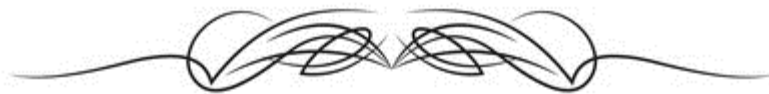


CHAPTER 1

INTRODUCTION



INTRODUCTION

As the necessity for higher levels of security rises, technology is bound to swell to fulfill these needs. Any new creation, enterprise, or development should be uncomplicated and acceptable for end users in order to spread worldwide. This strong demand for user-friendly systems which can secure our assets and protect our privacy without losing our identity in a sea of numbers, grabbed the attention and studies of scientists toward what's called biometrics.

Biometrics is the emerging area of bioengineering; it is the automated method of recognizing person based on a physiological or behavioral characteristic. There exist several biometric systems such as signature, finger prints, voice, iris, retina, hand geometry, ear geometry, and face. Among these systems, facial recognition appears to be one of the most universal, collectable, and accessible systems.

Biometric-based techniques have emerged as the most promising option for recognizing individuals in recent years since, instead of authenticating people and granting them access to physical and virtual domains based on passwords, PINs, smart cards, plastic cards, tokens, keys and so forth, these methods examine an individual's physiological and/or behavioral characteristics in order to determine and/or ascertain his/her identity. Passwords and PINs are hard to remember and can be stolen or guessed; cards, tokens, keys and the like can be misplaced, forgotten or duplicated; magnetic cards can become corrupted and unreadable. However, an individual's biological traits cannot be misplaced, forgotten, stolen or forged.



Figure 1.1: Example of Face recognition in video

1.1. Background

Automated face recognition is a relatively new concept. Developed in the 1960s, the first semi-automated system for face recognition required the administrator to locate features (such as eyes, ears, nose, and mouth) on the photographs before it calculated distances and ratios to a common reference point, which were then compared to reference data. In the 1970s, Goldstein, Harmon, and Lesk used 21 specific subjective markers such as hair color and lip thickness to automate the recognition. The problem with both of these early solutions was that the measurements and locations were manually computed.

In 1988, Kirby and Sirovich applied principle component analysis, a standard linear algebra technique, to the face recognition problem. This was considered somewhat of a milestone as it showed that less than one hundred values were required to accurately code a suitably aligned and normalized face image.

In 1991, Turk and Pentland discovered that while using the eigenfaces techniques [7], the residual error could be used to detect faces in images; a discovery that enabled reliable real-time automated face recognition systems. Although the approach was somewhat constrained by the environmental factors, the nonetheless created significant interest in furthering development of automated face recognition technologies. The technology first captured the public's attention from the media reaction to a trial implementation at the January 2001 Super Bowl, which captured surveillance images and compared them to a database of digital mugshots. This demonstration initiated much-needed analysis on how to use the technology to support national needs while being considerate of the public's social and privacy concerns.

Today, face recognition technology is being used to combat passport fraud, support law enforcement, identify missing children, and minimize benefit/identity fraud.

1.2. Proposed system

Recent years have seen significant progress in this area owing to advances in face modeling. Though machine recognition of faces has reached a certain level of maturity after more than 35 years of research, e.g., in terms of the number of subjects to be recognized, technological challenges still exist in many aspects. 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. So far, a robust face recognition algorithm that works successfully in all the above mentioned varying environments has not yet been designed. As a result, the problem of face recognition still remains attractive to many researchers across multiple disciplines.

We wish to propose a system that analyzes various existing algorithms so as to perform a comparative study on the basis of input size, performance and the various challenges that they overcome and so on. We wish to achieve a better accuracy than the existing methods.

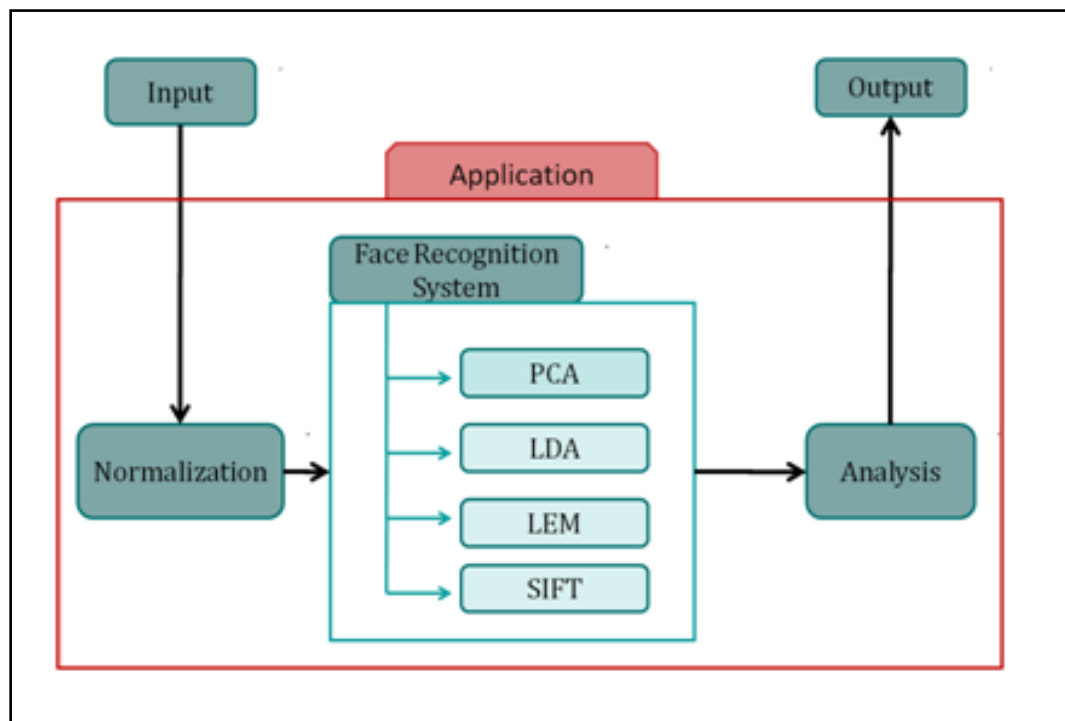


Figure 1.2: Architecture of Proposed System

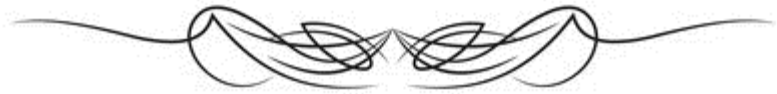
The proposed system must (a) Implement the existing face recognition algorithms and (b) runs a few tests on these algorithms to analyze their performance and accuracy under varying conditions such as, occlusion, illumination, expression changes etc.

As the project was for a short duration of 4 months, we have chosen four existing face recognition algorithms for analysis purposes. Out of these four algorithms, two are feature-based approaches (i.e. SIFT and LEM) while the other two are holistic approaches (i.e. PCA and LDA). To help understand the functionalities of the proposed system, its architecture is shown in the figure 1.1.

1.3. Objective

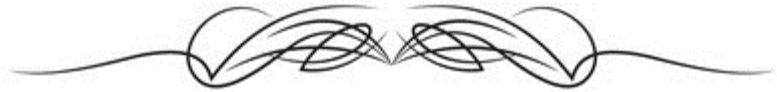
The main objectives of our project are:

- Survey of existing face recognition algorithms based on the key challenges that they address.
- Implementation of a few existing face recognition algorithms and their analysis.



CHAPTER 2

LITERATURE SURVEY



LITERATURE SURVEY

2.1. What is face recognition?

Face recognition is a task of identification of an already detected face as a known face or an unknown face. 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:

- Face Verification (or authentication), and
- 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 compare a query face image against all the template images in the database to determine the identity of the query face. Hence, given still or video images of a scene, face recognition identifies or verifies one or more persons in the scene using a stored database of faces.

2.2. Why face recognition?

Biometric-based technologies include identification based on physiological characteristics (such as face, fingerprints, finger geometry, hand geometry, hand veins, palm, iris, retina, ear and voice) and behavioral traits (such as gait, signature and keystroke dynamics). Face recognition appears to offer several advantages over other biometric methods, a few of which are outlined here: Almost all these technologies require some voluntary action by the user, i.e., the user needs to place his hand on a hand-rest for finger printing or hand geometry detection and has to stand in a fixed position in front of a camera for iris or retina identification. However, face recognition can be done passively without any explicit action or participation on the part of the user since face images can be acquired from a distance by a camera. This is particularly beneficial for security and surveillance purposes.

Furthermore, data acquisition in general is fraught with problems for other biometrics: techniques that rely on hands and fingers can be rendered useless if the epidermis tissue is damaged in some way (i.e., bruised or cracked). Iris and retina identification require expensive equipment and are much too sensitive to any body motion. Voice recognition is susceptible to background noises in public places and auditory fluctuations on a phone line or tape recording. Signatures can be modified or forged. However, facial images can be easily obtained with a couple of inexpensive fixed cameras. Good face recognition algorithms and appropriate preprocessing of the images can compensate for noise and slight variations in orientation, scale and illumination. Finally, technologies that require multiple individuals to use the same equipment to capture their biological characteristics potentially expose the user to the transmission of germs and impurities from other users. However, face recognition is totally non-intrusive and does not carry any such health risks.

2.3. Applications of face recognition

There are numerous application areas in which face recognition can be exploited, a few of which are:

- Security – to restrict access to buildings, airports, ATM machines etc.
- Surveillance – to identify and monitor the movements of known criminals, drug-offenders, thief's etc., via CCTVs.
- Auto attendance system – in colleges or schools to capture the students in a class room and recognize them for attendance purposes.
- Witness faces reconstruction.
- Image database investigations – searching image databases of licensed drivers, benefit recipients, missing children, immigrants and police bookings.
- Video indexing – to label faces in a video.
- General identity verification – used in companies for employee identification and for passports, driving license etc.



Figure 2.1: Criminal identification using MORIS

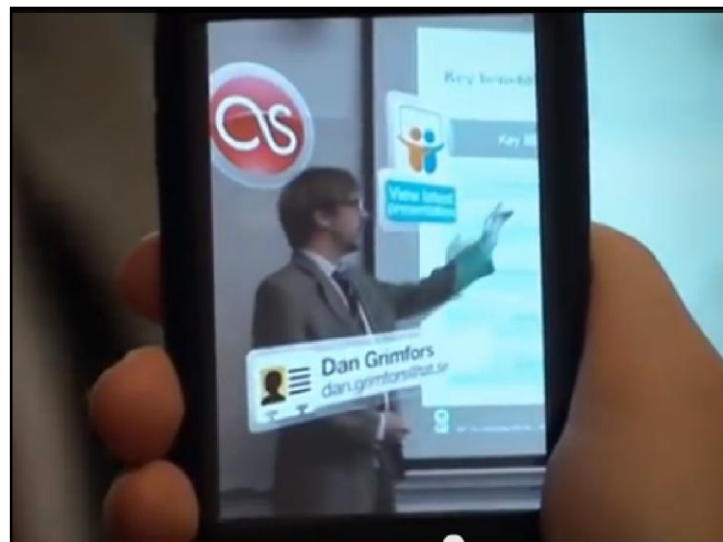


Figure 2.2: Example of video – indexing using face recognition

2.4. Face Recognition System

Facial recognition system is a type of biometric software application that can identify a specific individual in a digital image by analyzing and comparing patterns.

A face recognition system generally consists of four modules:

- *Face detection* segments the face areas from the background of a still or video image.
- *Face alignment* is aimed at achieving more accurate localization and at normalizing faces. Facial components, such as eyes, nose, and mouth and

facial outline, are located; based on the location points, the input face image is normalized with respect to geometrical properties, such as size and pose, using geometrical transforms or morphing.

- *Feature extraction* is performed to provide effective information that is useful for distinguishing between faces of different persons and stable with respect to the geometrical and photometrical variations. It is performed after a face is normalized geometrically and photometrically.
- *Face matching*, where the extracted feature vector of the input face is matched against those of enrolled faces in the database; it outputs the identity of the face when a match is found with sufficient confidence or indicates an unknown face otherwise.

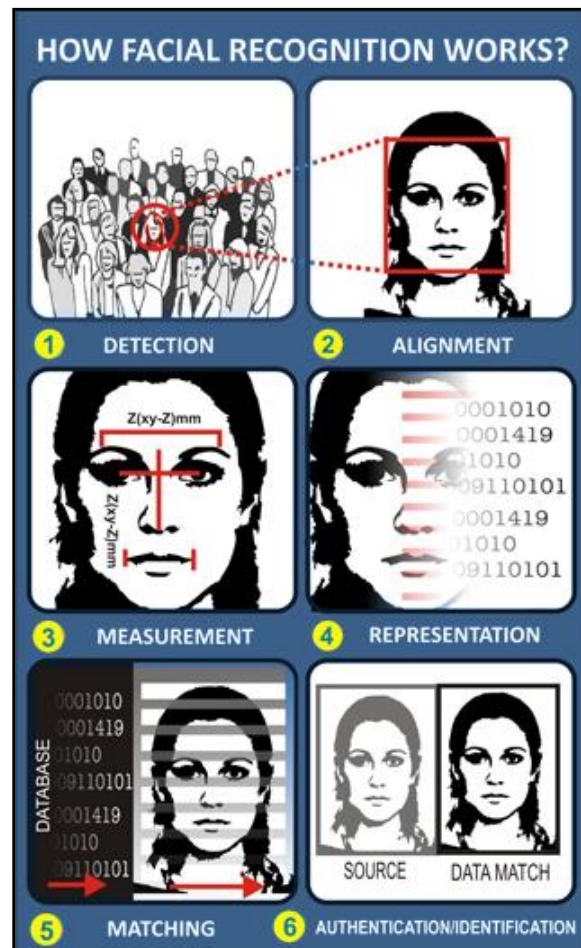


Figure 2.3: Working of a face recognition system

Face recognition is a challenging problem in the field of image analysis and computer vision that has received a great deal of attention over the last few years because of its many applications in various domains. Research has been conducted vigorously in this area for the past four decades or so, and though huge progress has been made, encouraging results have been obtained and current face recognition systems have reached a certain degree of maturity when operating under constrained conditions; however, they are far from achieving the ideal of being able to perform adequately in all the various situations that are commonly encountered by applications utilizing these techniques in practical life. The ultimate goal of Researchers in this area are to enable computers to emulate the human vision system.

2.5. Face Recognition Techniques

The method for acquiring face images depends upon the underlying application. Therefore, depending on the face data acquisition methodology, face recognition techniques can be broadly divided into three categories:

2.2.1. Face Recognition from Intensity Images

Face recognition methods for intensity images fall into two main categories: feature-based and holistic.

- **Feature-based :**

Feature-based approaches first process the input image to identify and extract (and measure) distinctive facial features such as the eyes, mouth, nose, etc., as well as other fiducial marks, and then compute the geometric relationships among those facial points, thus reducing the input facial image to a vector of geometric features. Standard statistical pattern recognition techniques are then employed to match faces using these measurements.

A well-known feature-based approach is the elastic bunch graph matching method. This technique is based on Dynamic Link Structures. A graph for an individual face is generated by choosing a set of fiducial points on the face, which are nodes of a full connected graph, and are labeled with the Gabor filters'

responses. Each arch is labeled with the distance between the correspondent fiducial points. A representative set of such graphs is combined into a stack-like structure, called a *face bunch graph*. Once the system has a face bunch graph, graphs for new face images can then be generated automatically by Elastic Bunch Graph Matching. Recognition of a new face image is performed by comparing its image graph to those of all the known face images and picking the one with the highest similarity value.

The advantages of a feature based system are: (a) robust to position variations in input image, hence it can be made invariant to image size, orientation and illumination, (b) high speed matching. The main disadvantage is that automatic feature detection would require implementer to decide which features are important enough to be extracted.

- **Holistic :**

Holistic approaches attempt to identify faces using global representations, i.e., descriptions based on the entire image rather than on local features of the face. The image is represented as a 2D array of intensity values and recognition is performed by direct correlation comparisons between the input face and all the other faces in the Database. This approach has been shown to work under limited circumstances (i.e., equal illumination, scale, pose, etc.), it is computationally very expensive and suffers from the usual shortcomings such as sensitivity to face orientation, size, variable lighting conditions, background clutter, and noise. The major hindrance to the direct matching methods is that they perform classification in a space of very high dimensionality. To counter this curse of dimensionality, several schemes have been proposed that employ dimensionality reduction methods to obtain and retain the most meaningful feature dimensions before performing recognition. These subspace methods are elaborated in Ref [3, 4]. A few of them are:

- i. **Principal Component Analysis (PCA):** it reduces the dimensionality and effect of noise by representing M face images in the form of K eigenface vectors, derived from the covariance matrix. These K eigenface vectors

contain all the important features required to represent a face image. The input image is also represent as an eigenface vector and a comparison of distances of the vectors would yield the recognized face, if the distance is beyond the threshold otherwise, the face is an unknown face. Ref [20] shows the methodology for using PCA on infrared images.

- ii. ***Linear Discriminant Analysis (LDA)***: it maximizes the ratio of the between-class scatter and the within-class scatter and is thus purportedly better for classification than PCA. The Fisherfaces method, which uses subspace projection prior to LDA projection (to prevent the within-class scatter matrix from becoming degenerate), is better at simultaneously handling variations in lighting and expression.
- iii. ***Independent component analysis (ICA)***: it is a generalization of PCA, is one such method that has been employed for the face recognition task. ICA aims to find an independent, rather than an uncorrelated, image decomposition and representation.

The main advantage of the holistic approaches is that they do not destroy any of the information in the images by concentrating on only limited regions or points of interest. Consequently, these techniques are not only computationally expensive but require a high degree of correlation between the test and training images, and do not perform effectively under large variations in pose, scale and illumination, etc.

2.2.2. Face Recognition from Video Sequences

A video-based face recognition system typically consists of three modules: one for detecting the face; a second one for tracking it; and a third one for recognizing it. Most of these systems choose a few good frames and then apply one of the recognition techniques for intensity images to those frames in order to identify the individual. Image sequences are captured in which the pose of the person's face changes from -90 degrees to 90 degrees. The sequences are projected into eigenspace to give a prototype trajectory for each known individual. During the recognition

phase, an unknown face trajectory is compared with the prototype trajectories to determine the identity of the individual.

Recently, some approaches have employed a video-to-video paradigm in which information from a sequence of frames from a video segment is combined and associated with one individual. This notion involves a temporal analysis of the video sequence and a condensation of the tracking and recognition problems. Such schemes are still a matter of ongoing research since the reported experiments were performed without any real variations in orientation and facial expressions.

Dynamic face recognition schemes appear to be at a disadvantage relative to their static counterparts in general, since they are usually hampered by one or more of the following: low quality images, cluttered backgrounds, the presence of more than one face in the picture and a large amount of data to process, which complicate face detection. Furthermore, the face image may be much smaller than the size required by most systems employed by the recognition modules. However, dynamic schemes do have the following advantages over static techniques: video provides temporal continuity, so classification information from several frames can be combined to improve recognition performance; video allows the tracking of face images such that variations in facial expressions and poses can be compensated for, resulting in improved recognition. Dynamic schemes also have an edge over static ones when it comes to detecting the face in a scene, since these schemes can use motion to segment a moving person's face.

2.2.3. Face Recognition from Other Sensory Inputs

Though the bulk of the research on face recognition has been focused on identifying individuals from 2D intensity images, in recent years some attention has nevertheless been directed towards exploiting other sensing modalities, such as 3D or range data and infra-red imagery, for this purpose.

- **3D Model Based Methods :** 3D information for face recognition appears to be that it allows us to exploit features based on the shape and the curvature of the face (such as the shape of the forehead, jaw line, and cheeks) without being plagued by the variances caused by lighting, orientation and background clutter

that affect 2D systems. One of the earliest of such approaches is where the principle curvatures of the face surface are calculated from range data, after which this data - supplemented by a priori information about the structure of the face - is used to locate the various facial features (i.e., the nose eyes, forehead, neck, chin, etc.). The faces are then normalized to a standard position and re-interpolated onto a regular cylindrical grid. The volume of space between two normalized surfaces is used as a similarity measure.

- **Infra-red:** Since thermal infra-red imagery of faces is relatively insensitive to variations in lighting, such images can hence be used as an option for detecting and recognizing faces. Furthermore, since infra-red facial images reveal the vein and tissue structure of the face which is unique to each individual (like a fingerprint), some of the face recognition techniques for the visible spectrum should therefore yield favorable results when applied to these images. However, there exist a multitude of factors that discourage the exploitation of such images for the face recognition task, among which figure the substantial cost of thermal sensors, the low resolution and high level of noise in the images, the lack of widely available data sets of infra-red images, the fact of infra-red radiation being opaque to glass (making it possible to occlude part of the face by wearing eye glasses) and, last but not least, the fact that infra-red images are sensitive to changes in ambient temperature, wind and metabolic processes in the subject make it difficult to use infra-red based techniques.

Reference [4, 2, 3] presents a detailed information of more number of face recognition techniques.

2.6. Challenges of Face Recognition

The human face is not a unique, rigid object. Indeed, there are numerous factors that cause the appearance of the face to vary. These factors are:

1. **Intrinsic factors:** These factors depend on the physical nature of face and are of the observer. These factors can again be divided into:

- **Intrapersonal factors:** These factors depend on the variance in facial appearance of the same person i.e. due to age, facial expressions, use of glasses etc.
 - **Interpersonal factors:** These factors depend on the variance in facial appearance of different people. This can be due to ethnicity, gender etc.
2. **Extrinsic factors:** These factors depend on the interactions of a person's face with external elements like light, noise etc. Hence these factors dependent on the observer's view of a face. E.g. Illumination, pose, resolution, noise, etc.

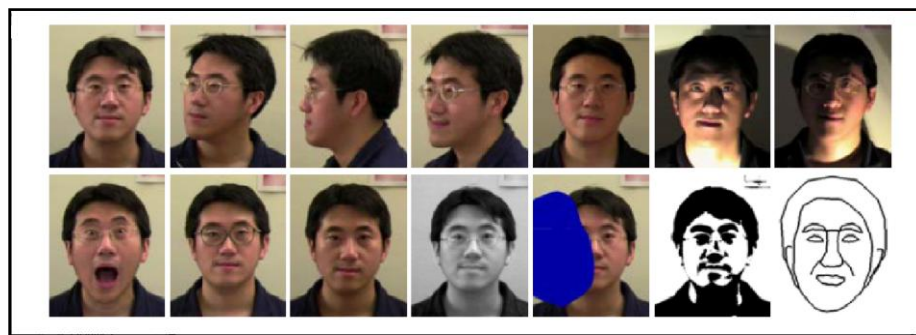


Figure 2.4: Challenges in face recognition

Among all these challenges, some of the key challenges are:

2.6.1. Illumination

Varying illumination is considered the main problem because the differences caused due to the variation is more significant than the differences between individuals. The variation in illumination causes variation in direction and energy distribution of the ambient illumination, and leads to major differences in the shading. The variations of both global face appearance and local facial features also cause problems for automatic face detection/localization, which is the prerequisite for the subsequent face recognition stage. Moreover, in an application, the illumination variation is always coupled with other problems such as pose variation and expression variation, which increase the complexity of the automatic face recognition problem. There are two main approaches to Illumination processing: Active approach, Passive approach.

In active approaches the image is captured in a consistent illumination condition using various techniques, so as to decrease the effect of variation of illumination during the recognition phase. The use of additional devices (optical filters) is required in this approach to acquire different modalities of face images that are not affected by illumination.

Passive approaches attempt to overcome illumination variation in face images due to environment illumination change. These approaches can be classified into 4 categories, namely:

- Illumination invariant modeling models the process of image formation by either using a statistical model or a physical model. Statistical model does not require assumptions regarding the surface properties and analysis techniques like PCA, LDA, ICA etc are applied on a training set containing images of varying illumination, so as to define a subspace which covers all the illumination variations. On the other hand, the Physical model requires assumptions regarding the surface reflectance properties. Ref [17] presents the list of generative models that can be used for illumination invariant face recognition.
- Illumination invariant feature extraction extracts facial features that are robust against illumination variations. There are several illumination invariant representations of faces, some of them are edge map [23], image intensity derivatives, local binary pattern (LBP) [15] and image convolved with a 2D Gabor-like filter. These representations can also be used with a log function to generate variations in the representations. However these representations by themselves are not sufficient to overcome images with varying illumination.
- Often images are not in the correct form to be plugged straight into a recognition algorithm. The image may contain more information than one single face, and the lighting conditions in the test image may not be the same as in the sample data for training the algorithm. This can greatly affect the effectiveness or the recognition rate of the algorithm. Therefore,

to obtain the best possible results, it is necessary to pre-process an image to normalize lighting and remove noise before inserting it into a recognition algorithm. One of the most famous technique for image pre-processing is Histogram Equalization (HE) which enhances the contrast of an image. Ref [14] encompasses the methodologies used for histogram equalization.

- The 3D morphable model is based on a vector space representation of faces. In this vector space, any convex combination of shape and texture vectors of a set of examples describes a human face. The shape and texture parameters can be separated from the illumination information. This proposed model can handle illumination and viewpoint variations, but they rely on manually defined landmark points to fit the 3D model to 2D intensity images. Ref [26, 27] provides more information on 3D morphable models.

More information on the above techniques [6, 8] and other illumination normalization methods used to make robust illumination invariant face recognition possible are discussed in Ref [10, 11, 12, and 13].

2.6.2. Pose

A good face recognition system should perform well even in an uncontrolled environment and when uncooperative subjects are involved. Pose variation is one of the key challenges and it attracts a lot of interest from the Computer Vision and Pattern Recognition community. A few methods have been proposed in solving the problem of pose variation but none of them are free from limitations and do not solve the problem completely.

Pose invariant face recognition algorithms are particularly useful in an uncontrolled environment and uncooperative subjects. For example, face recognition is important for providing security in public places. It is possible to maintain a database of the people in the public place at any given point of time. Using this it can be determined whether a person of interest is present or not.

The most straightforward approach to this problem would be to maintain a gallery of images of each individual in all possible poses to compensate for the pose variation. However, it is tedious and difficult to gather this gallery of images for each individual. For example, if the database maintains only a single image per person in its database. This leads us to the conclusion that the pose variation problem involves images of a subject that have a different pose as compared to the image stored in the database. If an algorithm is pose invariant, this makes face recognition a non-intrusive technique. This gives it a critical advantage over other biometric techniques.

Pose invariant face recognition algorithms can be classified into two categories: 2D techniques, 3D methods. 2D techniques can be further classified into three groups - (a) pose-tolerant feature extraction, (b) real view-based matching, (c) 2D pose transformation. The approaches that involve pose-tolerant feature extraction either find face classifiers or perform pre-processing of linear/non-linear mapping in the image space which is robust to variations in pose. Real view-based matching works with multiple images to cover all possible poses for face recognizer. When multiple images of a person are not present in the database real view-based matching performs poorly. In these cases 2D pose transformation can be used to alter the appearance of a known face image to unknown poses to build virtual views to enhance the face recognizer.

Face recognition using 3D models has been one of the most successful approaches recently. They are particularly useful when pose and illumination variations are involved. These approaches are successful because human faces are 3D objects with a particular structure. 3D face models can be reconstructed using either feature or image based techniques. The basic pose recovery method was proposed [17]. It is based on a generic cylindrical shape of the face. Images in arbitrary positions are mapped onto the cylindrical shape and the frontal virtual views are recovered. Given a rotated input probe image, the eyes are first detected then the face boundary and vertical symmetry line. All this is included in the normalization phase of the implementation. Further, PCA is used as a face recognizer and provided very good results.

2.6.3. Aging

Aging is a factor that has recently attracted attention with respect to the face recognition research. This factor is both complex and unpredictable as it involves a lot of changes to the properties of the face like skull size, facial texture, facial hair growth, etc. Most algorithms are not designed to overcome this particular challenge. Hence there is an increased interest in overcoming the challenge presented by aging.

Face recognition across ages is an important problem and has many applications, such as passport photo verification, image retrieval, surveillance, etc. This is a challenging task because human faces can vary a lot over time in many aspects, including facial texture (e.g., wrinkles), shape (e.g., weight gain), facial hair, presence of glasses, etc. In addition, Image acquisition conditions and environment often undergo large changes, which can cause non-uniform illumination and scale changes.

2.6.4. Occlusion

Occlusion in an image refers to hindrance in the view of an object. It can be natural as well as synthetic. Natural occlusion refers to obstruction of views between the two image objects without any intention while synthetic occlusion refers to artificial blockade of intentionally covering the image's view with a white/black solid rectangular block. Partial occlusion has been found in many areas of image processing. It is seen in iris recognition where the eyelashes occlude the iris; identification via ear can also be occluded by the presence of earrings. Even in real time application face image becomes occluded via accessories (sunglasses/scarf/ hair or even by hand). Other than biometric image processing, it is also encountered in medical field where the arteries may be occluded due to high cholesterol level.

Literature studies reveal that faces can be recognized in a restricted environment with high accuracy but in real environments it is still challenging such as illumination, pose variation and occlusions need to be overcome in which occlusions such as sunglasses, scarf etc., is more important.

Face recognition methods whether linear or non-linear are classified into three groups handling occlusion in face images i.e., feature based methods that deal with features like eyes, mouth, nose and establish a geometric correspondence between them. The second category is appearance-based methods that focus on the holistic features of face images by considering the whole face region and a third class deals with the hybrid local and global features of face images to be used for recognition purpose.

2.6.5. Expression

It has been noticed that facial expressions usually affects the performance of a face recognition system. To deal with it, some existing methods rely on extracting stable face features e.g., extracted line segment and geometric invariants.

Recently some newly-developed face recognition systems based on video sequences can handle a wider range of facial expressions. Compared to a single face image a video sequence consists of a large number of successive images and carries much more information, making the recognition task easier. However, a conventional face recognition system based on single image is still valuable in many real applications, e.g., creating personal photo book from digital image collections, where video sequences consistently capturing a face are not generally available.

2.7. Standard Face Databases

When benchmarking an algorithm it is recommendable to use a standard test data set for researchers to be able to directly compare the results. While there are many databases in use currently, the choice of an appropriate database to be used should be made based on the task given (aging, expressions, lighting etc). Another way is to choose the data set specific to the property to be tested (e.g. how algorithm behaves when given images with lighting changes or images with different facial expressions). If, on the other hand, an algorithm needs to be trained with more images per class (like LDA), Yale face database is probably more appropriate than FERET. Some prominent databases are:

2.7.1. The Color FERET Database, USA

The FERET program set out to establish a large database of facial images that was gathered independently from the algorithm developers. Dr. Harry Wechsler at George Mason University was selected to direct the collection of this database. The database collection was a collaborative effort between Dr. Wechsler and Dr. Phillips. The images were collected in a semi-controlled environment. To maintain a degree of consistency throughout the database, the same physical setup was used in each photography session. Because the equipment had to be reassembled for each session, there was some minor variation in images collected on different dates. The FERET database was collected in 15 sessions between August 1993 and July 1996. The database contains 1564 sets of images for a total of 14,126 images that includes 1199 individuals and 365 duplicate sets of images. A duplicate set is a second set of images of a person already in the database and was usually taken on a different day. For some individuals, over two years had elapsed between their first and last sittings, with some subjects being photographed multiple times. This time lapse was important because it enabled researchers to study, for the first time, changes in a subject's appearance that occur over a year.

2.7.2. The Yale Face Database

This database contains 165 grayscale images in GIF format of 15 individuals. There are 11 images per subject, one per different facial expression or configuration: center-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, and sleepy, surprised, and wink.

2.7.3. The Yale Face Database B

This database contains 5760 single light source images of 10 subjects each seen under 576 viewing conditions (9 poses x 64 illumination conditions). For every subject in a particular pose, an image with ambient (background) illumination was also captured.

2.7.4. AT&T "The Database of Faces" (formerly "The ORL Database of Faces")

This database consists of ten different images for each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

2.7.5. Indian Face Database

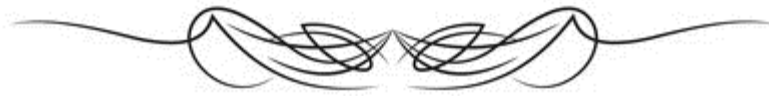
The database contains a set of face images taken in February, 2002 in the IIT Kanpur campus. There are eleven different images of each of 40 distinct subjects. For some subjects, some additional photographs are included. All the images were taken against a bright homogeneous background with the subjects in an upright, frontal position. The files are in JPEG format. The size of each image is 640x480 pixels, with 256 grey levels per pixel. The images are organized in two main directories - males and females. In each of these directories, there are directories with name as a serial numbers, each corresponding to a single individual. In each of these directories, there are eleven different images of that subject, which have names of the form abc.jpg, where abc is the image number for that subject. The following orientations of the face are included: looking front, looking left, looking right, looking up, looking up towards left, looking up towards right, looking down. Available emotions are: neutral, smile, laughter, sad/disgust.

2.7.6. Labeled Faces in the Wild

Labeled Faces in the Wild is a database of face photographs designed for studying the problem of unconstrained face recognition. The database contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the database. The only constraint on these faces is that they were detected by the Viola-Jones face detector.

2.7.7. PIE Database, CMU

It is a database containing 41,368 images of 68 people with each person under 13 different poses, 43 different illumination conditions and with 4 different expressions.



CHAPTER 3

SYSTEM ANALYSIS



SYSTEM ANALYSIS

3.1. Software requirements

The software requirements play an important role in understanding the basic software's needed of the system. The proposed system is coded on python using many of the exiting computer vision libraries and packages. The detailed list of requirements is enlisted below:

- System Back-end:

Programming Language	:	Python
Computer Vision Library	:	OpenCV
Packages	:	Numpy, Scipy, PIL.

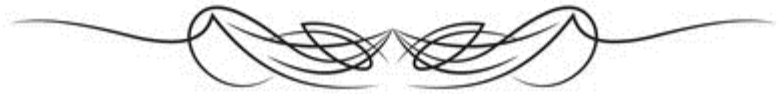
- System Front-end:

Web framework	:	Flask
Programming Language	:	HTML, CSS, JQuery

3.2. Hardware requirements

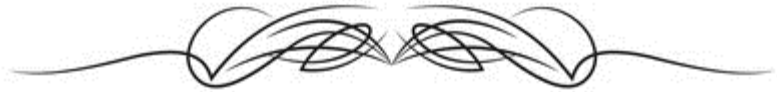
As the proposed system was implemented on our laptops, the hardware requirements are that of a basic computer system. We have enlisted them below.

RAM	:	2GB or Higher
Hard Disk	:	40GB
Operating system	:	Windows 7 32-bit/64-bit, Linux.



CHAPTER 4

SYSTEM DESIGN



SYSTEM DESIGN

The proposed system is designed to be robust with changing datasets, i.e., the algorithms are designed to work with changing input size and type. Initially, there is a database from which datasets are retrieved depending on the test being carried out. With this chosen dataset we execute certain preprocessing methods to help prepare the images for recognition purposes. The dataset is divided into testing dataset and training dataset. The training dataset is transformed to an intermediate representation and is used to train the algorithm for recognition purposes.

The testing dataset is taken one image at a time. The testing image is converted to the intermediate representation and is used to compare with the data stored from the training phase. This comparison makes use of certain distance comparing methods. The pair of images that return the minimum distance is considered to be matched. This is done repeatedly for each image in the testing dataset. In the analysis and output stage, the matches are categorized into success and failure and displayed. The accuracy is also computed for each test and displayed as result.

Along with this we do comparative study of the algorithms with respect to accuracy, efficiency and challenges it handles. Using this information we generate tables that contain the results.

The overall system design can be better understood by dividing it into modules. For this purpose the system is divided into five modules, namely:

- Input module
- Preprocessing module
- Face Recognition module
- Analysis module
- Output module

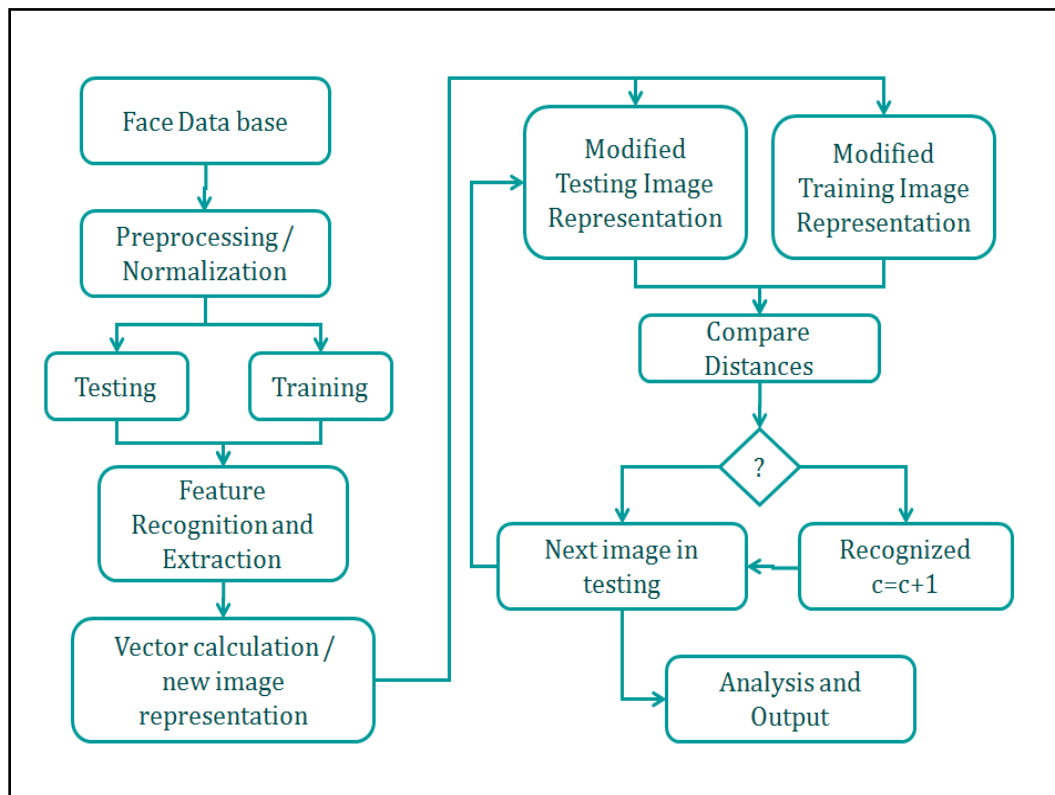


Figure 4.1 Data flow diagram of proposed system

4.1. Input Module

This module contains methods that read datasets from the face database. Each database has different versions of the method that will read a set of images from it depending on the test chosen. The tests are based on challenges such as, illumination, pose, expression, glasses, etc. The images read from the database is divided into two datasets (training and testing), which is later used for face recognition.

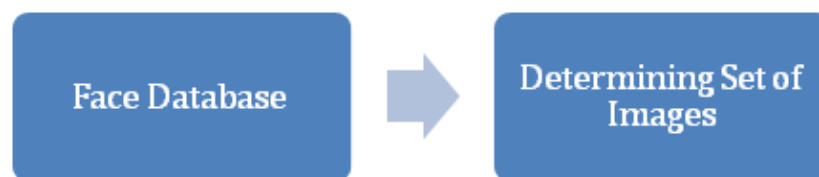


Figure 4.2: Representation of input module

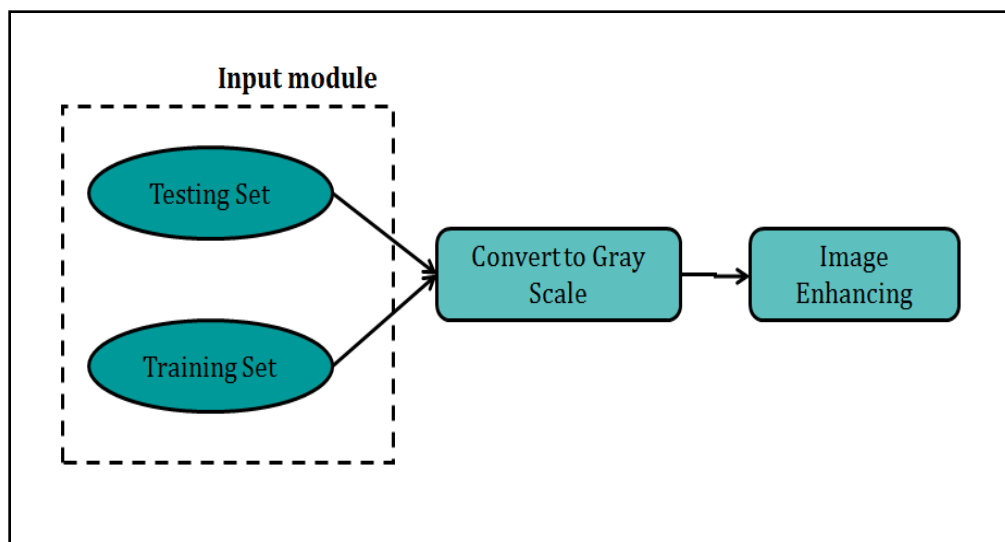
In this system, we run all our algorithms on two standard databases for comparison purposes. The databases chosen are, Yale Face Database A and ORL Database. The specifications of both are provided in the table below.

Database	Images per person	Number of subjects	Total number of images	Size of each image	Challenges
ORL	10	40	400	92 * 112 px	Pose
Yale A	11	15	165	320 * 243 px	Illumination Expression Glass Testing

Table 4.1: Specifications of Databases

4.2. Preprocessing module

To improve the performance of the face recognition algorithms, certain preprocessing methods are used to enhance the image quality by applying threshold techniques. Along with this few methods are used to extract important features (key points) from the enhanced face images. These key points are later supplied to the face recognition algorithms for recognition purpose.

**Figure 4.3: Representation of preprocessing module**

4.3. Face recognition module

As explained earlier in section 2.1, a face recognition system consists of 4 steps, namely detection, alignment/preprocessing, feature extraction and feature matching. Out of these the last two form an integral part of face recognition. Each face recognition algorithm implements these steps in different ways. Section 2.7 gives a general idea of the algorithms used in this system. Depending on the algorithm, the functions used to perform feature extraction and face matching vary. The figure below gives a basic representation of the face recognition stage

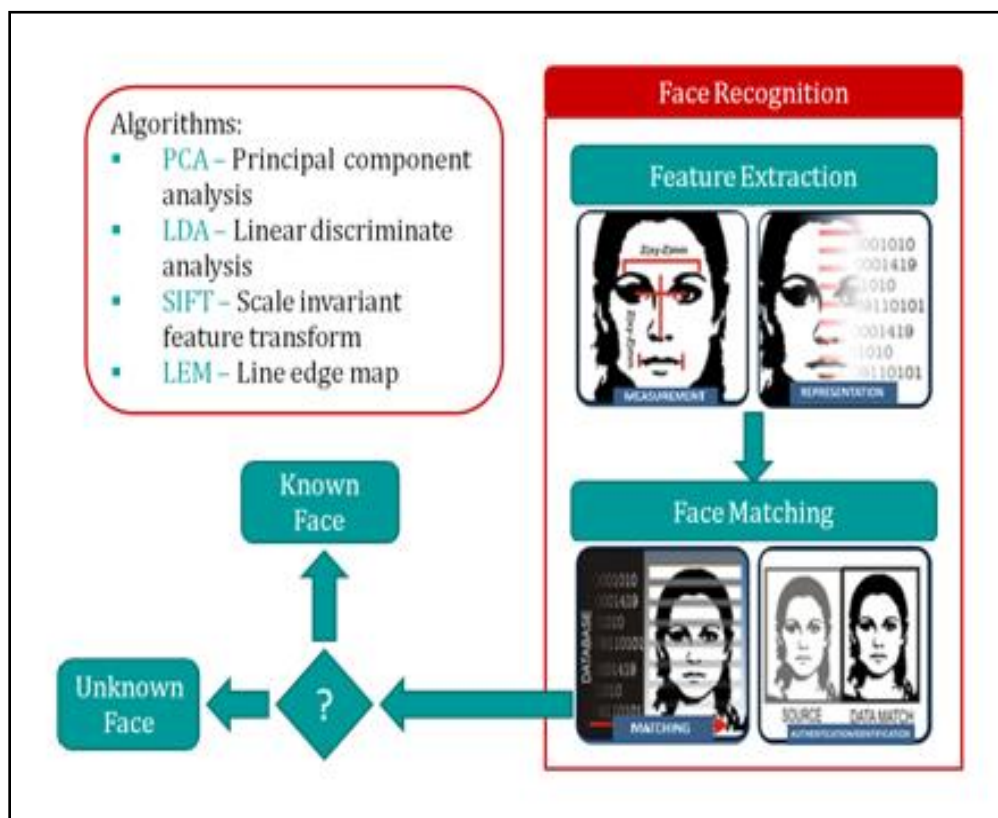


Figure 4.4: Representation of face recognition module

4.4. Analysis module

Two lists success and failure are generated with both of them containing matches that are correct and wrong respectively. Also, the accuracy and the efficiency of each algorithm are computed here. The accuracy is given by the number of correct matches over total number of testing images.

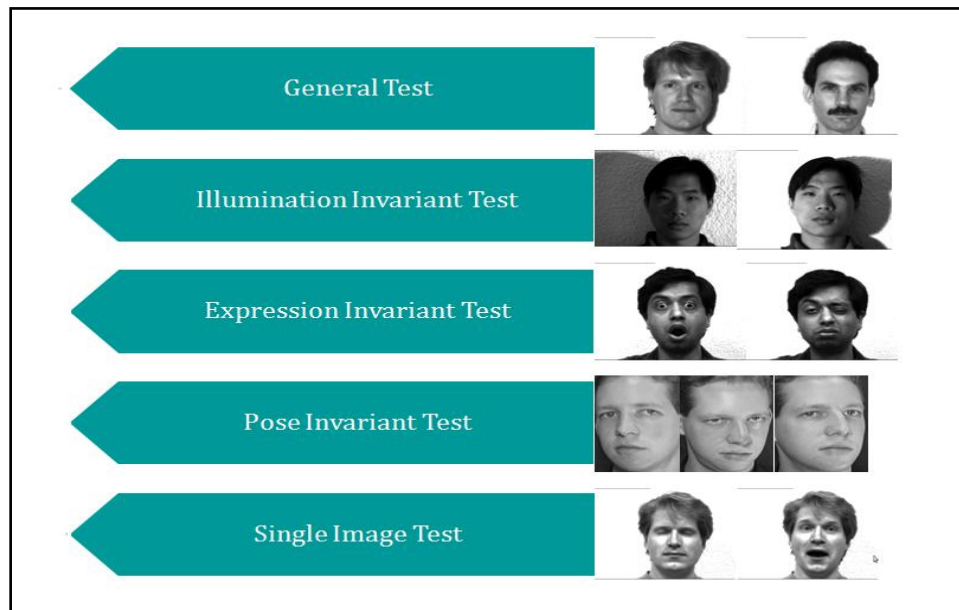


Figure 4.5: List of tests implemented in analysis module

4.5. Output module

From each list (success and failure) we generate plots that contain pairs of images, the test image and the matched image. This is done for each type of test. The accuracy and efficiency is tabulated for each algorithm based on various tests.

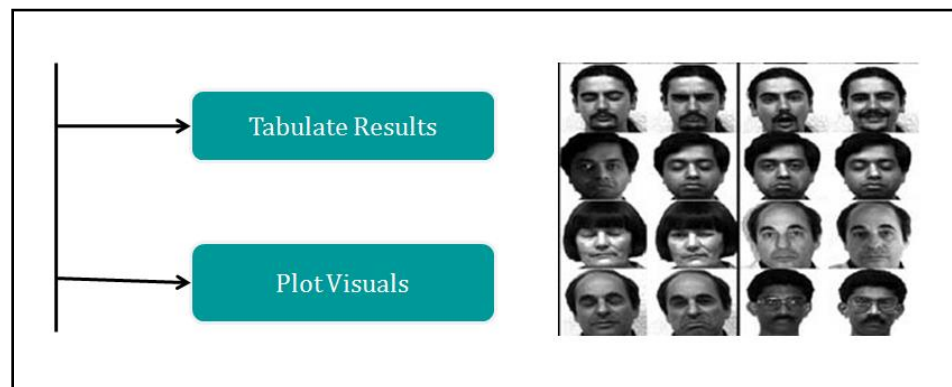
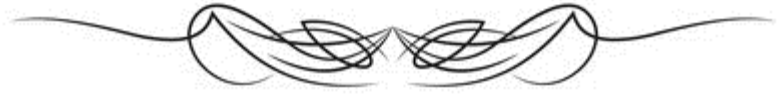
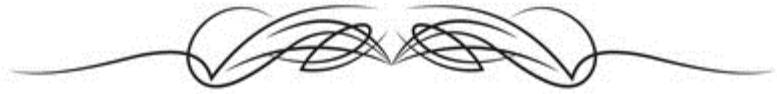


Figure 4.6: Representation of output module



CHAPTER 5

IMPLEMENTATION



IMPLEMENTATION

5.1. Implementation of Back-end

The back-end of the proposed system consists of the system's core modules that were explained in chapter 4. Implementation of these modules is elaborated in the sections below.

5.1.1. Libraries and packages used

There are existing computer vision libraries that came handy during the implementation of the modules and they are:

- **OpenCV :**

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android.



Figure 5.1: Logo of OpenCV library

OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can

take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Mobile robotics
- Object Identification
- Motion tracking

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting (meta-algorithm)
- Decision tree learning
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are now full interfaces in Python, Java and MATLAB/OCTAVE (as of version 2.5). The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Ch, Ruby have been developed to encourage adoption by a wider audience. All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 7

million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

For this project we used openCV 2.4. Functions like `cv2.imread ()`, `cv2.cvtColor ()`, `cv2.GaussianBlur ()` and many more, came handy in reading the images and enhancing them.

- **Python :**

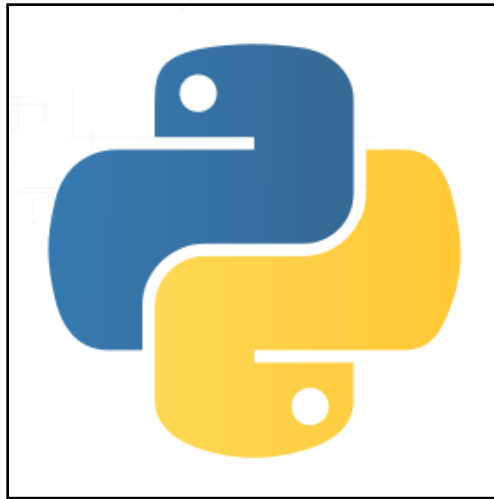


Figure 5.2: Logo of Python programming language

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts. Using third-party tools, Python code can be packaged into standalone executable programs (such as Py2exe, or Pyinstaller). Python code can be packaged into

standalone executable programs. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is free and open source software and has a community-based development model, as do nearly all of its alternative implementations.

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and there are a number of language features which support functional programming and aspect-oriented programming (including by meta programming and by magic methods). Many other paradigms are supported using extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.

The design of Python offers only limited support for functional programming in the Lisp tradition. The language has `map()`, `reduce()` and `filter()` functions, comprehensions for lists, dictionaries, and sets, as well as generator expressions. The standard library has two modules that implement functional tools borrowed from Haskell and Standard ML.

Python 2.0 was released on 16 October 2000, with many major new features including a full garbage collector and support for Unicode. With this release the development process was changed and became more transparent and community-backed. We have used Python 2.7 as the base programming language in this project.

- **Python Imaging Library (PIL) :**

The Python Imaging Library adds image processing capabilities to a Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general

image processing tool. Few standard procedures for image manipulation offered by PIL are:

- per-pixel manipulations,
- masking and transparency handling,
- image filtering, such as blurring, contouring, smoothing, or edge finding,
- image enhancing, such as sharpening, adjusting brightness, contrast or color,
- Adding text to images and much more.

Some of the file formats supported by PIL include PPM, PNG, JPEG, PGM, GIF, TIFF, and BMP. In the project we use PIL mainly for reading images (of type PGM, GIF or BMP) and converting the images to a standard size.


Python Imaging Library	
Original author(s)	Fredrik Lundh
Developer(s)	Secret Labs AB
Initial release	1995; 19 years ago ^[1]
Stable release	1.1.7 / November 15, 2009; 4 years ago ^[2]
Preview release	1.2a0 ^[3] / 2011; 3 years ago
Written in	Python, C
Type	Library for image processing
License	Python Imaging Library license ^[1]
Website	www.pythonware.com/products/pil/ 

Figure 5.3: Details of PIL from Wikipedia

It is available for Windows, Mac OS X and Linux. The latest version of PIL is 1.1.7, was released in September 2009 and supports Python 1.5.2–2.7, with Python 3 support to be released "later".

Development appears to be discontinued with the last commit to the PIL repository coming in 2011. Consequently, a successor project called Pillow has forked the PIL repository and added Python 3.x support. This fork has been adopted as a replacement for the original PIL in Linux distributions including Debian and Ubuntu (since 13.04).

- **Numpy :**

NumPy is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of Numarray into Numeric with extensive modifications. NumPy is open source and has many contributors.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode compiler/interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy seeks to address this problem by providing multidimensional arrays and functions and operators that operate efficiently on arrays. Thus any algorithm that can be expressed primarily as operations on arrays and matrices can run almost as quickly as the equivalent C code.



Figure 5.4: Numpy logo

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional

toolboxes, notably Simulink; whereas NumPy is intrinsically integrated with Python, a more modern, complete, and open source programming language. Moreover complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality.

The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are strided views on memory.[2] In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. The only limitation is that NumPy's arrays must be views on contiguous memory buffers. A replacement packages called Blaze attempts to overcome this limitation.

- **Matplotlib :**

Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. matplotlib can be used in python scripts, python and ipython shell, web application servers, and six graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc, with just a few lines of code.



Figure 5.5: Logo of matplotlib

It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB. SciPy makes use of matplotlib.

The pylab interface makes matplotlib easy to learn for experienced MATLAB users, making it a viable alternative to MATLAB as a teaching tool for numerical mathematics and signal processing. Some of the advantages of the combination of Python, NumPy, and matplotlib over MATLAB include:

- Based on Python, a full-featured modern object-oriented programming language suitable for large-scale software development
- Free, open source, no license servers
- Native SVG support
- Nice default plot styles: less code to polish the figure
- Deep integration with Python
- Matlab style programming interface

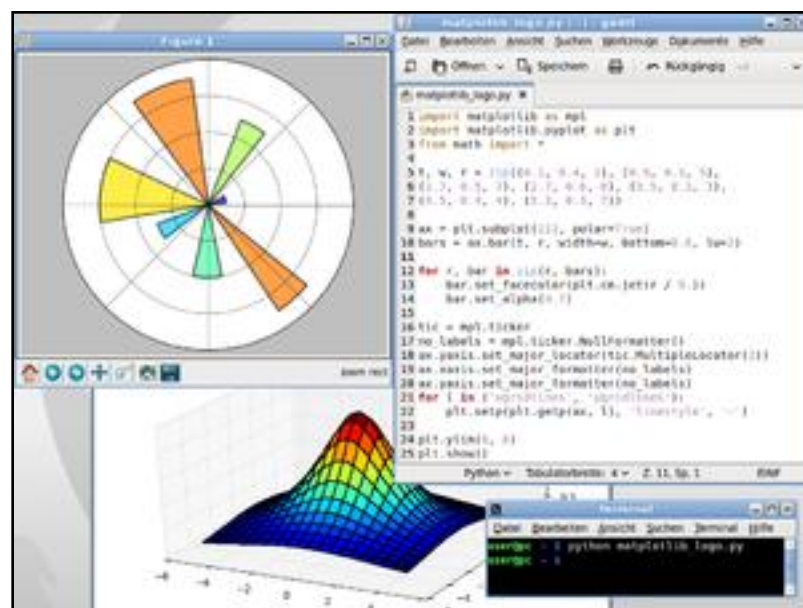


Figure 5.6: Example of matplotlib

In this project we use matplotlib to generate a success or failure image for display. A success or failure image consists of the input image and the matched image aligned horizontally. To do this we need to merge the two images into a single image, which is done using matplotlib. This library is also used to merge multiple success or failure images into a success or failure set. This image set is a single image that is displayed at the end as the result of a test.

5.1.2. Input module

The input module consists of functions to read images from Yale database A and ORL database. The images are stored in the respective database folders. The file path for the images is supplied to the `cv2.imread()` function which reads the image, fixes the size (if necessary) and the depending on the type of test chosen by the user (which is passed as parameter to the function that reads images), the array representation of the image is stored either in a training list or a testing list. Along with this, the filenames of images are stored in another list called `training_answer` and `testing_answer` depending on the set it belongs to. This comes handy in displaying the matching results and in plotting the visuals, as filenames are required to extract an image at a particular location and merge it with images for display of result.

Each database has a particular format in which the images are organized; hence, there are separate read functions for each of the databases to which the type of test is passed as a parameter. In case of ORL database each subject has a folder by the name “sXX” where XX represents the number of the subject. Each folder contains 10 images of the subject taken in varied angles (pose). Each image is named as “YY.pgm” where YY is again the number of the photo. For reading images from ORL database the `read_orl_images()` function first traverses through the directory to find all the sub-directories, it then traverses through the sub-directories to read the “.pgm” image files. In case of ORL database only 3 types of test can be conducted and they are:

- **General test** – In this test the first 5 images are put into training and the next five are put into testing.

- **Pose test** – In this test the images ‘2.pgm’, ‘4.pgm’, ‘5.pgm’, ‘6.pgm’, ‘9.pgm’, ‘10.pgm’ are put into testing as they vary in pose while the rest is put into training set.
- **Single image test** – This is a test for the examiner to choose an image of his choice from the testing set and match that single image with the images in the training set. For this test the even numbered images are put into testing set while the odd numbered images are put into training set.

The Yale database A contains 165 images of 15 subjects with 11 images per subject. Each image is named as “subjectXX” (where XX is subject’s number) followed by the extensions i.e. 'centerlight', 'glasses', 'happy', 'leftlight', 'noglasses', 'normal', 'rightlight', 'sad', 'sleepy', 'surprised', 'wink'. The `read_yale_images()` function takes the type of test as a parameter and read the images from the database file and categorizes the images based on test. The tests that are conducted on this database are:

- **General test** – In this test the first 6 images are put into training and the next 5 are put into testing.
- **Illumination test** – In this test, images with extensions as “centerlight”, “rightlight”, “leftlight” (i.e varying in illumination) are put into testing while the normal images are put into training. Here normal images are the ones with no variation and are front posed (i.e. 'glasses', 'noglasses', 'normal').
- **Expression test** – In this test images with variation in expressions are put into testing i.e images with extensions as 'happy', 'sleepy', 'sad', 'wink', 'surprised'; while normal images are put into training.
- **Glasses test** – In this test images with extension as glasses is put into training while images with extensions as “noglasses” and “normal” are put into training.
- **Single image test** – This is same as the test explained above for ORL database except that the first 5 images are put into training and next 6 images are put into testing.

5.1.3. Preprocessing module

This module uses function to enhance an image before the face recognition process can be applied to an image. As of now this module uses basic techniques to enhance an image for SIFT and LEM algorithms. Functions parts of this module are:

- **cv2.cvtColor ()** – This function converts an image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). We mainly use this function to convert the images to gray scale as most algorithms work on gray scaled images.
- **cv2.GaussianBlur ()** - Blurs an image using a Gaussian filter. The function convolves the source image with the specified Gaussian kernel so as to blur the image. This is used in LEM to find edge amp using canny edge detection algorithm.
- **cv2.canny ()** – This function is used to find an edge map of the input image by applying canny edge detection algorithm. This is used in LEM.
- **cv2.bitwise_and ()** - Calculates the per-element bit-wise conjunction of two arrays or an array and a scalar.

More preprocessing techniques can be used as part of future work.

5.1.4. Face recognition module

Four different face recognition algorithms that are implemented in this module are:

- **Principal Component Analysis (PCA)**

The Principal Component Analysis (PCA) is one of the most successful techniques that have been used in image recognition and compression. PCA is a statistical method under the broad title of factor analysis. The purpose of PCA is to reduce the large dimensionality of the data space to the smaller intrinsic dimensionality of feature space, which are needed to describe the data

economically. This is the case when there is a strong correlation between observed variables.

It reduces the dimensionality and effect of noise by representing M face images in the form of K eigenface vectors, derived from the covariance matrix. These K eigenface vectors contain all the important features required to represent a face image. The input image is also represent as an eigenface vector and a comparison of distances of the vectors would yield the recognized face, if the distance is beyond the threshold otherwise, the face is an unknown face.

The steps in implementing Principal component analysis are enlisted below:

- Get M face images, each of size $N \times N$ – In PCA size of all the images must be same.
- Convert each face image in the training set to a face vector i.e. a column vector or row vector of dimension N^2 .
- Normalize the face vectors by removing the features common to all the face images in training set from each face image so that each face image is left with only unique features (principal components). This can be done by the following steps:
 - Find the mean of the face images to obtain an average matrix that represents all the features in the face images of the database. The below formula can be used to derive (Global mean) for a dataset.

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

- Subtract the average face vector from each face vector to get normalized face vector.
- Calculate the covariance matrix (S) - using the formula:

$$S = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)(x_i - \mu)^T$$

Where μ is the global is mean and x_i is a face vector.

- Calculate eigenvectors λ_i and eigenvalues v_i - by using the covariance matrix derived from above equation.

$$Sv_i = \lambda_i v_i, i = 1, 2, \dots, n$$

- Select the K best eigenfaces such that $K < M$ and K should represent the whole training set. This can be done by ordering the eigenvectors in descending order by their eigenvalue. The k principal components are the eigenvectors corresponding to the k largest eigenvalues.

The k principal components of the observed vector x are then given by:

$$y = W^T(x - \mu)$$

Where $W = \{v_1, v_2, \dots, v_k\}$ i.e. it contains all the eigenvectors computed in the previous step.

- Find the weight vector for each face image – so as to represent the images as a linear combination of all the K eigen vectors and the average vector.
- The original face vector from the eigenface can be reconstructed by using formula below:

$$x = Wy + \mu$$

Where W is an array of eigenvalues corresponding to eigenvectors, y is the principal component and μ is the global is mean.

- The solution for the optimization problem for PCA is given by –

$$W_{pca} = \arg \max_W |W^T S W|$$

Where, S is the covariance matrix, W is an array of eigenvalues corresponding to eigenvectors and W^T is its transpose.

- Apply the same procedure to the images in the testing set – the weight vectors of images in testing set are compared to weight vectors of images in testing dataset.

- Use the Euclidean distance to measure the distance between the weight vectors of two images. The images with lowest distance between their weight vectors are considered as a match for the image in the testing set.

● **Linear Discriminant Analysis (LDA)**

PCA appears to work well when a single image of each individual is available, but when multiple images per person are present, then Belhumeur et al argue that by choosing the projection which maximizes total scatter, PCA retains unwanted variations due to lighting and facial expression. As stated by Moses et al. “the variations between the images of the same face due to illumination and lighting direction are almost always larger than image variations due to a change in face identity”. Therefore, they propose using Fisher’s Linear Discriminant Analysis, which maximizes the ratio of the between-class scatter and the within-class scatter and is thus purportedly better for classification than PCA.

The standard eigenfaces and the Fisherfaces approaches assume the existence of an optimal projection that projects the face images to distinct non-overlapping regions in the reduced subspace where each of these regions corresponds to a unique subject. However, in reality, that assumption may not necessarily be true since images of different people may frequently map to the same region in the face space and, thus, the regions corresponding to different individuals may not always be disjoint.

The PCA finds a linear combination of features that maximizes the total variance in data. While this is clearly a powerful way to represent data, it doesn't consider any classes and so a lot of discriminative information may be lost when throwing components away. Imagine a situation where the variance is generated by an external source, let it be the light. The components identified by a PCA do not necessarily contain any discriminative information at all, so the projected samples are smeared together and a classification becomes impossible.

In order to find the combination of features that separates best between classes the Linear Discriminant Analysis maximizes the ratio of between-classes to within-classes scatter. The idea is simple: same classes should cluster tightly

together, while different classes are as far away as possible from each other. This was also recognized by Belhumeur, Hespanha and Kriegman and so they applied Discriminant Analysis to face recognition

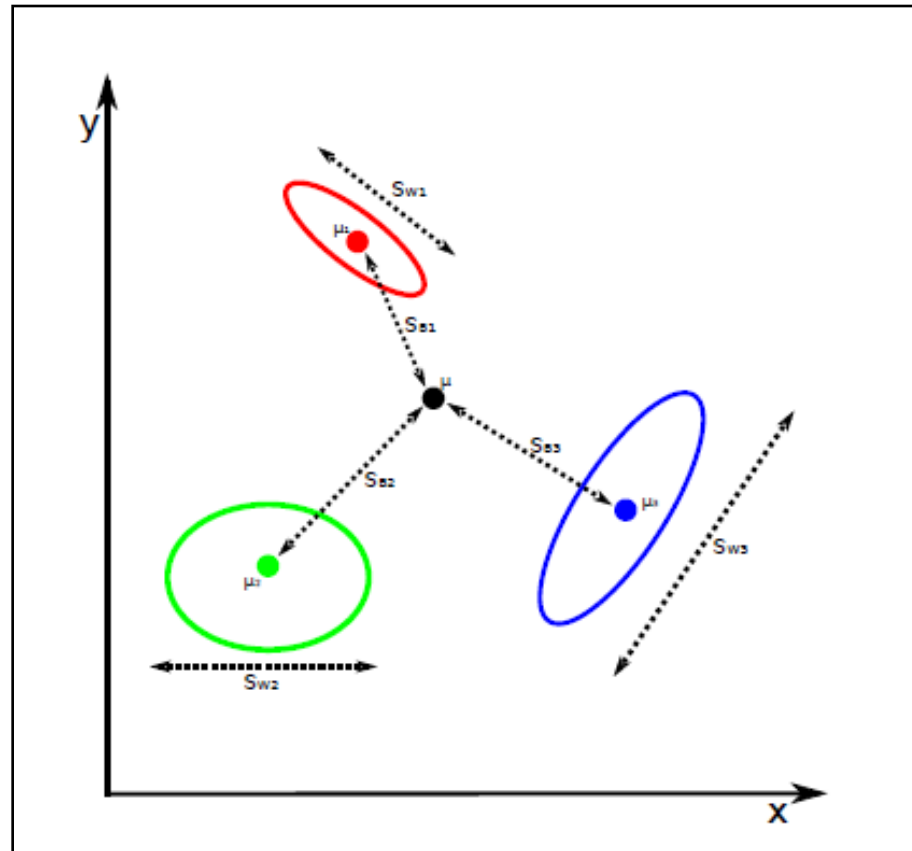


Figure 5.7: Scatter matrices S_B and S_W for a 3 class problem

The steps in implementing Fisher's linear discriminant analysis (FLDA) are enlisted below:

- Get M face images, each of same size – In LDA size of all the images must be same.
- Group the images into classes where each class contains N samples – grouping images into class's plays a vital role in the algorithm as this establishes how to distinguish the face images. Better the classification, better the results in matching. A good way to classify the face images would be to place all the images pertaining to a particular subject in one class.

$$X = \{X_1, X_2 \dots X_C\}$$

$$X_i = \{x_1, x_2 \dots x_n\}$$

Where, $i = 1, 2 \dots n$ and n is the number of samples in a class.

- Convert each face image in the training set to a face vector i.e. a column vector or row vector.
- Calculate the global mean of the face images in all the classes to obtain an average matrix that represents all the features in the face images of the database. The below formula can be used to derive (Global mean) for a dataset.

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

- Calculate the class mean μ_i of each class to obtain an average matrix that represents all the features in face images present in a class. The below formula can be used to derive (class mean) for a dataset.

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j$$

- Calculate the covariance matrix (S_B) that gives the scatter between classes - using the formula:

$$S_B = \sum_{i=1}^c N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

Where μ is the total mean and μ_i is the mean of class $i \in \{1, \dots, c\}$.

- Calculate the covariance matrix (S_W) that gives the scatter within classes - using the formula:

$$S_W = \sum_{i=1}^c \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

Where μ_i is the mean of class $i \in \{1, \dots, c\}$, x_j is the face vector belonging to class j .

- Calculate the eigenvectors by using the formula given below:

$$\begin{aligned} S_B v_i &= \lambda_i S_w v_i \\ S_w^{-1} S_B v_i &= \lambda_i v_i \end{aligned}$$

Where, S_B is the between class scatter matrix, S_w is the within class scatter matrix, v_i is the eigenvector and λ_i is its eigenvalue.

- The solution is given by –

$$W_{opt} = \arg \max_W \frac{|W^T S_B W|}{|W^T S_w W|}$$

The rank of S_w is at most $(N - c)$, with N samples and c classes. In pattern recognition problems the number of samples N is almost always smaller than the dimension of the input data (the number of pixels), so the scatter matrix S_w becomes singular. This is solved by performing a Principal Component Analysis (PCA) on the data and projecting the samples into the $(N - c)$ dimensional space. A Linear Discriminant Analysis is then performed on the reduced data, because S_w isn't singular anymore. The optimization problem can be rewritten as:

$$W_{fld} = \arg \max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_w W_{pca} W|}$$

Where, S_B is the between class scatter matrix, S_w is the within class scatter matrix, W_{pca} is an array of eigenvalues corresponding to eigenvectors calculated using PCA and W_{pca}^T is its transpose and W is an array of eigenvalues corresponding to eigenvectors calculated using LDA and W^T is its transpose.

The transformation matrix W , that projects a sample into the $(c-1)$ dimensional space is then given by:

$$W = W_{fld}^T W_{pca}^T$$

- Apply the same procedure to the images in the testing set – the weight vectors of images in testing set are compared to weight vectors of images in testing dataset.
 - Use the Euclidean distance to measure the distance between the weight vectors of two images. The images with lowest distance between their weight vectors are considered as a match for the image in the testing set.
- **Scale Invariant Feature transform (SIFT)**

For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. To perform reliable recognition, it is important that the features extracted from the training image be detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as object edges.

Another important characteristic of these features is that the relative positions between them in the original scene shouldn't change from one image to another. For example, if only the four corners of a door were used as features, they would work regardless of the door's position; but if points in the frame were also used, the recognition would fail if the door is opened or closed. Similarly, features located in articulated or flexible objects would typically not work if any change in their internal geometry happens between two images in the set being processed. However, in practice SIFT detects and uses a much larger number of features from the images, which reduces the contribution of the errors caused by these local variations in the average error of all feature matching errors. David G Lowe elaborates on recognition of images and objects using SIFT in his paper [21, 22].

SIFT key points of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually

comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors.

- **Line Edge Map (LEM)**

Edge information is a useful representation feature that is insensitive to certain changes. A suitable face feature representation, Line Edge Map (LEM), was proposed by Gao and Leung, [23, 24] which extracts as features line segments from a face edge map. In which the faces were encoded into binary edge map using sobel edge detection algorithm. A line edge map approach extracts lines from a face edge map as features. This can be considered as a combination of template matching and geometrical feature matching. LEM integrates the structural information with spatial information of a face image by grouping pixels of face edge map to line segments. After thinning the face edge map, a polygonal line fitting process known as the dynamic two strip algorithm is applied to generate the LEM of a face.

They also introduced the primary line segment Hausdorff distance (HpLHD) and the complete version of line segment Hausdorff distance (LHD), which they used to measure the similarity of face LEMs. In mathematical terms, The LEM representation which records only the end points of line segments on curves, further reduces the storage requirement. Efficient coding of faces is a very important aspect in a face recognition system. LEM is also expected to be less sensitive to illumination changes due to the fact that it is an intermediate - level image representation derived from low level edge map representation. The basic unit of LEM is the line segment grouped from pixels of edge map.

Advantages of using edge maps for face recognition are:

- Partially illumination-invariant
- Low memory requirement as LEMs as such do not store the line information instead just store information regarding the points.
- Recognition performance of template matching.

Steps in performing face recognition using LEM:

- Read image from database.
- Convert the image to gray-scale to get a binary image.
- Apply edge-detector algorithm to detect edges of the face in the image.
- Determine I which is a set of points along the edges..
- Use Hausdroff distance to measure the similarity between 2 separate set of points I_1 and I_2 . The formula for Hausdroff distance is given by:

$$H(I, J) = \max(h(I, J), h(J, I))$$

Where I is the LEM of a face image from the database while J is the LEM of a face images from the testing set and $h(I, J)$ calculates the total distance between the two LEMS. It is given by:

$$h(I, J) = \frac{1}{\sum_{i \in I} \|i\|} \sum_{i \in I} \|i\| \cdot \min_{j \in J} d(i, j)$$

In the above equation, i and j are 2 line segments whose distance from each other is being calculated by iterating over the line segments of both LEMs at a time.. Where $\|i\|$ is the length of the line segment i and $d(i, j)$ is a vector that comprises of the orientation distance, parallel distance and perpendicular distance between the 2 line segments.

$$\bar{d}(m_i^l, t_j^l) = \begin{bmatrix} d_\theta(m_i^l, t_j^l) \\ d_{//}(m_i^l, t_j^l) \\ d_\perp(m_i^l, t_j^l) \end{bmatrix}$$

The vector $d(i, j)$ can be calculated by using the below equation:

$$d(m_i^l, t_j^l) = \sqrt{d_\theta^2(m_i^l, t_j^l) + d_{//}^2(m_i^l, t_j^l) + d_\perp^2(m_i^l, t_j^l)}$$

In this project we use canny edge detector algorithm to improve the sharpness of the edges extracted. We then use cv2.goodFeaturesToTrack to extract good key points along the edges. Since LEMs are maintained as points rather than lines, we use the extracted points to match LEMs by using Hausdroff distance.

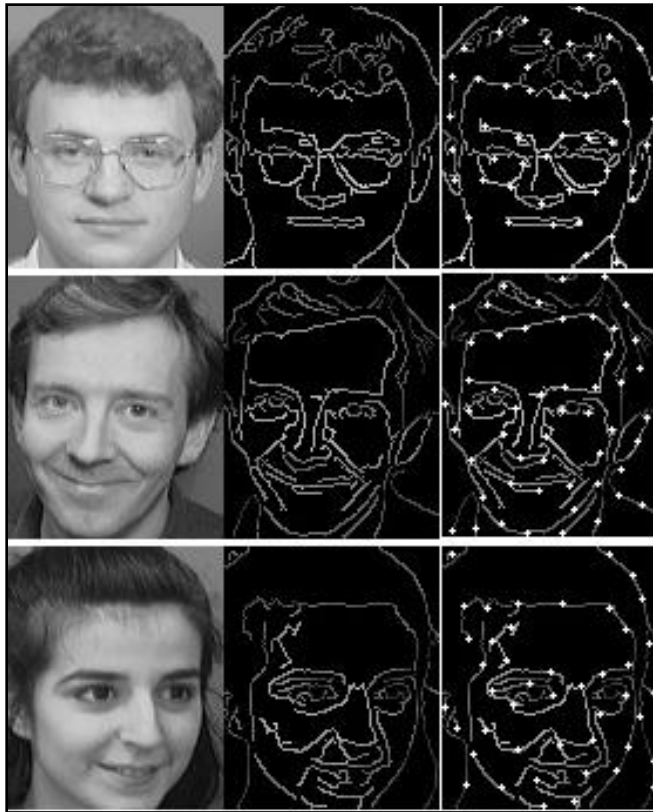


Figure 5.8: Transformation of images using LEM

The above figure shows the original image on the left, followed by the image obtained after applying canny edge detector algorithm. The rightmost image shows the points extracted along the edges of the previous image so as to supply these points for matching of LEMs

5.1.5. Analysis module

For the purpose of analysis we use the clock module to calculate the time an algorithm takes for matching the images in testing dataset with that of the training dataset. This yields the total time taken for the matching process, but since we are more interested in the time taken to match a single image, we divide the total time by the number of images in the testing dataset. The formula used is:

$$\text{Single image matching time (secs)} = \text{total time} / \text{len (testing)}$$

This number format representation of time taken for single matching is then converted into a string format and sent to the server to be displayed on the html page for the user to view.

Accuracy of an algorithm is measured by calculating the total number of matched hits and the dividing it by the total number of images in the testing dataset. The formula used is given below:

$$\text{Accuracy (\%)} = \text{float (count of hits)} / \text{len (testing set)} * 100$$

Along with these above results, the type of the test, number of images in training and testing dataset etc are stored in a table format for each algorithm for analysis purposes.

5.1.6. Output module

It is not enough to display the file name of the test image and the matched image on the command line, a better approach would be to display them graphically. This is what the output module does. Output module consists of functions that plot the visuals for the user to view. This module uses matplotlib extensively.

During the testing the path of test image, its matched counterpart and the location for storing the merged images is sent to a plotting function. The plotting function places the test image next to the successfully matched or unsuccessfully

matched image to create a new combined image of 'png' format. This plotted image is later displayed to the user on the browser based on his choice of view.

5.2. Implementation of Front-end

The front - end of the application is simple and implemented using the common languages used for web development. A user interface using the client – server architecture facilitates the user to send requests (i.e. run an algorithm or choose input dynamically, etc) and receive response in terms of obtained results. The process and frameworks used are explained in the following sections.

5.2.1. Web frameworks used

The web frameworks used to provide a web user interface for the design and analysis of face recognitions algorithms are:

- **Flask :**

Flask is a micro framework for Python based on Werkzeug, Jinja 2. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask takes the flexible Python programming language and provides a simple template for web development. Once imported into Python, Flask can be used to save time building web applications.

Flask is part of the categories of the micro-framework. Some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins. In the case of Flask, its dependencies are Werkzeug a WSGI utility library and jinja2 which is its template engine.



Figure 5.9: Logo of flask framework

Flask is called a microframework because it keeps the core simple but extensible. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light; there is little dependency to update and watch for security bugs. Cons is that some time the programmer will have to increase the list of dependencies by adding plugins by himself as there is no database abstraction layer, form validation, or any other components where third-party libraries already exist to provide common functionality. However, Flask supports extensions, which can add such functionality into an application as if it was implemented in Flask itself. There are extensions for object-relational mappers, form validation, upload handling, various open authentication technologies, and more. The features of Flask include:

- Contains development server and debugger
- RESTful request dispatching
- Uses Jinja2 templates
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Google App Engine Compatibility

● **HTML :**

We use HTML to design the web pages for the web user interface for this project. HTML or HyperText Markup Language is the standard markup language used to create web pages. The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. HTML tags most commonly come in pairs like `<h1>` and `</h1>`, although some tags represent empty elements and so are unpaired, for example ``. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags).



Figure 5.10: HTML5 logo

The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

- **CSS :**

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. Web browsers refer to Cascading Style Sheets (CSS) to define the look and layout of text and other material. The W3C, maintainer of both the HTML and the CSS standards, encourages the use of CSS over explicit presentational HTML. CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content.



Figure 5.11: CSS3 logo

CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice, on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed.

CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable. The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998), and they also operate a free CSS validation service.

- **Javascript and jQuery :**

JavaScript (JS) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It can also be used in server-side programming, game development and the creation of desktop and mobile applications. Its syntax was influenced by C.

JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics.

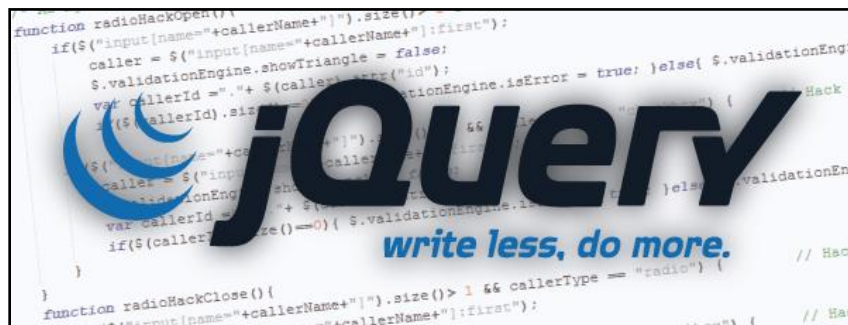


Figure 5.12: JQuery logo

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

5.2.2. User Interface

Initially we implemented an UI using Tkinter library for python. To enhance the UI we shifted from Tkinter to flask. As explained in above sections, flask is a web micro-framework that helps in developing a website.

We used HTML, CSS and javascript to design web pages that displays different information related to testing. For example, there's a HTML page for

starting the test, a page for choosing databases, a page for choosing the type of test to be run, etc.

Flask has a structure for storage of documents. All the static files that are used to render the HTML page (javascript, CSS files and images) are to be stored in the “static” directory. All the HTML pages are supposed to be stored in the “templates” directory. The main python files along with all the other python scripts needed for the execution of the application are to be stored outside above two folders. When a path is requested the function for that specified path is executed, the HTML pages are rendered by using the specified HTML page in the templates directory and by using specified CSS and javascript files in the static directory

We import the flask module to a python file and define paths for function invocations. Whenever a path for a function invocation matches the path give by the user as URL (redirected using <a> tags of HTML) the function is called and executed on the command line. The results if any to be displayed are returned to the HTML document and the HTML document is rendered at the end of function execution. Thus, the front – end provides a REStful web service, where all the interactions between the server and the user are based on the unique resource identifiers (URIs). Flask is simple to use and easy to learn. It brought much clarity to the UI code and also helped us provide styles for the web pages.

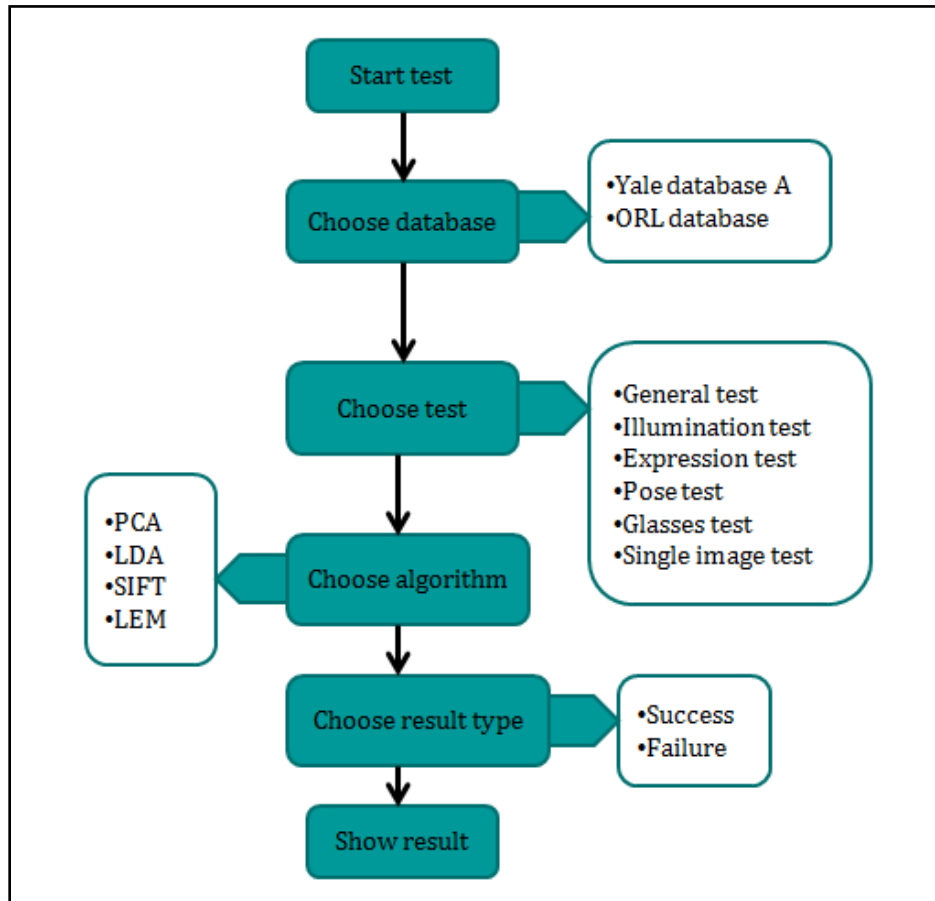
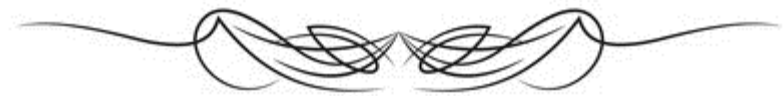
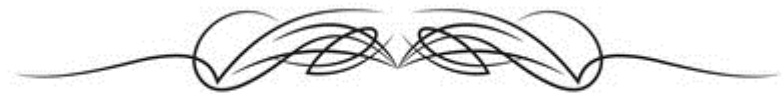


Figure 5.13: Page Transitions of the web based GUI



CHAPTER 6

TESTING



Chapter 6

TESTING

Once source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and code generation. In this test plan all major activities are described below:

- Unit testing.
- Integration testing.
- System testing.

6.1. Unit Testing

Unit testing focuses on verification of the unit of software design or module using the unit test plans, prepared in the design phase of the system development as a guide, important control paths are tested to uncover errors within the boundary of the modules. The interfaces of each of the module were tested to ensure proper flow of the information into and out of the modules under the consideration.

In our project the functions used to implement each algorithm act as units of the system. The functions were checked for correctness b using suitable temporary inputs.

6.2. Integration Testing

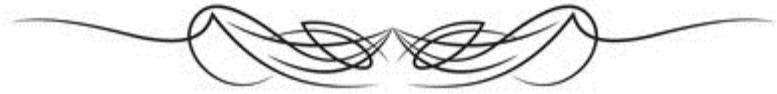
Data can be lost across an interface: one module can have an adverse effect on another's sub-functions, when combined together they may not produce the desired function or they may not work as they did in unit testing. Hence, integration testing becomes a very important part of testing to ensure that all the components of the system work well after integration.

All the modules used for implementing the individual algorithms were integrated and tested for correctness by using random input image set.

6.3. System Testing

System testing is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls in the scope of black box testing, and should require no knowledge of the inner design of the code or logic.

We run predefined tests on each algorithm. These tests are Illumination test, Pose test, General test, etc. The results or performance of each algorithm is tabulated as the comparison study of the four face recognition algorithms and is illustrated in the next chapter.



CHAPTER 7

ANALYSIS AND RESULTS



ANALYSIS AND RESULTS

7.1. Yale A Database

There are four tests that can be run on Yale database and they are: General test, Expression test, Illumination test and Glasses test. The results of these tests for different algorithm are provided in the sections below.

7.1.1. General Test :

In this test the length of training set is 90 and the length of testing set is 75. This is divided on the basis of taking the first 6 images for training and the last 5 images for testing.

Algorithm	Total images matched	Accuracy (%)	Total time taken to match	Time taken to match single image
PCA	66	88	1.519s	0.020s
LDA	65	86.66	0.469s	0.006s
SIFT	71	94.66	7m 16s	5.822s
LEM	59	78.66	13m 21s	10.692s
Length of training set		90		
Length of testing set		75		

Table 7.1: Analysis of General test on Yale database

7.1.2. Illumination Test:

In this test the length of training set is 45 and the length of testing set is 45. This is divided on the basis of taking the images with extensions as “centerlight”, “rightlight”, “leftlight” (i.e. varying in illumination) into testing while the normal images are put into training. Here normal images are the ones with no variation and are front posed (i.e. 'glasses', 'noglasses', 'normal').

Algorithm	Total images matched	Accuracy (%)	Total time taken to match	Time taken to match single image
PCA	20	44.44	0.420s	0.009s
LDA	21	46.66	0.219s	0.004s
SIFT	28	62.22	2m 1s	2.695s
LEM	10	22.22	3m 55s	5.241s
Length of training set		45		
Length of testing set		45		

Table 7.2: Analysis of Illumination test on Yale database**7.1.3. Expression Test:**

In this test images with variation in expressions are put into testing i.e images with extensions as 'happy', 'sleepy', 'sad', 'wink', 'surprised'; while normal images are put into training.

Algorithm	Total images matched	Accuracy (%)	Total time taken to match	Time taken to match single image
PCA	68	90.66	0.480s	0.006s
LDA	64	85.33	0.319s	0.004s
SIFT	75	100	2m 20s	1.871s
LEM	66	88	4m 46s	3.737s
Length of training set		30		
Length of testing set		75		

Table 7.3: Analysis of Expression test on Yale database**7.1.4. Glasses Test:**

In this test images with extension as glasses is put into training while images with extensions as “noglasses” and “normal” are put into training.

Algorithm	Total images matched	Accuracy (%)	Total time taken to match	Time taken to match single image
PCA	12	80	0.090s	0.006s
LDA	11	73.33	0.070s	0.004s
SIFT	15	100	28.549s	1.903s
LEM	14	93.33	57.189s	3.812s
Length of training set		30		
Length of testing set		15		

Table 7.4: Analysis of Glasses test on Yale database

7.2. ORL Database

There are two tests that can be run on ORL database and they are: General test and Pose test. The results of these tests for different algorithms are provided in the sections below.

7.2.1. General Test:

In this test the first 5 images are put into training and the next five are put into testing. The total number of images in training set is 200 and the total number of images in testing set is 200.

Algorithm	Total images matched	Accuracy (%)	Total time taken to match	Time taken to match single image
PCA	180	90	2.810s	0.014s
LDA	181	90.5	1.579s	0.007s
SIFT	198	99	6m 24s	1.920s
LEM	133	66.5	19m 38s	5.892s
Length of training set		200		
Length of testing set		200		

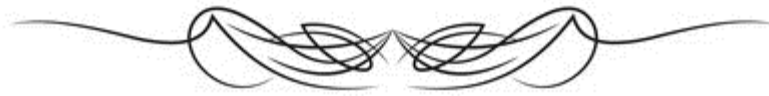
Table 7.5: Analysis of General test on ORL database

7.2.2. Pose Test:

The total number of images in training set is 160 and the total number of images in testing set is 240. In this test the images '2.pgm', '4.pgm', '5.pgm', '6.pgm', '9.pgm', '10.pgm' are put into testing as they vary in pose while the rest is put into training set.

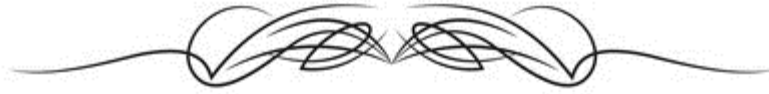
Algorithm	Total images matched	Accuracy (%)	Total time taken to match	Time taken to match single image
PCA	225	93.75	2.580s	0.010s
LDA	226	94.16	1.619s	0.006s
SIFT	232	96.66	5.560s	1.483s
LEM	172	71.66	18m 27s	4.619s
Length of training set		160		
Length of testing set		240		

Table 7.6: Analysis of Pose test on ORL database



CHAPTER 8

CONCLUSION AND FUTURE WORK



CONCLUSION AND FUTURE WORK

8.1. Conclusion

It can be observed from the tables in the previous section that SIFT performs better in all the tests on both the database in terms of accuracy while LDA takes least amount of time for matching.

As observed, PCA performs very well under variation in Expression and Pose. It also performs well on large datasets such as ORL. LDA performs well under variation in Pose and it takes the least amount of time for matching. SIFT provides high accuracy for Expression test, Pose test and Glasses test. Although SIFT takes more time to perform a match, it provides a better accuracy than all other algorithms in terms of matching.

LEM performs well under variation in Expression and glasses. LEM is the slowest algorithm. This is because of the number of key points taken to perform a match. There is a trade-off between the accuracy and the time taken to match the images in case of LEM. If the number of key points extracted from each image is increased, a better accuracy is obtained at the cost of time.

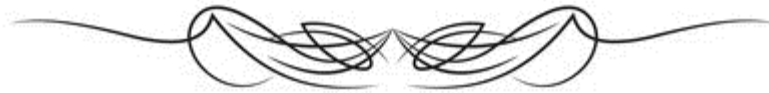
It can be observed that SIFT, LDA and PCA perform well for general test while all algorithms perform badly under variations in illumination. To increase the accuracy for matching images with illumination changes, better preprocessing methods need to be implemented. This can be implemented as part of future work.

8.2. Future work

Due to the time constraints, we chose to implement four existing algorithms for doing a comparative study. But for the future, lot of new features can be added to the developed application to improve on it. A few of those features are:

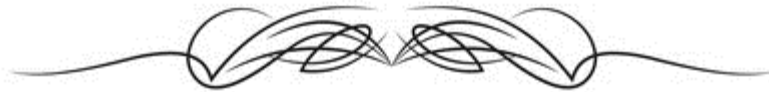
- Complex preprocessing techniques like Local binary patterns (LBPs), histograms, adaptive histograms, etc could be used to enhance the image to a greater extent so as to improve the results.

- More algorithms can be implemented and included in the comparative study.
- Other standard databases can also be used to perform the analysis so as to get a better view of how the algorithm performs on various databases.
- OpenCV 3.0 which is still under development provides more set of ready to use functions for the complex functions used in the existing application. Once openCv 3.0 is released, it can be used to make the existing code much simpler.



CHAPTER 9

BIBLIOGRAPHY

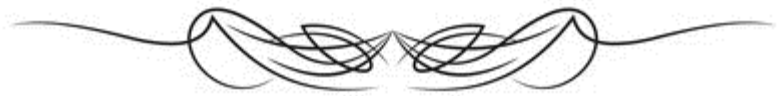


Chapter 9**BIBLIOGRAPHY**

- [1] Rama Chellappa and Wenyi Zhao, “*Face Processing, Advanced Modeling and methods*”
- [2] Andrea F. Abate, Michele Nappi, 2007, “*2D and 3D face recognition: A survey*”, Pattern Recognition Letters, Volume 28, Issue 14, pages: 1885–1906.
- [3] Ashok Rao, S. Nousath, 2010, “*Subspace methods for face recognition*”, Computer Science Review, Volume 4, Issue 1, pages: 1-17.
- [4] Rabia Jafri and Hamid R. Arabnia, 2009, “*A Survey of Face Recognition Techniques*” Journal of Information Processing Systems, Vol.5.
- [5] Wenyi Zhao and Rama Chellappa, “*Image based Face Recognition Issues and Methods*”, a technical report.
- [6] U. K. Jaliya and J. M. Rathod, “*A survey on Human Face Recognition Invariant to Illumination*”, 2013, International Journal of Computer Engineering and Technology, Volume 4, Issue 2, pp. 517-525.
- [7] M. Turk, A. Pentland, Eigenfaces for recognition, 1991, Journal of Cognitive Neuroscience 3.
- [8] Josef Kittler, Kieron Messer and Xuan Zou, “*Illumination Invariant Face Recognition: A survey*”.
- [9] Slobodan Ribaric and Marijo Maracic, “*Eigenphase-based face recognition: a comparison of phase-information extraction methods*”.
- [10] Yongping Li, Chao Wang and Xinyu Ao , “*Illumination Processing in Face Recognition*”.
- [11] Vishwakarma, V.P.; Pandey, S. & Gupta, M.N. “*A novel approach for face recognition using DCT coefficients re-scaling for illumination normalization*”, Dec. 2007, Proceedings of International Conference on Advanced Computing and Communications, pages. 535-539, ISBN: 0-7695-3059-1.

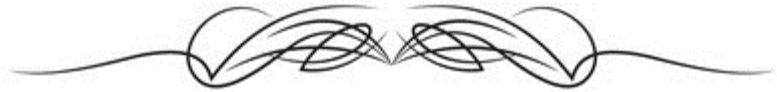
-
- [12] Chen, W.; Er, M.J. & Wu, S. “*Illumination compensation and normalization for robust face recognition using discrete cosine transform in logarithm domain*”, 2006, IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol. 36, No. 2, pages 458-466, ISSN: 1083-4419.
- [13] Shan, S.; Gao, W.; Cao, B. & Zhao, D. “*Illumination Normalization for Robust Face Recognition against Varying Lighting Conditions*”, 2003. Proc. of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures, 17, pp. 157-164
- [14] Bhawna Mittal, Sheetal Garg and Rajesh Garg, “*Histogram Equalization techniques for Image Enhancement*”, 2011, IJECT, Volume 2, Issue 1.
- [15] T. Ojala, M. Pietikainen, and T. Maenpaa, “*Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*”, 2002, IEEE Transaction, Pattern Anal. Mach. Intell., Volume. 24, Issue no. 7 and pp. 971–987.
- [16] David Kriegman, Jeffrey Ho and Kuang-Chih Lee, “*Acquiring Linear Subspaces for Face Recognition under Variable Lighting*”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, Volume 27, Issue No.5.
- [17] A. Georgiades, D. Kriegman, and P. Belhumeur, “*From Few to Many: Generative Models for Recognition under Variable Pose and Illumination,*” IEEE Trans. Pattern Analysis and Machine, Intelligence, vol. 40, pp. 643-660, 2001.
- [18] Mrinal Kanti Bhowmik, Kankan Saha, Sharmistha Majumder, Goutam Majumder, Ashim Saha, Aniruddha Nath Sarma, Debotosh Bhattacharjee, Dipak Kumar Basu and Mita Nasipuri, “*Thermal Infrared Face Recognition – a Biometric Identification Technique for Robust Security System*”.
- [19] J. Kittler, A. Hilton, M. Hamouz, and J. Illingworth, “3D assisted face recognition: A survey of 3d imaging, modelling and recognition approaches”, 2005 In Proc. IEEE conf. CVPR.
- [20] Xin Chen, “PCA-Based Face Recognition in Infrared Imagery: Baseline and Comparative study”, 2003.
- [21] David G. Lowe, “*Distinctive image features from scale-invariant keypoints*” International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

-
- [22] David G. Lowe, "*Object recognition from local scale-invariant features*" , 1999, International Conference on Computer Vision, Corfu, Greece, pp. 1150-1157.
 - [23] Yongsheng Gao, Maylor K.H Leung , "*Face Recognition Using Line Edge Map*", 2002, IEEE Transaction On Pattern Analysis And Machine Intelligence, Vol24, No.6
 - [24] Yongsheng Gao, Maylor K.H. Leung (2000), "*Line Segment Hausdroff distance on face matching*", 2000, pattern Recognition society, 0031-3203/01.
 - [25] Mihir Jain, Suman K. Mitra, Naresh D. Jotwani (2008), "*Eye Detection using Line Edge Map Template*".
 - [26] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. IEEE Trans. PAMI, 25(9):1063–1073, 2003.
 - [27] Weyrauch, B.; Heisele, B.; Huang, J. & Blanz, V. (2004). Component-based Face Recognition with 3D Morphable Models. CVPRW.



CHAPTER 10

SNAPSHOTS



SNAPSHOTS

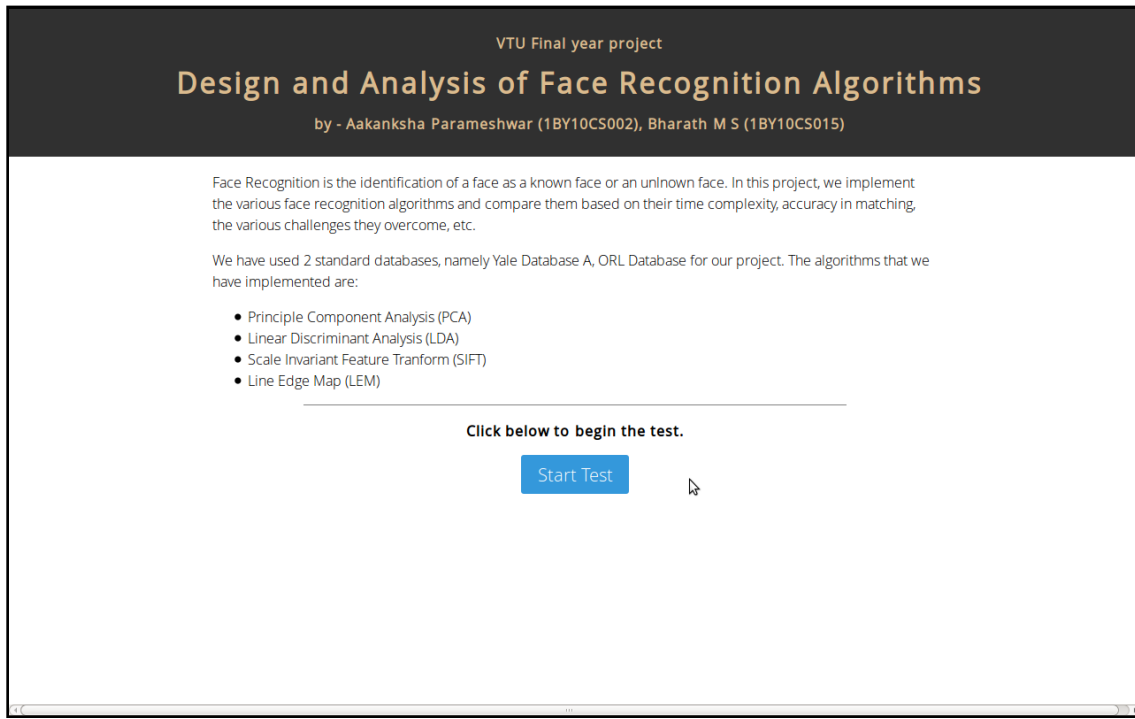


Figure 10.1: Index page of the application

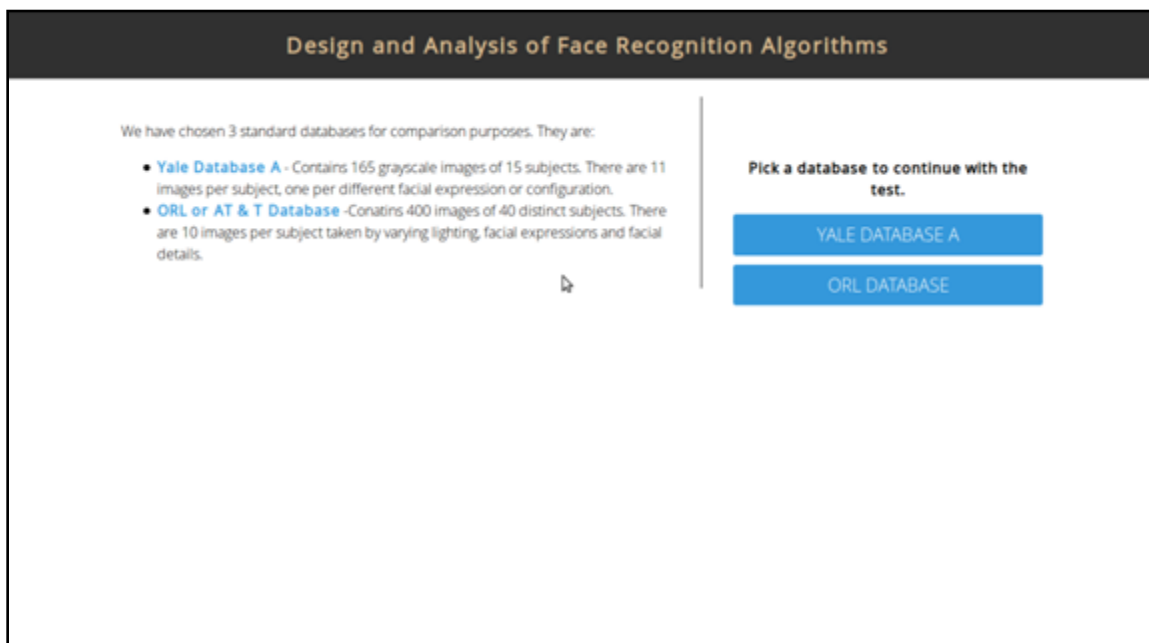


Figure 10.2: Page to choose the database

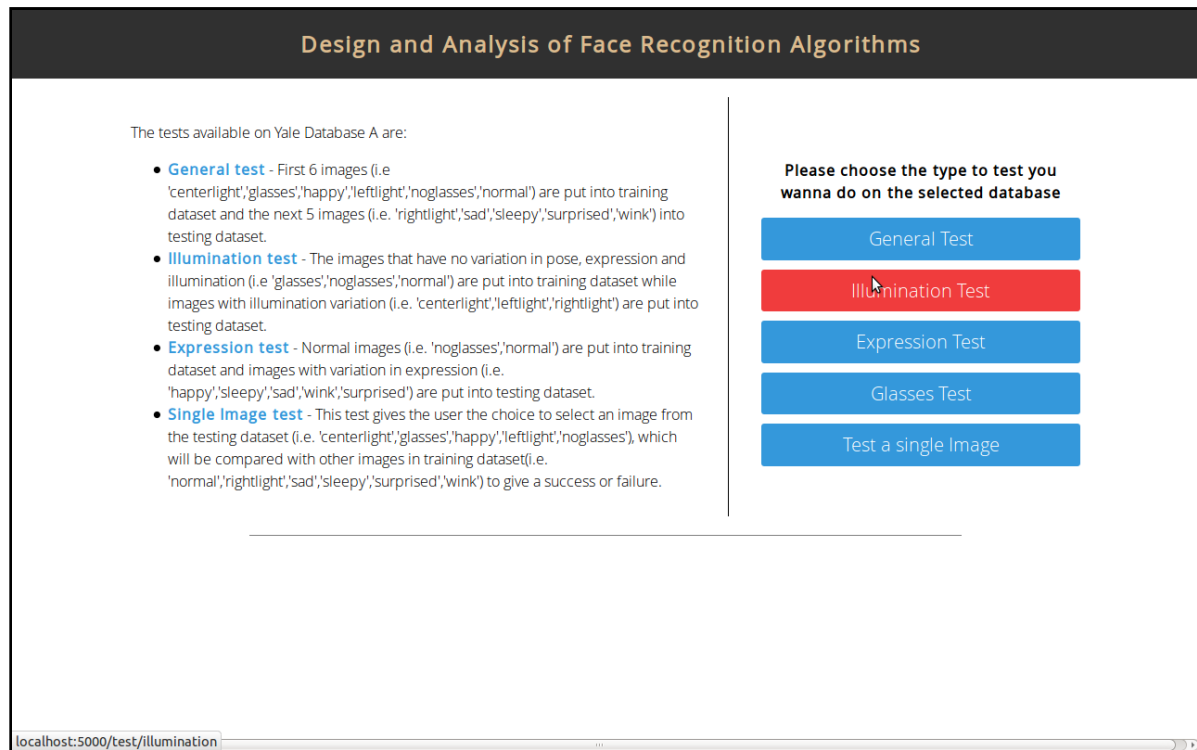


Figure 10.3: Web page to choose tests in Yale database

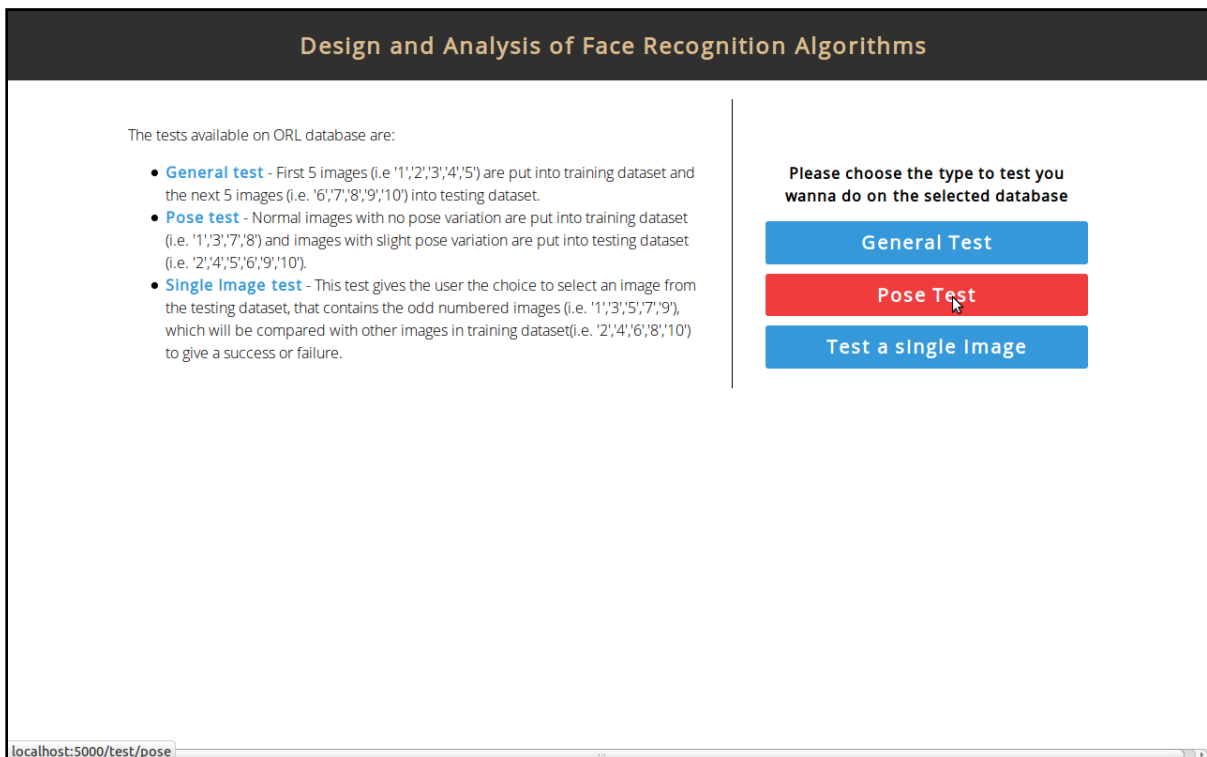


Figure 10.4: Web page to choose tests in ORL database

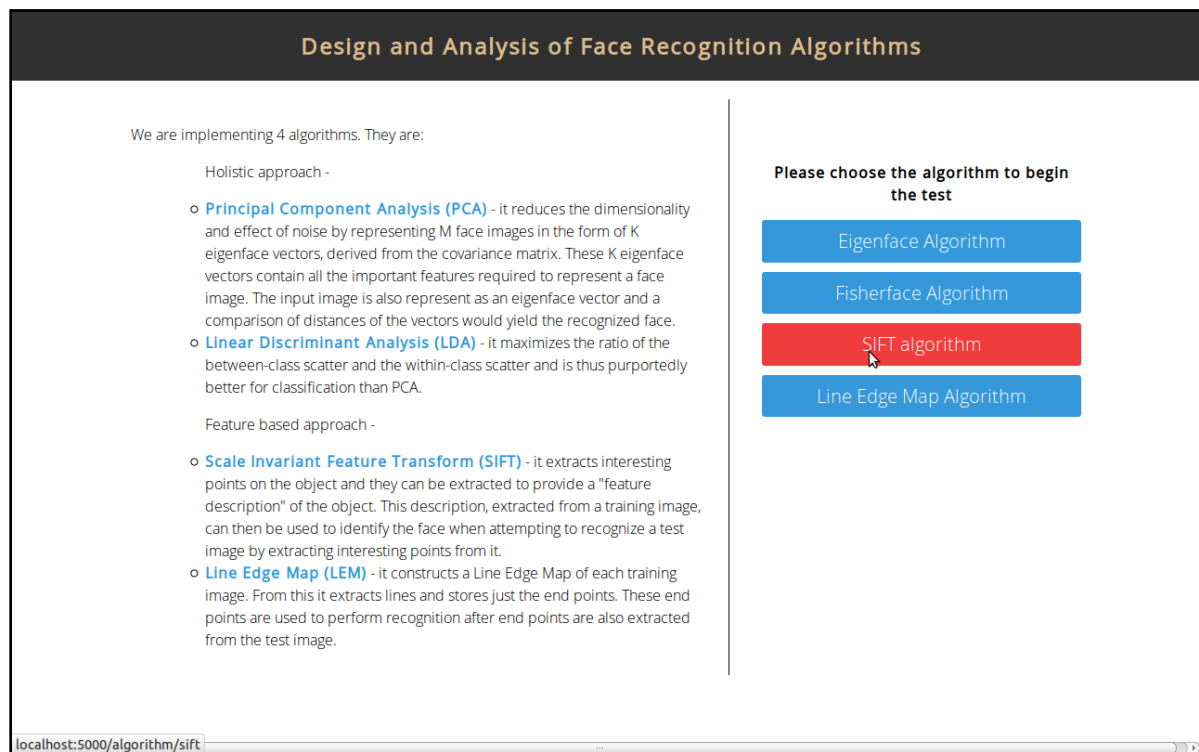


Figure 10.5: Web page to choose algorithms

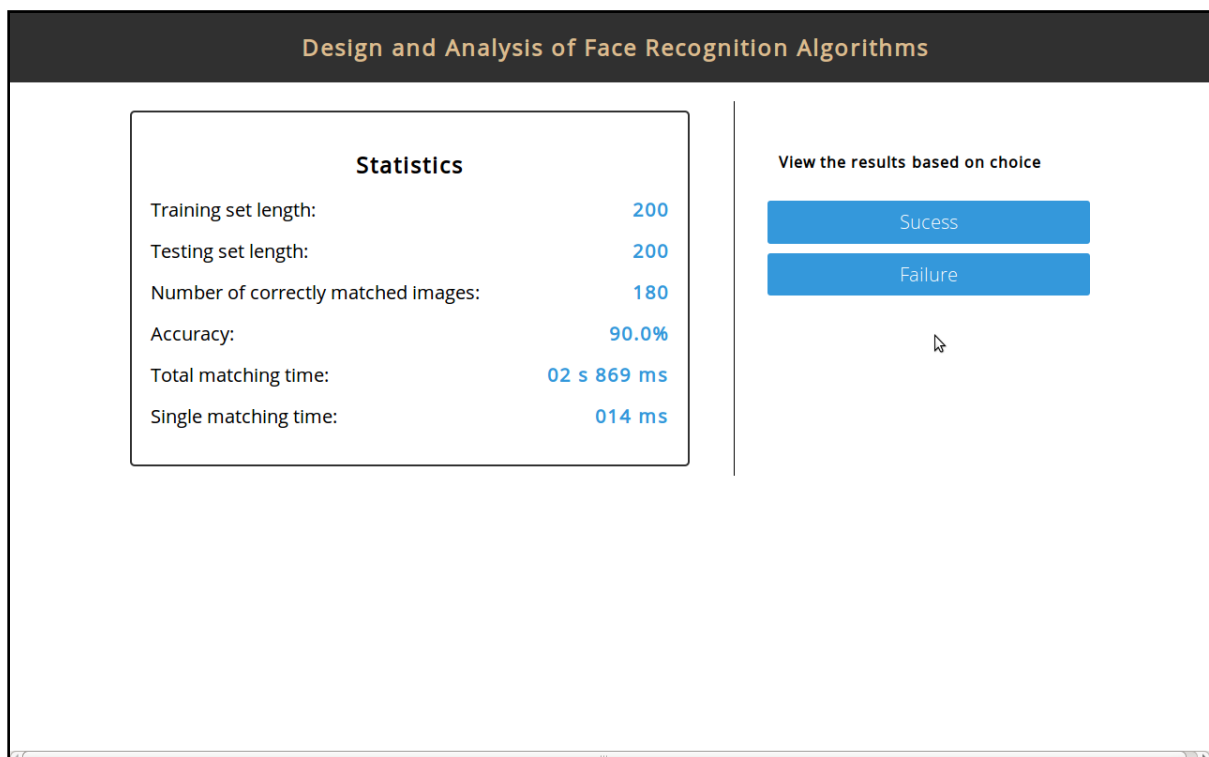


Figure 10.6: Shows the statistical results displayed after a test

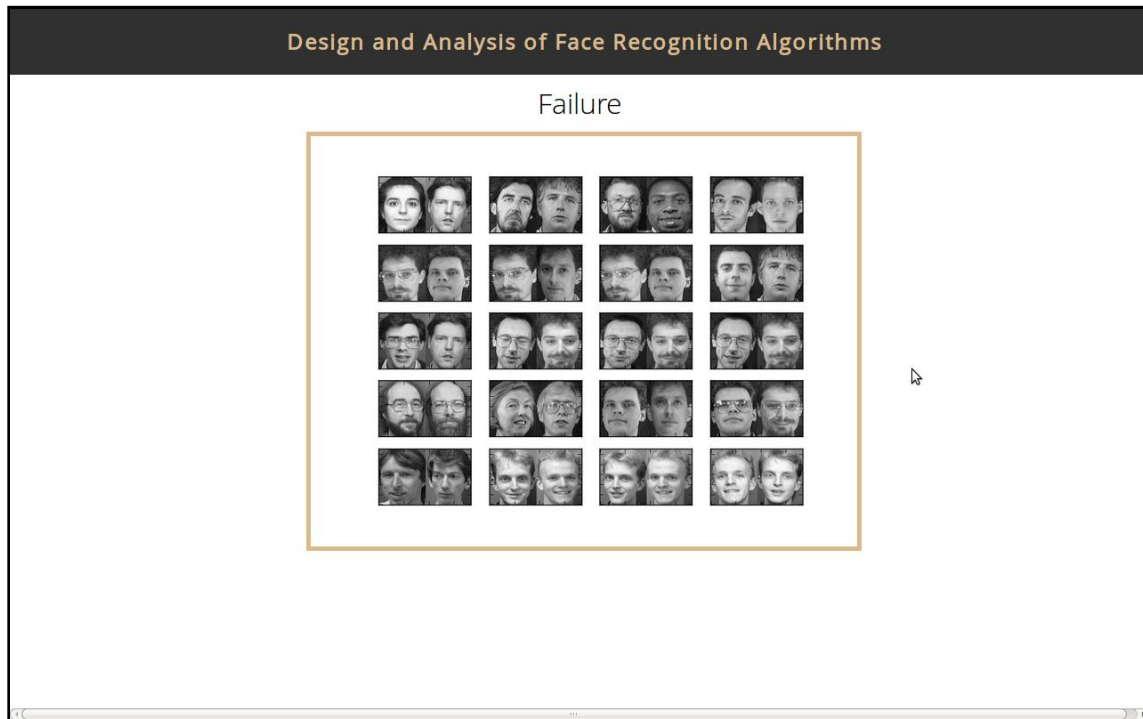


Figure 10.7: Page displaying the failed results

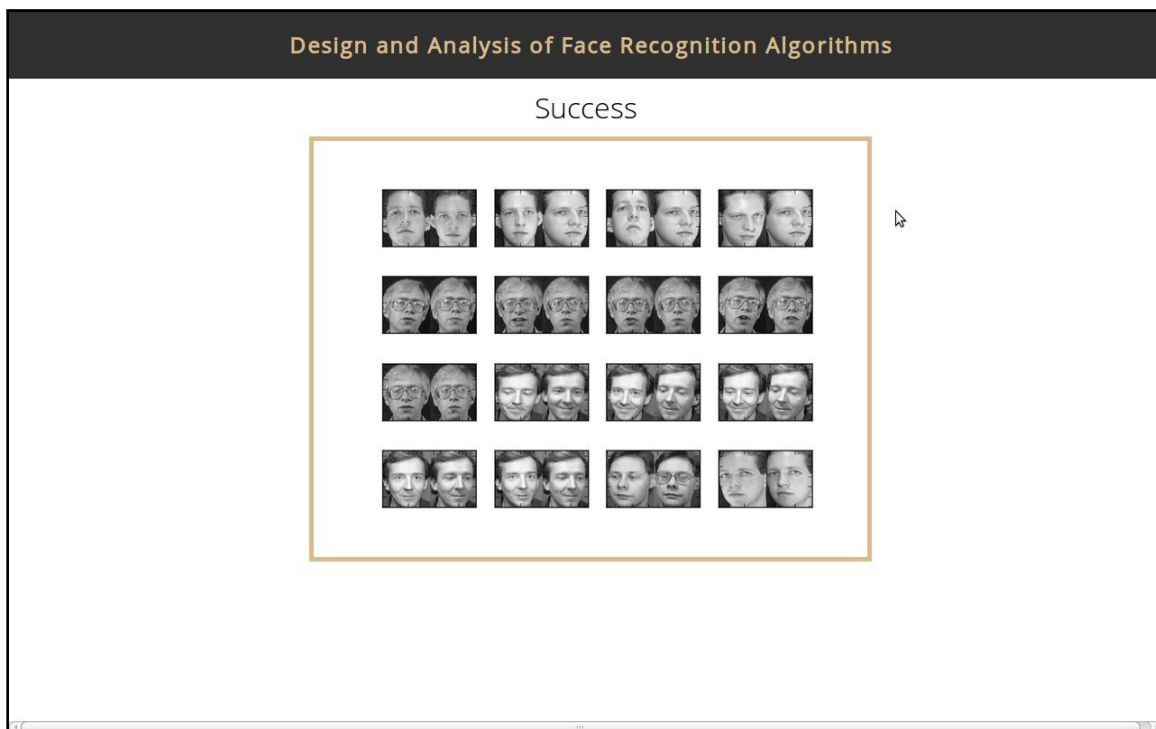


Figure 10.8: Page displaying the succeeded results

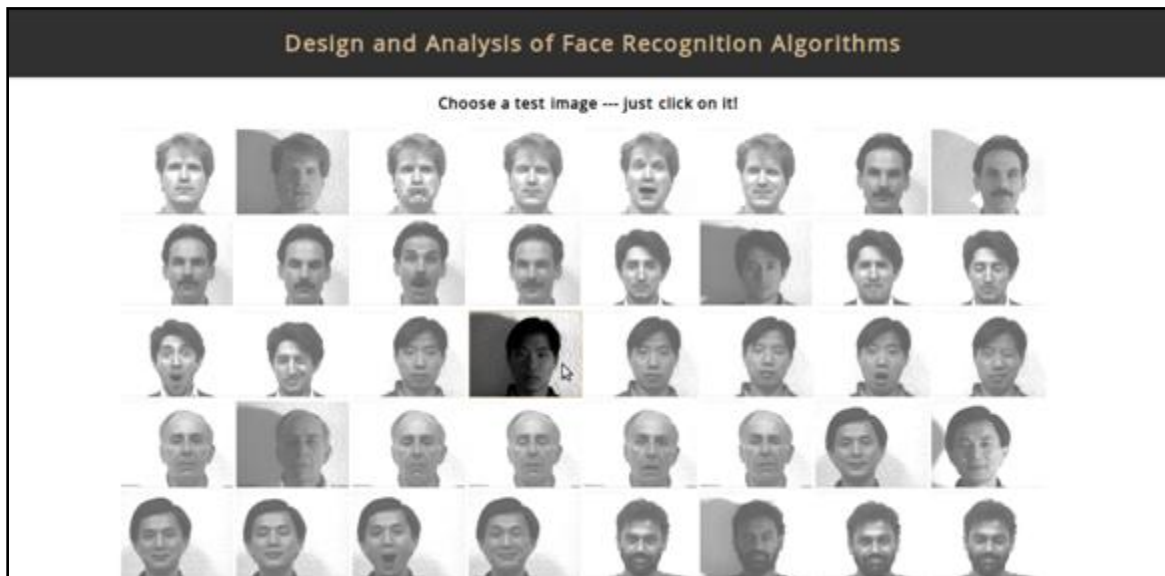


Figure 10.9: UI to choose an image for single image test

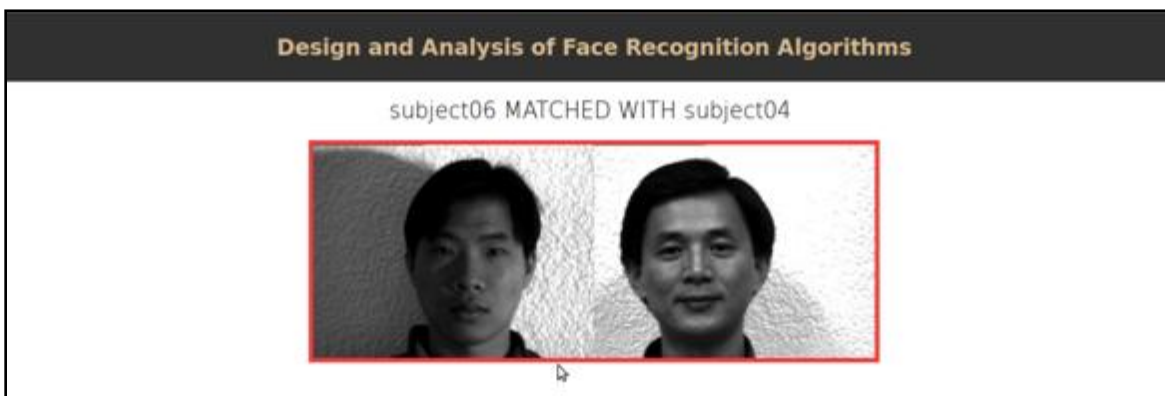


Figure 10.10: Shows failure result for Single image test



Figure 10.11: Shows success result for Single image test