REAL-TIME LOW-RESOLUTION FACE RECOGNITION USING LOCAL BINARY PATTERN HISTOGRAMS, EIGENFACE, AND FISHERFACE ALGORITHMS

by

Kamal Chandra Paul, B.Sc.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
December 2018

Committee Members:

Semih Aslan, Chair

William Stapleton

Bahram Asiabanpour

DEDICATION

This M.Sc. thesis is delightfully dedicated to my beloved parents (Khagendra Nath Paul & Amala Rani Paul), my dear wife (Samonty Das), and my beloved sister (Rikta Paul) for their endless support and without whom this achievement could never be accomplished. Their continuous encouragement and love have always energized me to go through this entire journey of my higher studies. May God bless you all.

ACKNOWLEDGMENTS

I want to express my sincere gratitude to my committee chair and thesis advisor, Dr. Semih Aslan, for his continuous guidance, moral support, wisdom, patience, encouragement, consideration, and for advising me on each step of the journey that without all his supports this research would never have been accomplished fruitfully. I am grateful to my thesis committee member, Dr. William Stapleton, for his constructive criticism and making valuable suggestions and comments on every step of this thesis. In addition, I would like to convey my special thanks to my thesis committee member, Dr. Bahram Asiabanpour and graduate advisor, Dr. Vishu Viswanathan for their attention, time, guidance and excellent support throughout this research.

On top of all, I would like to express my utmost gratitude is to my beloved parents and my wife Samonty Das for their endless mental support and encouragement. I am in debt to you all and without you all, my advanced degree would not have been possible.

TABLE OF CONTENTS

Page
ACKNOWLEDGMENTS vii

LIST OF TABLES xi

LIST OF FIGURES xiii

LIST OF ABBREVIATIONS xvi

ABSTRACT xviii

CHAPTER

1. INTRODUCTION 1

2. LITERATURE REVIEW 5

2.1 Face Detection 7

2.2 Face Recognition Algorithms 11

2.2.1 Local Binary Pattern Histograms 11

2.2.2 Eigenface 14

2.2.3 Fisherface 18

2.3 Image Enhancement Techniques 20

2.3.1 Histogram Equalization 20

2.3.2 Adaptive Histogram Equalization 22

2.3.3 Contrast Limited Adaptive Histogram Equalization 23

3. PROPOSED FACE RECOGNITION SYSTEM 25

3.1 Overall Proposed Face Recognition System 25

3.2 Specification of Devices and Cameras Used 26

3.3 Creating the Face Database and Starting the Training Process 27

3.4 Face Recognition using LBPH 34

3.5 Face recognition using Eigenface Algorithm 36

3.6 Face recognition Using Fisherface Algorithm 39

3.7 Combined LBPH and Fisherface Algorithm 40

4. EXPERIMENTAL RESULTS AND DISCUSSION 42

4.1 Effect of CLAHE on Poorly Illuminated Image 42

4.2 Recognition Results using LBPH Method 42

4.2.1 Face Recognition under Different Low-resolutions 43

4.2.2 Face Recognition with Different Angular Positions 46

4.3 Recognition Results Using Eigenface Algorithm 48

4.3.1 Face recognition Under Different Low-resolutions 48

4.3.2 Face recognition with Different Angular Positions 51

4.4 Recognition Result using Fisherface Method 54

4.4.1 Face Recognition Under Different Low-Resolution 54

4.4.2 Face Recognition with Different Angular Positions 59

4.5 Recognition Results Using the Combined LBPH and Fisherface Algorithm 62

4.5.1 Face recognition with Different Number of Images Per Person 62

4.5.2 Face recognition with Different Number of Subjects 66

5. CONCLUSION 70

APPENDIX SECTION 73

REFERENCES 89

LIST OF TABLES

Table **Page**

1. Previous Works on Face Recognition 6

2. Specification of the camera of the Android phone 26

3. Specification of the webcam of the laptop 27

4. Description of the developed Android app 29

5. Recognition Rate using the LBPH Method on the LRD200 database 44

6. Recognition Rate Based on the LRD100 Database 44

7. Recognition rate using LBPH method on the LRD200 database with the different angular positions of the face 46

8. Recognition rate using the Eigenface method at different resolution of the input image on the LRD200 database 48

9. Recognition rate using the Eigenface method under different resolution conditions on the LRD100 Database 49

10. Recognition rate using the Eigenface method for the different angular positions of the face based on the LRD200 database 52

11. Face recognition using the Fisherface method under different resolutions of the input image based on the LRD100 database. 54

12. Face recognition using the Fisherface with CLAHE method under different resolutions of the input image based on the LRD100 database. 55

13. Face recognition using the Fisherface algorithm under different resolutions of the input image based on the LRD200 database. 56

14. Face recognition using Fisherface with CLAHE method under different resolutions of the input image based on the LRD200 database. 56

15. Recognition rate, with the different deflection angles of the face, using the Fisherface method based on the LRD100 database at 45 px of the input image. 61

16. Recognition rate, with the different deflection angles of the face, using the Fisherface with CLAHE method based on the LRD100 database at 45 px of the input image. 62

17. Experimental results of face recognition using combined LBPH & Fisherface algorithm.......... 64

18. Recognition rates (at 45 px of the input image) using the combined LBPH and Fisherface algorithm with CLAHE method based on the database having 50 images per person and a varying number of subjects. 67

LIST OF FIGURES

Figure **Page**

1. Basic Face Recognition Framework 1
2. Formation of the integral image (ABCD) and Haar-like features (a-i) 7
3. Relevant Haar Feature for face detection........ 8
4. Formation of the integral image (b) from the image table (a) 8
5. A cascade classifier of 25 stages and the decision tree (T=Ture, F=False) [18] 9
6. The Flowchart of a Real-time Face Detection System 10
7. A 3x3 LBP operator 11
8. Circular neighborhoods of the center pixel with differ neighbor pixels 13
9. The formation of the final histogram using LBPH 13
10. Sample eigenfaces corresponding to a different number of eigenvalues [20] 16
11. Visualization of Eigenface Algorithm 17
12. Constructing a face by adding eigenvectors 18
13. Classes in PCA and LDA where LDA has well-segregated classes than PCA [24]. 19
14. Histograms of an image before and after Histogram Equalization is applied [28] 21
15. Histograms of an image before and after Histogram Equalization is applied. 21
16. Redistribution of the clipped part among all the histogram bins. 24
17. Block diagram of the overall proposed system 25
18. Sample face image of LRD200 database 28

19. A few snapshots of the My New Cam Android app showing various screens. 30

20. Pictures of the Android app showing the total number of images
in the gallery (left) and a few images in the gallery (right). 31

21. Block diagram of the overall process of database creation
and automatic training. 32

22. Flowchart of the process of creating a database and automatic training. 33

23. Flowchart for face recognition.. 35

24. Schematic Diagram of Eigenface method of Face Recognition. 37

25. Flowchart for a real-time face recognition using the Eigenface method. 39

26. Face recognition flowchart using combined LBPH & Fisherface method. 41

27. Real-time observation of the effect of CLAHE and
Filtered CLAHE from a video frame. 42

28. Face images of different resolutions for recognition. 45

29. Face recognition accuracy with image resolution (pixel). 46

30. Face recognition with different angular deflection (i) front facing,
(ii) right 30^0 facing, (iii) left 30^0 facing... 47

31. Face recognition at 15 px and corresponding low-resolution image of 15 px. 50

32. Real-time face recognition: known and unknown 50

33. Graphical representation of recognition accuracy with
different image resolution (pixel). 51

34. Recognition in Eigenface algorithm with i) front facing ii) 30^0 right facing
& iii) 30^0 left facing.. 53

35. Face recognition using the Fisherface algorithm (45px). The person on the left side is not included in the database and is marked as unknown, whereas the other person is recognized correctly. The red color text is for the Fisherface only, and pink color text is for the Fisherface with CLAHE method. 57

36. A comparative representation of facial recognition rate using the Fisherface algorithm with and without CLAHE. 58

37. An image frame taken from a real-time video feed is showing that the (Fisherface + CLAHE) algorithm can recognize the face correctly (pink text) whereas the Fisherface algorithm alone cannot (red text). 59

38. Recognition with a different angular position in Fisherface (red text) and Fisherface with CLAHE (pink text) method at 15 px. 60

39. Graphical representation of various recognition rates using the combined algorithm without CLAHE method. 65

40. Graphical representation of various recognition rates using the combined algorithm and CLAHE. 66

41. Graphical representation of face recognition rates with a different number of subjects. 67

42. Real-time face recognition using combined LBPH and Fisherface algorithm with CLAHE where the unknowns are not included in the database. 68

LIST OF ABBREVIATIONS

Abbreviation	**Description**
AHE	Adaptive Histogram Equalization
CLAHE	Contrast Limited Adaptive Histogram Equalization
FLDA	Fisher Linear Discriminant Analysis
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPGA	Field Programmable Gate Array
FPR	False Positive Rate
HE	Histogram Equalization
HOG	Histogram of Oriented Gradients
ID	Identification or Identity
LBP	Local Binary Pattern
LBPH	Local Binary Pattern Histogram
LDA	Linear Discriminant Analysis
LDR100	Low-Resolution Database 100
LDR200	Low-Resolution Database 200
MLBPH	Modified Local Binary Pattern Histogram

PCA	Principal Component Analysis
px	Pixels
Th	Threshold
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate

ABSTRACT

Nowadays, face recognition plays a vital role in various security, access control, and surveillance systems. In computer vision, automatic face recognition is a challenging job in the last couple of decades. Changes in illumination, image resolutions, the variation of light and posture of the face are still significant problems in face recognition. This research presents an improved real-time face recognition system at low-resolution of 15 pixels (px) with the pose, emotion, and resolution variations. We designed our datasets, named LRD200 and LRD100, and used them for training and classification. The Viola-Jones algorithm was used to detect the faces, and the face recognition part receives the face image from the face detection part to process it using the Local Binary Pattern Histograms (LBPH), Eigenface, and Fisherface algorithms with image preprocessing using Contrast Limited Adaptive Histogram Equalization (CLAHE) technique. The face database in this system can be updated through a standalone Android application (app) along with automatic restarting of training and recognition process with the updated database. At 15 px, real-time face recognition accuracies using LBPH, Eigenface, and Fisherface algorithms along with CLAHE were 78.40%, 72.25%, and 81.40% respectively. At 45 px the accuracies were 98.05%, 90.11%, and 93.92% respectively.

Using the combination of the LBPH and the Fisherface algorithm along with CLAHE method, an optimum of 96.55% face recognition accuracy was achieved with 50 images per person in the database. The face recognition accuracy decreased with the increase in the number of subjects in the database. The accuracy was 97.24% at 45 px with

5 persons in the database. With 15 persons in the database, the recognition rate was 91.19% using the combined algorithm and CLAHE.

This face recognition system can be employed for law enforcement purposes where the surveillance cameras capture low-resolution images because of the distance of the person from the camera. It can also be used as a surveillance system in crowded places such as airports or bus stations to reduce the risk of possible criminal threats.

1. INTRODUCTION

The process of detecting and locating a face from a single or series of images and identifying the face is known as face recognition. It is a biometric artificial intelligence-based technology which can uniquely identify a person from digital images, real-time videos as well as pre-recorded video clips by analyzing patterns of the person's facial shape and textures. As a human, we are very good at detecting and recognizing faces; however, it's a difficult job for computers to detect and recognize faces. Face recognition has diversified applications including video surveillance systems, security, access control, law enforcement, general identity verification [1], gender recognition [2] or missing person identification.

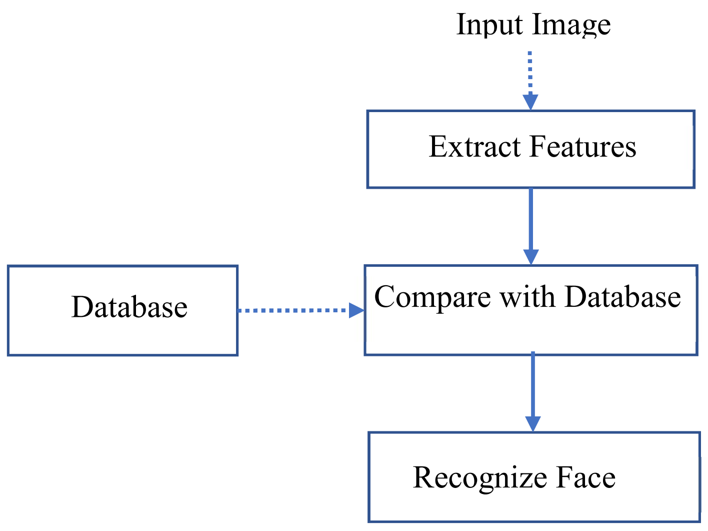


Figure 1: Basic Face Recognition Framework

Regardless of the differences (algorithms, based-model, input form, feature extraction processes, etc.), every face recognition system is based on the same basic framework. The very basic outline of a typical face recognition system can be observed in

Figure 1. The facial features are extracted from an input image to be recognized. Afterward, it is compared with a prebuilt database. If matching of the input image is found in the database, then it is recognized as known. Otherwise, the input image is marked as unknown. In extracting features of the faces, the procedure of the extraction needs to be decided carefully. The accuracy of the face recognition system is dependent upon how unique the features are for every different face. For example, many faces may have the same size, and hence the size of the face cannot be a suitable feature to extract. The extracted features should be tolerant of various environmental conditions, illumination, pose and expression. The method of feature extraction should be such that it improves the performance and efficiency of the recognition system with reduced complexity.

After deciding the method of feature extraction, a face database of different persons, those who will be identified through the system, is created. For a system of larger database size, feature extraction as well as training is time-consuming and requires a lot more processing power.

Next step is to compute the features of the incoming face and compare it with every face image of the database. Extracted features are represented by some values. Thus, while comparing an incoming face with the database of face images, the calculated distances of that face with all the database faces are utilized. The lowest distance, which is also known as the best match, is taken into consideration for recognition.

The input and output of the recognition system also vary with the purpose of the recognition system. The input image can be taken from continuous video frames, cameras or can be imported from a more extensive database. The output can be a name, number, picture or a signal depending upon the requirement.

In implementing the face recognition applications, face detection comes before recognition. Face recognition can be classified into two categories, namely, face verification and face identification. Face verification refers to the process which can detect whether a pair of pictures belongs to the same individual or not. On the other hand, face identification refers to the labeling target face with respect to a training set or database. Face recognition can be categorized into three types namely- i) holistic matching or appearance-based method, ii) structural or feature-based method, and iii) hybrid method [3]. In holistic matching approach (e.g., Eigenfaces, Principal Component Analysis (PCA) [4], [5] or Linear Discriminant Analysis (LDA) [6]), the global information of the face is considered for face recognition. On the other hand, in structural or feature-based methods [7], salient features like nose, mouth, eyes and their locations as well as native statistics are considered for recognition. Hybrid methods utilize a combination of holistic and structural methods.

When using a face recognition system, an input face image is compared to those face images that have been recorded previously and kept in the training database set of the system. In appearance-based face recognition, if the input images, taken for training and recognizing, are of higher dimensions then the computation becomes more complex and time-consuming and requires higher computing power. Therefore, dimensionality reduction is performed to obtain a small vector representation of the face(s). For low-resolution images, the size of the images becomes smaller, and the vector representation of the face also gets smaller. Moreover, in current applications of face recognition like video surveillance systems where the subject images are sometimes away from the cameras and are captured under a low light condition, the image quality degrades. A low-resolution image is a notable impediment of effective face recognition.

Over the past few decades, face recognition has been an active research field in computer vision. Nevertheless, there are some challenging problems such as variation of illumination [8], different pose, the direction of light source, environment, and changes in expression in face recognition system. There are different methods adopted by researchers to eliminate the illumination variation problem and low-resolution image recognition.

This research has presented a real-time low-resolution face recognition system using LBPH algorithm, PCA based Eigenface algorithm and Fisherface algorithm and has demonstrated an improved face recognition accuracy, even with various attitude deflection, with a reduced number of face images in the database. In this study, two sets of image databases have been created for training and recognition. One database consists of 200 images per person and another consists of 100 images per person. The face database can be updated with the help of an Android app. Using the app, the training and recognition process can automatically be restarted with the updated database.

2. LITERATURE REVIEW

Over the past few years, many researchers have developed various kinds of face recognition algorithms including PCA, LDA, Local Binary Pattern Histograms (LBPH), Histogram of Oriented Gradients (HOG) [9], and Sparse Coding [10] algorithm. The Viola-Jones algorithm [11] is one of the most popular face detection algorithms, and it is the basis of many other face detection techniques. In this algorithm, Haar-like rectangular features are used for detecting a face. Chakrasali and Kuthale [12] used Haar features and AdaBoost algorithm for face detection. They have used an image size of 320x240 pixels and achieved comparatively inferior performances with respect to the software-based system executed on Central Processing Units (CPU).

In the face recognition process, face detection plays a very important role. In some recognition systems, holistic face recognition processes have been used. A Principle Component Neural Network (PCNN) based face recognition system has been implemented by Mohan *et all.* [13]. The system is capable of recognizing up to 1400 faces in an image frame and is suitable for access control and video surveillance. However, the presented system showed a real-time image of low-resolution of 32x32 pixels only. The face recognition system proposed by Jammoussi *et al.* [14] supports a low-resolution image of 60x60 pixels, but a real-time application is not specified there. Endluri *et al.* [15] reported a real-time embedded platform of face recognition system based on the PCA method, which supports an image resolution of 320x340 pixels. However, the system cannot recognize more than two faces at a time. Recently, Schaffer et al. presented [16] a face recognition system which utilized a software-based (MATLAB) Viola-Jones face detection algorithm and FPGA based PCA algorithm for the face recognition part. They have reported a real-time face recognition at an accuracy of about 95%, which can process 13026 faces per second. In this system, 20 face images from each of 153 people have been considered for the recognition database. Zhao and Wei [17] presented a Modified LBPH (MLBPH) based real-time face recognition system with various facial deflections and attitude. In the study (see Table 1), they achieved a recognition accuracy ranging from 48-55% for 30^0 angular positions on either side of the face compared to the camera position.

Table 1: Previous Works on Face Recognition

	Conde et al. (2003)	**Zhao and Wei** [17] **(2017)**	**Ahmed** ***et al.*** [10] **(2018)**
Method	PCA+SVM	MLBPH	LBPH
Real-time	No	Yes	Yes
Number of Image per person	12	Not mentioned	500
Minimum image pixel	17x18	Not mentioned	35px
Recognition at minimum pixel	99.52% (True Positive)	Not mentioned	90%
Database	Own	FERRET & Own	LR500
Number of Subjects	29	Not mentioned	5
Recognition with facial deflection	-	(48-55)%	-

Ahmed et al. [10] used LBPH architecture for face recognition at a low-resolution of 35px. Using 500 images per person in the database, they have stated a recognition efficiency of

94% at 45px and 90% at 35px of the input image. PCA vs. low-resolution recognition system has been shown in [18] where the image resolution was 17x18 pixels, but a real-time recognition was not presented.

2.1 Face Detection

This section presents an overview of a face detection technique which employs the Viola-Jones [7] algorithm. This algorithm uses Haar-like rectangular features to construct the classifiers as shown in Figure 2.

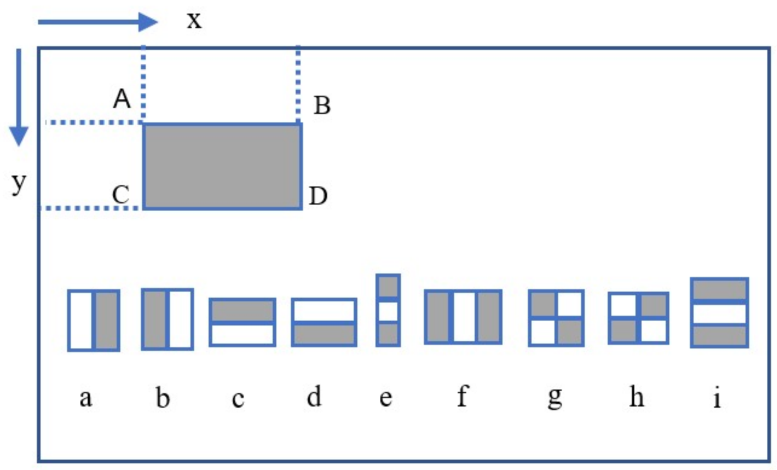


Figure 2: Formation of the integral image (ABCD) and Haar-like features (a-i)

For example, Haar-like rectangle *c* in Figure 2 is used to detect the eyes of a human face (Figure 3) because the area covered by the eyes are darker than the area just above the cheeks. Haar feature *f* in Figure 2 can be used to detect the nose feature because the junction area of the nose is brighter with respect to its two chick sides (Figure 3).

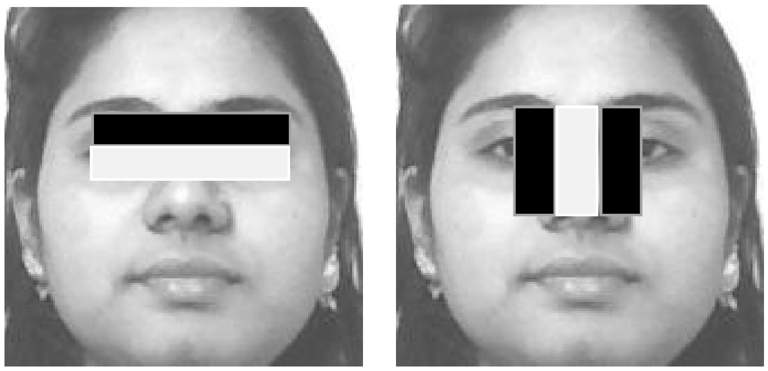
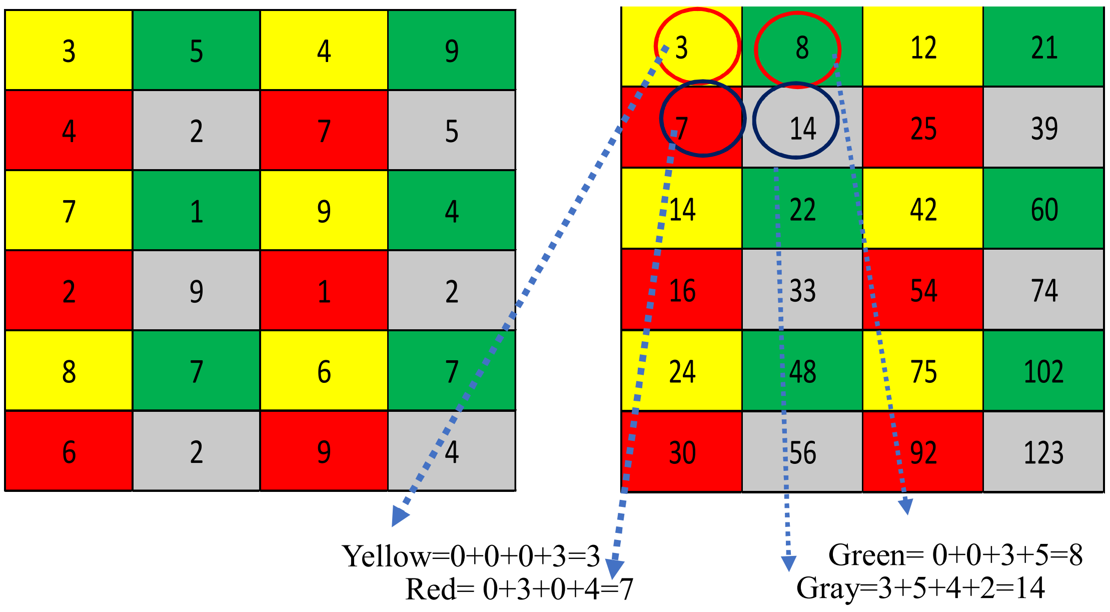


Figure 3: Relevant Haar Feature for face detection

A crucial part of the Viola-Jones algorithm is to compute the rectangular features very quickly to form an integral image. The integral image is constructed in such a way so that at any location (x, y) of the integral image, it contains the sum of all the pixels to the left side and above that point in the main image. The formation of the integral image from the image table as shown in Figure 4.



3	5	4	9
4	2	7	5
7	1	9	4
2	9	1	2
8	7	6	7
6	2	9	4

3	8	12	21
7	14	25	39
14	22	42	60
16	33	54	74
24	48	75	102
30	56	92	123

(a) image table (b) Integral Image

Figure 4: Formation of the integral image (b) from the image table (a)

A cascade classifier is used in this algorithm, which is constructed as a sequence of stages. At each stage, a set of selected rectangular features are used to slide over a sub-window (see Figure 5) to check whether there is a face or not. Using a threshold check, a sub-window region is either rejected as a face candidate or is pushed to the next state for further processing.

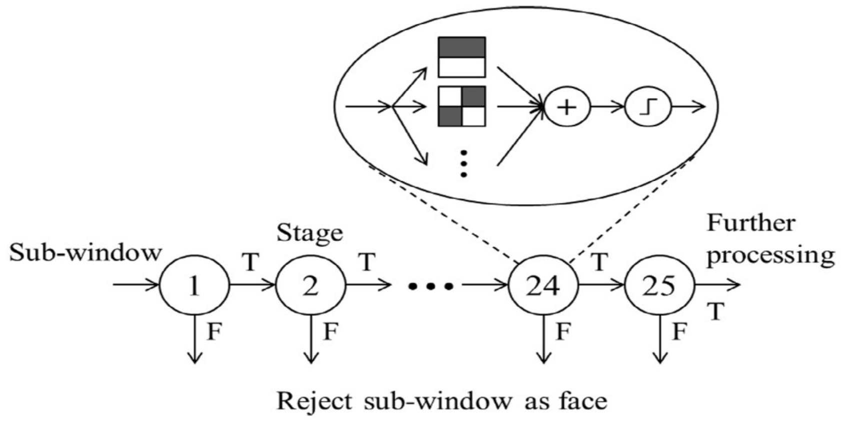


Figure 5: A cascade classifier of 25 stages and the decision tree (T=Ture, F=False) [18]

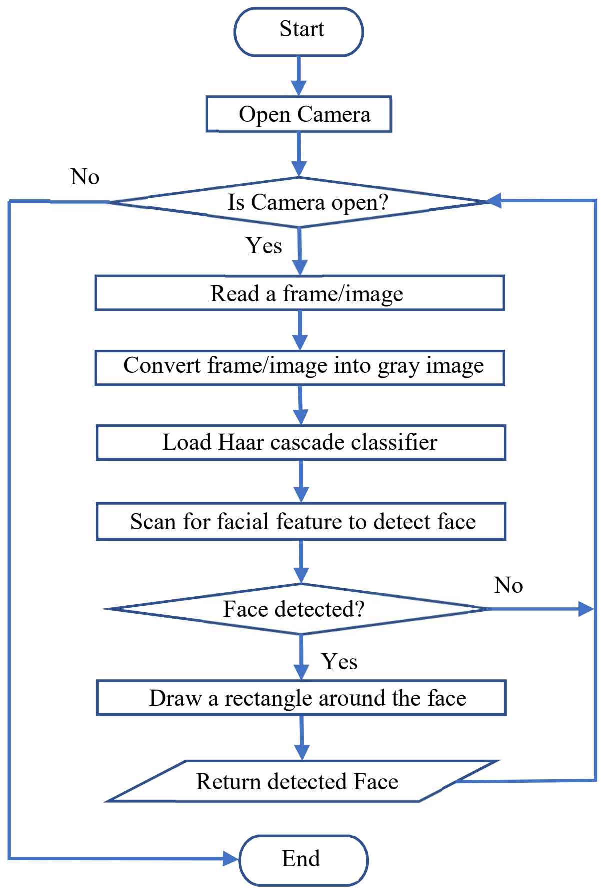

Start
Open Camera
No
Is Camera open?
Yes
Read a frame/image
Convert frame/image into gray image
Load Haar cascade classifier
Scan for facial feature to detect face
No
Face detected?
Yes
Draw a rectangle around the face
Return detected Face
End

Figure 6: The Flowchart of a Real-time Face Detection System

A pyramid of scaled images is used to detect faces of different sizes. The image pyramid consists of the same set of rectangular features but different sizes to slide over the initial image until all faces are found. Finally, all the faces are marked with colored rectangles in the original test image. The flowchart of a real-time face detection technique is depicted in Figure 6. If the real-time face images are captured using a camera, the system checks whether the camera is opened. Being confirmed that the camera is opened, an image frame is read to scan facial features in it. In the image frame, if a face is detected, the region is marked with a rectangle.

2.2 Face Recognition Algorithms

2.2.1 Local Binary Pattern Histograms

Local Binary Pattern (LBP), a powerful feature for texture classification in computer vision, is a simple yet very efficient operator to describe the contrast information of a pixel with respect to its neighboring pixels. By thresholding the neighboring pixels, LBP labels the pixels of an image and considers the result as a binary number. LBP, when combined with Histograms of Oriented Gradients (HOG) descriptor, considerably improves performance on some datasets. LBP combined with HOG descriptor can represent a facial image as a simple data vector, and it can be used for face recognition purpose.

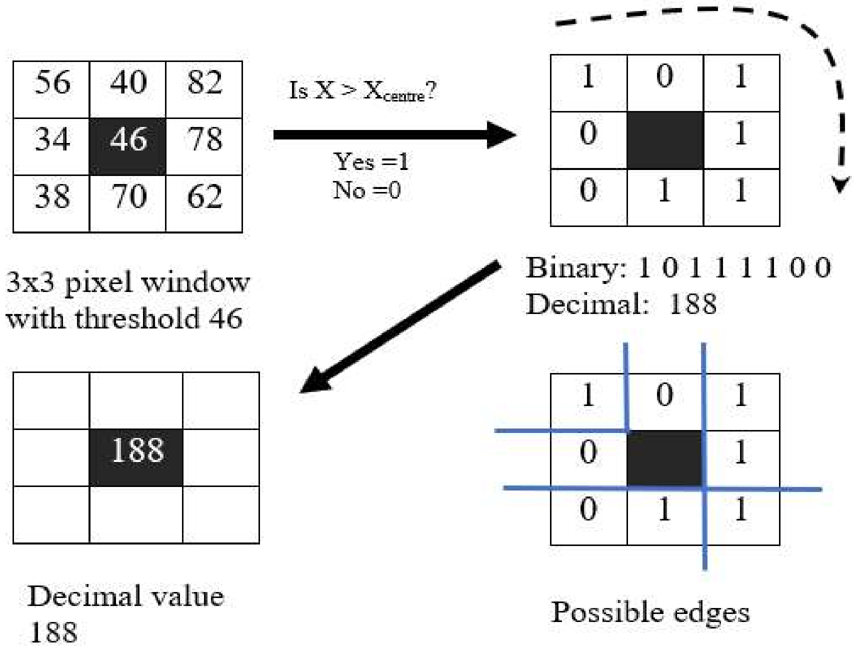


Figure 7: A 3x3 LBP operator

In Local Binary Pattern Histograms (LBPH) algorithm, to recognize a face, the face image is converted into a grayscale image. The grayscale image is then divided into 3x3 window cells to extract features. The center pixel of the cell is compared with each of the surrounding 8 pixels either in a clockwise or a counter-clockwise direction. If any surrounding pixel value is greater than the center pixel value, then it is replaced with a 1 otherwise it is replaced with a 0. Thus, in the resultant 3x3 window, if counted in a clockwise manner, a binary number of 8 bits except the center value is obtained. The center pixel in the original cell is then replaced with the decimal equivalent of the binary number. This value is used to reflect the texture feature of that region. A 3x3 LBP operator and its working procedure are shown in Figure 7.

Let g_c and g_0, g_1, g_2, …. g_{p-1} denote the values of the center pixel and neighbor pixels respectively; then, the 8-bit LBP code with respect to the center pixel at position (x, y) can be obtained using Equation (1).

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} s(g_c - g_p)2^p \tag{1}$$

The threshold function s(z) can be given by Equation (2).

$$s(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{2}$$

In the LBPH algorithm, the histogram which is used as a texture descriptor is a collection of the LBP codes of all the pixels for an input image, i.e.,

$$LBPH(i) = \sum_{x_c, y_c} \delta\{i, LBP(x_c, y_c)\}, \qquad i = 0, 1, \ldots., 2^{p-1} \tag{3}$$

where $\delta(.)$ Is known as Kroneck product function.

The LBPH method allows us to make the LBP operator of different radius and neighborhoods, which are known as circular LBP operators. An example of an LBP operator with various number neighbors and radii is given in Figure 8, where P denotes the number of neighboring pixels and R denotes the radius of the circular LBP operator.

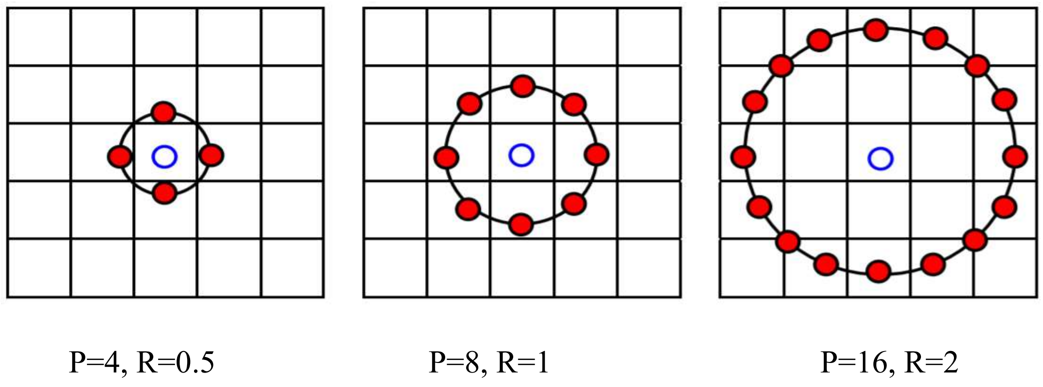


Figure 8: Circular neighborhoods of the center pixel with differ neighbor pixels

The whole gray face image is subdivided into several sub-regions in the LBPH algorithm, and then, the LBP feature vectors of each sub-region are extracted. After that, a histogram of each sub-region is calculated from the LBP feature vectors. The formation of the final histogram using the LBPH method is presented in Figure 9.

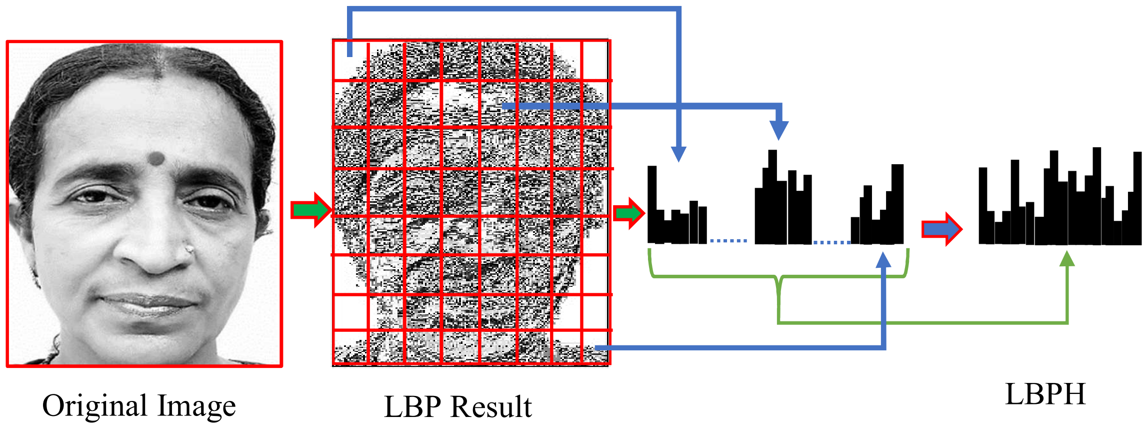


Figure 9: The formation of the final histogram using LBPH

All the histograms of the sub-regions are then concatenated to get the bigger size histogram of the full image which represents the main characteristic of the image.

2.2.2 Eigenface

This section presents an overview of the Eigenface [19] approach of face recognition. The Eigenface method is also known as the Principal Component Analysis (PCA) approach. The PCA algorithm was first introduced by Turk and Pentland in 1991. Less optimal separation in classes is the main weakness of this algorithm.

The PCA algorithm has three parts: i) Creating a Database image; ii) Calculating Eigen Faces; and iii) Recognition of a new face. All the images in the database must be of the same size because of the computational purpose. A face image *I* is of dimension *Y*Y* contains grayscale 0-255 values. At first, a column vector is created from the face images to calculate Eigenfaces, and then the mean face (Ψ) is calculated using Equation (4).

$$\Psi = \frac{1}{N}\sum_{n=1}^{N} I_n \tag{4}$$

The differences (Φ_i) of the database images (I_i) from the mean image (Ψ) is calculated using Equation (5).

$$\Phi_i = I_i - \Psi \tag{5}$$

N numbers of orthogonal vectors (v_k) can be found using the differences. The kth eigenvector can be found by using Equation (6).

$$\lambda_k = \frac{1}{N}\sum_{n=1}^{N}(v_k^T \Phi_n)^2 \tag{6}$$

Here, $v_l^k v_k = 1, if\ l = k$

= 0 otherwise.

λ_k is the kth eigenvalue of the covariance matrix C.

$$C = \frac{1}{N}\sum_{n=1}^{N} \Phi_n \Phi_n^T = QQ^T \quad (7)$$

where the matrix $Q = [\Phi_1 \Phi_2 \ldots\ldots \Phi_N]$.

The dimension of the matrix C is $Y^2 * Y^2$ which requires a lot of resources as well as computation time. In the PCA method, to reduce computation, dimension reduction is performed using a Z matrix where $Z = Q^T Q$ of dimension $N * N$. Taking the eigenvectors *v*i, the ith corresponding eigenvalue, *u*i can be calculated using equation (8) and (9).

$$Q^T Q v_i = \mu_i v_i \quad (8)$$

$$QQ^T = \mu_i Q v_i \quad (9)$$

In Equation (9) the eigenvector of the original covariance matrix is Qv_i, and μ_i can be defined by Equation (10).

$$u_i = \sum_{k=1}^{N} v_{ik}\, \Phi_k, \qquad i = 1,2, \ldots . N \quad (10)$$

During the recognition process, the new face image (I_{new}) is transformed into eigenface components by the operation given by Equation (11).

$$w_i = u_i^T (\, I_{new} - \Psi), \;\; i = 1,2, \ldots \ldots N \quad (11)$$

These components form a weighted vector Ω.

$$\Omega = [w_1 w_2 \ldots \ldots \ldots w_N] \quad (12)$$

The weighted vector (Ω) represents the contribution of each eigenface of the input face image. It can also be considered as a point in the N-dimensional vector space. The distance of the new input image is calculated with the database image using Equation (13).

$$d_k = \left|\left|\Omega_T - \Omega_k\right|\right|, \;\; where, k = 1,2, \ldots \ldots N \quad (13)$$

Representation of several eigenfaces corresponding to a different number of eigenvalues is shown in Figure 10.

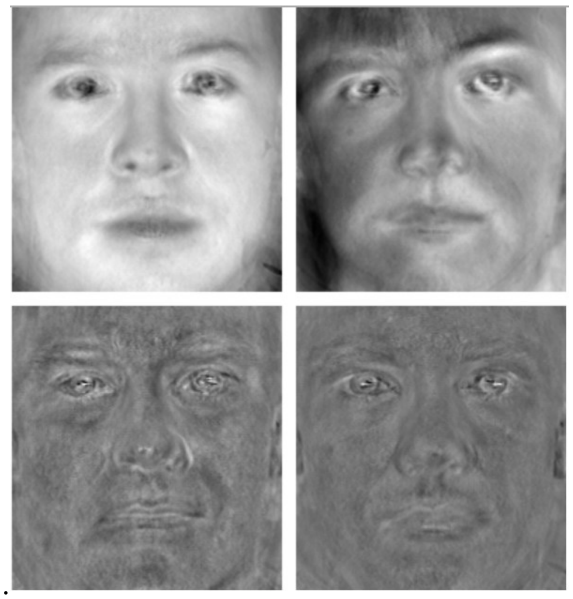


Figure 10: Sample eigenfaces corresponding to a different number of eigenvalues [20]

The input image is considered for recognition when the distance of that image compared to the database images is shortest. It is recognized if the distance is below a predefined threshold value. Otherwise, it is recognized as an unknown face.

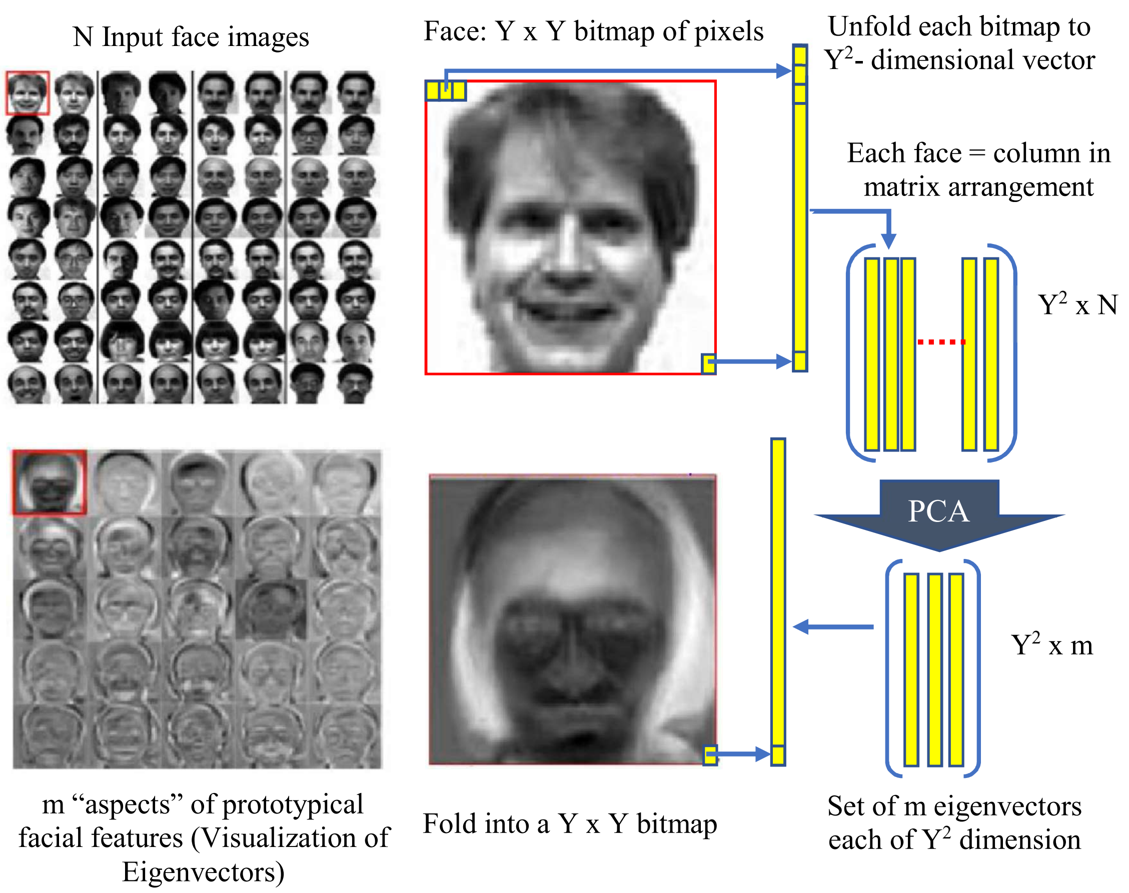

N Input face images
Face: Y x Y bitmap of pixels
Unfold each bitmap to Y²- dimensional vector
Each face = column in matrix arrangement
Y² x N
PCA
Y² x m
m "aspects" of prototypical facial features (Visualization of Eigenvectors)
Fold into a Y x Y bitmap
Set of m eigenvectors each of Y² dimension

Figure 11: Visualization of Eigenface Algorithm

The Eigenface algorithm can be visualized in Figure 11, where there are N numbers of input images. Each image in the database set is unfolded into a column matrix of dimension Y^2x1. Then, using all the database faces, the primary matrix is created having a dimension of Y^2xN. After computing the mean and the difference of each image from the mean, using the PCA technique, the dimension of the matrix is reduced to Y^2 x m where *m* represents the number of eigenvectors. Those *m* eigenvectors are *m* number of prototypical facial features. The image can be reconstructed by adding the mean face with those eigenvectors with different proportions (called weighted vectors). The formation of the main image from the eigenvectors is illustrated in Figure 12. The more numbers of eigenvectors are added with required weighted vectors, the more the image gets closer to the original image.

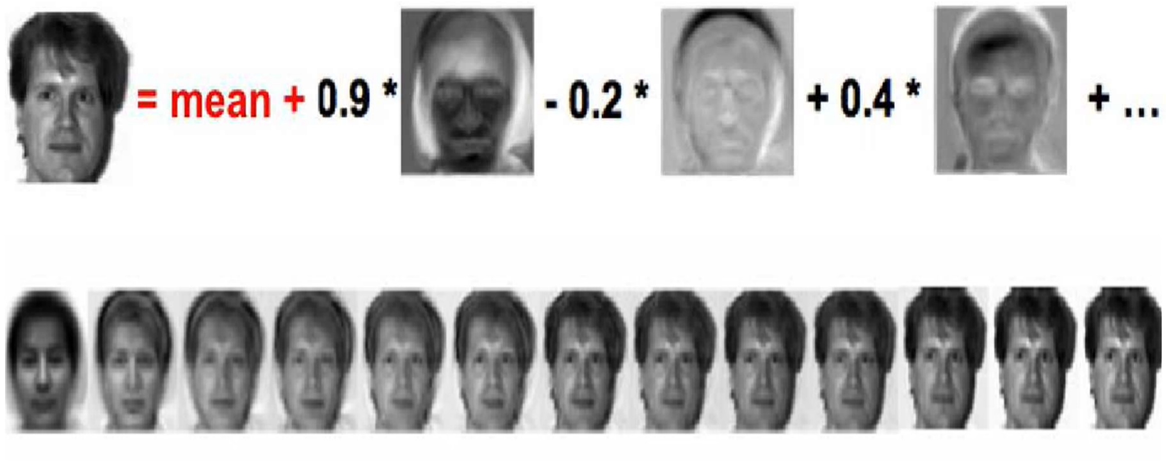


Figure 12: Constructing a face by adding eigenvectors

2.2.3 Fisherface

Fisherface [21]–[23] algorithm is another popular face recognition algorithm. It was developed in 1997. It is built upon the Eigenface and based on the Fisher Linear Discriminant Analysis (FLDA) derived from Ronald Fisher's linear discriminant analysis (LDA) technique. Although both the PCA and the LDA use linear projection for dimension reduction, the results are highly dissimilar. During dimension reduction, the PCA looks for the maximum total variation between individual sample points. On the other hand, the LDA seeks for the maximum divergence between groups. It necessitates the minimization of the variation amongst sample points within the same group.

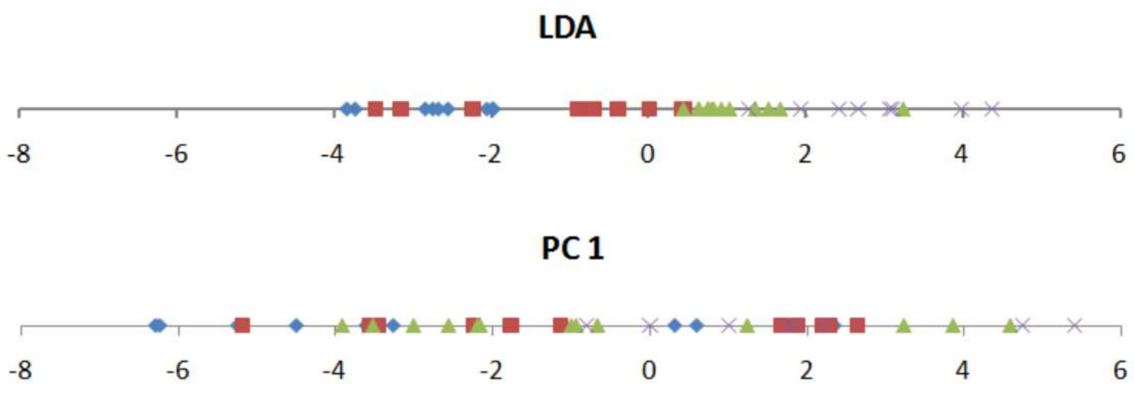


Figure 13: Classes in PCA and LDA where LDA has well-segregated classes than PCA [24].

The LDA maximizes the ratio of the between-class scatter matrix and within-class scatter matrix. For this reason, under different lighting conditions of the image, the Fisherface algorithm has limited effect on the classification process compared to the PCA. Projecting onto the first principle component in PCA and LDA can be well depicted in Figure 13. The groups in LDA are more scattered than that of PCA.

Let S_b be the between-class scatter matrix. It can be defined as

$$S_b = \sum_{i=1}^{m} N_i(\mu_i - \mu)(\mu_i - \mu)^T \tag{14}$$

Let S_w be the within-class scatter matrix. It can be defined as

$$S_w = \sum_{i=1}^{m} \sum_{X_j \in X_i} (\boldsymbol{X}_j - \mu_i)(\boldsymbol{X}_j - \mu_i)^T \tag{15}$$

where, $\boldsymbol{\mu}_\mathbf{i}$ denotes the mean image of class X_i, and N_i denotes the number of samples in class X_i. If S_w is nonsingular, the optimal projection W_{opt} is chosen in the matrix with orthonormal columns which maximizes the ratio of the determinant of the between-class scatter matrix of the projected samples to the determinant of the within-class scatter matrix of the projected samples [25], i.e.,

$$W_{opt} = \arg max_w \frac{|W^T S_b W|}{|W^T S_w W|} \tag{16}$$

$$= [w_1 w_2 w_3 \ \dots \ w_n] \tag{17}$$

Where $\{w_i | i = 1, 2, 3, \dots, n\}$ is the set of generalized eigenvectors of S_b and S_w corresponding to the n largest generalized eigenvalues $\{\lambda_i | i = 1,2,3,\dots, n\}$, i.e.,

$$S_b w_i = \lambda_i S_w w_i,\ i = 1, 2, 3, .., n \tag{18}$$

Note that, there are (m-1) nonzero generalized eigenvalues. The upper bound on n is (m-1), where m denotes the number of classes.

2.3 Image Enhancement Techniques

Image enhancement, especially contrast enhancement, is an important area in digital image processing. It is performed to increase the visual perception of humans and computers (Computer Vision). In image processing applications, the contrast enhancement technique plays a vital role [26]. It converts an image of poor contrast into an image which is more enhanced in contrast. Out of many contrast enhancement techniques, Histogram Equalization, Adaptive Histogram Equalization, and Contrast Limited Adaptive Histogram Equalization are discussed below.

2.3.1 Histogram Equalization

Histogram Equalization (HE) is a technique of contrast enhancement where the histogram of an image is used to enhance the digital image, and it improves the contrast globally. HE adjusts the intensity of the image which is done by spreading out the most frequent intensity values over the entire image [27]. This method is efficient for image enhancement where both the background and foreground of the image are either dark or bright [28].

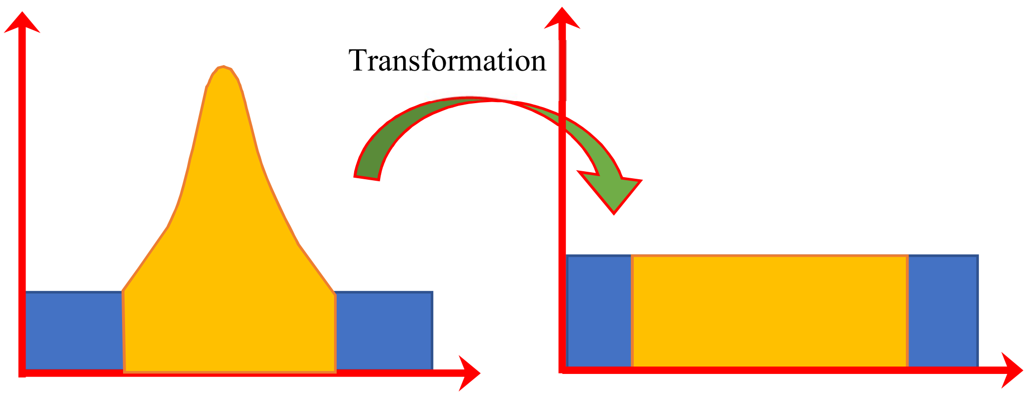


Figure 14: Histograms of an image before and after Histogram Equalization is applied [28]

The basic concept of HE can be visualized in Figure 14 and Figure 15. The intensities of the image can be adjusted by distributing them on the histogram. HE is done by spreading out the most frequent intensity values of the image, and it results in the lower contrast to gain a higher contrast.

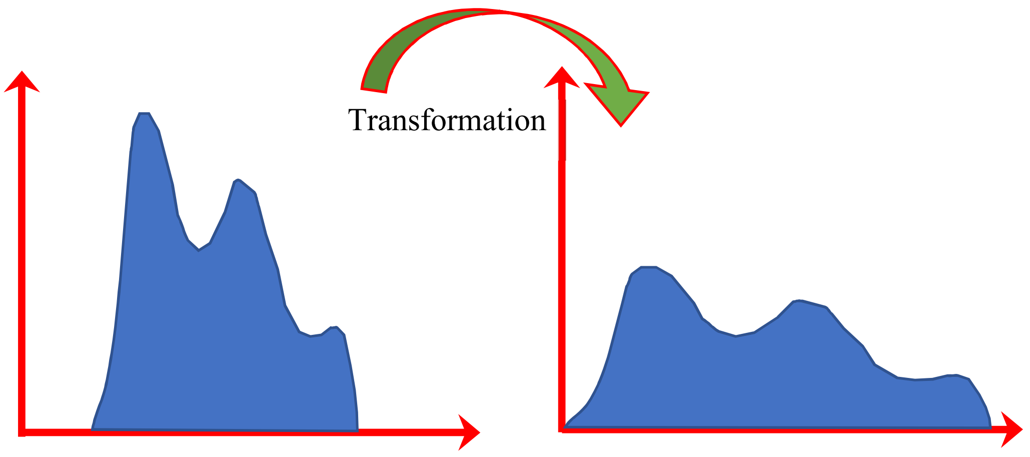


Figure 15: Histograms of an image before and after Histogram Equalization is applied.

Let I be an image of dimension m_r x m_c and L be the number of possible intensity levels in an image. L ranges from 0 to 255 which gives a total of 256 intensity levels. If p denotes the normalized histogram of the image I, then the probability assigned to each intensity level can be given by,

$$p_r(r_i) = \frac{n_i}{N} \quad 0 \leq n_i \leq 1, i = 0, 1, 2, \dots .(L-1) \tag{19}$$

where r_i is the normalized intensity value, n_i is the number of pixels with the intensity level r_i. N denotes the total number of pixels in the image.

The Histogram Equalized image can be then defined by,

$$h_{i,j} = floor((L-1)\sum_{r=0}^{I_{i,j}} p_r) \tag{20}$$

where $floor(\,)$ operator rounds down the value to the nearest integer.

One disadvantage of this method is that it may increase the contrast of background noise present in the image while decreasing the usable pixels. HE often results in an unrealistic effect in the image.

2.3.2 Adaptive Histogram Equalization

Adaptive Histogram Equalization (AHE) is a modification of the HE technique. In AHE, the whole image is segmented in some distinct sections, and Histogram Equalization is applied to each of the regions. The contrast is adjusted according to their neighbor pixels. This process results in local contrast enhancement of the image. One disadvantage of this method is that it overamplifies the noise in relatively homogeneous regions of an image [28].

2.3.3 Contrast Limited Adaptive Histogram Equalization

When AHE is performed on a region of an image where the range of intensity is relatively smaller, the noise in that region gets more enhanced. The problem of noise amplification associated with the AHE method can be prevented using Contrast Limited Adaptive Histogram Equalization (CLAHE). The CLAHE is a modification of Adaptive Histogram Equalization. At an intensity level, the contrast enhancement is directly proportional to the slope of the Cumulative Distribution Function (CDF). So, the contrast enhancement can be restricted by limiting the slope of the CDF [29]. At any bin location, the slope of CDF can be determined by the height of the histogram at that location. Hence, to limit the amount of contrast enhancement, the height of the histogram is clipped at a predefined value before computing the CDF which, in turn, limits the slope of the CDF. In the CLAHE method, the enhancement function is applied to all the neighborhood pixels [30].

If any part of the histogram exceeds the clip limit, that part can be redistributed equally among the histogram bins rather than discarding that part (see Figure 16). This redistribution pushes some of the bins above the clip limit which results in an effective clip limit that is greater than the prescribed limit. If this becomes undesirable, then the redistribution process can be applied repeatedly until the clipped part becomes negligible. This redistribution sometimes becomes advantageous.

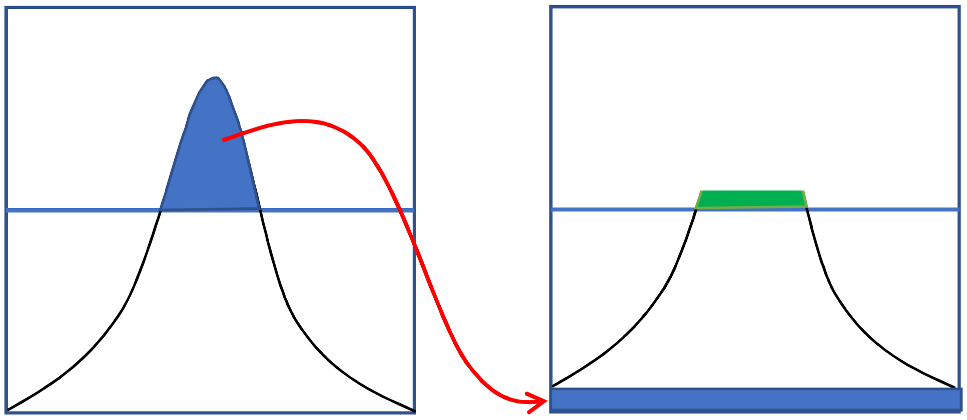


Figure 16: Redistribution of the clipped part among all the histogram bins.

If, N_{Xp} is the number of pixels in X direction of the sub-region and N_{Yp} is the number of pixels in the Y direction of the sub-region, then the average number of pixels can be given by,

$$N_{avg} = \frac{N_{Xp} N_{Yp}}{N_g}$$

The actual clip limit N_{CL} can be given by,

$$N_{CL=} N_{Clip} N_{avg}$$

Where, N_{Clip} is the clip limit.

The CLAHE can be applied to both grayscale and color images. Usually, for RGB true color image, the LAB color space is used to apply this enhancement technique.

3. PROPOSED FACE RECOGNITION SYSTEM

3.1 Overall Proposed Face Recognition System

The overall proposed face recognition system is presented in Figure 17, where the camera module attached to the laptop is used for capturing real-time video frames for recognition, and the database images are captured using an Android app.

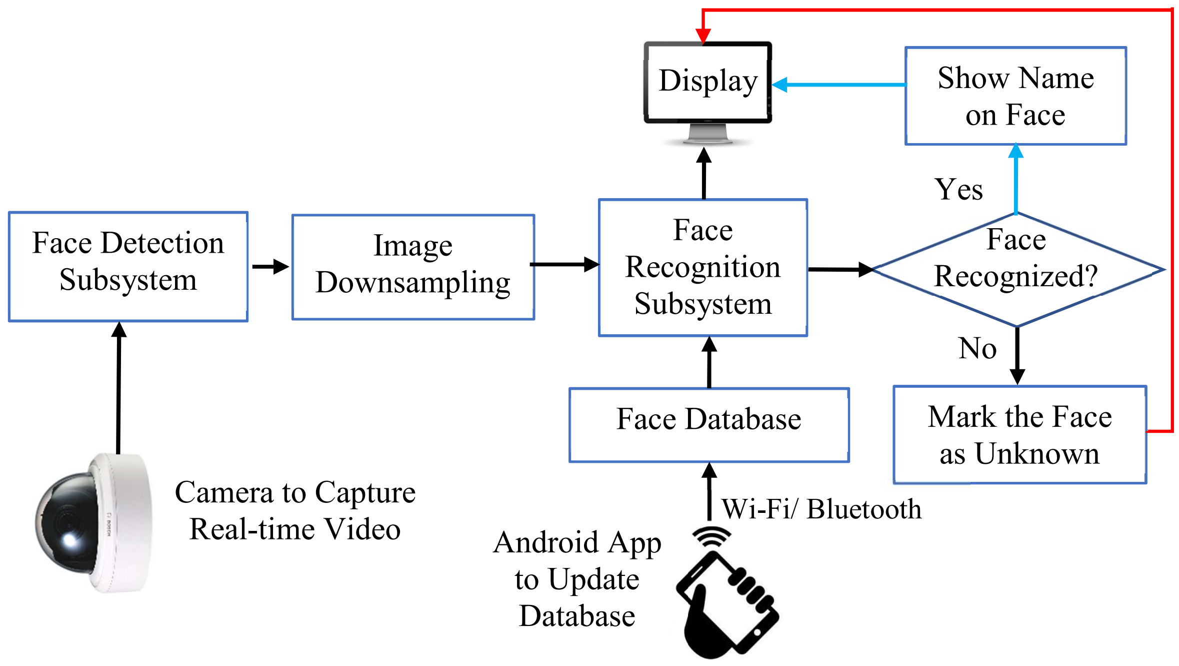


Figure 17: Block diagram of the overall proposed system

The image frames from the camera are passed through the face detection subsystem to locate the face in the image frame. Then some image preprocessing is done on each image frame before passing it through the face recognition subsystem. All the database images are captured using an Android app and then trained using each algorithm. The recognition is performed using three different algorithms: Local Binary Pattern Histograms, Eigenface, and Fisherface.

3.2 Specification of Devices and Cameras Used

The images of the databases are captured during daytime with the presence of daylight as well as fluorescent tube light. The front camera of the Android phone is of 8 MP with no flash enabled, and the back camera is of 13 MP with flash enabled. Both cameras have an image aspect ratio of 4:3 and do not have night vision feature. The specifications of the Android phone by which the database images were captured are given in Table 2.

Table 2: Specification of the camera of the Android phone.

Particulars	Specifications
Android Phone & Model Number	Oppo R7kf
Android Version	5.1.1
Front Camera	8 MP
Image Aspect Ratio (Front Camera)	4:3
Flash Status of Front Camera	No Flash Enabled
Back Camera	13 MP
Image Aspect Ratio (Front Camera)	4:3
Flash Status of Back Camera	LED Flash Enabled
Flash Use Status	No Flash Used During Capture
Night Vision Camera	Not Present
Processor	Qualcomm MSM8939 Octa Core

For real-time face recognition, the input images are captured from the continuous video frames from the webcam of a laptop. The webcam is 0.9 MP, and it does not have any flash in it. The image aspect ratio is 16:9. The complete specifications of the webcam of the laptop are given Table 3.

Table 3: Specification of the webcam of the laptop

Particulars	Specifications
Laptop Model Number	Lenovo ThinkPad Yoga 14
Webcam (Integrated)	0.9 MP
Flash Status	No Flash Enabled
Image Aspect Ratio	16:9
Night Vision Camera	Not Present
Processor	Core i5
Operating System	Windows 10

3.3 Creating the Face Database and Starting the Training Process

For this study, two sets of face databases consisting of 10 people are created. One database named LRD200 consists of 200 face images per person, has a total of 2000 face images, and the other database named LRD100 which consists of 100 face images per person, totaling 1000 face images in this database. Face database LRD100 is a subset of the LRD100 database. The images are captured in different illumination conditions and with different facial expressions and poseur to improve the recognition efficiency. A few sample images from the LRD200 database can be seen in Figure 18.

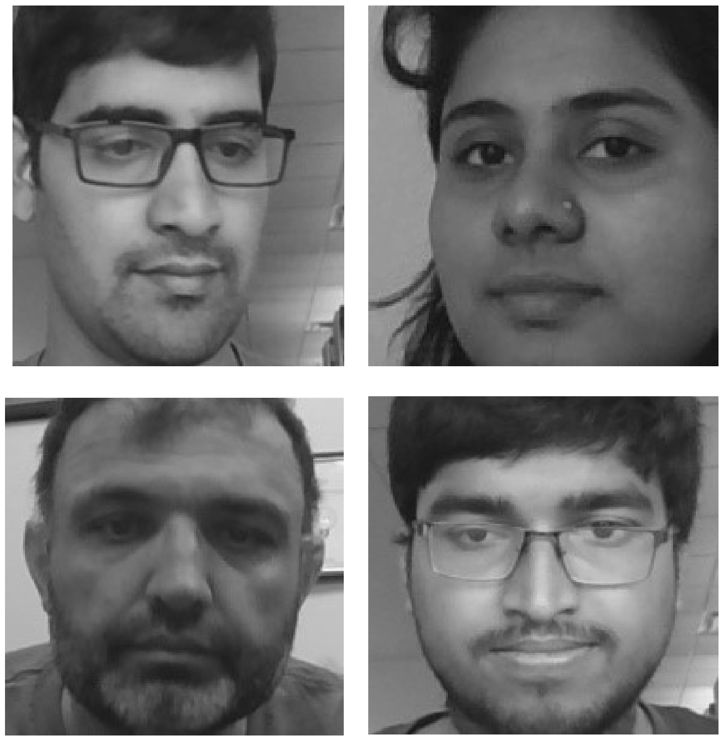


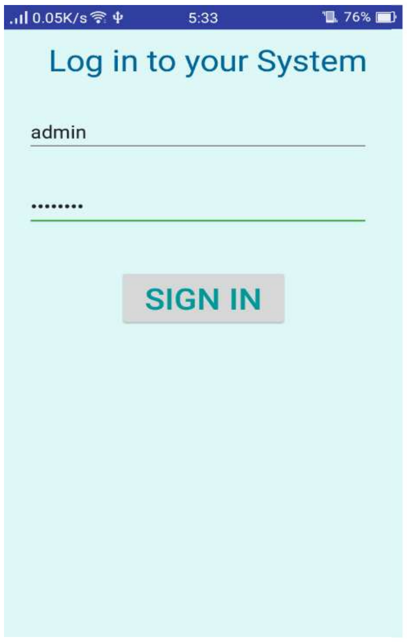
Figure 18: Sample face image of LRD200 database

An Android app named **My New Cam** is built to create the face database. The app takes pictures of the subject image. While taking pictures of the subject face, the user must put the name of the person. Entering the person's name in the name field results in the "CAPTURE IMAGE" button to be enabled. Later, this name is used as the subject name for face recognition. The subject images from the Android app are then sent to a Temporary Image Folder over Wi-Fi or Bluetooth communication system. The description of the developed android app is given in Table 4. Minimum SDK version is 18, and targeted SDK version is 28. The app has front and back camera option in it. It also has a single image gallery and a complete image gallery with zooming features. The camera app is built using the camera API 23. Java and XML are used to build the app.

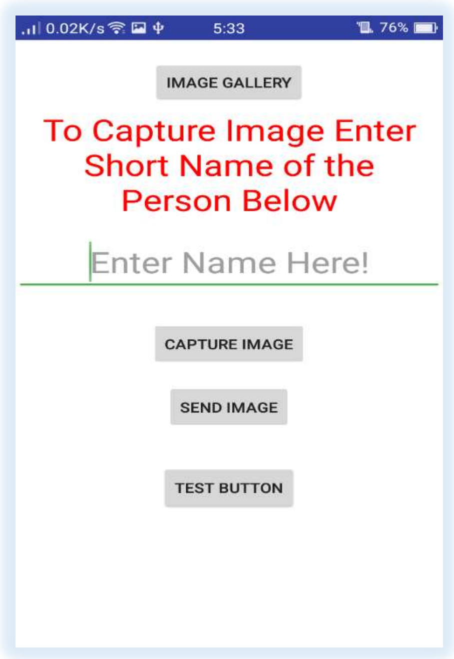
Table 4: Description of the developed Android app

Particulars	**Description**
Compile SDK Version	28
Camera API	23
Minimum SDK Version	18
Target Sdk Version	28
Programming Language	Java & xml
Development Platform	Android Studio
Features	Camera, Total Image Gallery, Single Image Gallery, Zoom view

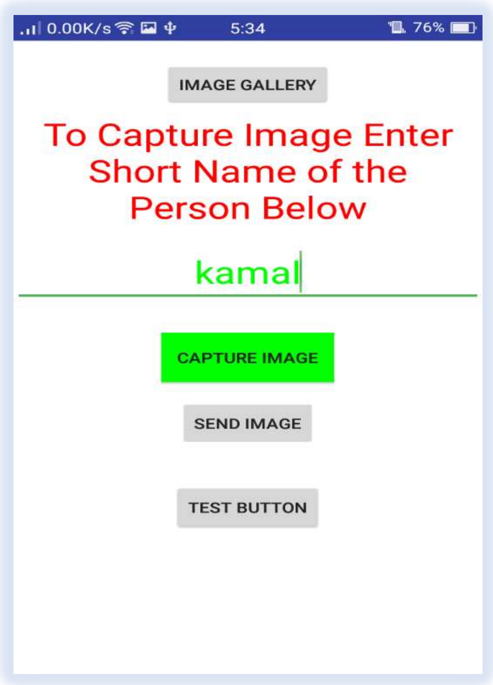
A few snapshots of the developed Android app are given in Figure 19 where the first one is showing the login screen. The user must put authenticated login information to use the app. By clicking the "CAPTURE IMAGE" button, the user can open the camera to capture face images.



(i) Login screen

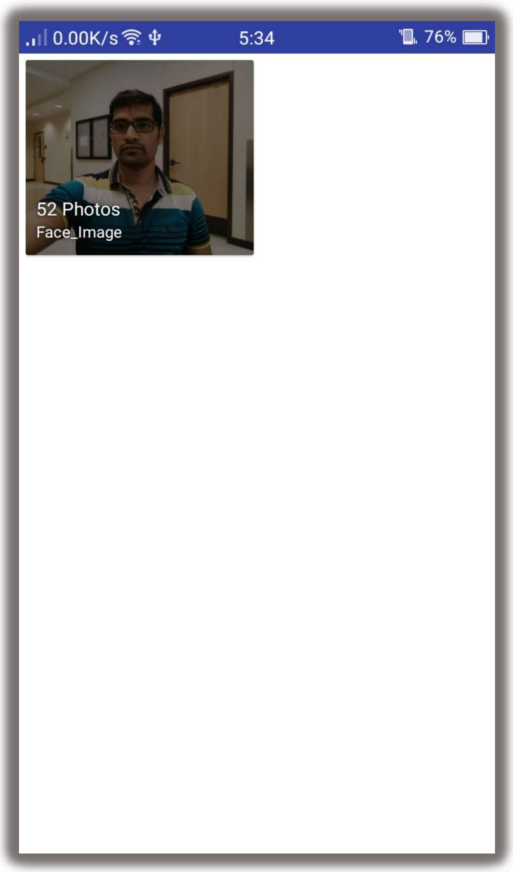


(ii) After login screen



(iii) Enabling "CAPTURE IMAGE" button by entering name

Figure 19: A few snapshots of the **My New Cam** Android app showing various screens.



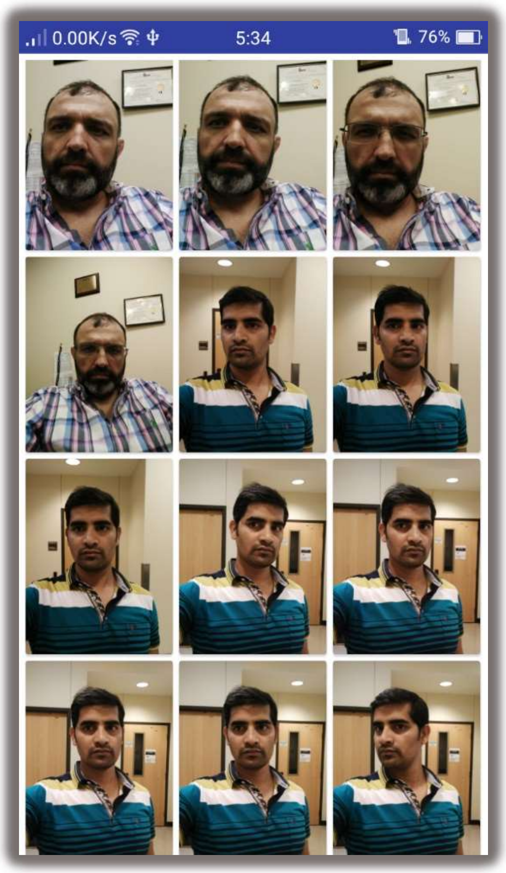


Figure 20: Pictures of the Android app showing the total number of images in the gallery (left) and a few images in the gallery (right).

The face image gallery showing the number of images and a few sample images in the Android app are shown in Figure 20.

In the system computer, a directory watcher program continuously monitors and detects any incoming face image in a temporary image folder. When it detects any incoming face image from the authorized Android device, a face detector program detects faces in the images. The faces, if found tilted, are then aligned in a vertical position so that the eyes are always in a horizontal position. To get a better accuracy of face recognition, the face images are then preprocessed using median filtering before cropping.

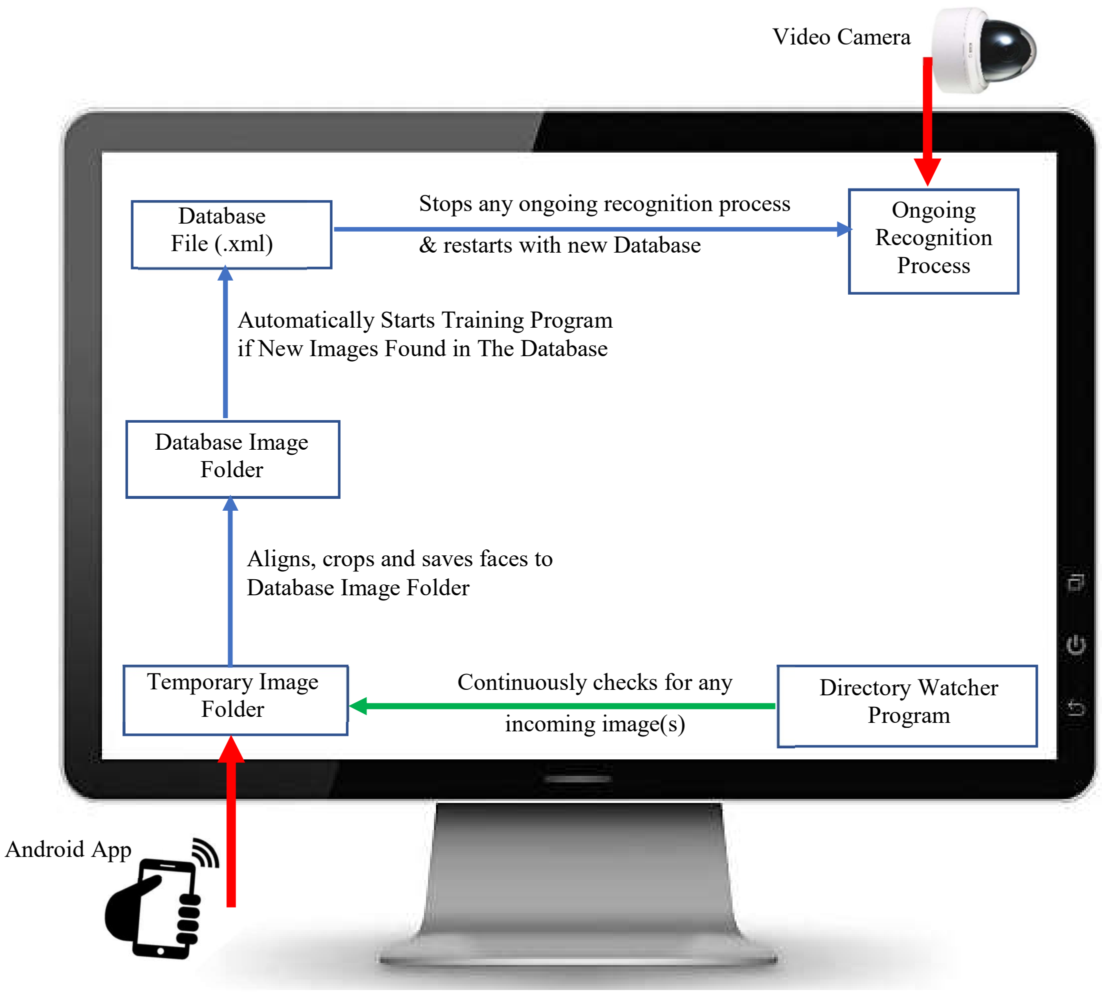

Video Camera
Database File (.xml)
Stops any ongoing recognition process & restarts with new Database
Ongoing Recognition Process
Automatically Starts Training Program if New Images Found in The Database
Database Image Folder
Aligns, crops and saves faces to Database Image Folder
Temporary Image Folder
Continuously checks for any incoming image(s)
Directory Watcher Program
Android App

Figure 21: Block diagram of the overall process of database creation and automatic training.

The overall block diagram of the process of creating a database along with automatic training and restarting the recognition process is presented in Figure 21. A median filtering process is applied to remove the noise present in the images. Afterward, the cropped face images along with the subject names are automatically saved to the "training-image" folder which contains all the database images. This process also includes the automatic starting of the training process when new face images are added to the training-image folder. After the completion of the training process, the training data are saved in a Recognizer_database.xml file which is used for recognition purpose. The training is done using the LBPH, the Eigenface and the Fisherface algorithms.

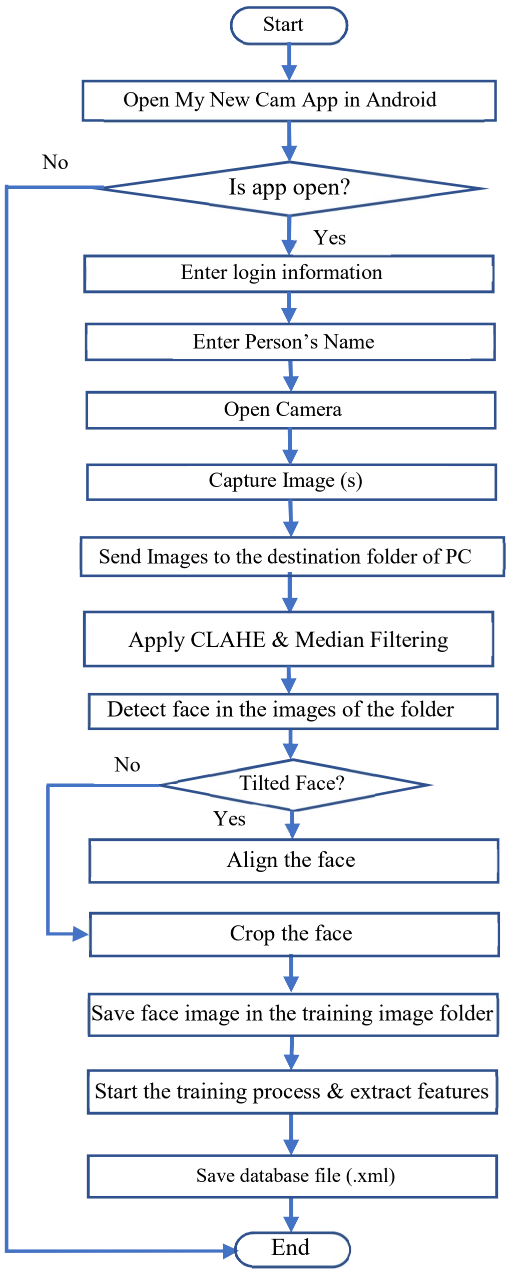

Start
Open My New Cam App in Android
Is app open?
No
Yes
Enter login information
Enter Person's Name
Open Camera
Capture Image (s)
Send Images to the destination folder of PC
Apply CLAHE & Median Filtering
Detect face in the images of the folder
Tilted Face?
No
Yes
Align the face
Crop the face
Save face image in the training image folder
Start the training process & extract features
Save database file (.xml)
End

Figure 22: Flowchart of the process of creating a database and automatic training.

New face images can always be added using the Android app, and the system automatically trains the image database and restarts the face recognizer program. OpenCV libraries and python are used for face detection and recognition. The flowchart of the entire process of creating a database and automatic restarting of the recognition program is given in Figure 22.

Besides the Android app, the system has the provision to capture database images using the laptop webcam which also requires the user to put the name of the person before capturing images so that the name is stored along with the file name of the subject image. The captured face images then are passed through the Face Detection procedure. Then, the images are aligned if found tilted. A median filtering process is applied after this step, and then the face recognition step comes into account.

3.4 Face Recognition using LBPH

We set up a real-time face recognition system where the input images are captured from the webcam video feed. Not every frame is used for recognition. Frames after every 300 milliseconds are considered for recognition. Before starting the webcam, the face database file for LBPH and NameList.txt files are loaded. Each frame read is passed through a Gaussian filtering process to reduce the noise present in the image during capture from the camera. The frame is then converted into a gray image for detecting a face. If the face image is found tilted, it is then aligned to keep both eyes in a horizontal position. The detected face is cropped and resized. The LBP feature vectors are extracted, and a histogram of the face image is obtained using the LBPH algorithm which represents the characteristic of the image.

The histogram of an input image is compared with the database histograms to recognize the image. The image is recognized as the subject image in the database having the closest histogram. The detailed flowchart for face recognition is illustrated in Figure 23.

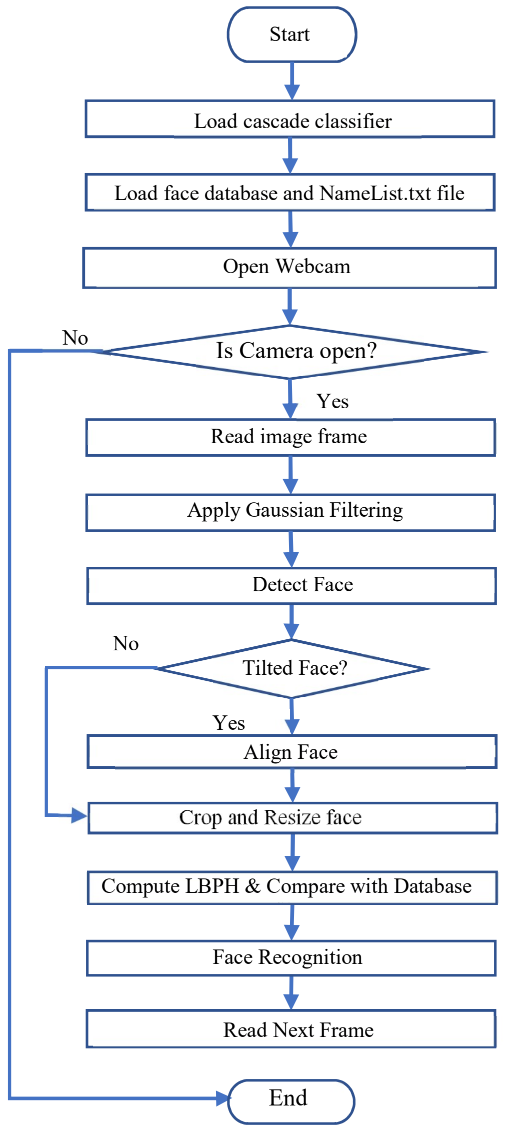


Figure 23: Flowchart for face recognition

A threshold value is set to identify an unknown person who is not included in the image database. Euclidean distance method [31] as shown in Equation (21) is used to compare histograms.

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2} \tag{21}$$

where D denotes the Euclidean distance between histograms.

3.5 Face recognition using Eigenface Algorithm

The webcam of a Lenovo Thinkpad Laptop is used to recognize faces in real-time. The input candidate images are captured from the real-time video feed of the webcam. The frames are read after every 300 milliseconds for recognition. Figure 24 shows the overall training and face recognition process using the Eigenface algorithm. The subject image frame is passed through a Gaussian Filtering process to reduce image noise present in the captured frame and to get improved recognition accuracy. Gaussian filtering [32] is done using equation (22).

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{22}$$

where x and y represent the distances from the origin in the horizontal axis and vertical axis respectively, and σ denotes the standard deviation of the Gaussian distribution. This filtering action preserves decent boundaries and edges in the images.

The frame is then passed through a face detection process where the detected face is aligned if found tilted. Face alignment is done to get better recognition result. The detected face is cropped and vectorized. Then the difference of the image with the mean database image is calculated. Also, eigenvectors for the incoming candidate image is obtained and projected to the Eigenface space.

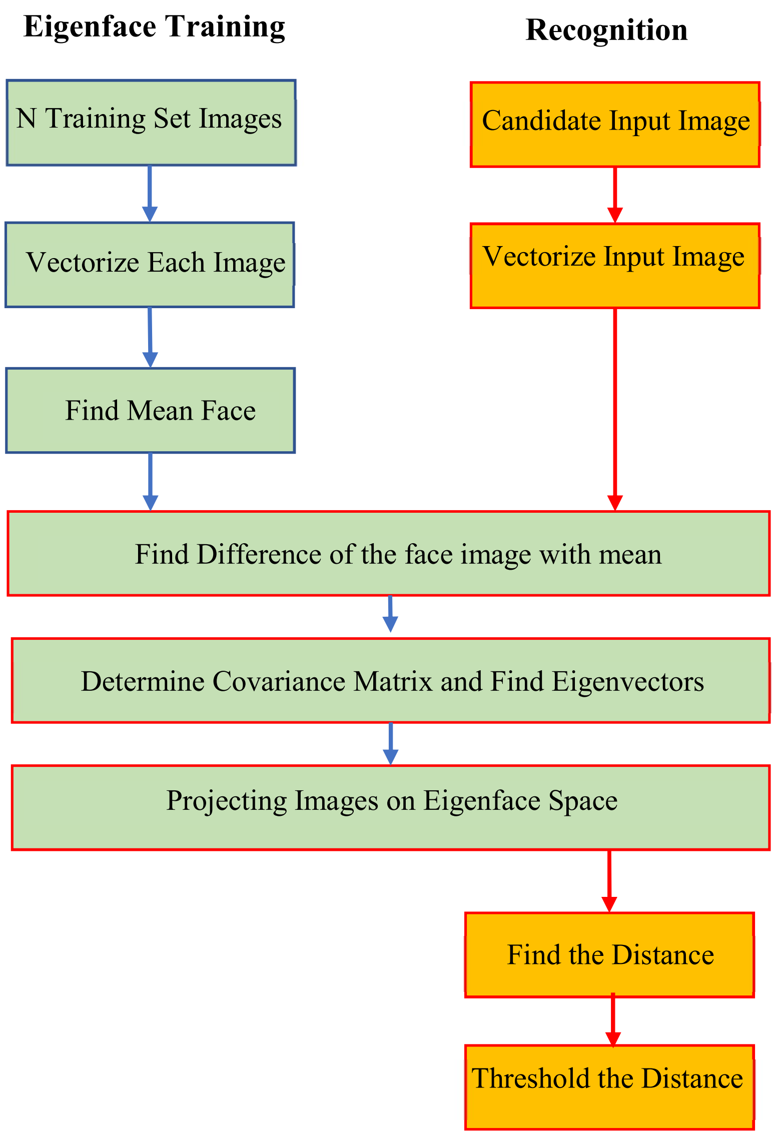


Figure 24: Schematic Diagram of Eigenface method of Face Recognition.

Then the distance of the incoming face with respect to the database image is calculated. The shortest distance found is then compared with a threshold value. If the distance is found below the threshold limit, then it is recognized, and the subject name is displayed on the computer screen. Otherwise, it is marked and displayed as an unknown face. Figure 24 shows the schematic diagram of training and recognition algorithm using the Eigenface method.

During the training period, the name part from each image file is separated to get subject names given during the image capturing time. A separate ID for each subject is assigned which corresponds to the position of the subject names in the list. Those subject names and IDs are stored in a text file to be used for recognition purpose. The overall flowchart of face recognition using Eigenface Method is presented in Figure 25. During the recognition time, the recognition system provides an ID after analyzing the input face image. Using that ID number, the system finds the corresponding subject name from the text file which was created during the training period. If it finds a subject name from the text file, the system displays the name of the person beside the face image on the screen. If the subject name corresponding ID number is not found, then the input face image is named as unknown on display.

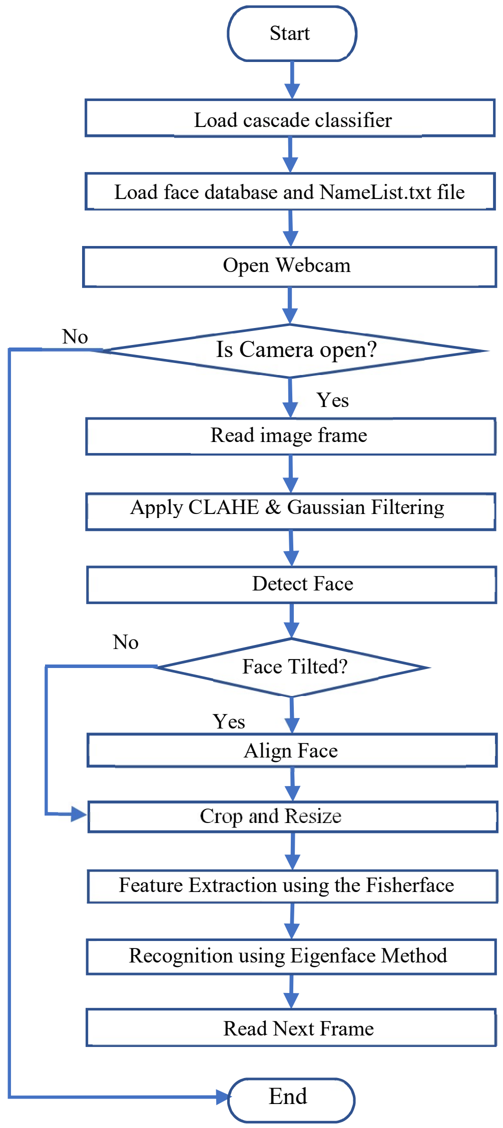


Figure 25: Flowchart for a real-time face recognition using the Eigenface method.

3.6 Face recognition Using Fisherface Algorithm

The Fisherface algorithm is another powerful algorithm which is comparatively less sensitive to light than the Eigenface algorithm. In the proposed system of face recognition, the input images are passed through the CLAHE process, and then a Gaussian filter is applied to the image to remove noise. The input images are captured from the webcam of the laptop. Frames after every 300 milliseconds are considered for recognition. If the input image is found tilted, it is aligned, and then the face is cropped for feature extraction using the Fisherface algorithm. Two sets of databases (LRD100 and LRD200) are used for training. The databases are created using the android app named My New Cam. The face recognition experiment is done with and without CLAHE. The overall flowchart of face recognition is the same as shown in Figure 25 except the feature extraction process which is done using the Fisherface algorithm.

3.7 Combined LBPH and Fisherface Algorithm

Finally, a face recognition system combining the LBPH and the Fisherface algorithm is proposed. The input image frames are preprocessed using the CLAHE method and filtered using a Gaussian filtering process. The faces are then aligned, cropped and resized for feature extraction using the LBPH and the Fisherface algorithm. The face recognition result using the LBPH method is compared to the result from the Fisherface algorithm. The flowchart of the proposed system is given in Figure 26. The final recognition result is the combined output of the two algorithms. If the outputs of both algorithms demonstrate the same result, the face is marked with the resultant subject name. Otherwise, the face is marked as unknown. To see the changes in the recognition accuracy, the number of images per person is also varied gradually from 10 to 100 with an increment of 10 images per person. This experiment is done based on the LRD100 database. At first, the best 10 images of each of the 10 subjects are chosen for training. The recognition rates are calculated using this training set. The images are chosen so that the database set contains the most possible expression and pose variations. The same procedure is repeated with 20, 30, 40, 50, 60 and 100 images of the subjects and the face recognition rates are recorded.

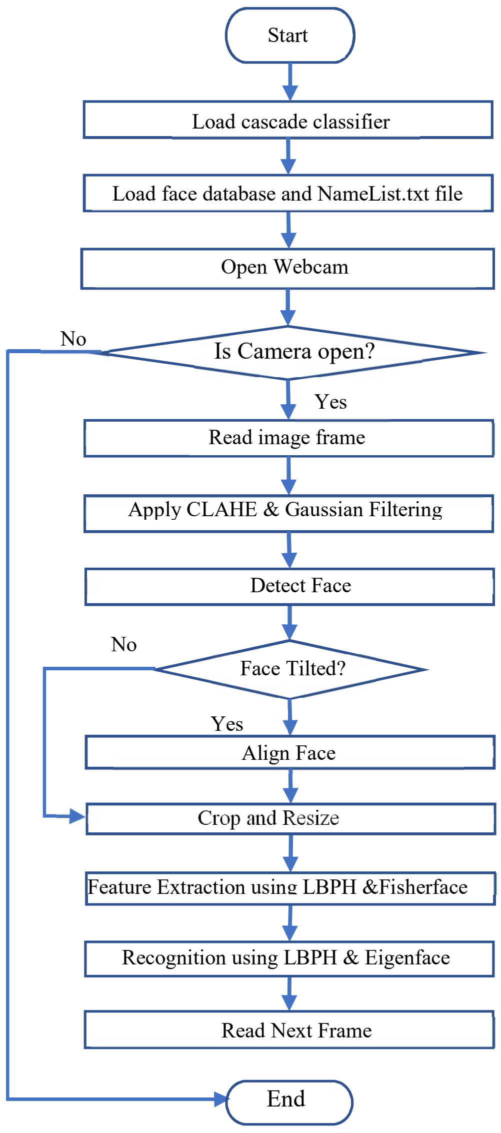
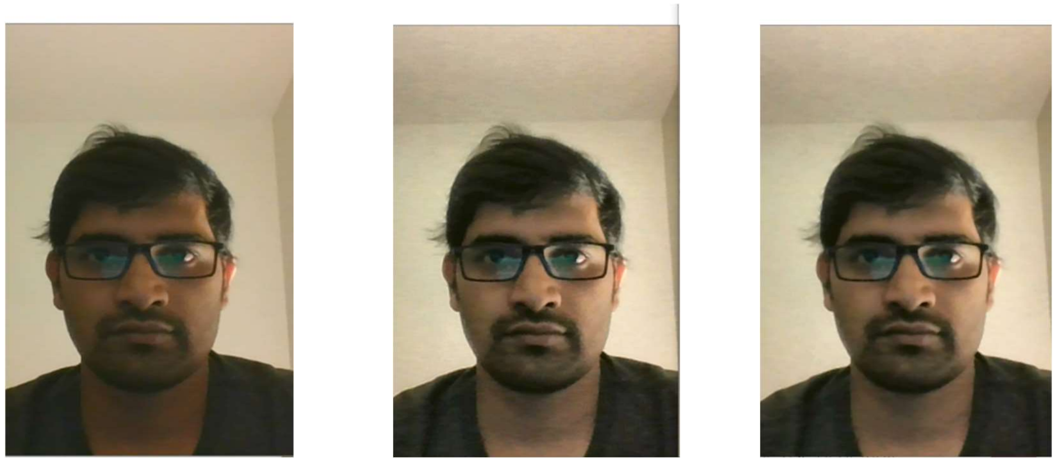

Start
Load cascade classifier
Load face database and NameList.txt file
Open Webcam
Is Camera open?
No
Yes
Read image frame
Apply CLAHE & Gaussian Filtering
Detect Face
Face Tilted?
No
Yes
Align Face
Crop and Resize
Feature Extraction using LBPH &Fisherface
Recognition using LBPH & Eigenface
Read Next Frame
End

Figure 26: Face recognition flowchart using combined LBPH & Fisherface method.

The average threshold value for the LBPH and the Fisherface algorithms change with the increase in the number of images in the database. Therefore, every time images are added to the database, the threshold values are adjusted manually.

4. EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Effect of CLAHE on Poorly Illuminated Image



(i) A frame from a video (ii) CLAHE of the frame (iii) Filtered CLAHE of the frame

Figure 27: Real-time observation of the effect of CLAHE and Filtered CLAHE from a video frame.

Use of the CLAHE method can enhance a poorly illuminated image. The effect of CLAHE is clearly shown in Figure 27. The leftmost image, taken from a real-time video frame, is poorly illuminated on the left side and below the chin region. After applying the CLAHE the illumination on the face part changes. The brightness of the darker side of the face is increased (see Figure 27 (ii)). The image frame is further enhanced using Gaussian Filtering process, which removes noise by smoothing, as shown in Figure 27 (iii).

4.2 Recognition Results using LBPH Method

It is required to have a subject name and corresponding ID of the subject to recognize a face using OpenCV libraries and Python. In this experiment, each of the database images has subject names with corresponding image file names, and a "_" is put after subject name fields. Therefore, to create the subject IDs, all the images in the training-image folder are read and the name part from the image files are extracted. The name from the files, which are subject names, are stored in a list sorted by alphabetical order. After that, an ID is given with each subject name corresponding to the position of the subject name in the list. The subject names from the list along with the corresponding IDs of the subjects are then saved in a text file named "NameList.txt". During the recognition process, when an ID of a subject name is obtained, the corresponding subject name is called from this text file.

4.2.1 Face Recognition under Different Low-resolution

For face recognition, the input images are taken from the webcam of a laptop. The recognition rates with different low-resolution image frames are recorded. During the recognition process, image frames are taken from the webcam at an interval of 300 milliseconds. Each time 400 image frames are used for recognition, and the process is repeated 5 times.

Then the recognition rate is calculated out of those 2000 image frames. The recognition rates with our created database LRD200 and at a different resolution of the input image are tabulated in Table 5. The rates are found 78.40% at 15 px, 92.10% at 20 px, 95.95% at 30 px, 96.60% at 35 px, and 98.05% at 45 px of the input image. During recognition time the head is rotated around the camera with different expressions and angular positions.

Table 5: Recognition Rate using the LBPH Method on the LRD200 database

Recognition	Using Database LRD200		
	True Positive	***Incorrect Times***	***True Positive Rate***
At 15 px	1568	432	78.40%
At 20 px	1842	158	92.10%
At 30 px	1919	81	95.95%
At 35 px	1932	68	96.60%
At 45 px	1961	39	98.05%

The recognition rates with LDR100 database are tabulated in Table 6 which are 60.60% at 15 px, 81.65% at 20 px, 84.55% at 30 px, 92.75% at 35 px, and 95.00% at 45 px of the input image. With the increase in image resolution the recognition rate increases. Also, the number of images in the database plays a vital role in determining the recognition accuracy.

Table 6: Recognition Rate Based on the LRD100 Database

Recognition	Using Database LRD100		
	True Positive	***Incorrect Times***	***True Positive Rate***
At 15 px	1212	642	60.60%
At 20 px	1633	367	81.65%
At 30 px	1691	309	84.55%
At 35 px	1855	145	92.75%
At 45 px	1900	100	95.00%

Figure 28 shows the input face image under various resolution condition while recognizing. The image at 15 px is very hard to recognize for a human.

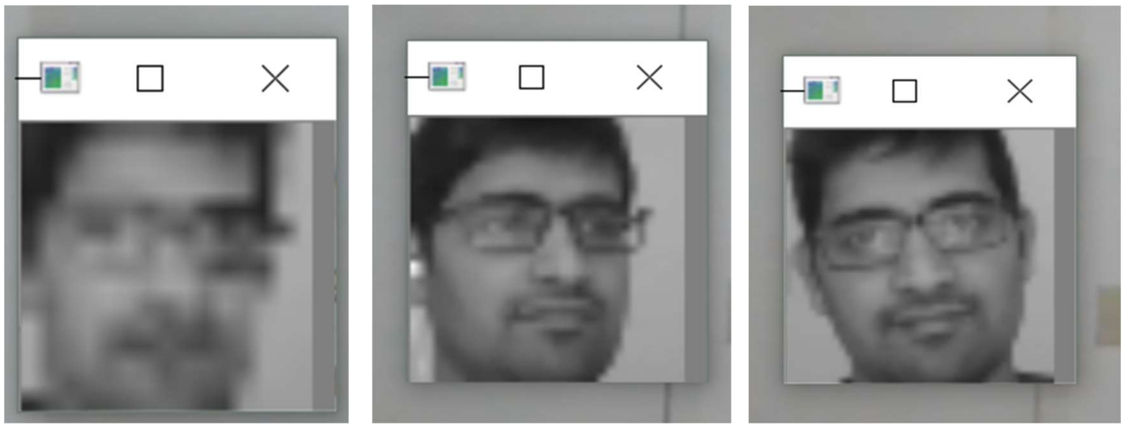


Figure 28: Face images of different resolutions for recognition.

Figure 29 demonstrates a graphical representation of face recognition rate for various low-resolution images. It also represents the trending of recognition accuracy with respect to the number of images per person in the database. Recognition rate increases as the input image pixel increases and the recognition rates are higher with the database of LRD200 than LRD100. With the increase in the number of images per person in the database, the recognition rate is increased.

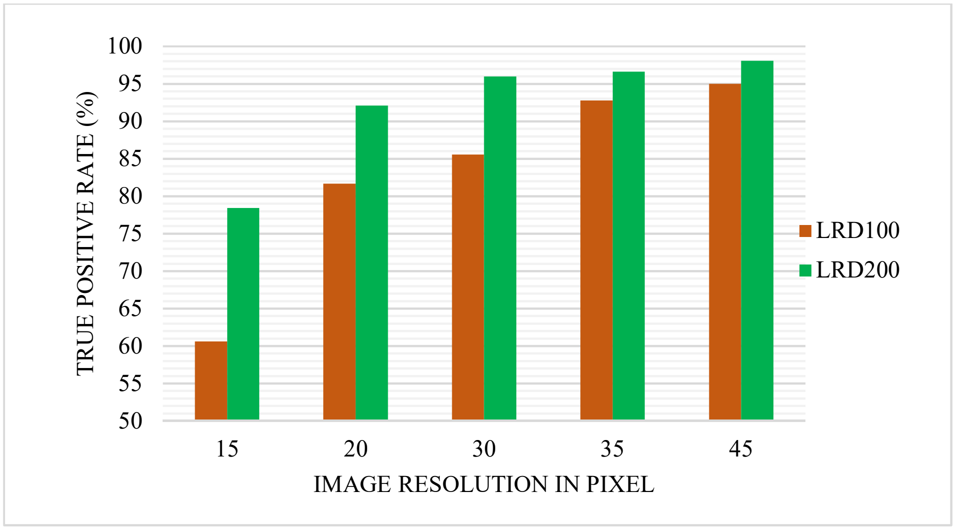


Figure 29: Face recognition accuracy with image resolution (pixel).

4.2.2 Face Recognition with Different Angular Positions

The recognition accuracy varies with the various angular positions of the head with respect to the camera. With higher deflection angle the recognition rate deteriorates.

Table 7: Recognition rate using LBPH method on the LRD200 database with the different angular positions of the face

Recognition at 45 px	**Using Database LRD200**		
	True Positive	***Incorrect Times***	***True Positive Rate***
Front Facing	1993	7	99.65%
Facing 30^0 Right	1637	383	81.85%
Facing 30^0 Left	1545	455	72.25%

The recognition rates are recorded with three angular position of the frontal head-front face, about 30^0 right and about 30^0 left positions as shown in Figure 30. While capturing image frames in front face position, the head is moved in the up and downward directions. Similarly, the head is moved in the up and downward directions while the facial deflections are about 30^0 left and about 30^0 right positions. The recognition rate is highest (99.65%) when the face is front faced. With 30^0 right facing, the recognition rate is 81.85% and with about 30^0 left facing the achieved recognition rate is 72.25%. Each time 400 frames, one frame in every 300 milliseconds, are used for recognition and the process is repeated 5 times to calculate the recognition accuracy out of 2000 frames.

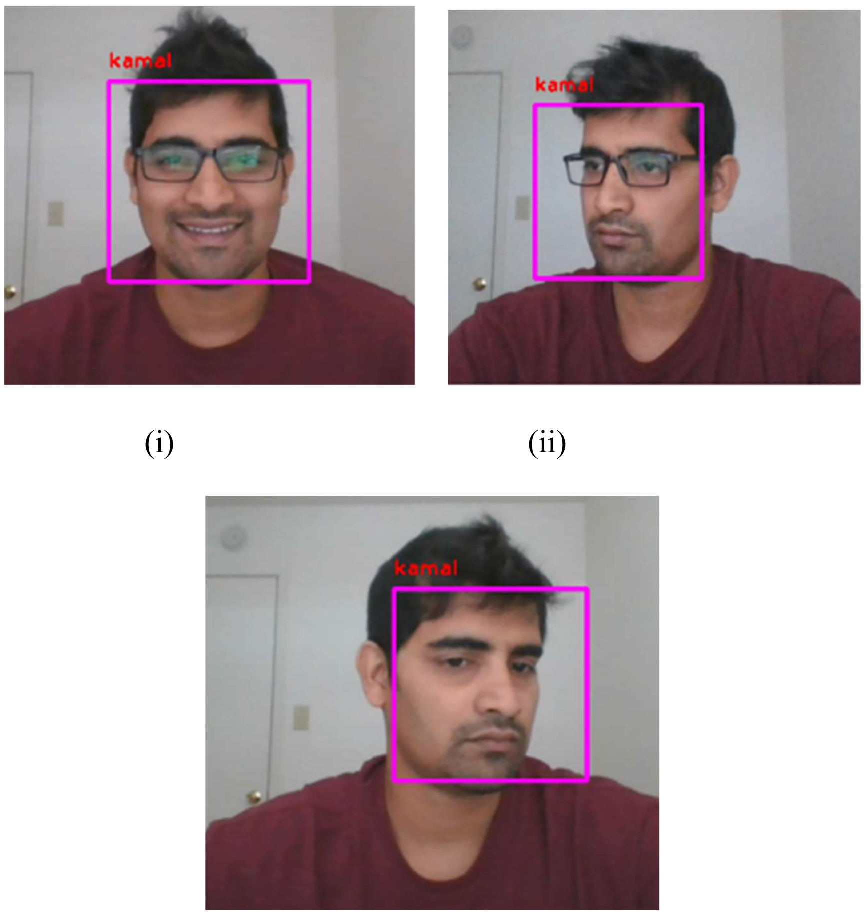


Figure 30: Face recognition with different angular deflection (i) front facing, (ii) right 30^0 facing, (iii) left 30^0 facing

4.3 Recognition Results Using Eigenface Algorithm

In the face recognition process, OpenCV libraries and python are used. The input face images are taken directly from a Lenovo Thinkpad Laptop's webcam video feed for real-time face recognition. The frames after every 300 milliseconds are considered for recognition.

4.3.1 Face recognition Under Different Low-resolutions

Face recognition rates under the different low resolution of the input image are recorded. The input image resolution ranged from as high as 110 px to as low as 15 px. The recognition rates are calculated for input images of 15 px, 20 px, 30 px, 35 px, 45 px, and 110 px. Table 8 shows the recognition rates under various resolution conditions on the LRD200 database. The achieved recognition rates are 72.25% at 15 px, 83.95% at 20 px, 86.60% at 30 px, 87.05% at 35 px and 90.11% at 45 px.

Table 8: Recognition rate using the Eigenface method at different resolutions of the input image on the LRD200 database

Recognition	**Using Database LRD200**		
	True Positive	***Incorrect Times***	***True Positive Rate***
At 15 px	1445	555	72.25%
At 20 px	1679	321	83.95%
At 30 px	1720	280	86.00%
At 35 px	1741	259	87.05%
At 45 px	1805	198	90.11%
At 110 px	1821	179	91.05%

Each time 400 frames are considered for recognition, and the process is repeated 5 times to recognize in a total of 2000 frames for a specific resolution of input image frames. While capturing input images frames the head was rotated around the camera with pose variation to get more accurate recognition results.

The same procedure is repeated with the LRD100 database image for recognition with different resolution conditions, and the results are tabulated Table 9. The recognition rates are found 70.85% at 15 px, 74.50% at 20 px, 79.10% at 30 px, 86.05% at 35 px and 87.95% at 45 px on this database.

Table 9: Recognition rate using the Eigenface method under different resolution conditions on the LRD100 Database

Recognition	**Using Database LRD100**		
	True Positive	***Incorrect Times***	***True Positive Rate***
At 15 px	1417	583	70.85%
At 20 px	1490	510	74.50%
At 30 px	1582	418	79.10%
At 35 px	1721	279	86.05%
At 45 px	1759	241	87.95%
At 110 px	1798	202	89.90%

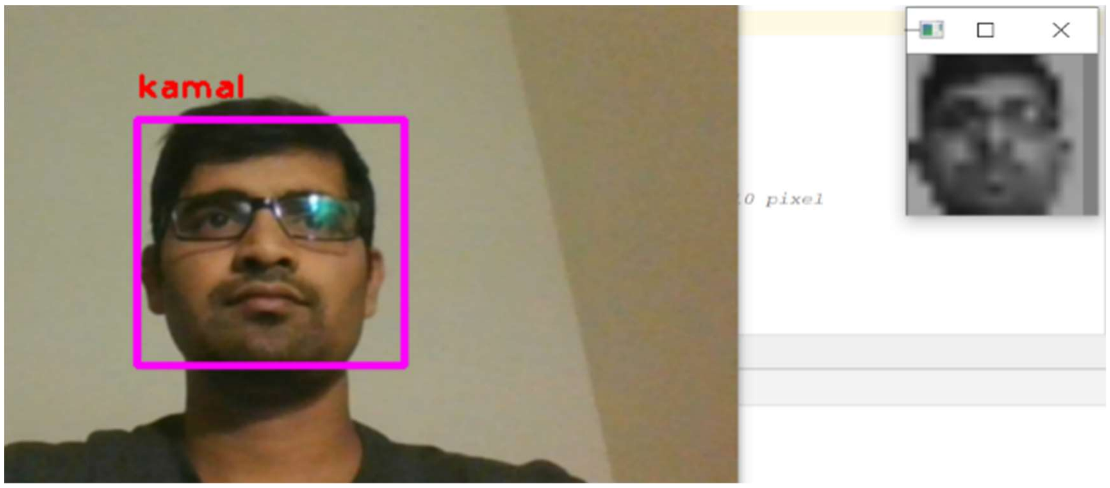


Figure 31: Face recognition at 15 px and corresponding low-resolution image of 15 px.

A real-time face recognition at 15 px resolution of the input image frame is shown in Figure 31. When a person is not included in the database image set is marked as unknown.

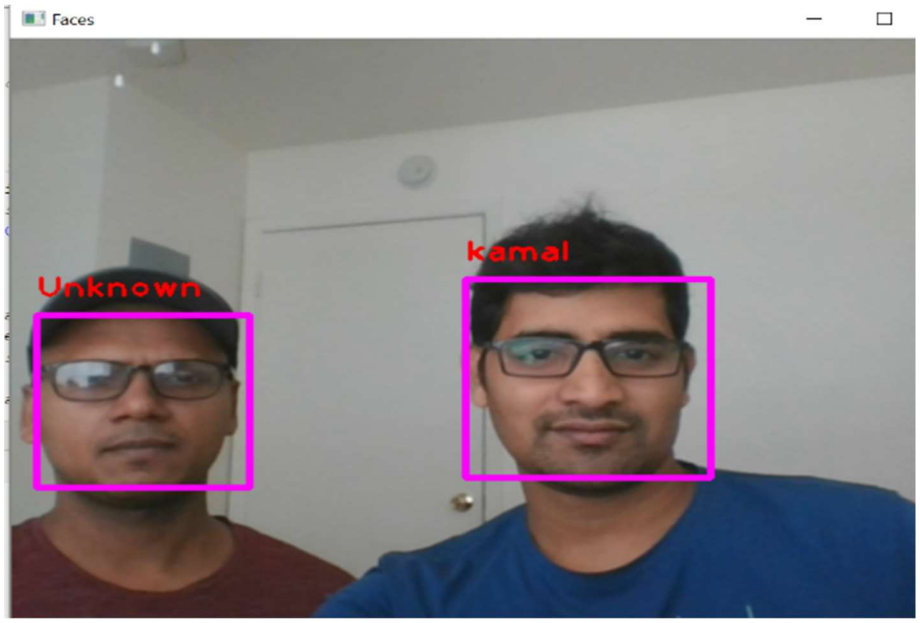


Figure 32: Real-time face recognition: known and unknown

The person on the left side of Figure 32 is removed from the database and the face recognition system marked him as unknown whereas the other person is correctly recognized and marked with his name. A graphical representation of the recognition rates with different image resolution and the different database is shown in Figure 33. The recognition rate increased with the increase in the resolution of the input image and with the increase in the number of images per person in the database.

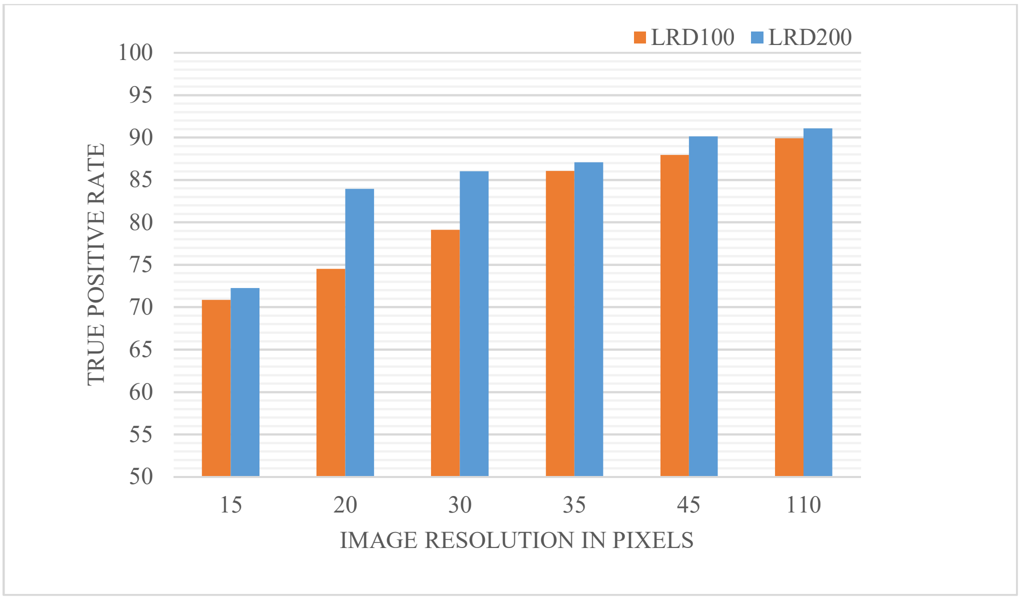


Figure 33: Graphical representation of recognition accuracy with different image resolution (pixel).

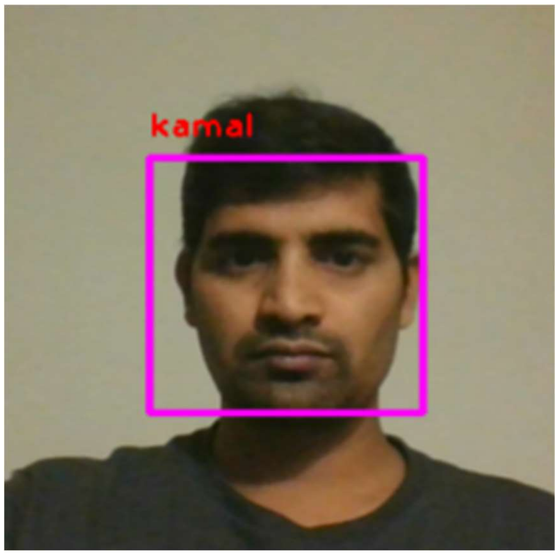
4.3.2 Face recognition with Different Angular Positions

The recognition accuracy changes with the deflection angle of the face with respect to the position of the camera. The recognition rate with front-facing input image is higher compared to the rate with left or right facing image. Face recognition rates with the various deflection angles of the face are given in Table 10. The recognition rate deteriorates with the increase in deflection angle.

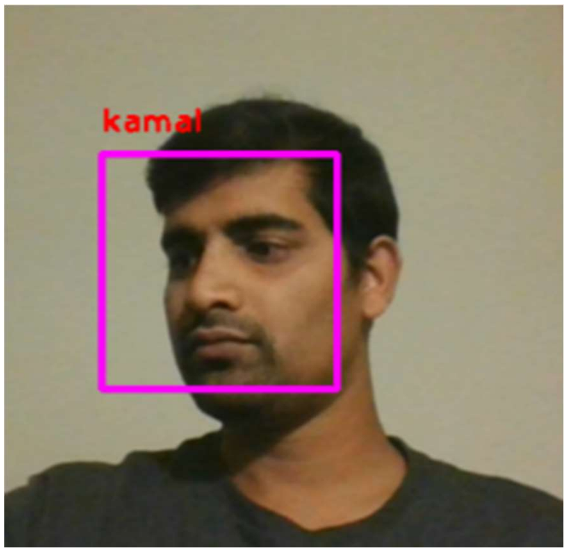
Table 10: Recognition rate using the Eigenface method for the different angular positions of the face based on the LRD200 database

Recognition at 45 px	**Using Database LRD200**		
	True Positive	***Incorrect Times***	***True Positive Rate***
Front Facing	1852	147	92.65%
Facing 30^0 Right	1681	319	84.05%
Facing 30^0 Left	1798	202	89.90%

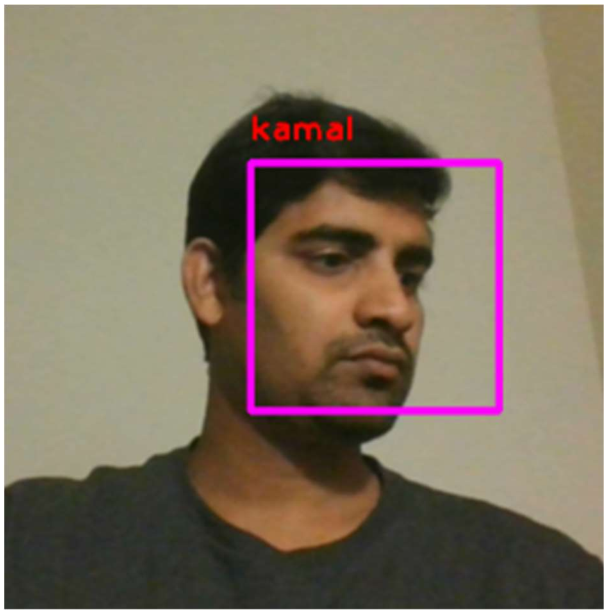
The recognition rates are recorded with 45 px resolution of the image and with three angular positions of the face: face-front facing, about 30^0 right, and about 30^o left position as shown in Figure 34. While recognizing faces in various angular position the head is moved in the up and downward directions. This experiment is done on the LRD200 database. The recognition rates are 92.65%, 84.05% and 89.90% front facing, 30^0 right deflection and 30^0 left deflection respectively. Each time with 400 frames are considered for recognition, and the process is repeated 5 times to calculate the recognition accuracy out of 2000 frames.



(i) Front Face



(ii) About 30^0 right facing



(iii) About 30^0 left facing

Figure 34: Recognition in Eigenface algorithm with i) front facing ii) 30^0 right facing & iii) 30^0 left facing

4.4 Recognition Result using Fisherface Method

4.4.1 Face Recognition Under Different Low-Resolution

The face recognition rates with two different database image sets and with the different resolution of the input image are recorded. The recognition rates using the Fisherface method based on the LRD100 database set are 55.44% at 15 px, 64.00% at 20 px, 69.48% at 30 px, 72.76% at 35 px, 76.84% at 45 px, and 84.00% at 110 px of the input image. The result is tabulated in Table 11.

Table 11: Face recognition using the Fisherface method under different resolutions of the input image based on the LRD100 database.

Recognition (Fisherface)	**Using Database LRD100**		
	True Positive	***Incorrect Times***	***True Positive Rate***
At 15 px	1386	1114	55.44%
At 20 px	1600	900	64.00%
At 30 px	1737	763	69.48%
At 35 px	1819	681	72.76%
At 45 px	1921	579	76.84%
At 110 px	2100	400	84.00%

Using the Fisherface algorithm along with CLAHE and Gaussian filtering, the face recognition rate increased. On the same database, LRD100, but with CLAHE the recognition rates are 80.96% at 15 px, 89.00% at 20 px, 90.64% at 30 px, 90.48% at 35 px, 93.92% at 45 px, and 94.92% at 110 px of the input image frames (see Table 12). The face is rotated around the camera during the recognition time and each time about 500 image frames are considered for recognition. The process is repeated 5 times to calculate the recognition rate out of 2500 image frames. The image frames are taken from the real-time video of the integrated webcam of the laptop.

Table 12: Face recognition using the Fisherface with CLAHE method under different resolutions of the input image based on the LRD100 database.

Recognition (Fisherface + CLAHE)	**Using Database LRD100**		
	True Positive	***Incorrect Times***	***True Positive Rate***
At 15 px	1868	632	74.72%
At 20 px	2063	437	82.52%
At 30 px	2191	309	87.64%
At 35 px	2262	238	90.48%
At 45 px	2348	152	93.92%
At 110 px	2373	127	94.92%

The same experiment is performed on the LRD200 database but this time using the Fisherface algorithm only. The recognition rates are found 55.80% at 15 px, 76.32% at 20 px, 92.72% at 30 px, 92.44% at 35 px, 92.52% at 45 px, and 93.64% at 110 px of the input image frames (see Table 13).

Table 13: Face recognition using the Fisherface algorithm under different resolutions of the input image based on the LRD200 database.

Recognition (Fisherface)	**Using Database LRD200**		
	True Positive	***Incorrect Times***	***True Positive Rate***
At 15 px	1395	1105	55.80%
At 20 px	1908	592	76.32%
At 30 px	2010	490	80.40%
At 35 px	2299	201	91.96%
At 45 px	2313	187	92.52%
At 110 px	2341	159	93.64%

Table 14: Face recognition using Fisherface with CLAHE method under different resolutions of the input image based on the LRD200 database.

Recognition (Fisherface + CLAHE)	**Using Database LRD200**		
	True Positive	***Incorrect Times***	***True Positive Rate***
At 15 px	1929	571	77.16%
At 20 px	2269	231	90.76%
At 30 px	2344	156	93.76%
At 35 px	2364	136	94.56%
At 45 px	2397	103	95.88%
At 110 px	2447	53	97.88%

The same experiment is performed on the LRD200 database, but CLAHE is introduced with the Fisherface method. This results in an increase of the recognition rate. The achieved recognition rates are 81.44% at 15 px, 90.76% at 20 px, 93.76% at 30 px, 94.56% at 35 px, 95.88% at 45 px, and 97.88% at 110 px of the input image frames (see Table 14).

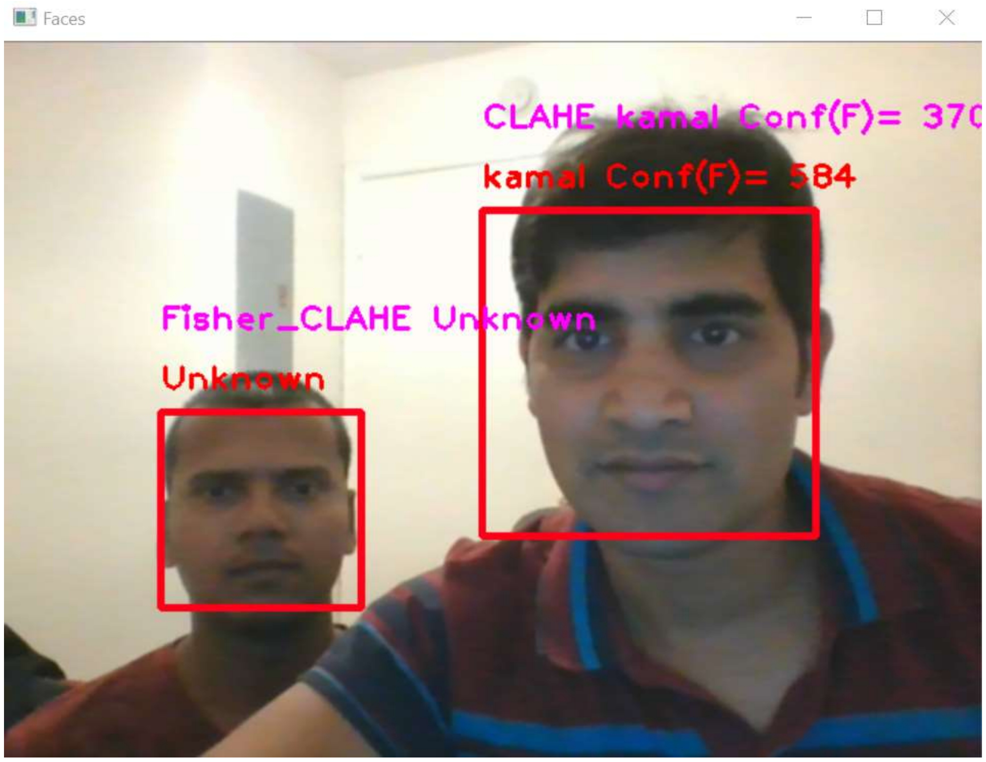


Figure 35: Face recognition using the Fisherface algorithm (45px). The person on the left side is not included in the database and is marked as unknown, whereas the other person is recognized correctly. The red color text is for the Fisherface only, and pink color text is for the Fisherface with CLAHE method.

A real-time face recognition using Fisherface and Fisherface with CLAHE method is displayed in Figure 35. The red text indicates recognition result using the Fisherface algorithm only, and the pink colored text indicates the recognition result using the Fisherface along with CLAHE method. In the figure, the person on the left side is excluded from the database and hence is marked as unknown in both methods. On the other hand, the person on the right side of the figure is in the face database and is recognized with his name displayed above the face.

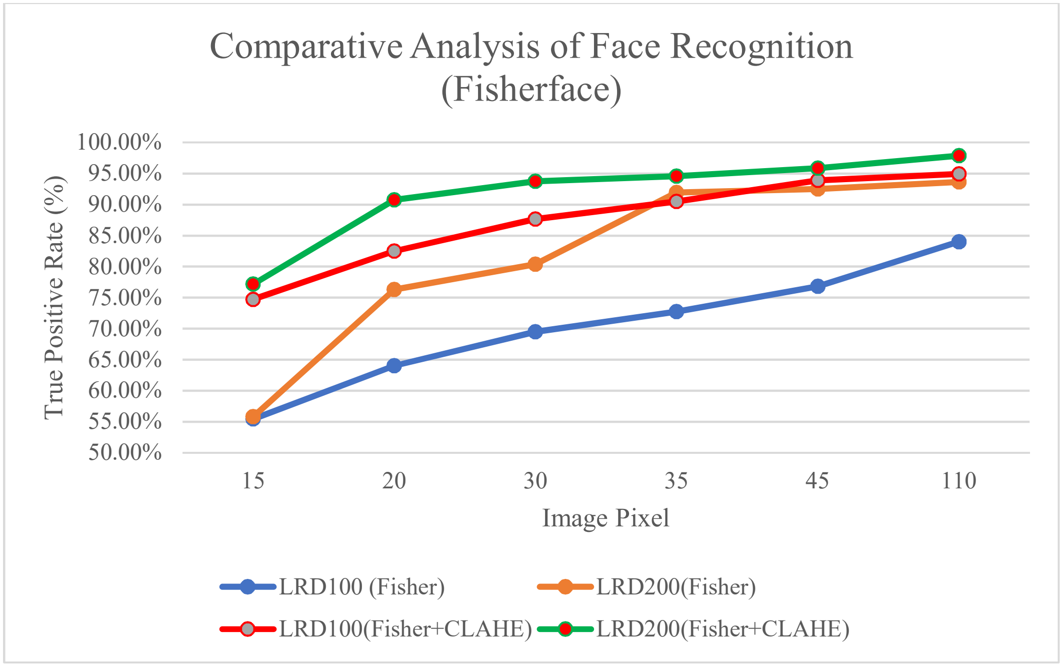


Figure 36: A comparative representation of facial recognition rate using the Fisherface algorithm with and without CLAHE.

A comparative analysis of different face recognition rates with respect to various database image under the various low-resolution conditions is illustrated graphically in Figure 36. From the graphical representation, we can see that the highest recognition (green line in Figure 36) is found with the LRD200 database and with the application of CLAHE along with the Fisherface algorithm.

It is evident from this experiment that introducing the CLAHE method with the Fisherface algorithm gives better recognition accuracy. An example is given in Figure 37 where the Fisherface algorithm along with the CLAHE algorithm can recognize the face from a real-time video frame, but the Fisherface algorithm alone fails to do it.

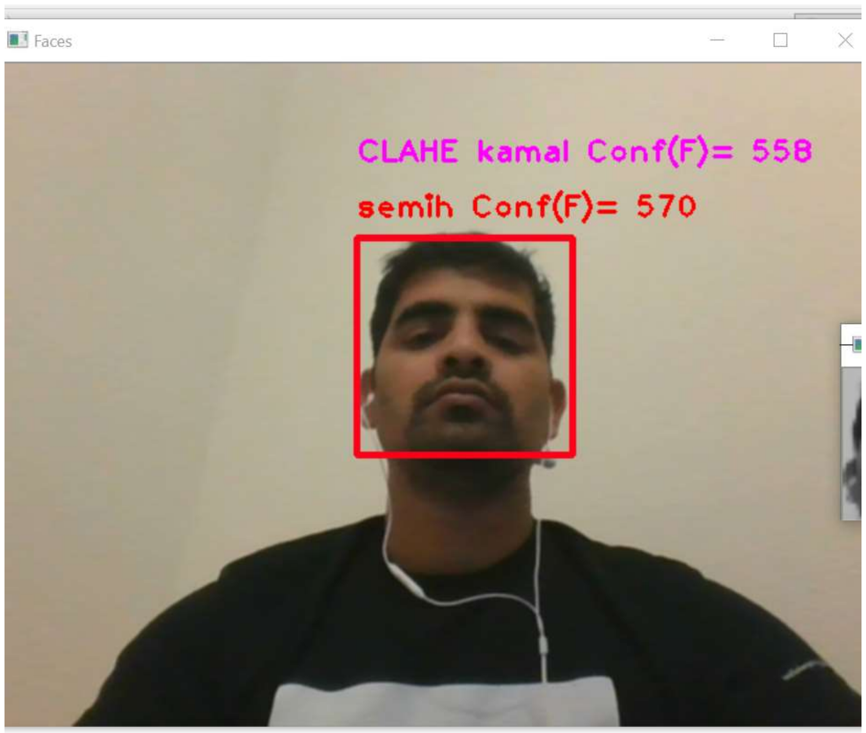
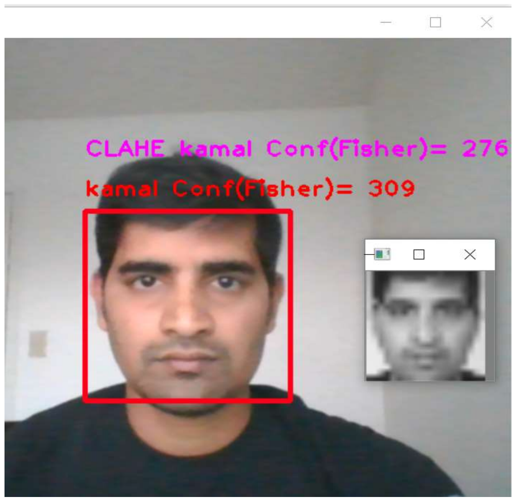


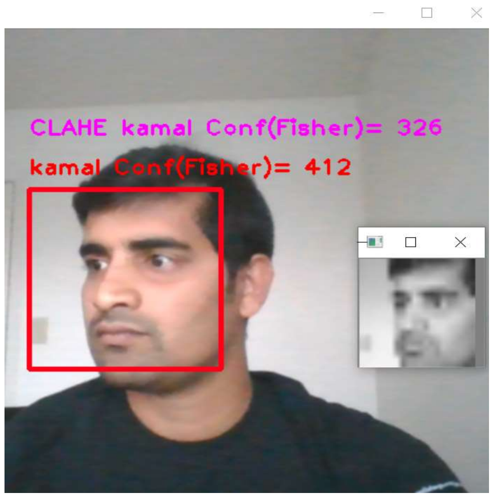
Figure 37: An image frame taken from a real-time video feed is showing that the (Fisherface + CLAHE) algorithm can recognize the face correctly (pink text) whereas the Fisherface algorithm alone cannot (red text).

4.4.2 Face Recognition with Different Angular Positions

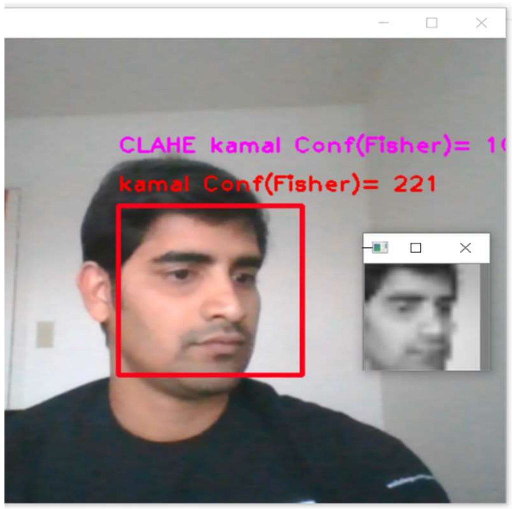
The recognition rates are recorded for different angular positions of the face with respect to the camera. The recognition rate decreases with the increase in the deflection angle.



(i) Front Face



(ii) About 30^0 right facing



(ii) About 30^0 left facing

Figure 38: Recognition with a different angular position in Fisherface (red text) and Fisherface with CLAHE (pink text) method at 15 px.

Real-Time face recognition with a different deflection angle of the face is displayed in Figure 38. In each position (front facing, about 30^0 right and about 30^0 left) of the face the head is moved in the up and downward directions while recognizing the faces. The recognition rate is highest when the face is directed straight towards the camera. Based on the LRD100 database and using Fisherface algorithm the recognition rate is found 88.76% with front facing, 71.88% with 30^0 right facing, and 70.92% with about 30^0 left facing (see Table 15).

Table 15: Recognition rate, with the different deflection angles of the face, using the Fisherface method based on the LRD100 database at 45 px of the input image.

Recognition at 45 px with Fisherface	**Using Database LRD100**		
	True Positive	***Incorrect Times***	***True Positive Rate***
Front Facing	2219	281	88.76%
Facing 30^0 Right	1797	703	71.88%
Facing 30^0 Left	1773	727	70.92%

Using the same database (LRD100) and with Fisherface algorithm along with the CLAHE method, the recognition rates are recorded (see Table 16). The face recognition rates are comparatively higher while the Fisherface algorithm is combined with the CLAHE method than using Fisherface only. The rates are 93.80% with front facing, 74.88% with about 30^0 right facing, and 90.12% with about 30^0 left facing. Application of the CLAHE method resulted in an increased recognition rate. The highest recognition rate is found with the front facing of the head.

Table 16: Recognition rate, with the different deflection angles of the face, using the Fisherface with CLAHE method based on the LRD100 database at 45 px of the input image.

Recognition at 45 px (Fisherface + CLAHE)	**Using Database LRD100**		
	True Positive	***Incorrect Times***	***True Positive Rate***
Front Facing	2345	155	94.80%
Facing 30^0 Right	1872	628	74.88%
Facing 30^0 Left	2069	431	82.76%

4.5 Recognition Results Using the Combined LBPH and Fisherface Algorithm

This experiment is carried out with 3808 image frames taken from the input video feed. One experiment is performed using a different number of images per person in the database. For this experiment, 10 different subjects are considered in the database image set. The other experiment is performed with a different number of subjects. The face recognition results are calculated using 5, 10 and 15 subjects in the database. During the entire experiment, input images of 45 px are considered for recognition.

4.5.1 Face recognition with Different Number of Images Per Person

Using the combined LBPH and Fisherface algorithm, the face recognition rates are calculated. In this experiment, a total of 3808 frames are considered for recognition. The input face images are recognized at 45px based on the LRD100 database. At first 10 good quality faces are chosen of the 10 subjects, and recognition rates are calculated. Later, each time 10 images of each subject are added, and the recognition rates are calculated. The True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), False Negative Rate, and the face recognition accuracy using the combined method are calculated as shown in Table 18. Two different sets of experiments are performed. One experiment is performed without the CLAHE method and the other without CLAHE method.

Using the combined algorithm and without CLAHE method, a maximum of 94.25% recognition accuracy is obtained with 50 images per person in the database. The TPR, TNR, FPR and FNR at this point are 96.98%, 97.51%, 0.49%, and 3.02% respectively. Introducing the CLAHE method along with the combined algorithm, a maximum of 96.55% recognition accuracy is obtained with 50 images per person in the database. The TPR, TNR, FPR and FNR at this point are 96.43%, 96.67%, 3.33%, and 3.57% respectively.

Table 17: Experimental results of face recognition using combined LBPH & Fisherface algorithm

Method	IPP	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	LBPH Th[1]	Fisher[2] Th
LBPH + Fisherface	10	65.24	5.82	94.18	34.76	80.12	145	1400
	20	66.10	8.03	91.97	33.90	79.35	145	1200
	30	83.37	1.31	98.69	16.63	91.06	145	1000
	40	82.38	2.13	97.87	17.62	90.15	145	800
	50	96.98	0.49	97.51	3.02	**94.24**	145	600
	60	96.87	17.54	82.46	3.13	89.66	145	600
	100	90.62	12.81	87.19	9.38	89.11	145	600
LBPH + Fisherface + CLAHE	10	79.22	5.07	94.93	20.79	87.18	145	1400
	20	76.34	7.83	92.17	23.66	84.40	145	1200
	30	94.37	1.45	98.55	5.63	96.46	145	1000
	40	93.89	3.02	96.98	6.11	95.44	145	800
	50	96.43	3.33	96.67	3.57	**96.55**	145	600
	60	97.22	14.69	85.31	2.78	91.26	145	600
	100	98.63	11.62	88.38	1.37	93.65	145	600

[1] Th stands for Threshold.
[2] Fisher stands for Fisherface algorithm.

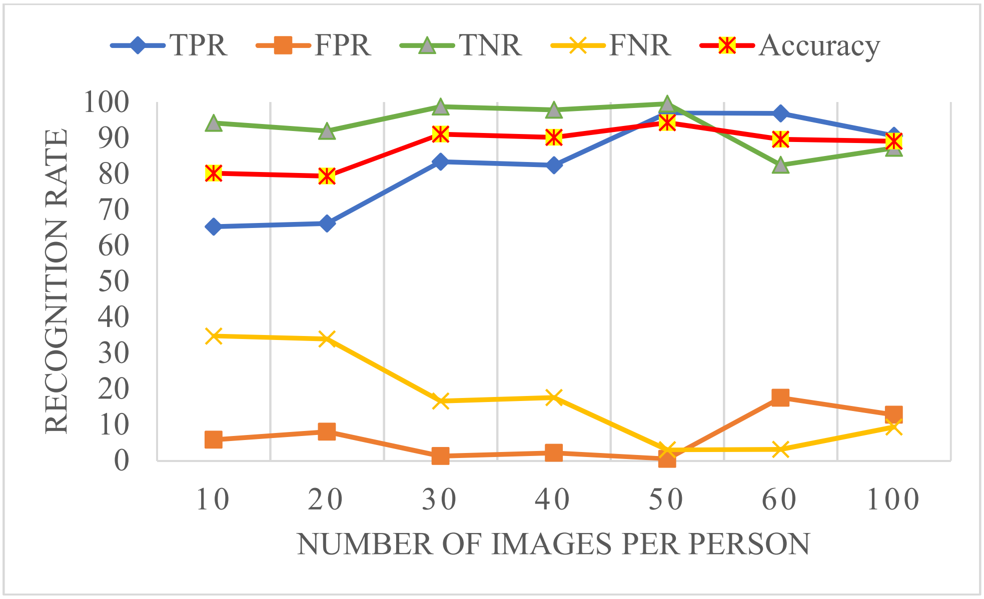


Figure 39: Graphical representation of various recognition rates using the combined algorithm without CLAHE method.

The various recognition rates using the combined algorithm and without CLAHE method are graphically represented in Figure 39. The TPR and TNR are very high as expected and the FPR and FNR are very low. Using the combined algorithm and CLAHE method the achieved recognition rates and face recognition accuracies are shown graphically in Figure 40. The maximum recognition accuracy is obtained with 50 images per person. The TPR and TNR are high whereas the FPR and FNR are low.

In both cases, with and without CLAHE, the maximum recognition accuracy is found where there are 50 images per person. The face recognition accuracy increased gradually as the number of images per person is increased. After 50 images per person, the recognition accuracy started to decrease. Therefore, using a minimum of 50 images per person in the database, with different pose and expression, we can get optimum recognition accuracy. At the optimum condition, the threshold values of LPBH and Fisherface algorithm are 145 and 600 respectively.

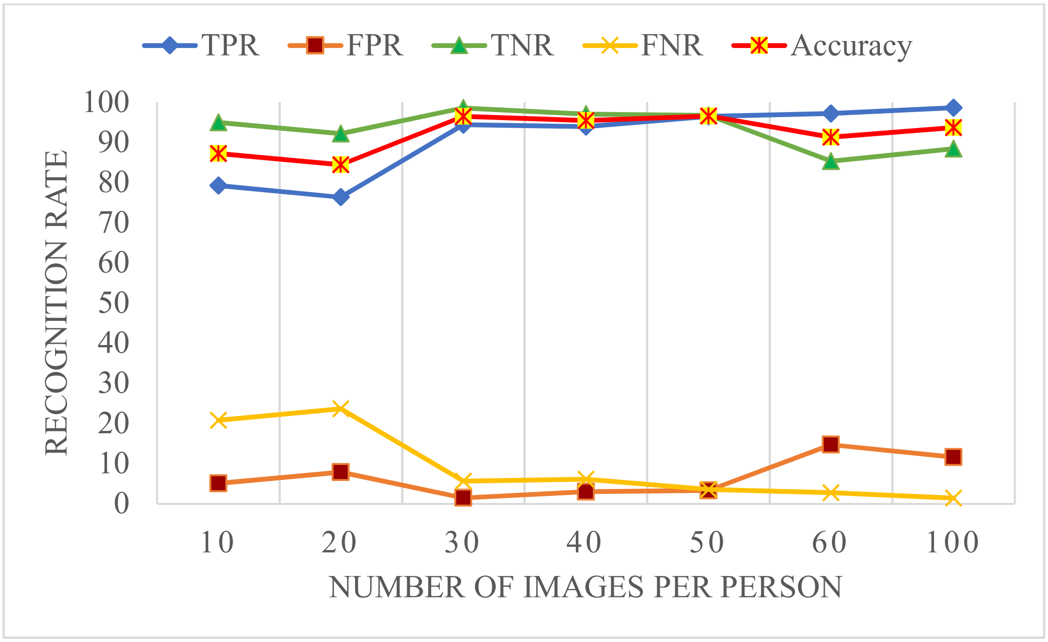


Figure 40: Graphical representation of various recognition rates using the combined algorithm and CLAHE.

4.5.2 Face recognition with Different Number of Subjects

This section represents the face recognition results using the combined LBPH and Fisherface algorithm along with the CLAHE method. The number of subjects is varied, and the recognition rates and accuracies are calculated. Initially, only 5 persons are considered in the database. The number of images per person is 50 for each case. The number of subjects is then changed to 10 and then to 15. For each case, the face recognition accuracy is calculated. The results are tabulated in Table 18. The recognition accuracy is maximum (97.16%) for 5 subjects in the database. It gradually decreases as the number of subjects is increased. With 15 subjects in the database, the achieved recognition accuracy is 91.91%.

Table 18: Recognition rates (at 45 px of the input image) using the combined LBPH and Fisherface algorithm with CLAHE method based on the database having 50 images per person and a varying number of subjects.

Subjects	TP	FP	TN	FN	TPR (%)	FPR (%)	TNR (%)	FNR (%)	Accuracy (%)	LBPH Th	Fisher Th
5	1607	45	1607	49	97.04	2.72	97.28	2.959	97.16	145	600
10	1595	55	1599	59	96.43	3.33	96.67	3.567	96.55	145	600
15	1939	54	1260	255	88.38	4.11	95.89	11.623	91.19	145	600

The recognition rates are illustrated graphically in Figure 41. The TPR and TNR are closer to 100%. On the other hand, the FPR and FNR are closer to zero. With the increase in the number of subjects, the False Positive Rate and the False Negative rate increased but the True Positive Rate and the True Negative rate decreased.

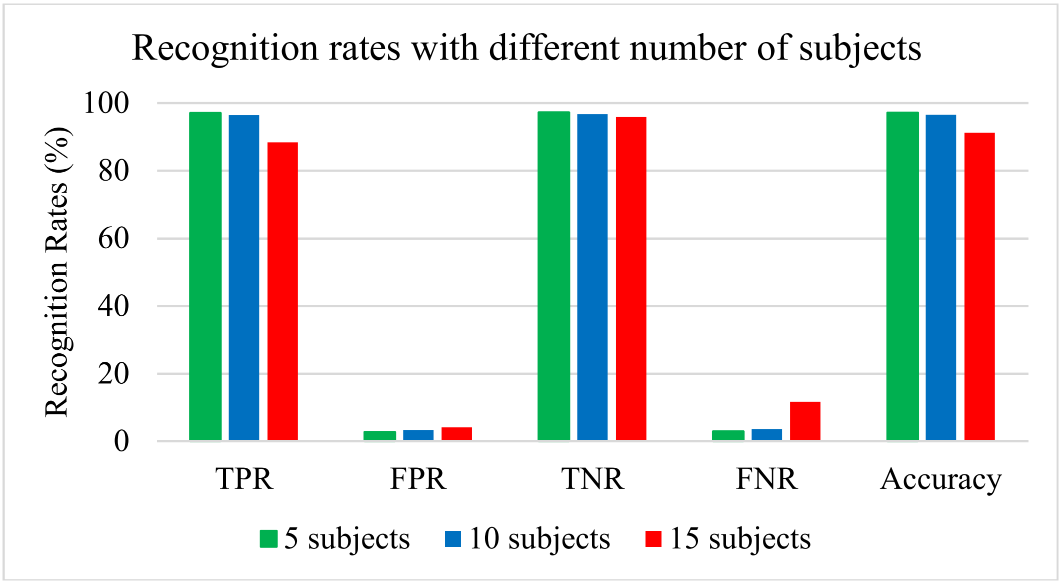
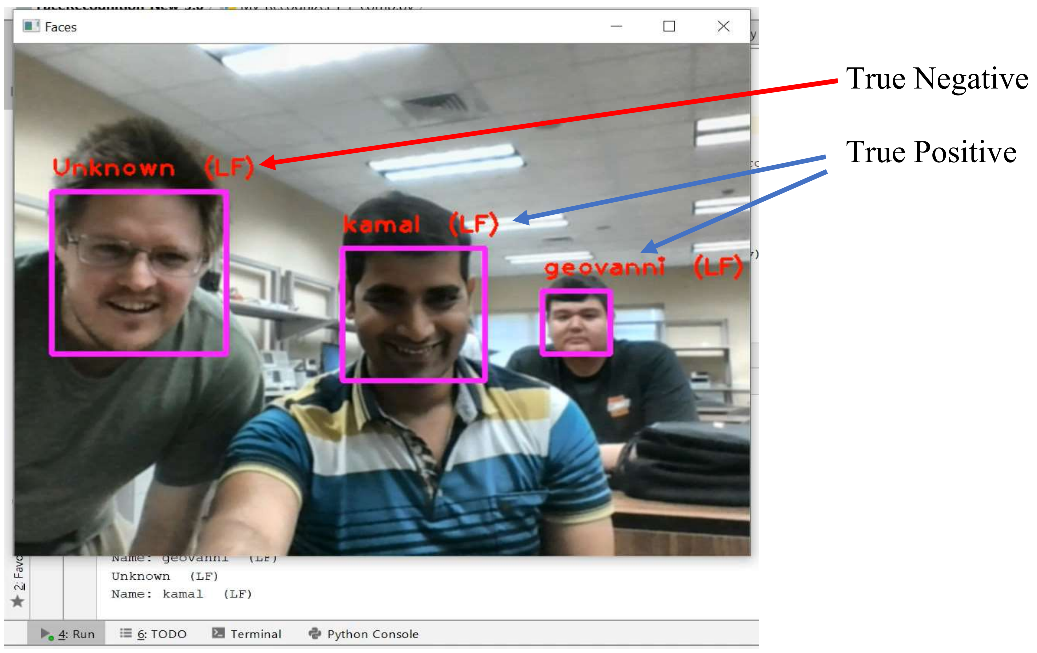
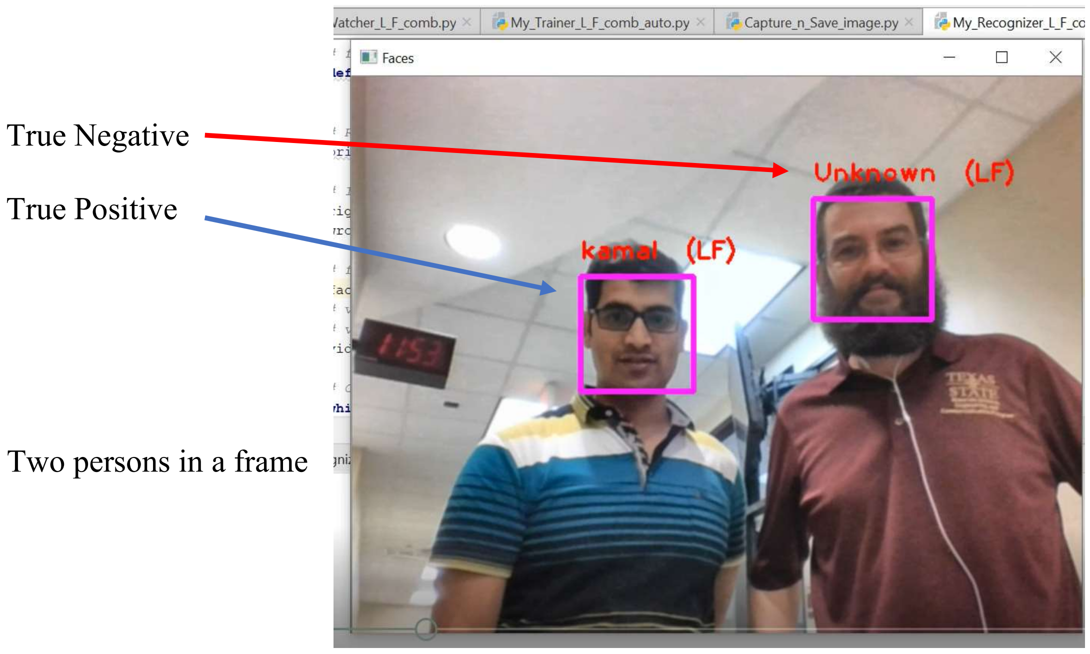


Figure 41: Graphical representation of face recognition rates with a different number of subjects.



```
Name: geovanni  (LF)
Unknown  (LF)
Name: kamal  (LF)
```

a) Three persons in a frame



b) Two persons in a frame

Figure 42: Real-time face recognition using combined LBPH and Fisherface algorithm with CLAHE where the unknowns are not included in the database.

A few snapshots of real-time face recognition are given in Figure 42. The leftmost person in Figure 42 (a) and the person in the right side in Figure 42 (b) are not included in the database, and hence they are marked as unknown. Other persons in the figure are included in the database, and their corresponding names are displayed on top of their faces.

5. CONCLUSION

The LBPH architecture of face recognition is a powerful algorithm to recognize faces under varying illumination conditions and at low-resolutions. Median filtering was applied for the database images, and the Gaussian filter was used for facial recognition. Our experiment results in a novel face recognition accuracy ranging from 78.40% to 98.05% with image resolution ranging from low-resolution of 15px to 45px with the LRD200 database. Also, with the LRD100 database, achieved face recognition accuracy ranged from 60.60% to 95% with the same pixel range of the input images. The recognition accuracy with facial deflection showed an improved result, and the accuracy ranges from 72.25% to 99.65% with facial deflection ranging from left 30 degrees to right 30 degrees with respect to the camera.

The Eigenface algorithm is useful in real-time face recognition systems, but this algorithm suffers from the problem of illumination variation. With the different illuminations and resolutions of the input image, the recognition rate degrades. Using the Eigenface algorithm the recognition accuracy was from 72.25% to 90.11% with the respective range of image resolution with our created database where there were 200 face images per person in the database. On the other hand, using 100 images per person in the database, the recognition accuracy ranged from 70.85% to 89.90% for the input image range of 15 px to 110 px. The accuracy ranged from 84.05% to 92.65% for various deflection of the face with the input image of 45 px. It is evident from this experiment that the number of images per person in the database image plays a vital role in face recognition accuracy.

The Fisherface algorithm along with the CLAHE method improved the recognition efficiency compared to the use of only the Fisherface algorithm. The True Positive rates were 81.44% at 15px and 95.88% at 45px of the input image which is better than that of the Eigenface algorithm. The recognition rates with different angular positions also showed improved accuracy compared to the Eigenface algorithm and the accuracy ranged from 74.88% to 93.80%.

Using the combined LPBH and Fisherface algorithm, at 45 px of the input image, the maximum face recognition accuracy (94.24% without CLAHE and 96.55% with CLAHE) was obtained using 50 images per person in the database of 10 different subjects. The recognition rate showed an improved result when CLAHE was used with the combined algorithm.

When the combined LBPH and Fisherface algorithm and CLAHE were used with a varying number of subjects, the recognition accuracy decreased with the increase in the number of subjects in the database. At 45 px of the input image, the recognition accuracies were 97.16%, 96.55% and 91.19% with 5, 10 and 15 subjects in the database respectively.

The use of an Android app in this research made the automatic training with new database images and face recognition process easier. The capturing of the subject images and sending the images from the Android app to the computer automate the retraining and restarting of the recognition process.

In sum, we can say that, among the three methods, the LBPH algorithm is best suitable for face recognition under illumination variation conditions as well as with the change in image resolutions. The application of the CLAHE method improves the recognition accuracy. The number of images per person in the training set database plays a crucial role in recognition accuracy at low resolution. Recognition rate increases with the increase of the number of images per person in the database. The face recognition accuracy is dependent upon the environment where the database images were taken and where the recognition system is implemented. Capturing the database images under different illumination conditions and with a variety of expression can improve the recognition rate.

The proposed face recognition system can help law enforcement officials to find and recognize criminals in various crowded places like bus and railway stations, airports and other crowded places more efficiently.

The Android app developed for this research can be used only to add face images to the database system and to restart the recognition system automatically, but it does not have the provision to delete images from the database. Our future work will focus on the Android app upgrade and integration with the recognition system so that the database can be controlled entirely using the mobile app.

APPENDIX SECTION

A snippet of a few codes used in this research is given below.

APPENDIX A

Python code for directory watcher.

```
# IMAGE FILE WATCHER PROGRAM BUILT FOR FACE RECOGNITION PROJECT.
# DETECTS FILES WITH SOME INTERVAL TIME.
# Here is the algorithm:
# IF New arrives then run another while loop until no new files are
added
# crop the faces and save it to new directory/ database files
directory.

import os, time
import psutil
import cv2
import datetime
import subprocess

# Directory to watch for incoming files
path_to_watch = r"C:\Users\kamal\OneDrive\Desktop\Photo\R7kf\photo"

# Move cropped face to training-images folder (Database Images)
move_to_path = r"C:\Users\kamal\OneDrive - Texas State
University\Research_Backup\FaceRecognition_New_3.6\training-images"

# Cascade Classifier for face detection
cascade_classifier = cv2.CascadeClassifier('opencv-
files/haarcascade_frontalface_default.xml')

# CLAHE OF IMAGE (Image Enhancement technique)
def clahe(img):
    # Converting to LAB format
    img_lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)

    # Splitting the image planes
    img_planes = cv2.split(img_lab)

    # creating CLAHE object (Arguments are optional).
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(4, 4))

    # apply clahe object on the first plane of the image.
    img_planes[0] = clahe.apply(img_planes[0])
    # Merge back the modified plane
    img_clahe_lab = cv2.merge(img_planes)

    img_clahe_rgb = cv2.cvtColor(img_clahe_lab, cv2.COLOR_LAB2BGR)
    return img_clahe_rgb  # Return the CLAHE image

img_file_ext = {"jpg", "png", 'jpeg', 'bmp', 'pgm', "gif"}

# Function To crop faces from the images of the watch directory and
move the face only files to new directory
def crop_face_n_move(watch_path):
    # List all the image files in the watch directory.
    img_file_names = os.listdir(watch_path)

    # Initialize a image counter
    counter = 0
    for files in img_file_names:  # Loop over all the image file in the
old directory
        # Each file path of old directory
        old_file_path = os.path.join(watch_path, files)  # Old file
paths

        ext = files.split(".")[-1].rstrip()
        if ext in img_file_ext:  # Checking file type image
            # Read Image file
            image = cv2.imread(old_file_path)
            # image = clahe(image)
            # Convert to gray and detect face
            img_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            faces = cascade_classifier.detectMultiScale(img_gray, 1.3,
10, minSize=(150, 150))
```

```
            for (x, y, w, h) in faces:  # Draw rectangle on face
                cv2.rectangle(image, (x, y), (x + w, y + h), (0, 25,
255), 2)
                # cv2.imshow('Face Detection', cv2.resize(image, (400,
500)))  # Show the images in the list
                cropped_face = img_gray[y:y + h, x:x + w]
                cropped_face = cv2.resize(cropped_face, (110, 110))
                cv2.imshow("Cropped Face", cropped_face)
                counter += 1

                # Cropped faces are saved and moved to new directory
with original file names.
                cv2.imwrite(os.path.join(move_to_path, files),
cropped_face)

        else:  # Continue and delete if the file is not an image file.
            os.remove(os.path.join(watch_path, files))
            continue
        # Remove image files from the old directory
        os.remove(os.path.join(watch_path, files))
        cv2.waitKey(1)
    print("Total %d faces found and moved." % counter)
crop_face_n_move(path_to_watch)

# Create a dictionary with all previous files
previous_files = dict([(file, None) for file in
os.listdir(path_to_watch)])

# While loop to continuously watch the directory.
while 1:
    start = datetime.datetime.now().time()
    print("New watch cycle started! Time: " + str(start))
    time.sleep(20)  # Sleep time in seconds.
    # List all the current files in a dictionary after every wait time.
    current_files = dict([(file, None) for file in
os.listdir(path_to_watch)])
    added_files = [file for file in current_files if not file in
previous_files]  # Files if added
```

```
    removed_files = [file for file in previous_files if not file in
current_files]
    if added_files:
        print("Receiving files. Please Wait a minute....")
        time.sleep(90)
        # print("90 second waited!")

        print("New files found in the watch directory!")
        print("Added: ", ", ".join(added_files))
        # print("Checking further for new files if arrives...")
        print("Detecting faces, cropping and moving to new
directory...")
        # Do Face Detection, cropping and moving to new directory here
        crop_face_n_move(path_to_watch)
        cv2.destroyAllWindows()

        # os.system("Python My_Trainer_L_F_comb_auto.py")
        subprocess.Popen("Python My_Trainer_L_F_comb_auto.py",
shell=True)

    if removed_files: print("Removed: ", ",".join(removed_files))
    # Do more stuffs here if needed.
    previous_files = current_files  # Stabilize the system by
equalizing the previous and current state.
    cv2.destroyAllWindows()
```

APPENDIX B

Python code to read images from the face database directory and training using the combined LBPH and Fisherface algorithm:

```
# ____________ THIS IS A FACE TRAINER PROGRAM __________________________
# ____________ TRAINING IS DONE COMBINING 2 METHODS: LBPH AND FISHER
FACE ALGORITHM _______________
# _______ THIS TRAINER TAKES ONLY FACE IMAGES AS INPUT AND NOT ANY FULL
IMAGE OTHER THAN FACE ____________
# ________ CREATED. BY KCPAUL. UPDATED ON : 11.05.2018
_________________________

import os, cv2
import psutil
import numpy as np

# Training Image Directory
# image_dir = "training-images_lrd10"
image_dir = "training-images"

# Enlist the images in the directory ( Used to find the names of the
persons)
image_list = os.listdir(image_dir)

Names = []  # List of persons : To store in the Names.txt file
IDs = []  # Corresponding ID of person : To store in the Names.txt file

# Finding and enlisting the Names of the persons and stores them in a
list named Names.
for name in image_list:
    nam = name.split("_")[0].rstrip().lower()  # Isolating the name
part from the image.
    if nam not in Names:
        Names.append(nam)
        IDs.append(Names.index(nam))

Names = sorted(Names)  # Sort the Name list with alphabetical order
print("Total Persons: " + str(len(Names)))
print(Names)
print(IDs)

# Store the names and corresponding IDs in a text file (Used to find
name and IDs during recognition).
name_list_file = open("Names.txt", "w+")
for n in Names:
    name_list_file.write(str(Names.index(n)) + "," + n + "\n")

name_list_file.close()

# Preparing Training data
path = "training-images"
imagePaths = sorted([os.path.join(image_dir, fil) for fil in
os.listdir(path)])  # Sorted list of training image paths
# print(imagePaths)

faces = []  # Array to hold all the faces
labels = []  # Array to hold all the IDs of the corresponding faces.
for image in imagePaths:
    face_image = cv2.cvtColor(cv2.imread(image), cv2.COLOR_BGR2GRAY)
    # face_image = Image.open(image).convert('L')       # Open image
and convert to gray
    face_image = cv2.resize(face_image, (110, 110))  # resize the image
so the EIGEN recogniser can be trained
    face_array = np.array(face_image, 'uint8')  # convert the image to
Numpy array

    # Image Nromalization
    # zeros = np.zeros((110, 110))
    # face_array = cv2.normalize(face_array, np.zeros((110, 110)), 0,
255, norm_type=cv2.NORM_MINMAX)

    # Here the image path is in this format: "training-
images\\kamal_1.jpg"
    # Isolating the name part from the image.
    nam = image.split("\\")[1].split("_")[0].rstrip().lower()
    ID = Names.index(nam)  # Retrieve the ID of the array
    faces.append(face_array)  # Append the Numpy Array to the list
    labels.append(ID)  # Append the ID to the Face_IDs list

    cv2.imshow('Training Image', face_array)  # Show the images in the
list
    cv2.waitKey(1)

cv2.destroyAllWindows()

print("Total number of Faces found = %d" % len(faces))
print("Total number of IDs found = %d" % len(labels))
print("Data Prepared.")

# Train the recognizer

# creating LBPH face recognizer
print("Training in LBPH method is running....")
LBPHface_recognizer = cv2.face.LBPHFaceRecognizer_create(4, 8, 8, 8,
145)  # Create LBPH face recognizer
LBPHface_recognizer.train(faces, np.array(labels))  # Train LBPH face
recognizer
LBPHface_recognizer.save('Trainer_Data/LBPHrecognizer_data.xml')  #
Save LBPH face recognizer data
print('Training in LBPH method is complete and data is saved.')

# Fisher Face Recognizer
print("Training in Fisher Face method is running....")
Fisherface_recognizer = cv2.face.FisherFaceRecognizer_create(5, 600)  #
Create Fisher face recognizer
Fisherface_recognizer.train(faces, np.array(labels))  # Train Fisher
face recognizer
Fisherface_recognizer.save('Trainer_Data/FisherFaceRecognizer.xml')  #
Save Fisher face recognizer data
print('Training in Fisher Face method is complete and data is saved.')

print('Training is Complete!')
cv2.waitKey(1)
cv2.destroyAllWindows()
```

```
print("Any running recognition program will be stopped!\n")
# ################## Stop the running Recognizer program
cam_pid = ""

# Read the txt file where the process ID is stored
file = open("Recognizer_PID.txt", "r+")
while True:
    line = file.readline()
    if line == "":
        break
    cam_pid = line

for proces in psutil.process_iter():
    pro_info = proces.as_dict(attrs=['pid', 'name'])
    procname = str(pro_info['name'])
    procpid = str(pro_info['pid'])
    if "python" in procname and procpid == str(cam_pid):
        print("Stopped Python Process ", proces)
        proces.kill()
        print("Stopping the Recognizer program having pid : " +
str(cam_pid))

print("Starting the Face recognizer program!")
os.system("Python My_Recognizer_L_F_comb.py")
```

APPENDIX C

Python code snippet of combined recognizer program.

```
# ___________ THIS IS A FACE RECOGNIZER PROGRAM, WHICH LOADS FACE
DATABASE FROM .XML FILE ___________________
# ___________ RECOGNITION IS DONE IN LBPH AND FISHER FACE ALGORITHM
(COMBINED) __________________
# ___________ CREATED. BY KCPAUL. UPDATED ON : 11.05.2018
_______________________
# ____________ Texas State University, San Marcos, Texas
____________________

# import OpenCV and other modules (As required)

import os
import cv2
import numpy as np

# Store process IDs in a text file (Used to terminate this program
while running).
# This is necessary to terminate this program when commanded by
My_Trainer_Auto program
file = open("Recognizer_PID.txt", "w+")
file.write(str(os.getpid()))
file.close()

print("PID for My_Recognizer_L_F_comb.py process is : " +
str(os.getpid()))

# CLAHE OF IMAGE (Image Enhancement technique)
def clahe(img):
    # Converting to LAB format
    img_lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)

    # Splitting the image planes
    img_planes = cv2.split(img_lab)
    # creating CLAHE object (Arguments are optional).
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(4, 4))

    # apply clahe object on the first plane of the image.
    img_planes[0] = clahe.apply(img_planes[0])
    # Merge back the modified plane
    img_clahe_lab = cv2.merge(img_planes)

    img_clahe_rgb = cv2.cvtColor(img_clahe_lab, cv2.COLOR_LAB2BGR)
    return img_clahe_rgb  # Return the CLAHE image

# Subjects are the names of the persons in the database.
subjects = []
name_file = open("Names.txt", "r")  # Subjects are loaded from the
previously saved Names.txt file.
while (True):
    line = name_file.readline()
    if line == "":
        break
    subjects.append(line.split(",")[1].rstrip())

subjects = sorted(subjects)  # Sort the name list.
no_of_subjects = len(subjects)  # Number of persons.

LBPHface_recognizer = cv2.face.LBPHFaceRecognizer_create(4, 8, 8, 8,
145)  # creating LBPH face recognizer
LBPHface_recognizer.read('Trainer_Data/LBPHrecognizer_data.xml')  #
Load/Read LBPH face recognizer data

Fisherface_recognizer = cv2.face.FisherFaceRecognizer_create(5, 600)  #
creating Fisher face recognizer
Fisherface_recognizer.read('Trainer_Data/FisherFaceRecognizer.xml')  #
Load/Read Fisher face recognizer data

# function to draw rectangle around face on the image
def draw_rectangle(img, rect):
    (x, y, w, h) = rect
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 255), 3)
```

```
# function to draw text (Name of the Recognized face)
def draw_text(img, text, x, y):
    cv2.putText(img, text, (x, y), cv2.FONT_HERSHEY_PLAIN, 1.5, (0, 0, 
255), 2)

# Recognizing Face
print("Recognizing face(s).")

# Initializing counter for efficiency calculation
true_positive = 0
false_positive = 0

true_negative = 0
false_negative = 0

# face_cascade=cv2.CascadeClassifier("opencv-
files/haarcascade_frontalface_alt2.xml")
face_cascade = cv2.CascadeClassifier("opencv-
files/haarcascade_frontalface_default.xml")

# vid_capture = 
cv2.VideoCapture("test_video/WIN_20181112_14_01_53_Pro.mp4")   # with 
Glasses
# vid_capture = 
cv2.VideoCapture("test_video/WIN_20181112_14_01_06_Pro.mp4")
# vid_capture = 
cv2.VideoCapture("test_video/WIN_20181112_14_02_27_Pro.mp4")

vid_capture = cv2.VideoCapture(0)

# Opening the camera object.
while vid_capture.isOpened():
    ret, frame = vid_capture.read()
    # avg = round(np.average(frame))  # Average Image pixel
    # print(avg)
    # if avg <= 200:  # For low resolution image apply clahe ( Under 
TEST, not finalized).
```

```
    frame = clahe(frame)
    frame = cv2.GaussianBlur(frame, (3, 3), 0)  # Applying image filter
to reduce noise effect.

    # Converting the frames into gray-scale image to recognize
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect Faces
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3,
minNeighbors=15)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 10, 255), 2)

        # Trial for low resolution image of size 20, 30 and 40 pixel
etc. (if any)
        face_area = cv2.resize(gray[y:y + h, x:x + w], (45, 45))
        # cv2.imshow("Reduced Pixel", face_area)

        face_area = cv2.resize(face_area, (110, 110))  # Resize the
face image to 110x 110 pixel
        # cv2.imshow("converted to 110", face_area)

        # Normalization
        # face_area = cv2.normalize(face_area, np.zeros((110, 110)), 0,
255, norm_type=cv2.NORM_MINMAX)

        rect = (x, y, w, h)
        draw_rectangle(frame, rect)

        # predict the face image using our face recognizer(s)
        LabelL, ConfL = LBPHface_recognizer.predict(face_area)

        LabelF, ConfF = Fisherface_recognizer.predict(face_area)

        print("LBPH: " + str(LabelL) + "  " + subjects[LabelL], "[",
round(ConfL), "]")
        print("Fisher: " + str(LabelF) + "  " + subjects[LabelF], "[",
round(ConfF), "]")
```

```
        # Recovering the name(s) from the IDs (Labels) using LBPH &
Fisherface
        if (LabelL >= 0) & (LabelL < no_of_subjects) & (LabelF >= 0) &
(LabelF < no_of_subjects):
            # Print Name of algorithm, Labels and conf of each
algorithm
            # print("LBPH  : " + str(LabelL) + "  " + subjects[LabelL],
"[", round(ConfL), "]")
            # print("Fisher: " + str(LabelF) + "  " + subjects[LabelF],
"[", round(ConfF), "]")

            # Conditioning for recognition if both gives same subject
name
            if subjects[LabelL] == subjects[LabelF]:
                label_text_comb = subjects[LabelL] + "  (LF)"
                print("Name: " + label_text_comb)
                # Print the Names and corresponding confidences on the
console
                draw_text(frame, label_text_comb, x, y - 15)
                print("Confs: " + "[" + str(round(ConfL)) + ", " +
str(round(ConfF)) + "]")
                # print("Fisher: " + str(LabelF) + "  " +
subjects[LabelF], "[", round(ConfF), "]")

                # Portion for efficiency calculation.
                if subjects[LabelL] == subjects[LabelF] == "kamal":  #
for specific user.
                    true_positive += 1
                else:
                    false_positive += 1
            else:
                Label_text = "Unknown  (LF)"
                print(Label_text)
                draw_text(frame, Label_text, x, y - 15)

                false_negative += 1
        else:
            Label_name = "Unknown  (LF)"
            print(Label_name)
            draw_text(frame, Label_name, x, y - 15)

            false_negative += 1
            # true_negative += 1

        print("True Positive=" + str(true_positive))
        print("False Positive =" + str(false_positive))
        print("True Negarive =" + str(true_negative))
        print("False Negative =" + str(false_negative))

        # Accuracy Calculation
        TPR = (true_positive / (true_positive + false_positive +
true_negative + false_negative) * 100)
        FPR = (false_positive / (true_positive + false_positive +
true_negative + false_negative) * 100)

        FNR = false_negative / (true_positive + false_positive +
true_negative + false_negative) * 100
        TNR = true_negative / (true_positive + false_positive +
true_negative + false_negative) * 100
        Accuracy = (true_positive + true_negative) / (
                true_positive + true_negative + false_positive +
false_negative) * 100

        # print("With the 45 Px input image.")
        print("True Positive =" + str(round(TPR, 2)) + " %")
        print("True Negative = " + str(round(TNR, 2)) + " %")
        print("False Negative = " + str(round(FNR, 2)) + " %")
        print("Accuracy = " + str(round(Accuracy, 2)) + " %")

    cv2.putText(frame, "Created by: Kamal Paul", (5, 20),
cv2.FONT_HERSHEY_PLAIN, 1.5, (50, 0, 255), 2)
    cv2.putText(frame, "Supervised by: Dr. Aslan", (5, 40),
cv2.FONT_HERSHEY_PLAIN, 1.5, (50, 0, 255), 2)
    cv2.imshow("Recognized Face(s)", frame)
    # cv2.imshow("Normalized", cv2.normalize(frame, np.zeros(())))

    if cv2.waitKey(100) & 0xFF == ord('q'):
        break

cv2.waitKey(1)
vid_capture.release()
cv2.destroyAllWindows()
```

APPENDIX D

Formula to calculate face recognition rates and accuracy:

True positive (TP) is equivalent with hit. True negative (TN) is equivalent with correct rejection. False positive (FP) is equivalent with false alarm which is type I error. False negative (FN) is equivalent with miss which is type II error.

Thus sensitivity, hit rate, or True Positive Rate (TPR) can be calculated using the following formula.

$TPR = TP/ (TP + FN) = 1-FNR$

Specificity or True Negative Rate (TNR) can be given by

$$TNR = TN/ (TN + FP) = 1 - FPR.$$

Miss rate or False Negative Rate (FNR) is calculated by

$$FNR = FN/ (FN + TP) = 1 - TPR.$$

Fall-out or False Positive Rate (FPR) is calculated using the following formula:

$$FPR = FP/ (FP + TN) = 1 - TNR.$$

Face recognition accuracy is calculated using

$Accuracy = (TP + TN) / (P + N) = (TP + TN)/ (TP+TN+FP+FN).$

REFERENCES

[1] G. Kaur, H. Kaur, and M. Kaur, “A Survey on Face Recognition Techniques,” in *Int. J. Adv. Res. Comput. Sci.*, 2017, vol. 8, no. 4, pp. 33–35.

[2] R. Ferizal, S. Wibirama, and N. A. Setiawan, “Gender recognition using PCA and LDA with improve preprocessing and classification technique,” in *2017 7th International Annual Engineering Seminar (InAES)*, 2017, pp. 1–6.

[3] D. N. Parmar and B. B. Mehta, “Face Recognition Methods & Applications,” *arXiv:1403.0485 [cs]*, Mar. 2014.

[4] S. K. Shiji, “Biometric prediction on face images using eigenface approach,” in *2013 IEEE Conference on Information Communication Technologies*, 2013, pp. 104–109.

[5] N. Morizet, F. Amiel, I. D. Hamed, and T. Ea, “A Comparative Implementation of PCA Face Recognition Algorithm,” in *2007 14th IEEE International Conference on Electronics, Circuits and Systems*, 2007, pp. 865–868.

[6] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, “Face recognition using LDA-based algorithms,” *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 195–200, Jan. 2003.

[7] B. S. Manjunath, R. Chellappa, and C. von der Malsburg, “A feature based approach to face recognition,” in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1992, pp. 373–378.

[8] G. Heusch, Y. Rodriguez, and S. Marcel, “Local binary patterns as an image preprocessing for face authentication,” in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, 2006, pp. 6 pp. – 14.

[9] T. Do and E. Kijak, “Face recognition using Co-occurrence Histograms of Oriented Gradients,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 1301–1304.

[10] A. Ahmed, J. Guo, F. Ali, F. Deeba, and A. Ahmed, “LBPH based improved face recognition at low resolution,” in *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 2018, pp. 144–147.

[11] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, vol. 1, pp. I–I.

[12] S. V. Chakrasali and S. Kuthale, “Optimized face detection on FPGA,” in *2016 International Conference on Circuits, Controls, Communications and Computing (I4C)*, 2016, pp. 1–6.

[13] A. R. Mohan, N. Sudha, and P. K. Meher, “An embedded face recognition system on A VLSI array architecture and its FPGA implementation,” in *2008 34th Annual Conference of IEEE Industrial Electronics*, 2008, pp. 2432–2437.

[14] A. Y. Jammoussi, S. F. Ghribi, and D. S. Masmoudi, “Implementation of face recognition system in virtex II Pro platform,” in *2009 3rd International Conference on Signals, Circuits and Systems (SCS)*, 2009, pp. 1–6.

[15] R. Endluri, M. Kathait, and K. C. Ray, “Face recognition using PCA on FPGA based embedded platform,” in *2013 International Conference on Control, Automation, Robotics and Embedded Systems (CARE)*, 2013, pp. 1–4.

[16] L. Schaffer, Z. Kincses, and S. Pletl, “FPGA-based low-cost real-time face recognition,” in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, 2017, pp. 000035–000038.

[17] X. Zhao and C. Wei, “A real-time face recognition system based on the improved LBPH algorithm,” in *2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)*, 2017, pp. 72–76.

[18] C. Conde, A. Ruiz, and E. Cabello, “PCA vs low resolution images in face verification,” in *12th International Conference on Image Analysis and Processing, 2003.Proceedings.*, 2003, pp. 63–67.

[19] M. Truk and A. Pentland, *Eigenfaces for Face Detection/Recognition*, vol. 3, no. 1. Journal of Cognitive Neuroscience, 1991.

[20] S. Zhang and M. Turk, “Eigenfaces,” *Scholarpedia*, vol. 3, no. 9, p. 4244, Sep. 2008.

[21] H.-J. Lee, W.-S. Lee, and J.-H. Chung, “Face recognition using Fisherface algorithm and elastic graph matching,” in *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, 2001, vol. 1, pp. 998–1001 vol.1.

[22] M. Sharkas and M. A. Elenien, “Eigenfaces vs. fisherfaces vs. ICA for face recognition; a comparative study,” in *2008 9th International Conference on Signal Processing*, 2008, pp. 914–919.

[23] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,” *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 19, no. 7, p. 10, 1997.

[24] M. X. He and H. Yang, “Microarray Dimension Reduction,” 2009. [Online]. Available: http://compbio.pbworks.com/w/page/16252905/Microarray%20Dimension%20Reduction. [Accessed: 15-Oct-2018].

[25] I. Zafar, E. A. Edirisinghe, S. Acar, and H. E. Bez, “Two-dimensional statistical linear discriminant analysis for real-time robust vehicle-type recognition,” presented at the Electronic Imaging 2007, San Jose, CA, USA, 2007, p. 649602.

[26] S. Muniyappan, A. Allirani, and S. Saraswathi, “A novel approach for image enhancement by using contrast limited adaptive histogram equalization method,” in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 2013, pp. 1–6.

[27] O. Patel, Y. P. S. Maravi, and S. Sharma, “A Comparative Study of Histogram Equalization Based Image Enhancement Techniques for Brightness Preservation and Contrast Enhancement,” *Signal & Image Processing : An International Journal*, vol. 4, no. 5, pp. 11–25, Nov. 2013.

[28] “Histogram equalization,” *Wikipedia*. 18-May-2018.

[29] “CS6640 - Project 2: Contrast Limited AHE.” [Online]. Available: http://www.cs.utah.edu/~sujin/courses/reports/cs6640/project2/clahe.html. [Accessed: 12-Oct-2018].

[30] G. Yadav, S. Maheshwari, and A. Agarwal, “Contrast limited adaptive histogram equalization based enhancement for real time video system,” in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2014, pp. 2392–2397.

[31] K. S. do Prado, “Face Recognition: Understanding LBPH Algorithm,” *Towards Data Science*, 10-Nov-2017. [Online]. Available: https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b. [Accessed: 11-Oct-2018].

[32] M. S. Nixon and A. S. Aguado, *Feature extraction and image processing*, 1st ed. Oxford ; Boston: Newnes, 2002.