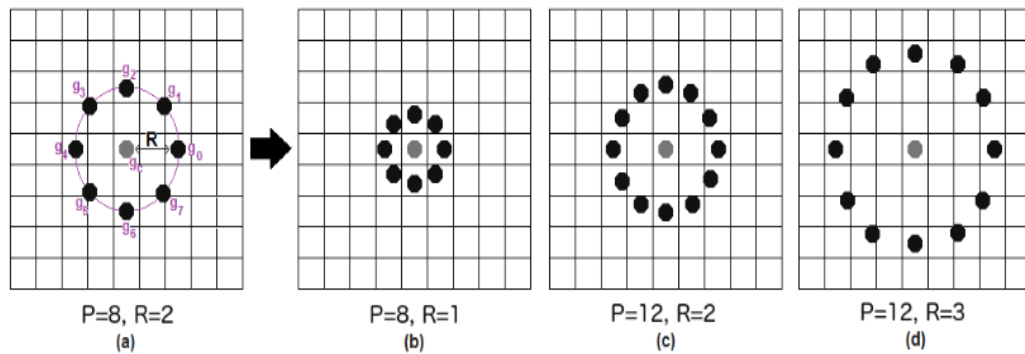


Figure 14- Histogram Plot

In this algorithm, we can change the default values of the neighbours as well as the radius to get better results. This in other words is known as the bilinear interpolation, which helps in assumptions of new data values.

4) *Extraction of Histograms*: After the image is processed, this step includes the use of *gridx* and *gridy* arguments. These values are used to split the image into separate graphs as given below:

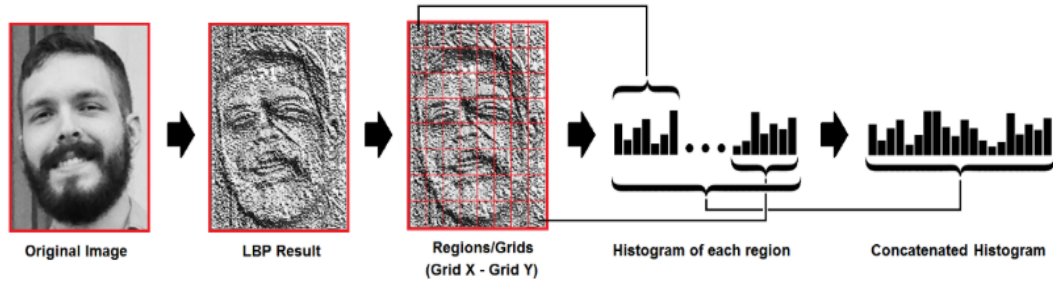


Figure 15- Histogram Graphs Plotted

Each histogram contains 0 to 255 points which denotes the number of times each intensity level occurs for all the pixels. All the histograms are generated using this for every region of the image and is concatenated to one bigger histogram. This histogram is the final result of the LBPH algorithm that represents all the features of the input image.

5) *FaceRecognition*: This is the final step of the LBPH algorithm, after the model is trained using the data provided i.e. the image sample. Another histogram is created for dataset and the above mentioned steps are followed again. Now a new histogram is received for the new image sample. Now we need to validate the face sample that we earlier collected with that of the input image given to us, so the two histograms need to be compared to each other. This returns us the image that is closest to the input histogram. This algorithm uses the **Euclidean distance** formulae to calculate the variation between two histograms, given below:

$$ED = \sqrt{\sum_{i=1}^n \{his1(i) - his2(i)\}^2} \quad 3.2$$

Here  $i$  represents the ID for each user that needs to be compared to the histogram generated using the input image given to it. This distance value is used to calculate the confidence value using the given equation:

$$confidence \% = (100 * (1 - ED/300)) \quad 3.3$$

A threshold percentage value is set as per the user's requirement, and this value

decides whether the detected image belongs to a user in the database or not. The higher the confidence percentage level the more is the level of features matching to the model trained. Therefore, the final result can be defined by using the confidence value after comparing it to the threshold value.

- b) Eigenface Recognizer:** *Eigenface algorithm* is the oldest of all the face recognition algorithms, found in the year 1991. This algorithm works on the principle of PCA that stands for Principal component analysis. It takes modules from the PCA that creates the Eigenface. To train an average level of model we need to use 80 as per the documentation provided by the OpenCV. It returns a value of -1 if the algorithm is unable to recognize the face. This value is calculated using a threshold value to calculate the distance from the Eigenface vector. The syntax used for this algorithm is `cv2.face.createEigenFaceRecognizer()`.
- c) Fisherface Recognizer:** This algorithm was designed in the year 1997, it works on the principle of LDA which stands for Latent Dirichlet Allocation that is based on the ‘bag-of-words’ model. Fisherface algorithm takes LDA modules as the input and returns the Fisherface as result. . To train an average level of model we need to use 0 as per the documentation provided by the OpenCV. It returns a value of -1 if the algorithm is unable to recognize the face. This value is calculated using a threshold value to calculate the distance from the Eigenface vector. The syntax used for this algorithm is `cv2.face.createFisherFaceRecognizer ()`. This algorithm is similar to that of the Eigenface except for the input components given to each one of them differs. Fisherface uses the LDA components whereas the Eigenface algorithm uses the PCA components as inputs. And the rest procedure is same to each one of them where a threshold value is set by the user and a confidence value is calculated. After the calculation of this value it is chosen whether the trained model matches to the input image or not. A vector file is created that stores the images that are imported to the program after converting the into numpy arrays. The training of the model is done using the `FaceRecognizer.train (NumpyImage, Labels)`. The resizing of images is not required

for LBPH algorithm. The model is trained, and *FaceRecognizer.save (FileName)* can be used to save our trained model as a XML file.

### 3) Stage III:

This stage is the final stage of facial recognition. The face recognition object that is created by the corresponding class after the user gives the most probable arguments to it. The face detection is done by the face detection algorithms and face recognition is done with the help of the above mentioned algorithms, out of which only the best suitable algorithm is to be used. OpenCV includes a function *FaceRecognizer.predict()* which is used to predict or calculate a confidence value by comparing the real-time video samples to the model trained using the image samples collected earlier. The calculation of confidence value is similar to all the algorithms that are described above. The confidence value is compared to a threshold value set by the user as per the requirement that validates the face given to it as the input. If the confidence value is greater than the threshold then we can say that the face matches to some extent while there are many factors that are responsible in affecting the confidence value. The Face Recognition value depends on the following factors: Face size, face features and face direction whereas the confidence value depends on the skin tone, as per a research by MIT darker skin tones were less precise when compared to other skin tones. The other factors include the surrounding light as it needs to be bright enough to capture all the face features as accurately as possible, moreover the low light images were unable to produce expected outputs. If the confidence level is lesser than the threshold value then the program prints unknown face. The following flowchart shows the step by step implementation of the Face Recognition system.

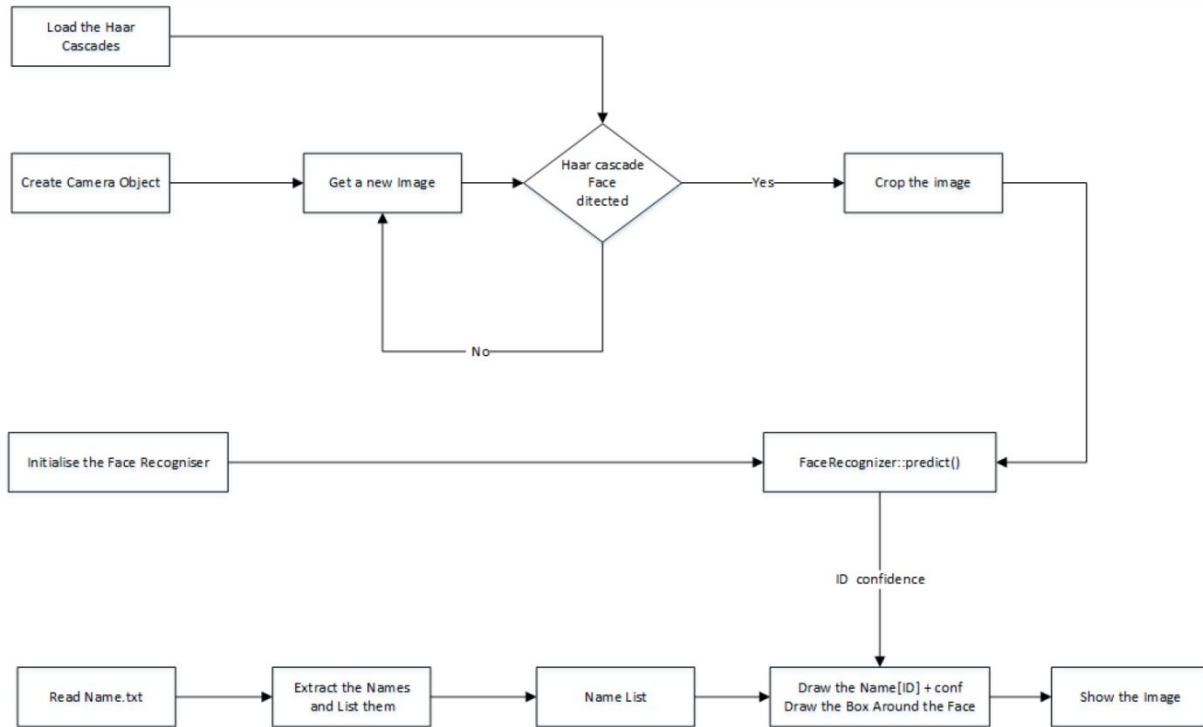


Figure 16: Face Recognition Flowchart

### 3.5) Libraries Used:

#### OpenCV: -

OpenCV stands for Open Source Computer Vision. It is used in many areas of reasearch as well as daily applications, some of the areas where it is used are image processing, video processing, feature extraction, video analysis, object detection, and many other machine learning areas. It is a vast library and is found in many other languages such as GO, Java, Python and R. In our project we use this library for face detection using a machine learning algorithm called HAAR-cascade in which a front face classifier was used to detect face and the cropped images were stored in a folder. The next task is to recognize the human face that was already stored in the directory. After the model is trained, these images were used by the OpenCV library to train a model. This library has inbuilt algorithm that reduces the human mathematical calculations and returns the result by applying the algorithms that we have already disussed before. All the maths is done by the computer itself to reduce human errors and save a few chunk of lines. The last step includes the



prediction of the face that is given as the input to the model trained. It calculates the euclidean distance value or in some books it is said to be the confidence value. This value is compared to the threshold value to check whether the algorithm provided us with the best of the results.

- The general methods used for collection of sample images:
  - i. `cv2.CascadeClassifier (classifier)`
  - ii. `cv2.cvtColor (image, colour)`
  - iii. `cv2.detectMultiScale (image, scale, minNeighbours)`
  - iv. `cv2.VideoCapture(source)`
  - v. `cv2.resize(image, (x, y))`
  - vi. `v2.imwrite(file_name_path, image)`
  - vii. `cv2.putText(image, string, (x, y), font)`
  - viii. `cv2.imshow(string, image)`
  - ix. `cv2.waitKey()`
  - x. `release()`
  - xi. `cv2.destroyAllWindows()`

- The functions used for face recognition:
  - i. `cv2.CascadeClassifier (classifier)`
  - ii. `cv2.face.LBPHFaceRecognizer()`
  - iii. `model.train(training_data, labels)`
  - iv. `model.predict (image)`

The other functions were already discussed.

## **NumPy:-**

NumPy's principle object is that the consistent multidimensional exhibit. it's a table of elements (normally numbers), everything of an identical kind, recorded by a tuple of non-negative whole numbers. In NumPy measurements are referred to as tomahawks. For instance, the directions of some extent in 3D house [1, 2, 1] has one pivot. That hub has three elements in it, thus we have a tendency to state it's a length of three. Within

the model given beneath, the cluster has two tomahawks. The first hub includes a length of two, the following hub includes a length of three. NumPy's cluster category is named ndarray. It's to boot well-known by the alias cluster. Note that numpy.array isn't comparable to the quality Python Library category array.array, which simply handles one-dimensional clusters and offers less utility.

- Some of the functions that were used are given below:
  - i. `numpy.asarray()`
  - ii. `numpy.uint8`

## Chapter 4:

### IMPLEMENTATION AND RESULT

#### 4.1) Test Cases:

After we have trained our model, we need to check whether the given model works properly.

The following are the given set of test plans:

**1) The user, for whom the model was trained:**

The first test plan includes to focus on the person for whom the model has been trained. The expected output is 'Verified' as it has been trained for that same user.

**2) Another user, for whom the model was not trained:**

The second test plan includes to focus on some other user, whose data has never been stored. The expected output is 'Unknown' as this user has to be a random person.

**3) When no face is given as input:**

The expected output for this case would be 'Face Not Found'.

#### 4.2) Collecting image samples:

The following images describe the collection of sample images:



The sample image collection starts collecting the sample images and saves them to a specified folder. The below image depicts the sample images being stored to a folder.



Figure: Collected Image Samples

### 4.3) Model Training and testing:

1. The model training was done by the OpenCV module.
2. The algorithm used to recognize the face is the LBPH algorithm.
3. The Accuracy achieved by the model is 81.6%.

## 4.4) Implementation:

### Code for collecting sample images:

```
import cv2
import numpy as np

face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

def face_extractor(img):

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces =face_classifier.detectMultiScale(gray,1.3,5)

    if faces is():
        return None

    for (x,y,w,h) in faces:
        cropped_face = img[y:y+h ,x:x+w]

    return cropped_face

cap =cv2.VideoCapture(0)

count = 0

while True:
    ret,frame = cap.read()
    if face_extractor(frame) is not None:
        count+=1
        face = cv2.resize(face_extractor(frame), (200,200))
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

        file_name_path = 'D:/1. MY DATA/_ML/PROJ/New/DataCollected/face1/user'+str(count)+'.jpg'
        cv2.imwrite(file_name_path,face)

        cv2.putText(face,str(count), (50,50),cv2.FONT_HERSHEY_COMPLEX,1, (0,255,0) ,2)
        cv2.imshow('face Cropper',face)
    else:
        print("Face not Found")
        pass

    if cv2.waitKey(1)==13 or count==100:
        break

cap.release()
cv2.destroyAllWindows()
print('Collecting Samples Complete!!! ')
```

## Code for Face Recognition:

```
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join

data_path = 'D:/1. MY DATA/_ML/PROJ/New/DataCollected/face1/'
onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path,f))]
Training_Data = []
Labels = []

for i, files in enumerate(onlyfiles):
    image_path = data_path + onlyfiles[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images, dtype =np.uint8))
    Labels.append(i)

Labels = np.asarray(Labels, dtype=np.int32)
model = cv2.face.LBPHFaceRecognizer()
model.train(np.asarray(Training_Data), np.asarray(Labels))
print("Model Train Complete !!!")
face_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

def face_detector(img, size = 0.5 ):
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces =face_classifier.detectMultiScale(gray,1.3,5)
    if faces is():
        return img, []
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,255),2)
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200,200))
    return img,roi

cap =cv2.VideoCapture(0)
while True:
    ret,frame = cap.read()
    image, face = face_detector(frame)
    try:
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        result = model.predict(face)
        if result[1] < 500:
            confidence = (100*(1-(result[1])/300))
            display_string = str(confidence)+'% Confidence it is user'
            cv2.putText(image,display_string, (100,120), cv2.FONT_HERSHEY_COMPLEX,1, (250,120,255),2)
        if confidence > 77 :
            cv2.putText(image, "Verified", (250,450), cv2.FONT_HERSHEY_COMPLEX,1, (0,255,0),2)
            cv2.imshow('Face Recognition', image)

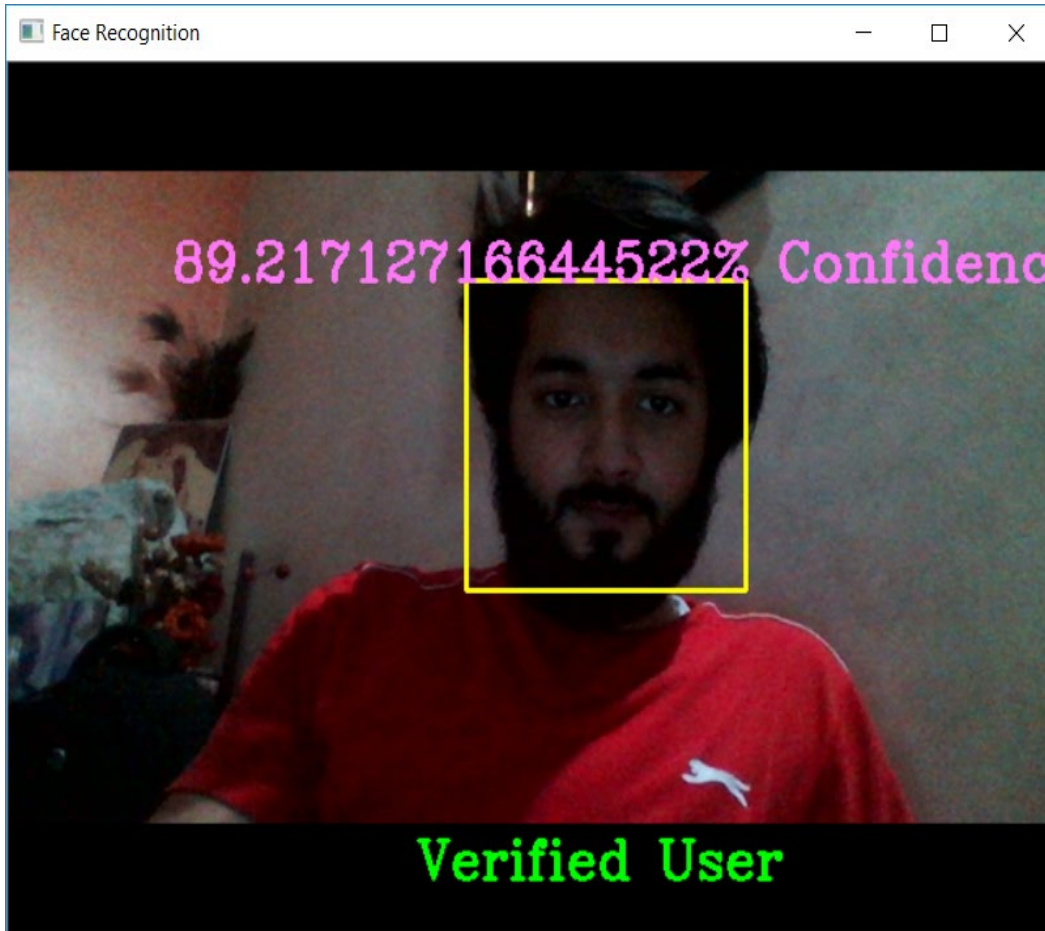
        else:
            cv2.putText(image, "Unknown", (250,450), cv2.FONT_HERSHEY_COMPLEX,1, (0,255,0),2)
            cv2.imshow('Face Recognition', image)
    except:
        cv2.putText(image, "Face not found", (250,450), cv2.FONT_HERSHEY_COMPLEX,1, (255,0,0),2)
        cv2.imshow('Show Your Face', image)
        pass

    if cv2.waitKey(33)==27:
        break

cap.release()
cv2.destroyAllWindows()
```

#### 4.4) Output: -

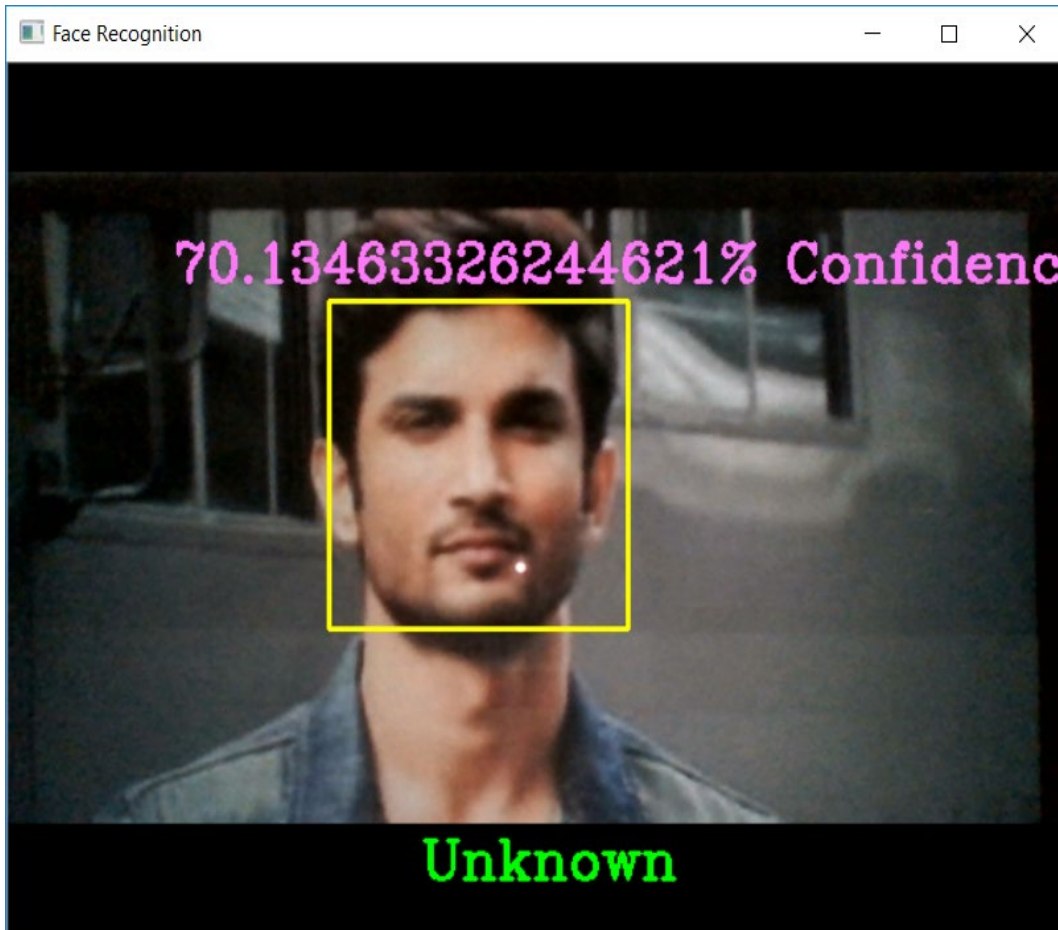
##### 1) The user, for whom the model was trained:



##### Console Output:

```
*Python 3.8.2 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\1. MY DATA\_ML\PROJ\New\Main.py =====
Model Train Complete !!!
```

**2) Another user, for whom the model was not trained:**

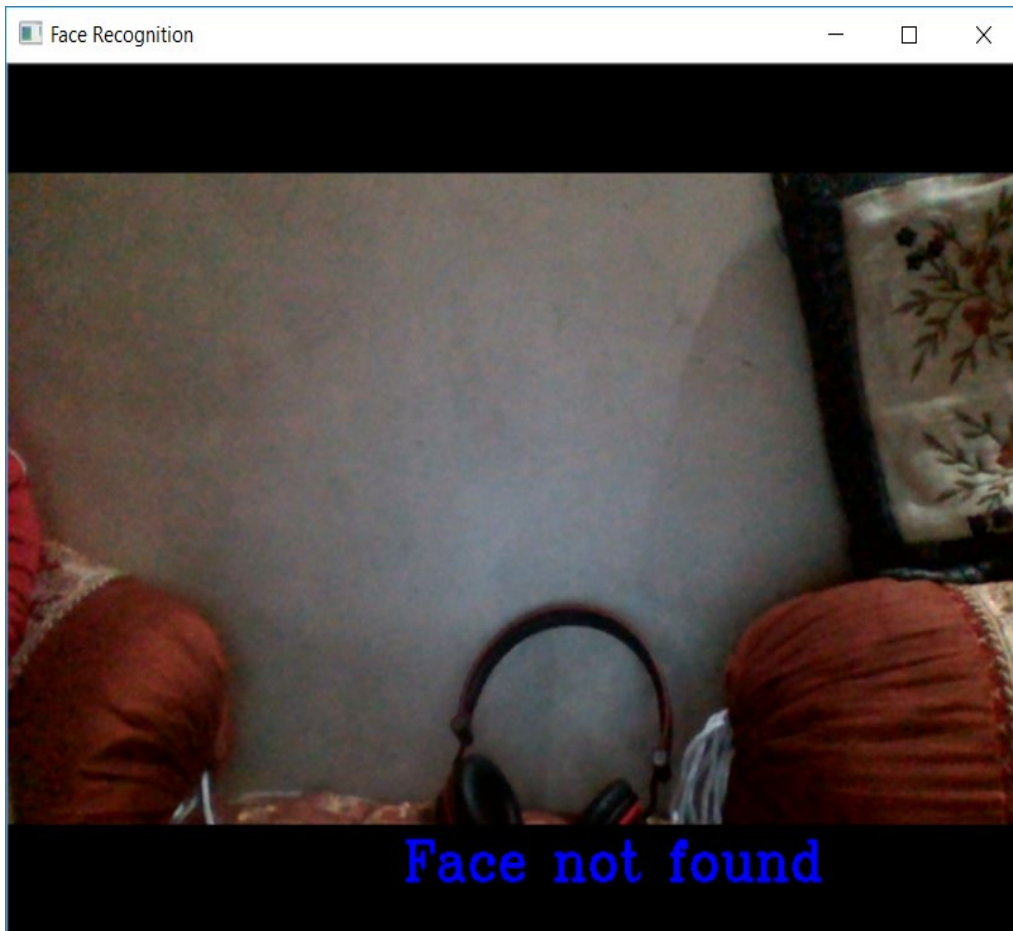


**Console Output:**

```
*Python 3.8.2 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\1. MY DATA\_ML\PROJ\New\Main.py =====
Model Train Complete !!!
```



### 3) When no face is given as input:



### Console Output:

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
===== RESTART: D:\1. MY DATA\ML\PROJ\New\Main.py ===== ^
Model Train Complete !!!
No face found
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
Ln: 14 Col: 0
```

## **Chapter 5:**

### **CONCLUSION**

#### **5.1) Conclusions:**

This paper describes the mini-project on face detection and face recognition. This project is divided into two separate areas of face detection and face recognition. Under the face detection part, Haar-cascades was used in detecting the user face and sample images were collected. In the second part, after the model was trained for the sample input images, LBPH algorithm was used in implementing of this project, as we have already discussed it earlier in this project. The major advantage of this project includes the HAAR-cascade combined with one of the finest face recognition algorithm LBPH makes it a cost effective as well as a reliable face recognition application. Face recognition has been studied since the last few years, not only in the security areas but also for various areas such as Attendance systems, validating identity at ATM's, finding criminals and many other areas.

#### **5.2) Future Scope:**

The future work includes the work on the security feature that would send the image captured to the admin's registered email address. As we have already discussed, there are many other algorithms such as Eigenface and Fisherface that could have also been used so a detailed analysis on which algorithm works the best would be a secondary goal for the project. This would not only help in the enhancing the accuracy of the face recognition algorithm but also help to increase the security features.

## REFERENCES

- [1]: A Comparative Study between LBP and Haar-like features for Face Detection Using OpenCV by *Kushsairy Kadir, Mohd Khairi Kamaruddin, Haidawati Nasir, Sairul I Safie, Zulkifli Abdul Kadir Bakti*
- [2]: Face Detection and Recognition Using OpenCV by *Maliha Khan, Sudeshna Chakraborty, Rani Astya, Shaveta Khepra* (ICCCIS/2009)
- [3]: HAAR like Feature-Based Car Key Detection Using Cascade Classifier by *Paawan Sharma, Mukul K. Gupta, Amit K. Mondal and Vivek Kaundal* (2016)
- [4]: The System of Face Detection Based on OpenCV by *Xianghua Fan, Fuyou Zhang, Haixia Wang, Xiao Lu*
- [5]: A Real-time Face Recognition System Based on the Improved LBPH Algorithm by *XueMei Zhao, ChengBing Wei* (IEEE/2017)
- [6]: Class Attendance Using Face Detection and Recognition with OPENCV by *K. Yamini, S. Mohan Kumar, S. Sonia, P. V. Yugandhar, T. Bharath kumar*
- [7]: A Study of LBPH, Eigenface, Fisherface and Haarlike features for Face recognition using OpenCV by *A. M. Jagtap, Vrushabh Kangale, and Kushal Unune*. (ICISS2019)
- [8]: <https://www.towardsdatascience.com>