

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'Hamed BOUGARA – Boumerdes



Institute of Electrical and Electronic Engineering
Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of
the Requirements for the Degree of

MASTER

In Electronics

Option: Computer Engineering

Title:

**Face Detection and Recognition in Real
Time**

Presented by:

- **KHELOUFI Anis**
- **NAGHECHE Bilal**

Supervisor:

Dr. CHERIFI Dalila

Registration Number:/2018

ABSTRACT

Face recognition Technology has received significant attention in the past several years in research and application, the existence of several algorithms makes a difficult choice for implementation.

The goal of this project is to implement a real time face recognition application, in the first part we implement a system using a global based algorithm (PCA) and we have tested and discussed performance of the application at each level (detecting, tracking, recognizing).

In the second part we tried to make an enhancement of the first algorithm by combining it with two different classifiers, (K-NN and Naïve Bayes), and tested the performance on three different static datasets and discussed results of the performance.

Finally, we have implemented a real time face-recognition system using a local-method based algorithm (LBPH) and displayed the results of our implementation.

DEDICATION

I dedicate this work to my parents, my teachers of life

To all my sisters and brothers,

In addition, to my friends

Anis.

I thank God Almighty my creator, my source of inspiration, knowledge and understanding. I dedicate this work to my teacher and godmother DR.CHERIFI DALILA who has encouraged me all the way and whose encouragement has made sure that I give it all it takes to finish what I have started. To my mother NORA, father ALI, my brothers, friends. Without forgetting my partner in this project KHELOUFI ANIS. And also I dedicate it to ILYES OUKAOUAK, the Indians and the creators of YouTube.

Bilal Nagheche.

ACKNOWLEDGMENTS

A special gratitude we give to our final year project Supervisor, Dr Cherifi Dalila whose contribution in stimulating suggestions and encouragement helped us to coordinate our project especially in writing this report and for her endless support and guidance. We have been extremely lucky to have a supervisor who cared so much about our work, and who responded to our questions and queries so promptly. We would like to express our deepest appreciation to all those who provided us the possibility to complete this report.

Furthermore, we would also like to acknowledge with much appreciation the crucial role of the staff of IGEE library, who gave the permission to use all required equipments and the necessary materials to complete the task, special thanks goes to William, who helped us to assemble the parts and gave suggestions. Finally yet importantly, many thanks go to the teachers of the Institute who inspired and taught us during our 5 years program and all those who contributed directly and indirectly in our 5 years journey at the institute. .

TABLE OF CONTENT

Abstract.....	ii
Dedication.....	iii
Acknowledgments	iv
Table of content	v
List of figures	viii
List of tables	x
List of abbreviations	x
General Introduction.....	1
CHAPTER 1: face recognition review.....	2
1.1. Introduction	2
1.2. Biometric perspective.....	3
1.3. Face Recognition and face detection.....	4
1.4. Face Recognition system.....	5
1.4.1 Face detection	5
1.4.2 Face tracking.....	6
1.4.3 Features extraction.....	6
1.4.4 Feature selection	7
1.4.5 Face classification.....	8
1.5.Summary.....	8
CHAPTER 2 : Face recognition algorithms and methods.....	9
2.1 Introduction:	9
2.2 Image representation	9
2.3 Face detection.....	10
2.3.1 Haar-cascade classifier	10
2.4 Face Tracking	12

2.4.1 Harris corner	12
2.4.2 KLT Feature tracker.....	12
2.5 Face Recognition	13
2.5.1 Principal Component Analysis (PCA):.....	13
2.5.2 Local Binary Patterns Histograms	20
2.6 K nearest neighbor classifier (KNN).....	22
2.7 Naïve Bayes Classifier:	23
2.8 Summary	24
Chapter 3 : Implementation and results	25
3.1 Introduction	25
3.2 Part 1:.....	28
3.2.1 Implementation methodology.....	25
3.2.1.a Face detection.....	25
3.2.1.b Face tracking	26
3.2.1.c Face Recognition.....	28
3.2.2 Results and discussion	29
3.2.2.a Face detection.....	29
3.2.2.b Face tracking	31
3.2.2.c Face recognition	32
3.3 Part 2 :.....	33
3.3.1 Database description	33
3.3.2 Results.....	33
3.3.2.a Expiriment with Datatest.mat.....	33
3.3.2.b Experiments with posetest.mat.....	34
3.3.2.c Experiments with illumination.mat	35
3.3.3 Discussion.....	35
3.4 Part 3 :.....	36

3.4.1 Implementation methodology	36
3.4.2 Results and discussion	39
3.5 Summary.....	41
General Conclusion	42
References	43

LIST OF FIGURES

CHAPTER I: face recognition review

Figure 1.1: Ascenarion of using biometric MRTD systems for passport control (left), and a comparison of various biometric features based on MRTD compatibility.....	4
Figure 1.2: face recognition processing flow.	5
Figure 1.3: face detection process.	6
Figure 1.4: Feature extraction diagram for face recognition with local binary pattern.	7
Figure 1.5: Different selected features for one image.	7

CHAPTER II: Face recognition algorithms and methods

Figure 2.1: Standard coordinates used to represent digital images.	9
Figure 2.2: Example of Haar-like features	10
Figure 2.3: Example of a Haar-like features combination	11
Figure 2.4: Frames from man-tracking video, courtesy of Kanade.....	13
Figure 2.5: Construction of face vector.	15
Figure 2.6: Reconstruction of face image.	18
Figure 2.7: express the image in pixels.....	20
Figure 2.8: Circular LBP.	21
Figure 2.9: express the image on histogram.	22
Figure 2.10: random different elements	23

Chapter III: Implementation and results

Figure 3.1: Face detection for a single target	29
Figure 3.2: multiple target face detection.....	30
Figure 3.3: face detection application with false positives	30
Figure 3.4: face tracking application with the targets head leaning to the right	31
Figure 3.5: face tracking application with the targets head leaning to the left.....	31

Figure 3.6: face recognition application in Matlab.	32
Figure 3.7: assigning ID number to the captured face.	36
Figure 3.8: captured face.	36
Figure 3.9: the 20 samples of the captured face.	37
Figure 3.10: how the files should be like.	37
Figure 3.11: A generated .yml trained file.	37
Figure 3.12: flow chart of recognition using LPBH.....	38
Figure 3.13: detection of multifactes.....	39
Figure 3.14: recognized face for trained subject	40
Figure 3.15: detected non trained subject.....	40
Figure 3.16: detected sbjects trained and non trained	40
Figure 3.17: detected and recognized subjects	40

LIST OF TABLES

Table 3.1: Accuracy results of PCA with Bayes classifier.....	33
Table 3.2: Table 3.2 : PCA results with K-NN classifier	34
Table 3.3: Table 3.3 : Result of PCA with BAYES and KNN classifier with posetest dataset	35
Table 3.4: Results OF PCA with Bayes and K-NN classifier with illumination dataset	35

LIST OF ABBREVIATIONS

IDLE : Integrated Development Learning Environment

KLT : Kanade lucas tracker

K-NN : K-Nearest Neighbor

LPB : Local Binary Pattern

LBPH : Local Binary Pattern Histogram

MRTD : Machine Readable Travel Document

PCA : Principale Component Analysis

PIN : Personel Identification Number

OpenCV : Open Computer Vision

PDA : Personal Digital Assistant

General Introduction

Face recognition is a task so common to humans, that the individual does not even notice the extensive number of times it is performed every day. Although research in automated face recognition has been conducted since the 1960's, it has only in the past several years caught the attention of the scientific community. Many face analysis and face modeling techniques have progressed significantly in the last decade [1]. However, the reliability of face recognition schemes still poses a great challenge to the scientific community.

The task of Face Recognition Algorithms is to compare two images and determine if they belong to the same person. Face Recognition systems are developed to detect and recognize a person that differ in characteristics. The Face Recognition Systems have evolved greatly during the last decades. Because of this development, there is an increase in algorithmic complexity, that takes long computation time and power. Many tools such as Principle Component Analysis, Linear Discriminant Analysis, Independent Component Analysis, Support Vector machine, Genetic algorithm have been used for face recognition systems [2].

Our aim is to implement face recognition methods for real time applications, in order to achieve the implementation of this project a research was conducted and is described in the report as follows :

Chapter 1 presents a general review on the Face Recognition Technology; includes the process steps of a face recognition system. **Chapter 2** provides a theoretical background and methods used in constructing face recognition application. **Chapter 3** is devoted to describe the developed application, and to present the obtained results. Finally, conclusions are extracted from the project and some guidelines for the future development of the system are presented.

Chapter 1:

Overview on face recognition

1.1. Introduction:

Face recognition is a task that humans perform routinely and effortlessly in their daily lives. Wide availability of powerful and low-cost desktop and embedded computing systems has created an enormous interest in automatic processing of digital images and videos in a number of applications, including biometric authentication, surveillance, human-computer interaction, and multimedia management.

Research in face recognition is motivated not only by the fundamental challenges this recognition problem poses but also by numerous practical applications where human identification is needed. Face recognition as one of the primary biometric technologies became more important owing to rapid advances in technologies such as digital cameras, Internet and mobile devices and increased demands on security. It has also several advantages over other biometric technologies where it is natural, nonintrusive and easy to use. A face recognition system is expected to identify faces present in images and videos automatically. It can operate in either or both of two modes: (1) face verification (or authentication) and (2) face identification (or recognition). Face verification involves a one-to-one match that compares a query face image against a template face image whose identity is being claimed. Face identification involves one-to-many matches that compares a query face image against all the template images in the database to determine the identity of the query face. Another face recognition scenario involves a watch-list check, where a query face is matched to a list of suspects (one-to-few matches). The performance of face recognition systems has improved significantly since the first automatic face recognition system was developed by Kanade. Furthermore, face detection, facial feature extraction and recognition can now be performed in real time for images captured under favorable (i.e., constrained) situations. Although progress in face recognition has been encouraging, the task has also turned out to be a difficult endeavor especially for unconstrained tasks where viewpoint, illumination, expression, occlusion, accessories and so on vary considerably. In the following sections, we give a brief review on technical advances and analyze technical challenges [3].

1.2. Biometric perspective:

Face as a biometric finds wide applications in authentication, security, and so on. One potential application is in the security systems used by governmental agencies, collecting visitor's fingerprints and face images. Biometric signatures enable automatic identification of a person based on physiological or behavioral characteristics.

Physiological biometrics are biological/chemical traits that are innate or naturally grown, while behavioral biometrics are mannerisms or traits that are learned or acquired. Biometrics technologies are becoming the foundations of an extensive array of highly secure identification and personal verification solutions. Compared with conventional identification and verification methods based on personal identification numbers (PINs) or passwords, biometrics technologies offer some unique advantages. First, biometrics are individualized traits while passwords may be used or stolen by someone other than the authorized user. Also, a biometric is very convenient since there is nothing to carry or remember. In addition, biometric technology is becoming more accurate and inexpensive.

Among all biometrics, face biometric is unique because face is the only biometric belonging to both physiological and behavioral categories. While the physiological part of the face biometric has been widely researched in the literature, the behavioral part is not yet fully investigated. In addition, as reported in, face has an advantage over other biometrics because it is a natural, non-intrusive and easy-to-use. For example, among the six biometrics : face, finger, hand, voice, eye, and signature in Figure 1.1, the face biometric ranks first in the compatibility evaluation of a machine readable travel document (MRTD) system on the basis of six criteria: enrollment, renewal, machine-assisted identity verification requirements, redundancy, public perception, storage requirements and performance. Probably the most important feature of a biometric is its ability to collect the signature from non-cooperating subjects. Besides applications related to identification and verification such as access control, law enforcement, ID, licensing and surveillance. face recognition is also useful in human-computer interaction, virtual reality, database retrieval, multimedia, computer entertainment, for a review of face recognition applications.

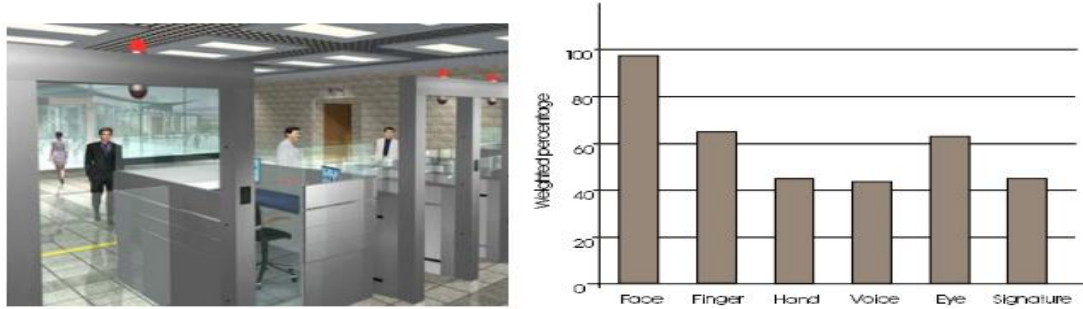


Figure 1.1: A scenario of using biometric MRTD systems for passport control (left), and a comparison of various biometric features based on MRTD compatibility[3]

1.3. Face recognition and face detection

Face recognition is a technique of recognizing faces but it is not necessary to freeze the user in order to take a picture. However, there is a problem with recognizing faces when the pose of the face is different, but in particular, there is a limit on face rotations in depth, which include left, right, up and down rotations. Face recognition itself is difficult because it is a fine discrimination task among similar objects, once we can find faces, which are quite similar. Adding pose variation naturally makes the problem more difficult. This is because the appearance of a person's face changes under rotation since the face has a complex 3D structure. At this point we have to distinguish face recognition from face detection. Many people think that these two terms are the same. Though even they have many similar techniques, are based on the same idea and algorithms, are two different systems. The main difference is the fact that face recognition is detecting faces and search through a dataset in order to find an exact match but on the other hand face detection is looking for any match and as soon as one is found then the search stops. This task of face recognition seems to be sequential and have traditionally often been treated as such. However, it is both computationally and psychophysically more appropriate to consider them as a set of co-operative visual modules with closed-loop feedback. In order to realize such a system, an integrated approach has been adopted to perform acquisition, normalization and recognition in a coherent way. Images of a dynamic scene are processed in real-time to acquire, normalize and align face sequence. In essence, this process is a closed-loop module that includes the computation and fusion of three different visual cues: motion, color and face appearance models [4].

1.4. Face Recognition system:

Face recognition is a visual pattern recognition problem. Therefore a face as a three-dimensional object, subjects to varying illumination: Pose, expression and so on. A face recognition system generally consists of four modules as depicted in Figure 1.2, where localization and normalization (face detection and alignment) are processing steps before face recognition (facial feature extraction and matching) is performed [5].

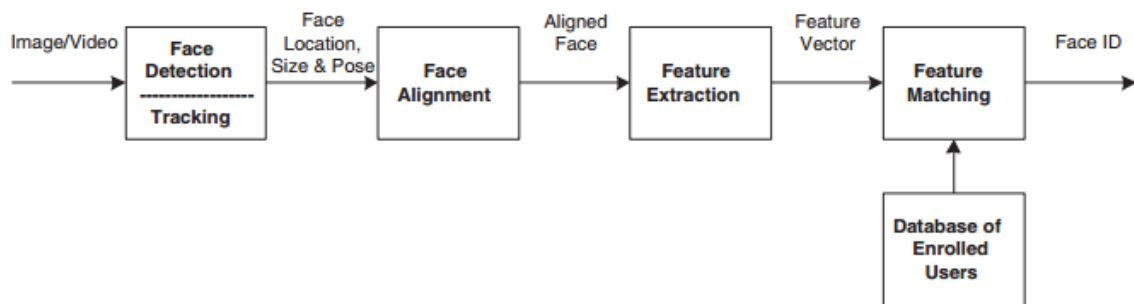


Figure 1.2: face recognition processing flow[5].

1.4.1. Face Detection:

The main function of this step is to determine whether human faces appear in a given image and where these faces are located. The expected outputs of this step are patches containing each face in the input image. In order to make further face recognition system more robust and easy to design, face alignment are performed to justify the scales and orientations of these patches. Besides serving as the pre-processing for face recognition, face detection could be used for region-of-interest detection, retargeting, video and image classification. Face detection is a concept that includes many sub-problems. Some systems detect and locate faces at the same time, others firstly perform

a detection routine and then, if positive, they try to locate the face. Then, some tracking algorithms may be needed. The following image shows how the face detection is done.

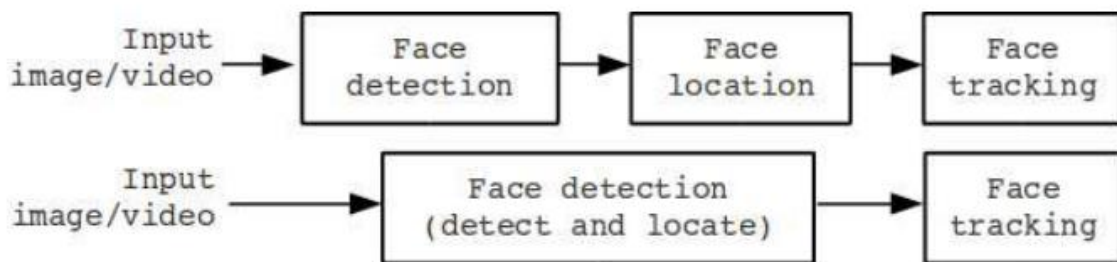


Figure 1.3: face detection process.

1.4.2. Face Tracking:

Many face recognition systems have a video frames as an input. Those systems may require to be capable of not only detecting but tracking faces. Face tracking is essentially a motion estimation problem. It can be performed using many different methods such as head tracking, feature tracking, image-based tracking and model-based tracking. The basic face tracking process seeks to locate a given image in a picture. Then, it has to compute the differences between frames to update the location of the face. There are many issues that must be faced: Partial occlusions, illumination changes, computational speed and facial deformations [6].

1.4.3. Feature Extraction:

Humans can recognize faces from the age of 5 years old. It seems to be an automated and dedicated process in our brains [7], though it is a much debated issue [8]. What it is clear is that we can recognize people we know even when they are wearing glasses or hats. We can also recognize men who have grown a beard. It is not very difficult for us to see our grandma's wedding photo and recognize her, although she was 23 years old. All these processes seem trivial, but they represent a challenge to the computers. In fact, face recognition's core problem is to extract information from photographs. This feature extraction process can be defined as the procedure of extracting relevant information from a face image. This information must be valuable to the later step of identifying the subject with an acceptable error rate. The feature extraction process must be efficient in terms of computing time and memory usage. The output should also be optimized for the classification step. [9]

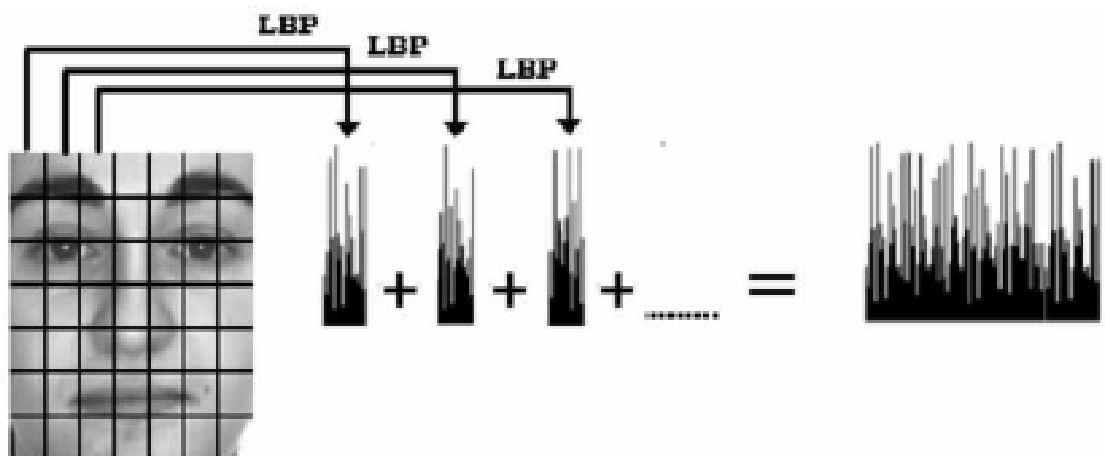


Figure 1.4: Feature extraction diagram for face recognition with local binary pattern[9].

Feature extraction involves several steps: dimensionality reduction, feature extraction and feature selection. These steps may overlap, and dimensionality reduction could be seen as a consequence of the feature extraction and selection algorithms. Both algorithms could also be defined as cases of dimensionality reduction.

1.4.4. Feature selection:

Feature selection algorithm's aim is to select a subset of the extracted features that cause the smallest classification error. The importance of this error is what makes feature selection dependent to the classification method used. The most straightforward approach to this problem would be to examine every possible subset and choose the one that fulfills the criterion function. However, this can become an unaffordable task in terms of computational time. Some effective approaches to this problem are based on algorithms like branch and bound methods. [10]

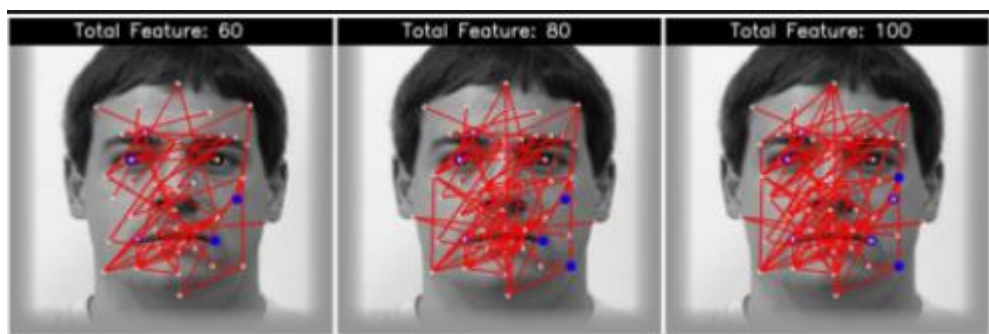


Figure 1.5: Different selected features for one image[10].

1.4.5. Face classification:

Once the features are extracted and selected, the next step is to classify the image using a wide variety of classification methods. Sometimes two or more classifiers are combined to achieve better results. On the other hand, most model-based algorithms match the samples with the model or template. Then, a learning method can be used to improve the algorithm

Classification methods are used in many areas like data mining, finance, signal decoding, voice recognition, natural language processing or medicine. Here classifiers will be addressed from a general pattern recognition point of view. Many face recognition applications include a tagged set of subjects.

In classification the similarity between faces from the same individual and different individuals after all the face images in database are represented with relevant features. Sometimes feature extraction & recognition process done simultaneously [11].

1.5. Summary:

The technology of face recognition is widely used in security systems, authentication, and individual identification. Through this chapter, we have presented a general overview on face recognition. We highlighted the importance of face recognition as a physiological biometric and its advantage over other biometrics. After explaining the difference between face recognition and detection, we presented a general diagram of face recognition system that contains three main blocks, face detection, tracking, and recognition that is the main focus of our project, we will cover these parts in details in the coming sections of this project

Chapter 2:

Face Recognition Algorithms

2.1 Introduction

This chapter provides a description of the theory behind the algorithms that we have implemented and used at each level of this project, where in the first section we described the algorithm used in the detection and then the tracking method that followed it and after the tracking we mentioned the face recognition methods. We used in this project PCA, LBPH and in the last section we described K-NN and Naïve Bayes classifiers

2.2 Image representation

A two dimensional digital image can be represented as function $I(x, y)$ where x and y are discrete coordinate quantities. For simple notational clarity and convenience, integer values are used for these discrete coordinates. The origin $(0, 0)$ is located on the top left of an image. The coordinate convention used to represent a digital image is shown in figure 2.1 [12].

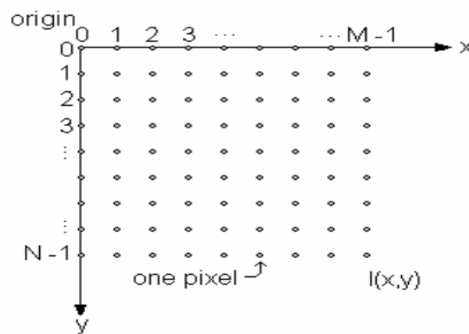


Figure 2.1: Standard coordinates used to represent digital images[12].

The representation of images can be written as a $N \times M$ matrix on the form

$$I(x, y) = \begin{bmatrix} I(0,0) & I(0,1) & \cdots & I(0,M-1) \\ I(1,0) & I(1,1) & \cdots & I(1,M-1) \\ \vdots & \vdots & \ddots & \vdots \\ I(N-1,0) & I(N-1,1) & \cdots & I(N-1,M-1) \end{bmatrix} = \begin{bmatrix} I_{0,0} & I_{0,1} & \cdots & I_{0,M-1} \\ I_{1,0} & I_{1,1} & \cdots & I_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ I_{N-1,0} & I_{N-1,1} & \cdots & I_{N-1,M-1} \end{bmatrix} \quad (2.1)$$

Each element of the matrix is called an image element, picture element or pixel. The terms image and pixel will be used throughout the rest of this project to denote a digital image and its elements. The pixels are registered by a digital sensor and may encode color or intensity.

Sometimes it can be more convenient to represent the image as a vector or array instead of a matrix. An image can be represented as a face vector with $p = N \times M$ elements. The vector is constructed by adding each row of $I(x, y)$

$$C = [(I_{0,0}, I_{0,1}, \dots, I_{0,M-1}) (I_{1,0}, I_{1,1}, \dots, I_{1,M-1}) \dots (I_{N-1,0}, I_{N-1,1}, \dots, I_{N-1,M-1})]^T \quad (2.2)$$

Each pixel of the image then corresponds to a coordinate in a p -dimensional space often referred to as the image space.

2.3 Face detection

Before recognizing a face, it is essential to detect and extract the faces from the original pictures. For recognition, the algorithms compare only faces and any other element in the picture that is not part of a face will deteriorate the recognition.

2.3.1 Haar-cascade classifier

Haar-cascade is a method, invented by Viola and Jones (2001), which trains a machine learning for detecting objects in a picture. In this context, it can be used to detect faces. The name of this method is composed of two important words, Haar and Cascade. Haar belongs to Haar-like features which is a weak classifier and will be used for the face recognition. A Haar-like feature is a rectangle which is split into two, three or four rectangles. Each rectangle is black or white. Figure 2.2 shows the different possible features. A Haar-cascade needs to be trained with various positive and negative pictures. The objective is to extract the combination of these features that represents a face. While a positive picture contains the object which has to be recognized, a negative picture represents a picture without the object. In the context of face detection, a positive picture possesses a face, and a negative picture does not. This machine learning requires a grayscale pictures. The intensity of gray will be used to detect which feature is represented. These features can be found by calculating the sum of the dark pixels in an area subtracted by the sum of the bright pixels [13].

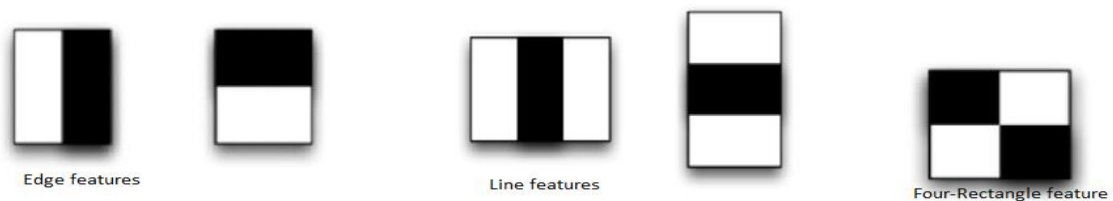


Figure 2.2: Example of Haar-like features[13]

The extracted combination of features from the training part will be used for detecting faces in a picture. To detect a face in an unknown picture the combination of the features will be researched. The features are tried to be matched only in a block of pixels defined by a scale. The scale can be for example a square of 24x24 pixels. Each feature of the combination will be tried to be matched one by one in the block. If one of the features does not appear in the block, the research in it will be stopped. The remaining features will not be tested because the machine concludes that there is no face in this block. Then, a new block is taken, and the process will be repeated. This method tests all the blocks of pixels with the researched combination in cascade which explains the second word in the name of the method. This method is efficient to detect an image without faces because only a few tests need to be run to infer that the image does not contain a face. A face is consequently detected when each feature of the combination has been recognized correctly in a block. Figure 2.3 is a rough representation of the features combination which will be tried to be matched in each block. We can see that the eyes are darker than the cheeks and the middle of the nose is brighter. All these features which were extracted from the training are used to find a pattern to represent a face [13].

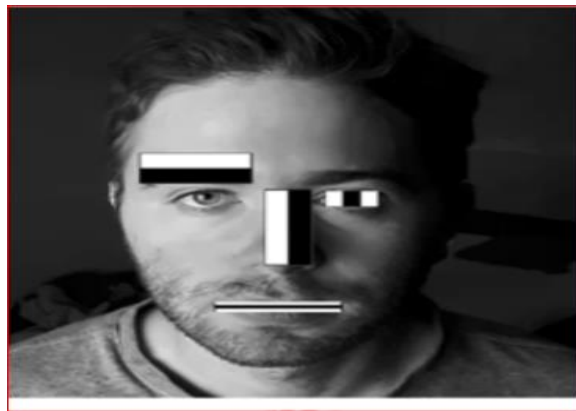


Figure 2.3: Example of a Haar-like features combination[13]

The process will proceed block by block until the last one. After checking the last block, the scale is increased, and the detection process starts again. The process is repeated several times with different scales to detect faces of different size. Only few pixels are different between two neighbor blocks. Therefore, each time a face is detected in a picture, the same face is detected in different blocks. All the detected faces that concern the same person are merged and are considered as one at the end of the entire

process. The accumulation of these weak classifiers builds a face detector able to detect faces very fast with a suitable accuracy. A Haar-cascade classifier has to be trained only once. Thus, it is possible to create one's own Haar-cascade or use one which has already been trained [13].

2.4 Face Tracking

The most popular method for feature point tracking is the KLT algorithm which was introduced by Lucas and Kanade [14]. The KLT algorithm automatically detects a scattered set of feature points having sufficient texture for tracking them reliably. Afterwards, by estimating for each point translation detected points are tracked that minimizes the dissimilarity between windows centered at current feature point position and the position of translation. In spite of being more than 20 years old, the KLT algorithm is still widely used as compared with other methods because it operates in a fully automatic way and its performance is competitive in terms of feature point quality [15].

2.4.1 Harris corner

The Harris corner detector is a popular interest point detector due to its strong invariance to: rotation, scale, illumination variation and image noise [16]. The Harris corner detector is based on the local auto-correlation function of a signal, where the local auto-correlation function measures the local changes of the signal with patches shifted by a small amount in different directions. A discrete predecessor of the Harris detector was presented by Moravec, where the discreteness refers to the shifting of the patches [17]

2.4.2 KLT feature tracker

The KLT feature tracker is an approach to feature extraction. it makes the use of spatial intensity information to direct the search for the position that yields the best match. Its algorithm is [18]:

1. Find a good point to track (harris corner)
 - Harris corner points have sufficiently large eigenvalues, so the optical flow equation is solvable.
2. For each Harris corner compute motion (translation or affine) between consecutive frames.
3. Link motion vectors in successive frames to get a track for each Harris point
 - If the patch around the new point differs sufficiently from the old point, we discard

these points.

4. Introduce new Harris points by applying Harris detector at every (10 or 15) frames
5. Track new and old Harris points using steps 2-3

In the following frames from tracking videos, arrows represent the tracking motion of the harris corners.

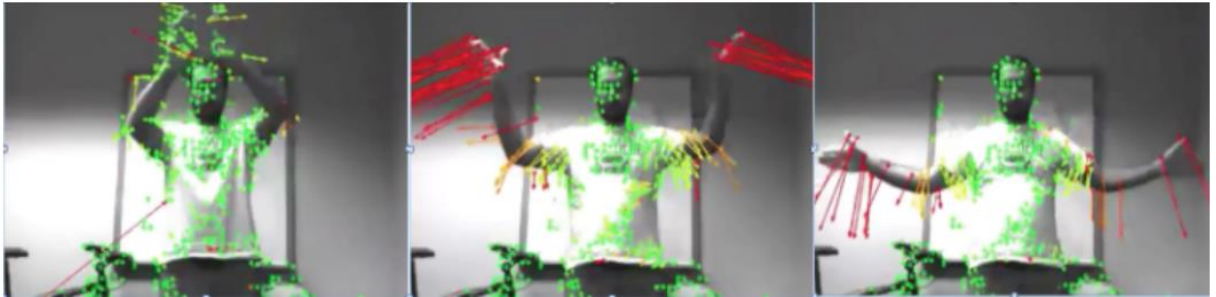


Figure 2.4: Frames from man-tracking video, courtesy of Kanade[18]

2.5 Face Recognition

This section describes the mostly used face recognition algorithms, and it is restricted to the implemented algorithms in this project and is meant to give a detailed description and functioning of it.

2.5.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is one of the most popular appearance-based methods used mainly for dimensionality reduction in compression and recognition problems [19]. Principal component analysis (PCA) also known as Karhunen-Loeve expansion, which has two useful properties when used in face recognition. The first is that it can be used to reduce the dimensionality of the feature vectors. The second useful property is that PCA eliminates all of the statistical covariance in the transformed feature vectors [20].

Sirovich and Kirby first used PCA to efficiently represent pictures of human faces. They showed that any particular face can be (i) economically represented along the eigenpictures coordinate space, and (ii) approximately reconstructed using just a small collection of eigenpictures and their corresponding projections ('coefficients'). Within this context, Turk and Pentland presented the well-known Eigenfaces method for

face recognition in 1991. Since then, PCA has been widely investigated and has become one of the most successful approaches in face recognition [21].

PCA (Eigenfaces method)

PCA based approaches typically include two phases: training and classification[19]

I. Training phase

In the training phase, an eigenspace is established from the training samples using PCA and the training face images are mapped to the eigenspace for classification. The steps involved in this phase are[19]:

Step1: A training set \mathbf{S} of M face images (I_i ; $i = 1, \dots, M$) is first acquired

$$I_i = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1N} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{N1} & \gamma_{N2} & \cdots & \gamma_{NN} \end{bmatrix}_{N \times N} \quad \text{for } i = 1, 2, \dots, M \quad (2.3)$$

Where $[\gamma_{ij}]_{i,j=1,2,\dots,N}$ are the grey values of the pixels.

Step2: Each face image I_i is expressed as a vector Γ_i ($i=1\dots M$) by concatenating each row (or column) into a long thin vector as shown in figure II.2. Since each image is ($N \times N$) matrix so Γ_i is ($N^2 \times 1$) vector.

$$I_i = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1N} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{N1} & \gamma_{N2} & \cdots & \gamma_{NN} \end{bmatrix}_{N \times N} \xrightarrow{\text{concatenation}} \Gamma_i = \begin{bmatrix} \gamma_{11} \\ \gamma_{12} \\ \vdots \\ \gamma_{1N} \\ \gamma_{21} \\ \vdots \\ \gamma_{2N} \\ \gamma_{N1} \\ \gamma_{N2} \\ \vdots \\ \gamma_{NN} \end{bmatrix}_{N^2 \times 1} \quad (2.4)$$

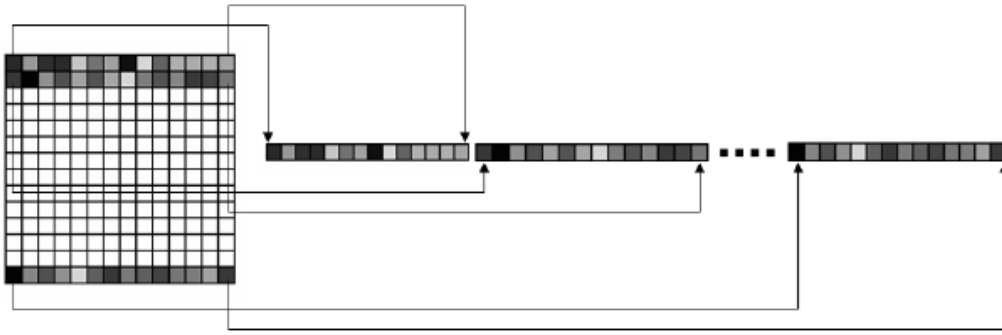


Figure 2.5: Construction of face vector[19].

Step3: Compute the average face vector of the set, which is given by:

$$\Psi = \frac{1}{M} \sum_{i=1}^M \Gamma_i \quad (2.5)$$

$$\Psi = \begin{bmatrix} \psi_{11} \\ \psi_{12} \\ \vdots \\ \psi_{1N} \\ \psi_{21} \\ \vdots \\ \psi_{2N} \\ \psi_{N1} \\ \psi_{N2} \\ \vdots \\ \psi_{NN} \end{bmatrix} = \frac{1}{M} \begin{bmatrix} \gamma_{11}^1 + \gamma_{11}^2 + \dots + \gamma_{11}^M \\ \gamma_{12}^1 + \gamma_{12}^2 + \dots + \gamma_{12}^M \\ \vdots \\ \gamma_{1N}^1 + \gamma_{1N}^2 + \dots + \gamma_{1N}^M \\ \gamma_{21}^1 + \gamma_{21}^2 + \dots + \gamma_{21}^M \\ \vdots \\ \gamma_{2N}^1 + \gamma_{2N}^2 + \dots + \gamma_{2N}^M \\ \gamma_{N1}^1 + \gamma_{N1}^2 + \dots + \gamma_{N1}^M \\ \gamma_{N2}^1 + \gamma_{N2}^2 + \dots + \gamma_{N2}^M \\ \vdots \\ \gamma_{NN}^1 + \gamma_{NN}^2 + \dots + \gamma_{NN}^M \end{bmatrix}_{N^2 \times 1} \quad (2.6)$$

Where M is the number of images in the training set.

Step 4: The images are mean centered by subtracting the mean image from each image vector.

$$\left[\phi_i = \Gamma_i - \Psi \right]_{N^2 \times 1} \quad i = 1, 2, \dots, M \quad (2.7)$$

The purpose of subtracting the mean image from each image vector is to be left with only the distinguishing features from each face and “removing” in way information that is common.

Step 5: Build the mean- subtract face matrix \mathbf{A} .

The mean- subtract faces are arrayed in a matrix \mathbf{A} , with one column per sample image.

$$\mathbf{A} = \begin{bmatrix} \Phi_1 & \Phi_2 & \Phi_3 & \dots & \Phi_M \end{bmatrix}_{N^2 \times M} \quad (2.8)$$

Step 6: This set of very large vectors is then subjected to principal component analysis, which seeks a set of M orthonormal vectors U_n , which best describes the distribution of the data. The k^{th} vector U_k is chosen such that

$$\lambda_k = \frac{1}{M} \sum_{n=1}^M (U_k^T \Phi_n)^2 \quad (2.9)$$

is a maximum, subject to

$$U_l^T U_k = \delta_{kl} = \begin{cases} 1 & \text{if } l = k \\ 0 & \text{otherwise} \end{cases} \quad (2.10)$$

Note: The vectors U_k and scalars λ_k are the eigenvectors and eigenvalues respectively.

Step 7: Getting the eigenvalues and eigenvectors.

The covariance matrix is given as:

$$\mathbf{C} = \frac{1}{M} \sum_{n=1}^M (\Phi_n \Phi_n^T) = \mathbf{A} \mathbf{A}^T \quad (2.11)$$

Where \mathbf{C} is $N^2 \times N^2$

The covariance matrix \mathbf{C} however is $N^2 \times N^2$ real symmetric matrix, and determining the N^2 eigenvectors and eigenvalues is an intractable task for typical image sizes. We need a computationally feasible method to find these eigenvectors.

Consider the eigenvectors V_i of $\mathbf{A}^T \mathbf{A}$ such that

$$\mathbf{A}^T \mathbf{A} V_i = \mu_i V_i \quad (2.12)$$

Premultiplying both sides by \mathbf{A} , we have

$$\mathbf{A} \mathbf{A}^T \mathbf{A} V_i = \mu_i \mathbf{A} V_i \quad (2.13)$$

$$\mathbf{C} \mathbf{A} V_i = \mu_i \mathbf{A} V_i \quad (2.14)$$

$$C U_i = \mu_i U_i \quad (2.15)$$

From which we see that AV_i are the eigenvectors of $C = AA^T$

Thus AA^T and $A^T A$ have the same eigenvalues and their eigenvectors are related as follows:

$$U_i = AV_i \quad (2.16)$$

Following these analysis, we construct the $M \times M$ matrix L

$$L = A^T A \quad (2.17)$$

Where $L_{mn} = \Phi_m^T \Phi_n$.

Step 8: Getting the Eigenfaces.

The M eigenvectors, V_l of L determine linear combinations of the M training set face images to form the eigenfaces U_l

$$U_l = \sum_{k=1}^M V_{lk} \Phi_k \quad l = 1 \dots M \quad (2.18)$$

AV_i needs to be normalized in order to be equal to U_i . Since we only sum up a finite number of image vectors, M , the rank of the covariance matrix cannot exceed $M - 1$ (The -1 come from the subtraction of the mean vector ψ).

Because U_l are the eigenvectors of the covariance matrix corresponding to the original face images, and because they are face-like in appearance, we refer to them as ‘Eigenfaces’.

With this analysis, the calculations are greatly reduced, from the order of the number of pixels in the images (N^2) to the order of the number of images in the training set (M). In practice, the training set of face images will be relatively small ($M \ll N^2$), and the calculations become quite manageable. The associated eigenvalues allow us to rank the eigenvectors according to their usefulness in characterizing the variation among the images.

Step 9: Compute for each face its projection onto the face space.

All the face images are projected into the subspace spanned by the computed M eigenvectors, the projection weight are found using the following formula:

$$W_{ik} = U_k^T(\Gamma_i - \Psi) \quad \text{for } k = 1, 2, \dots, M \quad i = 1, 2, \dots, M \quad (2.19)$$

$$W_{ik} U_k^T \Phi_i \quad (2.20)$$

Where U_k^T are the eigenvectors.

These weights form the feature vectors

$$\Omega_i^T = [W_{i1} \ W_{i2} \ \dots \ W_{iM}] \quad \text{for } i = 1, 2, \dots, M \quad (2.21)$$

The feature vector describes the contribution of each eigenfaces in representing the input face image, treating the eigenfaces as a basis set for face images.

Step 10: Compute the threshold θ .

A distance threshold θ , that defines the maximum allowable distance from a face class as well as from the face space, is set up by computing half the largest distance between any two face classes [22]:

$$\theta = \frac{1}{2} \max\{\|d_{ij}\|\} \quad i, j = 1 \dots M \quad (2.22)$$

Where $\|d_{ij}\|$ denotes the distance between training images i and j .

Rebuilding a face image with Eigenfaces.

A face image can be approximately reconstructed (see figure 2.6) by using the mean image and the eigenfaces as :

$$\Gamma'_i = \Psi + \sum_{k=1}^M W_{ik} U_k \quad i = 1 \dots M \quad (2.23)$$

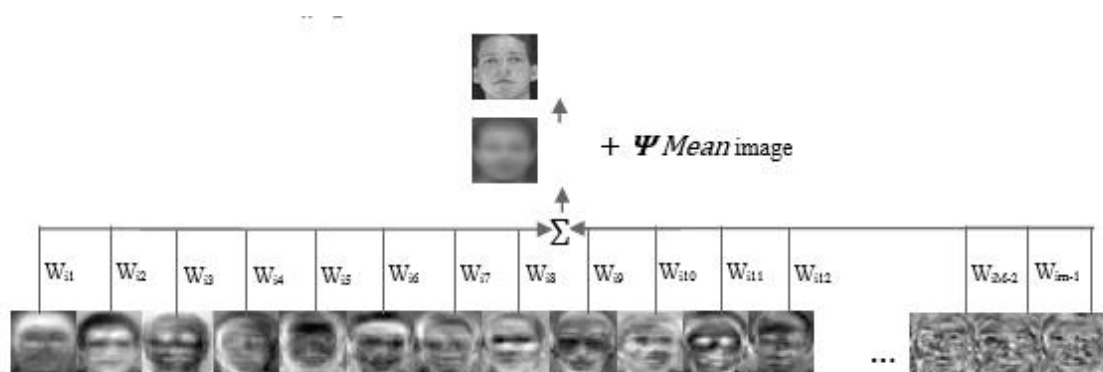


Figure 2.6: Reconstruction of face image.

II Classification phase

In the classification phase, an unknown face is presented to the system. The system projects it onto the same eigenspace and computes its distance from all the stored faces. The face is identified as being the same individual as the face which is nearest to it in face space. The steps involved in this phase are:

Step 1: The test face image to be recognized I_{test} is expressed as a vector Γ_{test} by concatenating each row (or column) into a long thin vector.

$$\Gamma_{test} = \begin{bmatrix} \gamma_{11} \\ \gamma_{12} \\ \vdots \\ \gamma_{1N} \\ \gamma_{21} \\ \vdots \\ \gamma_{2N} \\ \gamma_{N1} \\ \gamma_{N2} \\ \vdots \\ \gamma_{NN} \end{bmatrix}_{N^2 \times 1} \quad (2.24)$$

Step 2: Subtract the average face from it.

$$\Phi_{test} = \Gamma_{test} - \Psi \quad (2.25)$$

Step 3: Compute its projection onto the face space.

The weight of the test image is determined by the following formula:

$$W_k = U_k^T (\Gamma_{test} - \Psi) \quad \text{for } k = 1 \dots M \quad (2.26)$$

The weights form a feature vector,

$$\Omega_{test}^T = [W_1 \ W_2 \ \dots \ W_M] \quad (2.27)$$

Step 4: Computing the distance between the test face and all known faces.

The simplest method for determining, which face class provides the best description of an input face image is to find the face class k that minimizes the distance between the input face and all the stored faces. There are several methods of computing the distance between multidimensional vectors

Ω_{test} is the feature vector of the test face and $(\Omega_{test})_i$ is its i^{th} component.

Ω_k is a vector describing the k^{th} face class and $(\Omega_k)_i$ is its i^{th} component.

2.5.2 Local Binary Patterns Histograms

Local Binary Pattern (LBP) is a texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. This algorithm is a combination of LBP and Histograms Oriented Gradients (HOG), to use this algorithm we need firstly to train the data set with the facial images of the people we want to recognize then applying the LBP operation:

The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the radius and neighbors. The image below shows this procedure[23]:

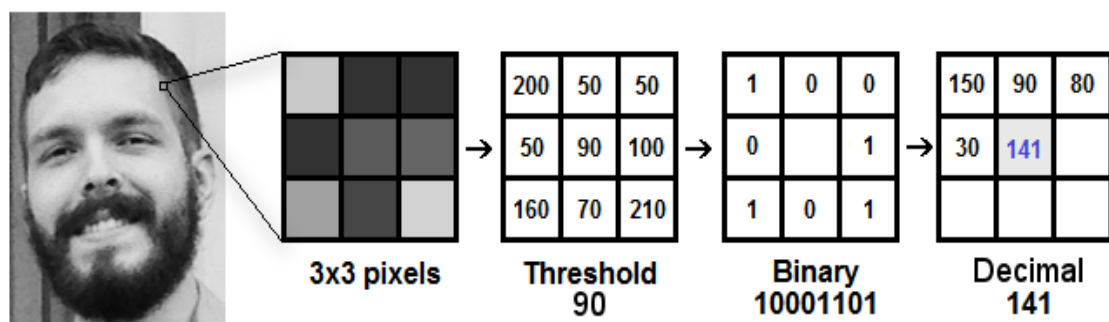


Figure 2.7: express the image in pixels[23]

Based on the image above, the following steps should be done:

- The facial image will be in grayscale.
- We can get part of this image as a window of 3x3 pixels and it can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).

- Taking the central value of the matrix to be used as the threshold, this value will be used to define the new values from the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

The LBP procedure it can be expanded to use a different number of radius and neighbors this method called Circular LBP.

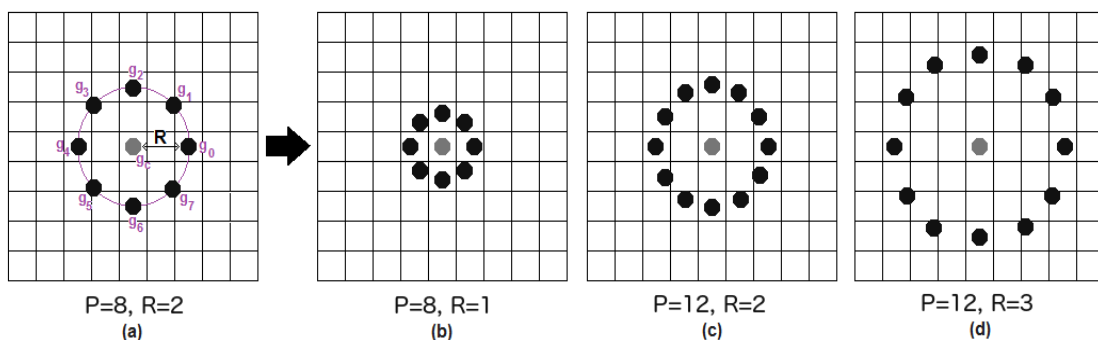


Figure 2.8: Circular LBP[23].

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

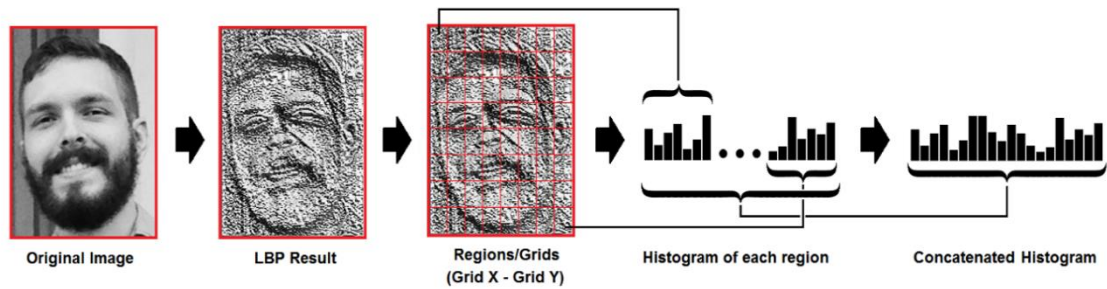


Figure 2.9: express the image on histogram[23].

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

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0- 255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have $8 \times 8 \times 256 = 16.384$ positions in the final histogram. The final histogram represents the characteristics of the image original image.

The algorithm is already trained, each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

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

2.6 K Nearest Neighbor Classifier (KNN)

In machine learning, we need to classify the elements or regress them, and one of the methods used is K-NN algorithm, where K is the number of the nearest elements and NN is a standard from the Nearest Neighbor. We commonly use this algorithm because of its easy of interpretation and low time complexity, the performance of the KNN is highly related to value of k, the number of elements and their topological distribution over the feature space. As shown in figure 2.10 we have blue, red and green elements distributed randomly and the aim is to know to where the green element belongs and this it will be depends to K. So if $K=1$ means only one nearest neighbor and in our case it will

be a blue element so the green element will belong to the blue, and if $K=7$ the majority nearest neighbor will be the red elements so the green element will belong to red.

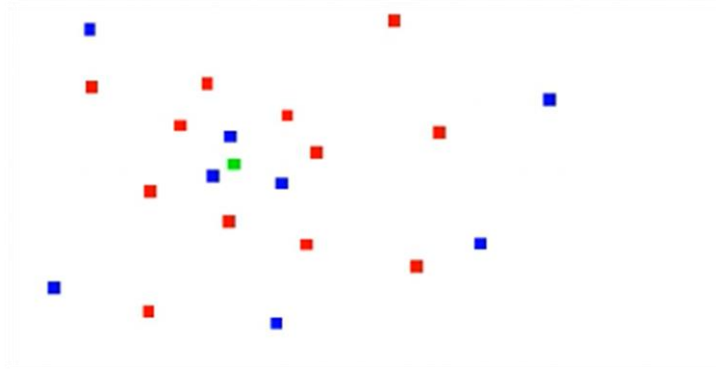


Figure 2.10: random different elements

To decide which element is near, a distance calculation will be done based on one of the following equations, where the first one is the city block distance, the second is Euclidean distance and the third one is cosine distance.

$$d_1(x, y) = \sum_{i=1}^N |x_i - y_i| \quad (2.28)$$

$$d_2(x, y) = \sqrt{\sum_{i=1}^N |x_i - y_i|^2} \quad (2.29)$$

$$d_{cos}(x, y) = 1 - \frac{x \cdot y}{|x| \cdot |y|} \quad (2.30)$$

2.7 Naïve Bayes Classifier

A Naïve Bayes classifier is a simple probabilistic classifier based on applying Bayes theorem with strong (naive) independence assumptions, this algorithm is used because of its real time application, multi class prediction, and fast and easy to use, it can be described by the following equation:

$$P(Y|X) = \frac{P(Y) \prod_{i=1}^d P(X_i|Y)}{P(X)} \quad (2.31)$$

$P(Y|X)$ is the posterior probability of class (Y, target) given predictor (X, attributes).

$P(Y)$ is the prior probability of class.

$P(X|Y)$ is the likelihood which is the probability of predictor given class.

$P(X)$ is the prior probability of predictor.

2.8 Summary

Throughout this chapter, we presented PCA as a global method used for the dimensionality reduction and a local method LBP we explained the used algorithms for each component of a face recognition system Detection Tracking and Recognition. We also presented two classification methods that we used in this project which are K-NN and Bayes Naïve.

Chapter 3:

Implementation and Results

3.1 Introduction

Different sets of experiments were conducted based on the study that was performed to understand the latest algorithms in the field. Matlab platform is used for the implementation of the applications and evaluation of some algorithms that are described in the following sections and an implementation of a real time face recognition system using python as shown in the last part of this chapter.

3.2 Part 1

In this part, we build a real-time face recognition system using Matlab platform with vision toolbox. In order to make the system work in real time, we need to have three components, which are face detection, face tracking, and lastly face recognition. Those components will be described in detail and how we proceeded building the three component showing some important code details and the used algorithms in each component and we discuss the performance and accuracy of the application at each level.

3.2.1 Implementation methodology

3.2.1.a Face detection

To detect people's face using Viola-Jones algorithm, we used vision Cascade Object Detector System object in vision toolbox of Matlab2017a. First, we set the acquisition to be a video input and we chose 640X480 as our input source from the web camera to save memory and processing time.

```
videoFileReader = imaq.VideoDevice('winvideo', 1, 'MJPG_640x480','ROI',[1 1 640  
480]);
```

We call the function of cascade detector object, for the cascading Object detector to work:

```
faceDetector = vision.CascadeObjectDetector();
```

The next step, we need to see if there is any object being detected in our camera by

```
calling bbox= step(faceDetector, videoFrame);
```

We make the machine to check every frame if it did not detect anything since we are working in a real time analysis by calling the following code:

```
while(size(bbox,1)<1)
    videoFrame= step(videoFileReader);
    bbox= step(faceDetector, videoFrame);
end
```

To get input from cam we step the video frame and check the size of the retuning box

```
x = bbox(1, 1); y = bbox(1, 2); w = bbox(1, 3); h = bbox(1, 4);
bboxPolygon = [x, y, x+w, y, x+w, y+h, x, y+h];
```

`bboxPolygon(1:2:end)` represents the x values and `bboxPolygon(2:2:end)` represents the y values

3.2.1.b Face tracking

Using KLT algorithm to do face tracking and eigenvalue algorithm to find corner points. Basically, this is Shi–Tomasi corner detection algorithm that detects the corner. It directly computes the value of eigenvalues to determine whether it is a point of interest or not. In real-time, when all corner points are gone, we run again the corner detection algorithm.

To detect the points inside of the face region, we use the following code:

```
points =detectMinEigenFeatures(rgb2gray(videoFrame), 'ROI', bbox);
```

Then we used PointTracker object in Matlab to track all the points. In addition, we let the face detection algorithm to run when the detected and transformed points decrease its number to 1. In order to track the points, we initialize a point tracker object with max bidirectional error to reduce noise impact.

```
pointTracker = vision.PointTracker('MaxBidirectionalError', 2);
```

We track the detected features by stepping through video frame after we initialize the point tracker with the points locatios.

```
[points, isFound] = step(pointTracker, videoFrame);
```

Next, we compare the old points with the new points found. We can get the old points which are still exist in the next frame from

```
oldInliers = oldPoints(isFound, :);
```

And the new locations of the points by:

```
visiblePoints = points(isFound, :);
```

to find the geomatrix transformation relation we use :

```
[xform, oldInliers, visiblePoints] = estimateGeometricTransform(... oldInliers,
visiblePoints, 'similarity', 'MaxDistance',4);
```

To figure out the geometric transformation, we used maxdistance =4 to identify the transformation. As we got the transformation of the points, we use the polygon to track our new location using:

```
[bboxPolygon(1:2:end),bboxPolygon(2:2:end)]..=transformPointsForward(xform,bboxPol
ygon(1:2:end)bboxpolygon(2:2:end));
```

To get the transformed polygon position using the geometrix transformation we compare the old inliner locations and the new point locations. We make the algorithm to continue recognizing the face after the first set of points are gone. So, we set up the way that if there's no points left on the image, we initialize everything using the next video frame.

If there is no face in the next frame, we wait until there is some faces for us to detect and track.

```
release(pointTracker);
```

and after resetting everything up we wait until we find a face

```
while(size(bbox,1)<1)
    videoFrame = step(videoFileReader);
    bbox= step(faceDetector, videoFrame);
    step(videoPlayer, videoFrame);
```


3.2.1.c Face Recognition

We used PCA to find a match in the database that we have created, after detecting the face the program captures 10 images and store them during the training, the original data will be in the form of the eigenvectors we found from the correlation matrix. After inputting an image, we will measure the difference between the eigenvectors in input images and the original images in the dataset, and then we need to determine which picture has the least difference to identify the input image.

The general eigenvector calculation requires that we subtract the mean of our database.

```
m=uint8(mean(v,2)); %m is the mean of all images, v is the database.
```

```
vzm=v-uint8(single(m)*single(O)); % vzm is v with the mean removed.
```

After subtracting the mean from our database set, we will need to find the correlation matrix, and find the eigenvectors to it.

```
L=single(vzm)'*single(vzm);
```

```
[V,D]=eig(L);
```

```
V=single(vzm)*V;
```

```
V=V(:,end:-1:end-(N-1)); %N =10 which means we are choosing the 10 largest eigenvectors.
```

V gives us the eigenvectors for L, whereas L is the correlation matrix, ordered by its eigenvalues in D from low to high. Single is just converting data to single precision. So, now we got the largest 10 eigenvectors for our whole database. We want to find the signature for each of our database images. We can multiply the eigenvectors by our mean subtracted image to get its signature.

```
cv(i,:)=single(vzm(:,i))'*V;
```

Where i is the value of different picture and vzm is the database with mean subtracted. We want to do the same thing for our input image to get its signature, then compare it with each images in the database.

```
p=r-m; s=single(p)'*V;
```

Here r is the input image, m is the main from our database, and V is the eigenvector of our database. We found s to be the signature of the input image. Then the only thing left to do is comparing the input image signature with what we already have in our database.

$$z=[z,norm(cv(i,:)-s,2)];$$

Where z is the matrix containing all differences between input image and the image in database. Later on, we will be able to find the minimal value of z to get the corresponding database image back, where i is the index of the matched database image.

$$[a,i]=min(z);$$

3.2.2 Results and discussion

3.2.2.a Face detection

Bu using vision library in matlab tool, our detection application has a very high accurarcy for a single and multitarget we estimate the detection rate to be greater than 80% under normal condition and the results of the detection are shown in figure 3.1 for single target.

And in figure 3.2 for multi targets, however there exist some false positives when exposed to strong light and that's shown in figure 3.3

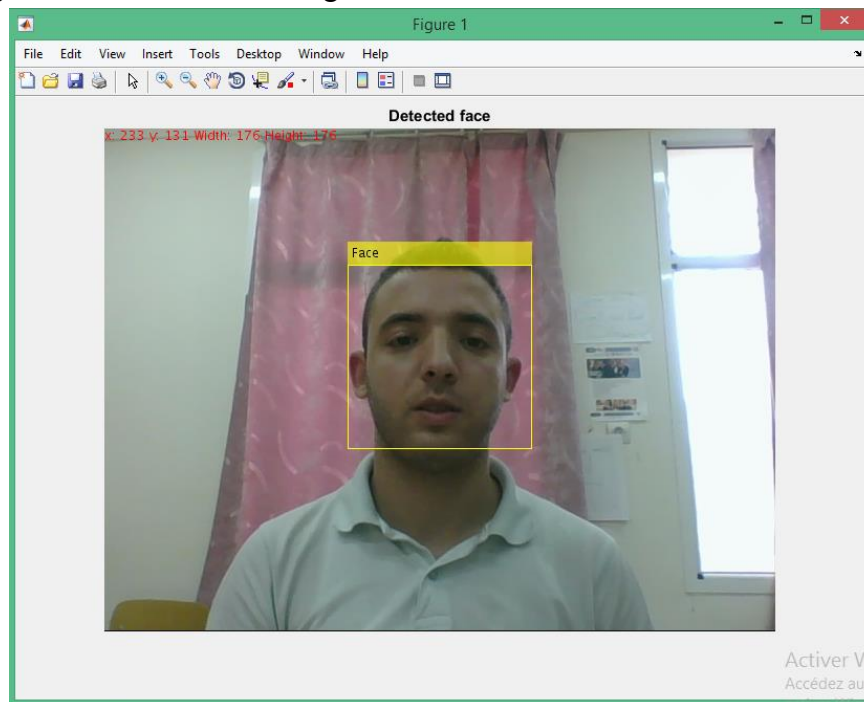


Figure 3.1: Face detection for a single target

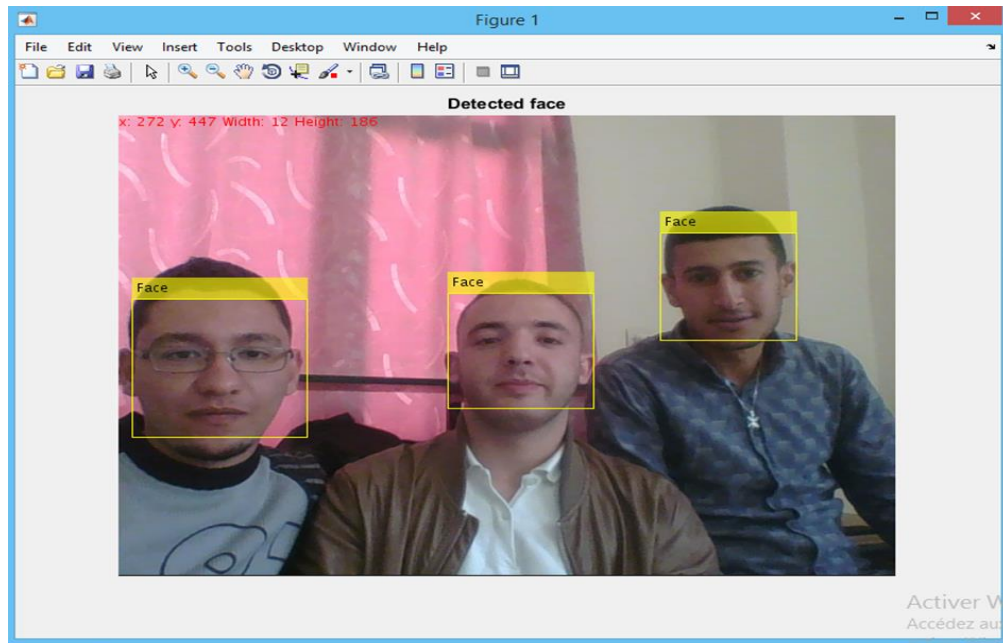


Figure 3.2 : multiple target face detection

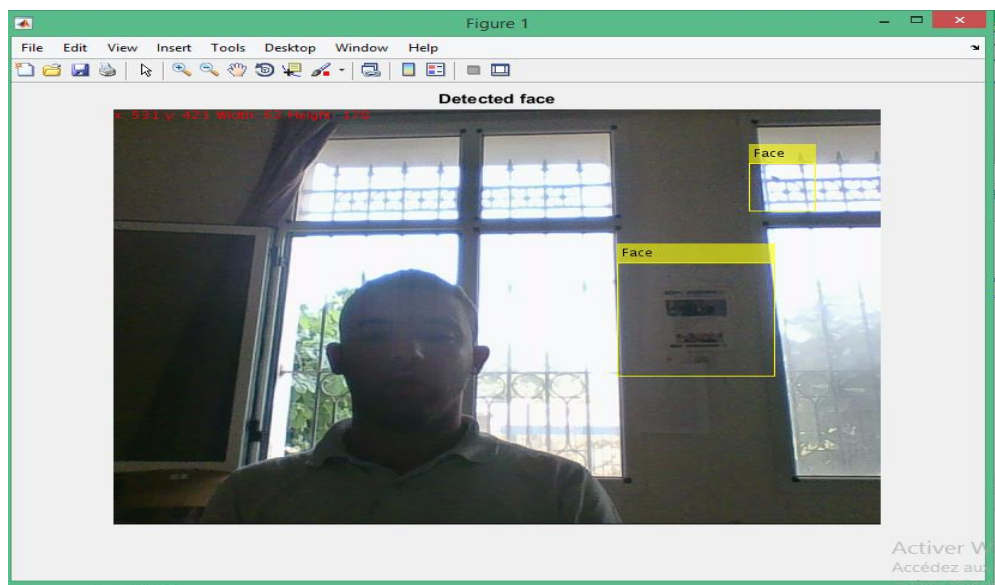


Figure 3.3: Face detection application with false positives

This last figure shows that our program detects faces that doesn't exist and that's a false positive detection, we qualify the detection in our application as performant even though this error, cause its exposed to extrem lighting as we clearly see it in the background of the image.

3.2.2.b Face tracking

On a single target the tracking tool of the system show high accuracy and performance. While rotating the face, however some corner points can be missing, and that the polygon window will continue move with our movement when we prevented face detection from running in every frame. The results show that we need to run the detection when losing the target, and this speeds up our face recognition system.

To stop the face tracking program, we just make a false positive face detection, and then the tracking algorithm fails because the face detection failed.

The system lose track of a face under extreme light circumstances like strong light or very dim light. But still, comparing this with human tracking system, we would not be able to track a person under such circumstances.

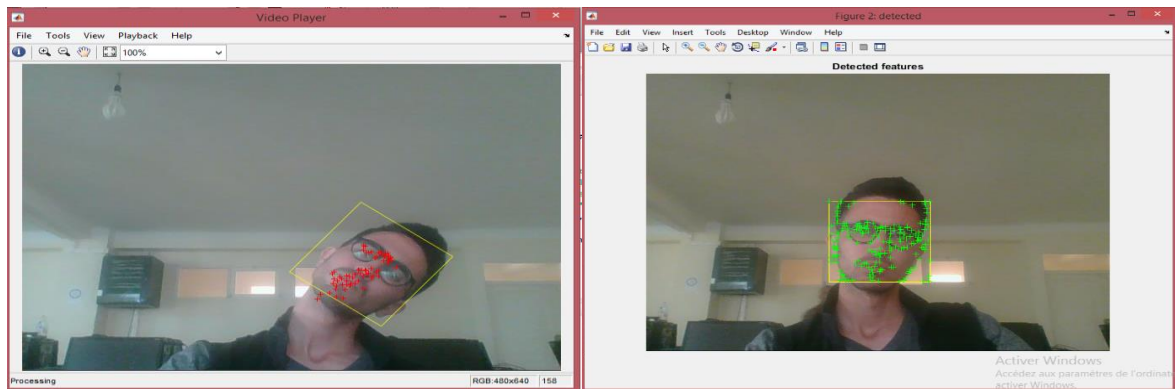


Figure 3.4: Face tracking application with the target's head leaning to the right

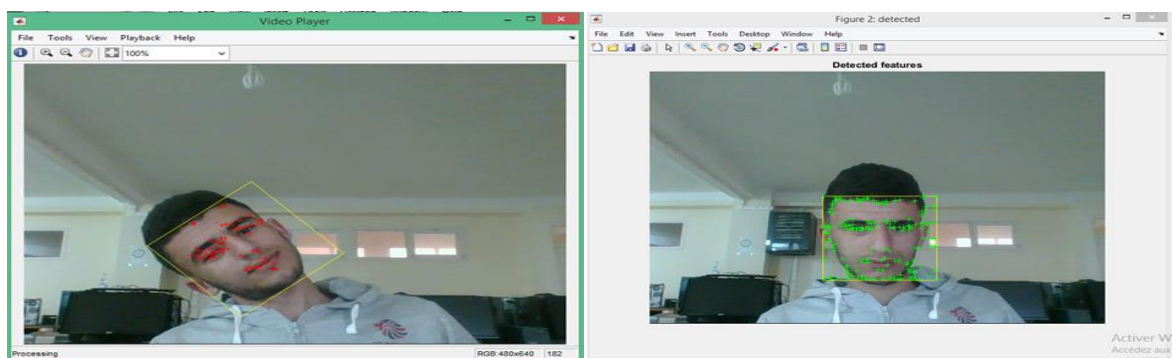


Figure 3.5: Face tracking application with the target's head leaning to the left

We need the tracking system to work on multiple targets, reduce the run time of the program, and identify the person's movements in the future.

3.2.2.c Face recognition

As shown on figure 3.6 the recognition is positive and the person is detected and identified.

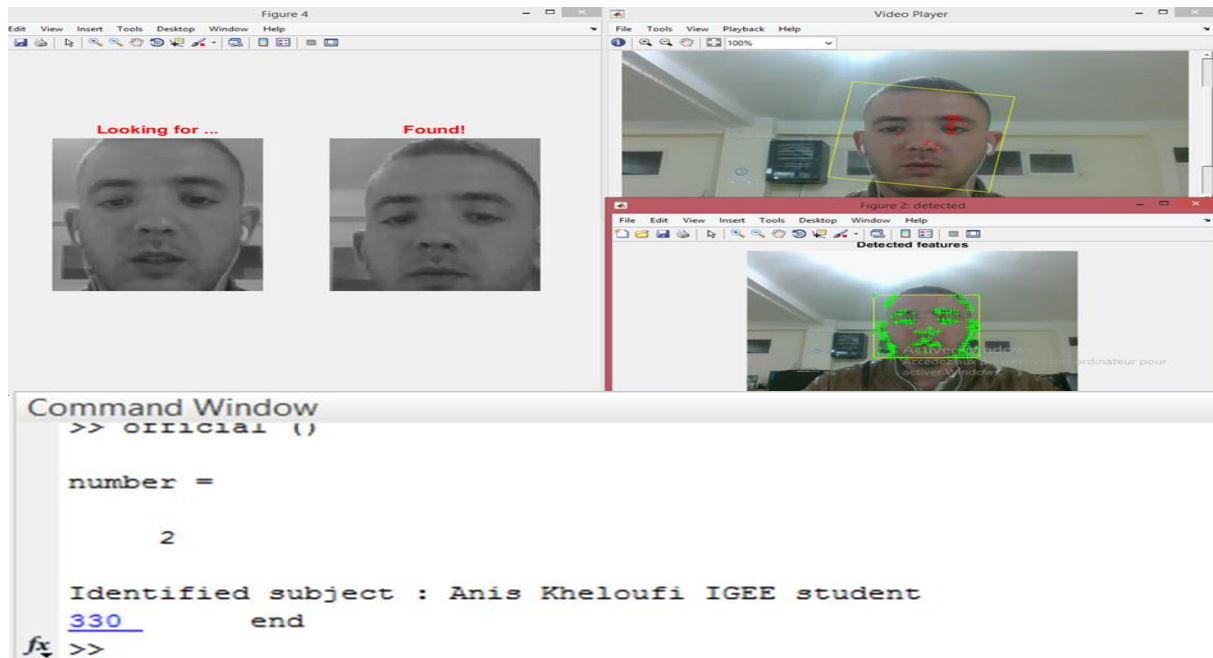


Figure 3.6: Face recognition application in Matlab.

The Recognition rate of our application is around 80%, in some cases the system failed in recognizing females with hair covering part of their face and that is understandable, since even for human beings it is a difficult task to recognize an individual with part of its face covered.

Another limitation of our application is extrem lighting, as we saw in the first part of our application that under extrem lightning the detection tool deteriorates and results if false positive detection which make the application fail at the next levels since we have no true detected face for tracking and recognition.

3.3 Part 2

In order to test the algorithms and enhance the performance of the recognition tool built in the first part, we implemented a combination of PCA algorithm with K-NN and Bayes classifiers. Where PCA Algorithm is used for dimensionality reduction, parameters were tried out on the three data sets (datatest.mat; posetest.mat and illumination.mat) obtained online “<https://www.kairos.com/blog/60-facial-recognition-databases> “and their accuracy results are reported bellow.

3.3.1 Database description

Datatest.mat: 200 subject three faces per subject size: 24x21

Posetest.mat: 68 subjects 13 images per subject (13 different poses) size: 48 x 40

Illumination.mat: illumination.mat 68 subjects 21 images per subject (21 different illuminations) size: 48x40

3.3.2 Results

3.3.2.a Experiment with Datatest.mat

✓ PCA with bayes

The following table gives the classification accuracy for PCA combined by the Bayes Rule for different number of principal components. It is seen that the accuracy does not vary by a large amount when increasing the number of principal components, which means that most of the energy of the signal is contained within the first 25 principal components.

Table 3.1: Accuracy results of PCA with Bayes classifier

Number of PCA	Accuracy
25	40.40%
50	40.40%
100	40.45%
150	40.90%
200	42.95%
300	42.95%

PCA is useful for dimensionality reduction if the size of the training set is too small for the number of dimensions of the data. However, if you are using all of the principal components, PCA will not improve the results of your linear classifier, if your classes were not linearly separable in the original data space.

✓ PCA with K-NN

The following table gives the classification results after PCA is applied to the image vector and by using the K-NN classifier. Results with $k=1, 2$ neighbors are displayed with varying number of principal components. Here we see an increase in the classification accuracy as the number of principal components used in the data representation increase.

Table 3.2: PCA results with K-NN classifier

K	Norm	PCA number	Accuracy
1		25	23.9 %
		100	39.9 %
		200	39.9 %
		300	43.9 %
		400	44.4 %
2	2 nd norm	25	18.4 %
		100	29.4 %
		200	33.4 %
		300	35.4 %
		400	35.4 %

3.3.2. b. Experiments with posetest.mat

(First) 9 images per subject (69% of data) are used for training and (last) 4 for testing (21% of data). In general, we can see that the classification accuracies for every method are more than the previous data set as the number of training/testing samples are more.

Table 3.3: Results of PCA with Bayes and K-NN classifier with posetest dataset (k=1, distance 2nd norms)

Classification Method	Parameters	Accuracy
PCA + Bayes	#Principal components = 300	55.9 %
PCA + kNN	#Principal components = 300	63.7 %

3.3.2. c. Experiments with illumination.mat

(First) 18 images per subject (85% of data) are used for training and (last) three for testing (15% of data).

Table 3.4: Results of PCA with Bayes and K-NN classifier with illumination dataset (k=1, distance 2nd norm)

Classification Method	Parameters	Accuracy
PCA + Bayes	#Principal components = 300	46.7 %
PCA + kNN	#Principal components = 300	39.6 %

3.3.3 Discussion

The Experiment under the (datatest.mat) database Resulted in an accuracy up to 40% for both classifier, the system uses three images per subject, and that is not a considerable number. However for the second dataset that contained images with different poses, gave better results up to 60% for K-NN classifier and 55% for Bayes. Since we are using 69% of the database for training among 884 images for the 68 subject, and for the last database that contained images with different illuminations resulted in 46% accuracy for the Bayes classifier even though 85% of the data is used for training the system and that is due to the variation illumination of the images.

PCA is used to extract Features then feed it to the system for classification, we see that as the amount of data increases the classification accuracy increases, and it is dependent on illuminiations and different poses.

A drawback is that it is sensitive for lighting conditions and the position of the head.

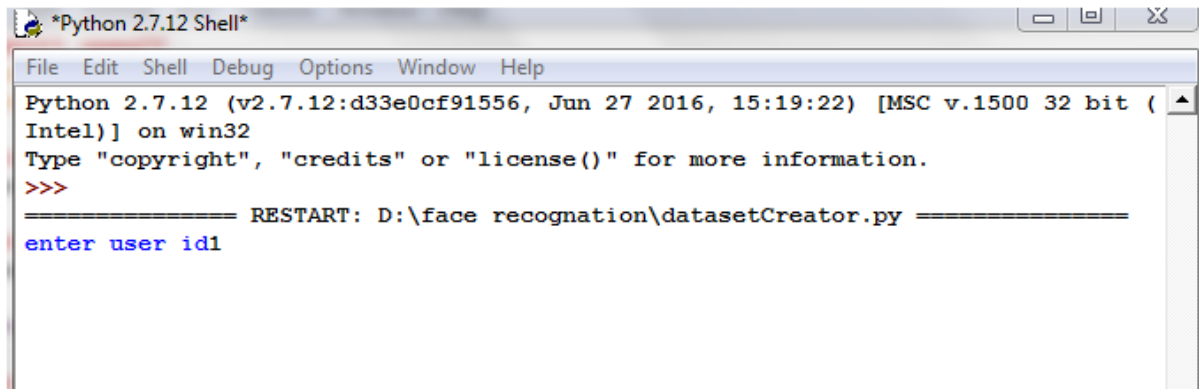
3.4 Part 3

To overcome the lighting limitation of the recognition method built in the previous part we build a real-time face recognition application with LBPH algorithms using python 2.7 and openCV 3.4. The program is constructed in three parts: Dataset Creator, Trainer and Recognizer. Steps of implementation are shown in the coming sections.

3.4.1 Implementation methodology

Dataset creation

We create the dataset by opening IDLE, create a new file and save it in project folder with making sure that we have the “*haarcascade_frontalface_default.xml*” file in the same folder. Then we need to import the library open CV, create a video capture and import the cascade classifier object. Our dataset generator is going to capture few sample faces of one person from the live video frame and assign an ID to it as shown in figure 3.7, and it will save those samples in a folder named “dataset”.



```
*Python 2.7.12 Shell*
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\face recognition\datasetCreator.py =====
enter user id1
```

Figure 3.7: Assigning ID number to the captured face.

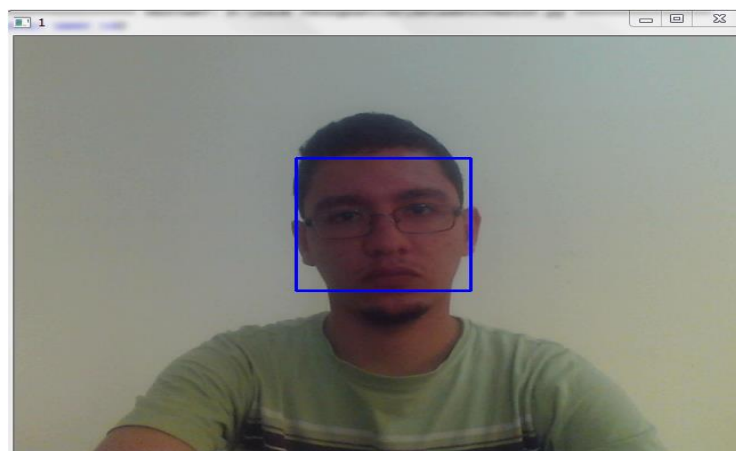


Figure 3.8: Captured face.

After that, 20 samples from the video are taken.



Figure 3.9: The 20 samples of the captured face.

Now we have our dataset so we can train the recognizer to learn the faces from this dataset.

Trainer

First we create a python “trainer.py” file in the same folder where we saved our dataset and then we create a folder in the same directory named “trainer ”, this folder is where we are going to save our recognizer after training.

Nom	Modifié le	Taille	Type
dataSet	29/05/2018 12:56		Dossier de fichiers
trainer	10/05/2018 16:17		Dossier de fichiers
datasetCreator	29/05/2018 12:47	1 Ko	Python File
haarcascade_frontalface_default	25/01/2015 07:31	909 Ko	Document XML
trainer	29/05/2018 13:18	1 Ko	Python File

Figure 3.10: How the files should be like.

After running the trainer code a “*trainer.yml*” file will be created inside the trainer folder, we will use this file to actually recognize the faces that we trained.

Nom	Modifié le	Type	Taille
trainingData	29/05/2018 11:44	YAML Document	278 Ko

Figure 3.11: A generated .yml trained file.

Recognizer

Flow-chart of the recognizer tool is shown in the following figure:

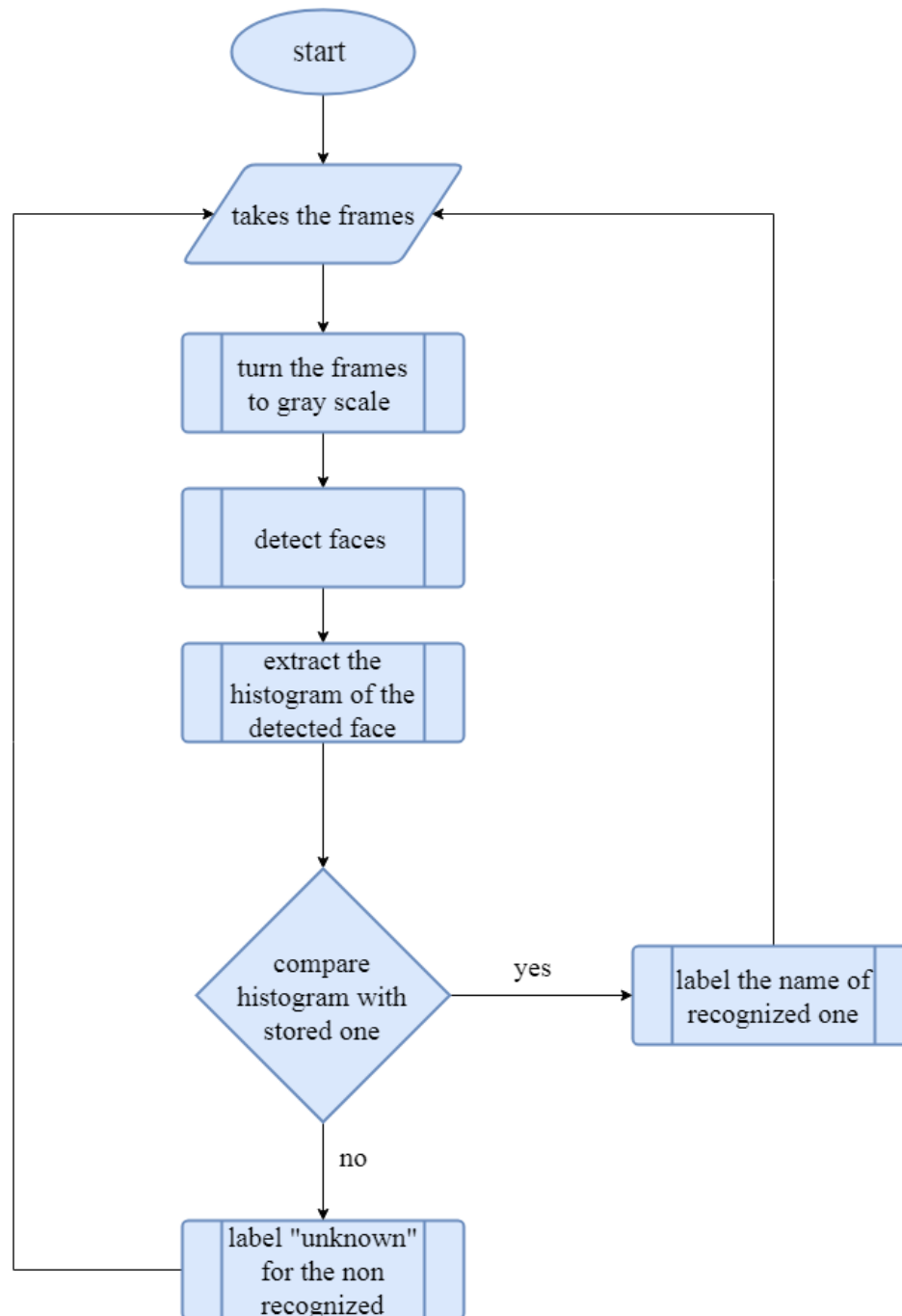


Figure 3.12: Flow chart of recognition using LPBH

3.4.2 Results and discussion

Detection

The detection tool is highly accurate, performant on single and multiple targets, however it is limited for fast moving objects (faces) and generally, that is due to the quality of the acquisition hardware. The figure below shows detection of multi-faces the detection tool is very performant, notice the flash of light in the background of this image and there are no false positives

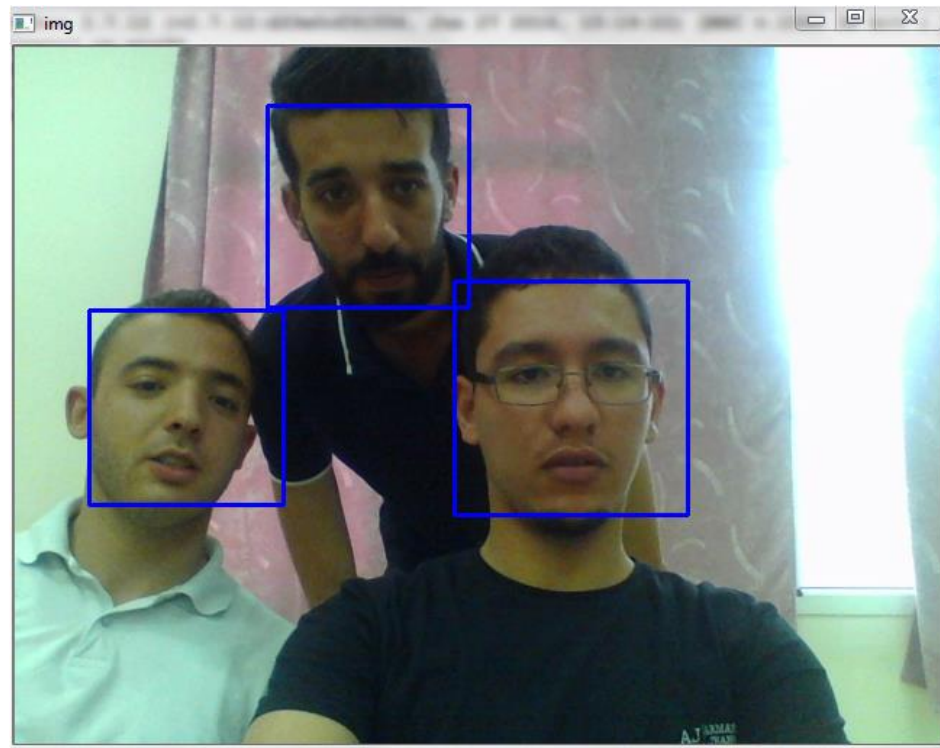


Figure 3.13: Detection of multifaces

Recognition

Figure 3.14 and figure 3.15 show a detected face where the trained subject's frames is instantly recognized and the software displays its name. In addition, the non-trained subject's frames are labeled "UNKNOWN".

For figure 3.16 multifaces are detected for trained and nontrained subject's frames, the software instantly recognize the trained subject while it labels " UNKNOWN" the non recognized subject, in figure 3.17 both subjects were recognized and labeled with their names after detecting their faces

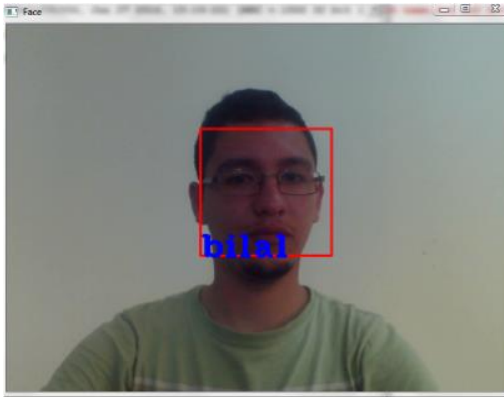


Figure 3.14: Recognized face for trained subject

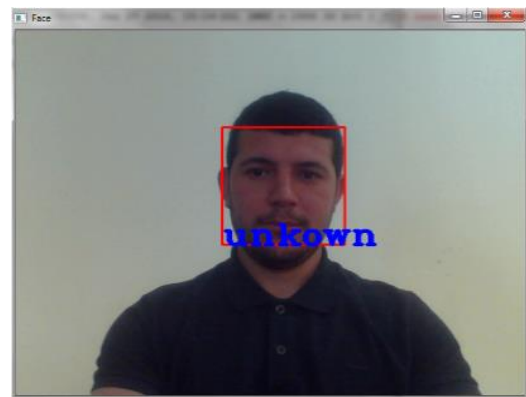


Figure 3.15 : Detected non trained subject

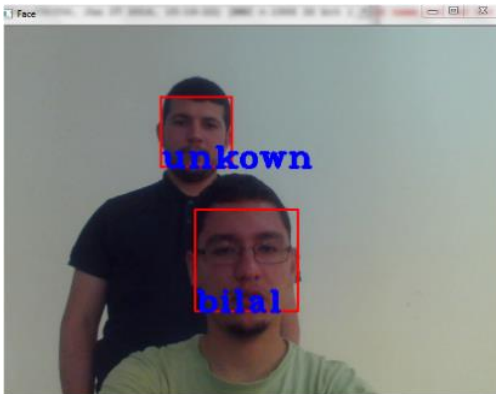


Figure 3.16: Detected subjects trained and non trained

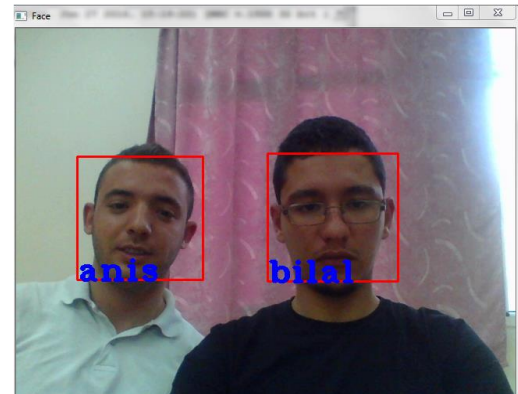


Figure 3.17: Detected and recognized subjects

Discussion

The accuracy of the software mainly depends on the recognition tool, once a subject's face is detected if under a controlled environment accuracy rate is very high. the application fail under extreme dark or absence of illumination, fast moving objects and that's understandable since even for human eye that's considered a hard task, another limitation of the software is subject standing at far distance from acquisition tool and that results in failure of detection . Hardware have a certain effect on accuracy of the software quality of used camera and processor speed and capacity that affects directly the execution time.

3.5 Summary

In this chapter we have realized an implementation of the studied Algorithms in the previous chapter of this project, the system consists of three tools; Detector, Tracker, and a Recognizer, details of the implementation were given with the tool's performance Results.

One of the major limitation of our System at the recognition stage built Using PCA Algorithm was its sensitivity to light. At extreme lighting conditions the recognition Deteriorates considerably.

In a trial to enhance the recognition method, we combined PCA Algorithm with K-NN and Bayes classifier, where PCA was used for dimensionality reduction. While K-NN and Bayes Algorithms are used for classification and recognition, results of this experiment are reported above, main points to consider from the experiment results is that when PCA is combined with Bayes, most of the energy signal is contained within 25 first Eigenfaces. Second when combining PCA with K-NN the recognition method performs better for $k=1$, and accuracy increase by increasing trained Data. Testing the Algorithm on the three different Datasets highlights the major limitation that is lighting.

To overcome the problem we implemented in the last part a recognition tool using a local based algorithm that performs better in lighting conditions and results were reported above.

General Conclusion

The aim of this project was to implement a real time face recognition system, following the processing schema; first step we started by implementing a face detection tool using Viola Jones Algorithm. In the second step, since we are in real time processing, we implemented a face tracker using KLT tracking algorithm. In the last step, we built a recognizer using PCA algorithm for dimensionality reduction and classification. After testing and evaluating our application we proposed to enhance the recognition method by combining PCA with K-NN and Bayes classifiers. The major drawback of our application was lighting conditions that deteriorates the recognition performance, to overcome the problem we used a local based Algorithm LBPH using python.

Researchers worldwide continue in implementing innovative and novel works concerning face detection and recognition techniques to achieve a better performance based on the traditional methods such as eigenface, wavelets and neural networks. These algorithms or methods have been integrated into mobile platform using mobile devices such as computers, PDAs and smart phones[24], our main contribution in this project was implementing a real time face recognition application that can be improved and implemented on mobile platforms.

As a further work, we will try to enhance the performance of the recognition method by using other algorithms and techniques for dimensionality reduction and classification, as we will make the application mobile and integrated to an embedded system that can be used for authentication and security.

References:

- [1] X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang. Face recognition using Laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005
- [2] Neel Ramakant Borkar Sonia Kuwelkar, e IEEE 2017 International Conference on Computing Methodologies and Communication, Real-Time Implementation Of Face Recognition System
- [3] JAIN, Anil K. et LI, Stan Z. *Handbook of face recognition*. New York : springer,2011.
- [4]http://personal.ee.surrey.ac.uk/Personal/T.Windeatt/msc_projects/tambasis/WEB/c2.htm
- [5] Kavita , Ms. Manjeet Kaur A Survey paper for Face Recognition Technologies *International Journal of Scientific and Research Publications*, Volume 6, Issue 7, July 2016
- [6] W. Zhao, R. Chellappa, A. Rosenfeld, and P. Phillips. Face recognition: A literature survey. *ACM Computing Surveys*, pages 399–458, 2003.
- [7] S. Bentin, T. Allison, A. Puce, E. Perez, and G. McCarthy. Electrophysiological studies of face perception in humans. *Journal of Cognitive Neuroscience*, 8(6):551–565, 1996.
- [8] R. Diamond and S. Carey. Why faces are and are not special. an effect of expertise. *Journal of Experimental Psychology: General*, 115(2):107–117, 1986.
- [9] Manuel Grana , Ion Marquez University of the Basque Country EHU Face recognition algorithms 2010
- [10] Jiliang Tang, Salem Alelyani and Huan Liu Feature Selection for Classification: A Review
- [11] Shraddha Arya, Arpit Agrawal, Devi Ahilya University Face Recognition with Partial Face Recognition and Convolutional Neural Network (IJARCET) , January 2018
- [12] Achim Sanjay Manikarnika, Norwegian University of Science and Technology, A General Face Recognition System 2006
- [13] Nicolas Delbiaggio ,Haaga-Helia University of applied science, A comparison of facial recognition’s algorithms 2017
- [14] Bruce D. Lucas and Takeo Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision. *International Joint Conference on Artificial Intelligence*, pages 674-679

- [15] Neha S. Sakure, Rushikesh T. Bankar, Suresh S. Salankar, **CAMPARATIVE ANALYSIS OF FACE TRACKING (IJARECE)** , March 2016
- [16] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, June 2000
- [17] Konstantinos G. Derpanis, university of York Canada **The Harris Corner Detector** 2004
- [18] Khaled Jedoui, Jamie Xie, Michelle Guo, Nikhil Cheerla, epartment of Computer ScienceStanford UniversityStanford, CA 94305
- [19] Dalila CHERIFI, Nadjet RADJI, Amine NAIT-ALI **Effect of Noise, Blur and Motion on Global AppearanceFace Recognition based Methods performance**, *International Journal of Computer Application* 2011
- [20] Sasan Karamizadeh , Shahidan M. Abdullah , Azizah A. Manaf , Mazdak Zaman, **An Overview of Principal Component Analysis** *Journal of Signal and Information Processing*, 2013
- [21] A. Pentland, “Looking at People: Sensing for Ubiquitous and Wearable Computing,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 107-119, Jan. 2000
- [22] K. Kalaiarasi, Aliza Tajudin, R. K. Subramanian* **School of Computer Science, University of Science Malaysia A NEAR REAL-TIME FACE RECOGNITION SYSTEM BASED ON EIGENFACES**
- [23] <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>
- [24] Nurulhuda Ismail and Mas Idayu Md. Sabri **World Academy of Science, (IJCIE)Vol:4, No:9, 2010 Mobile to Server Face Recognition: A System Overview.**