Face Recognition System

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Mr. Arvind Kumar

By

Apoorv Gupta

Roll No.: 111322

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "**Face Recognition System**", submitted by **Apoorv Gupta (111322)** in partial fulfillment for the award of degree of **Bachelor of Technology in Computer Science and Engineering** to **Jaypee University of Information Technology, Waknaghat, dist. Solan** has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Arvind Kumar

(Assistant Professor)
(Grade II)

Acknowledgement

I would like to express my gratitude to all those who gave me the possibility to complete this project. I want to thank the **Department of CSE & IT** in **JUIT** for giving me the permission to commence this project in the first instance, to do the necessary research work.

I am deeply indebted to my project guide **Mr. Arvind Kumar**, whose help, stimulating suggestions and encouragement helped me in all the time of research on this project. I feel motivated and encouraged every time I get his encouragement. For his coherent guidance throughout the tenure of the project, I am very thankful. I feel fortunate to be taught by him.

I am also grateful to **Mr. Amit Singh (CSE Project lab**) for his practical help and guidance.

Date: Apoorv Gupta

Table of Contents

List of Figures

Abstract

Eigenfaces approach for face recognition is implemented as our final project. Face recognition has been an active area of research with numerous applications since late 1980s. Eigenface approach is one of the earliest appearance-based face recognition methods, which was developed by M. Turk and A. Pentland [1] in 1991. This method utilizes the idea of the Principal Component Analysis and decomposes face images into a small set of characteristic feature images called Eigenfaces. Recognition is performed by projecting a new face onto a low dimensional linear "face space" defined by the eigenfaces, followed by computing the distance between the resultant position in the face space and those of known face classes. A number of experiments were done to evaluate the performance of the face recognition system we have developed. The results demonstrate that the Eigenfaces approach is quite sensitive to head/face orientation scale and illumination. At the end of the report, a couple of ways are suggested to improve the recognition rate. The report is organized as follows: the first part provides an overview of face detection algorithms; the second part states the implementation of the Haar Based approach for face detection; Part III focuses on overview of Face Recognition algorithms and Part IV focuses on the implementation part of face recognition.

Chapter 1

Introduction

In this chapter, a face detection approach is presented. Face detection is an essential application of visual object detection and it is one of the main components of face analysis and understanding with face localization and face recognition. It becomes a more and more complete domain used in a large number of applications, among which we find security, new communication interfaces, biometrics and many others.

The goal of face detection is to detect human faces in still images or videos, in different situations. In the past several years, lots of methods have been developed with different goals and for different contexts. I will make a global overview of the main of them and then focus on a detector which processes images very quickly while achieving high detection rates. This detection is based on a boosting algorithm called AdaBoost and the response of simple Haar-based features used by Viola and Jones [1]. The motivation for using Viola's face detection framework is to gain experience with boosting and to explore issues and obstacles concerning the application of machine learning to object detection.

1.1 General Approach

Automatic face detection is a complex problem which consists in detecting one or many faces in an image or video sequence. The difficulty resides in the fact that faces are non-rigid objects. Face appearance may vary between two different persons but also between two photographs of the same person, depending on the lightning conditions, the emotional state of the subject and pose. That is why so many methods have been developed during last years. Each method is developed in a particular context and we can cluster these numerous methods into two main approaches: image based methods and feature-based methods. The first one use classifiers trained statically with a given example set. Then the classifier is scanned through the whole image. The other approach consists in detecting particular face features as eyes, nose etc...

1.2 The chosen approach

We have chosen to work in a common context. The method used is both image based and feature based. It is image based in the sense that it uses a learning algorithm to train the classifier with some well-chosen train positive and negative examples. It is also feature based because the features chosen by the learning algorithm are for lots of them directly related to the particular features of faces (eyes positions, contrast of the nose bridge). The boosting techniques improve the performances of base classifiers by re-weighting the training examples. The learning using Boosting is the main contribution of this face detection. On the other hand, the simple classifiers used for the boosting are simples Haar-like features which permits a fast computation while good detection rates.

1.3 Overview of the project

In the next chapter, an overview of the main existing approaches for Face Detection is given. We first define precisely what the face detection task is and then detail the image based and feature based methods. Chapter 3 explains the developed algorithm. The main theory of Boosting is given as well as the use of the Haar-like masks, a new image representation and an implementation in cascade. Chapter 4 gives the overview of Face Recognition and the detailed explanation of the chosen approach. Chapter 5 focuses on the results and output of the face recognition and face detection algorithm, and finally Chapter 6 concludes the project and discusses about the future work.

Chapter 2

Overview of Face Detection

2.1 Introduction

In the following I will present different aspects of the face processing domain while reviewing the main existing methods. First of all, we need to define what face detection is, why it is an interesting objective and how it can be approached with various methods.

We can define the face detection problem as a computer vision task which consists in detecting one or several human faces in an image. It is one of the first and the most important steps of Face Recognition. Usually, the methods for face recognition or expression recognition assume that the human faces have been extracted from the images, but while the human visual system permit us to find instantaneously faces in our purview indifferently of the external conditions, doing the same automatically with a computer is a quite difficult task.

2.1.1 A brief history

Along face detection, many other parts of Face analysis present useful applications and the number of these applications is increasing considerably nowadays with the evolution of the automatic systems in the life of every one of us. Face Recognition, Face Localization, Face Tracking, Facial expression recognition are the main of these research domains.

Face recognition consists in identifying the people present in images, in other words, we want to assign one name to one detected face. It is used in security systems for example.

- Face localization is the problem of finding precisely the position of one face, whose presence is already known in a single image.
- Face tracking has a goal to follow a detected face in a sequence of images in a real world context in most of the cases.

- Facial expression recognition will try to estimate the affective state of detected people (happiness sadness etc...).

It is clear that the first step for all these problems is to find faces in images. For that various approaches have been developed and that is what will be detailed in this section.

The first face detection systems have been developed during the 1970's but the computation limitations restricted the approaches to techniques which could be efficient in only few applications as passport photograph identification for instance. It is only since the beginning of the 1990's that more elaborated techniques have been built with the progress in video coding and the necessity of face recognition. In the past years, lots of different techniques have been developed, in such a proportion that today we can count not less than 150 different methods.

2.1.2 Face detection difficulties

If automatic face detection has not been developed before, it is because it is particularly hard to build robust classifiers which are able to detect faces in different image situations and face conditions even if it seems really easy to do this with our human visual system. In fact, the object "face" is hard to define because of its large variability, depending on the identity of the person, the lightning conditions, the psychological context of the person etc...

The main challenge for detecting faces is to find a classifier which can discriminate faces from all other possible images. The first problem is to find a model which can englobe all the possible states of faces. Let's define the main variable points of the faces:

- **The face global attributes**. We can extract some common attributes from every face. A face is globally an object which can be estimated by a kind of ellipse but there are thin faces, rounder faces... The skin color can also be really different from one person to one another.

- **The pose of the face**. The position of the person in front of the camera which has been used to acquire the image can totally change the view of the face: the frontal view, the profile view and all the intermediate positions, upside down.
- **The facial expression**. Face appearance depends highly on the affective state of the people. The face features of a smiling face can be far from those of an indifferent temperament or a sad one. Faces are non-rigid objects and that will limit considerably the number of detection methods.

The background composition is one of the main factors for explaining the difficulties of face detection. Even if it is quite easy to build systems which can detect faces on uniform backgrounds, most of the applications need to detect faces in any background condition, meaning that the background can be textured and with a great variability. So our two class classification task is to assign an image to the face class or the non faces class. Given a set of examples, we can extract some properties of faces for representing the face class but it is impossible to find properties which can represent all the non-face class.

In this context, various approaches have been taken to detect faces in images. But as the face detection task is quite complex, each method is built in a precise context and we will now review the main existing methods. The next sections detail the two main face detection approaches:

- **Image- Based methods** which are built given a set of examples and uses a sliding window to perform the detection.
- **Geometrical- Based** methods which take in account geometric particularity of face structures.

2.2 Image Based Detection

2.2.1 Introduction

The Image-Based methods have been doubtless the more used until today. They are "Image-Based" because they are built using example images in opposition to some "template-methods" which need a prior knowledge about faces. In order to extract the features from some training examples, we will need to follow a statistical learning approach or other machine learning algorithms. The principle is to learn a face and a non-face distribution, given a set of positive and negative examples. For this, we will naturally be placed in a probabilistic context : An image or any input data is considered as an random variable x and the two classes face and Non face are characterized by their conditional density functions: p(x | f ace) and p(x | non f ace). It is obvious that these density functions are unknown and one of the main goals is to approximate them in order to discriminate faces and non faces. In this probabilistic approach, many different methods exist, among which Eigen faces, Fisher's Linear Discriminant and Neural network or support vector machines etc...

The main difficulty in this approach is that the example dimension, i.e. the dimension of x is often high so an important step will be to reduce this example space in order to find a discriminant function which separates positive and negative examples.

2.2.2 Eigenfaces

2.2.2.1 Definition

The principle of face detection using Eigenfaces is to extract these features from a set of images by Principal Components Analysis (PCA) and estimate if the extracted Eigenfaces correspond to typical face pattern. In fact all input images can be represented by a weighted vector of Eigenfaces in the Eigen space and the challenge is to determine if this linear combination is closer to one class or to the other.

2.2.2.2 Principal Component Analysis

The first step for this Eigenfaces classification is to extract the Eigenfaces from the original images. For this, the Principal Components Analysis (PCA) is used. PCA which is also known as the Karhunen-Loeve method reduces the input space dimensionality by applying a linear projection that maximize the scatter of all projected samples.

One serious point is that the main variance cause in an object class is the lightning variation. The optimal linear transformation given by PCA has the drawback to focus on components representing the illumination changes. One of the correction methods is to let out the first principal Eigenfaces considering that they contain almost all the variations due to lightning.

2.2.3Neural Network, SVM, HMM, Winnow

2.2.3.1 Neural Network

One of the best face detection system in term of false positive rate and detection rate is a Neural Network-Based face detection developed by Rowley [6]. It uses a retinally connected neural network which decides if a scanned windows is a face or not. The face detection system can be divided in two main steps:

- **A neural network-based filter.**

  The input of this first stage is a pre-processed square image (20x20) and the output of the neural network is a real value between -1 and +1. The preprocessing and neural network steps are shown in the figure:

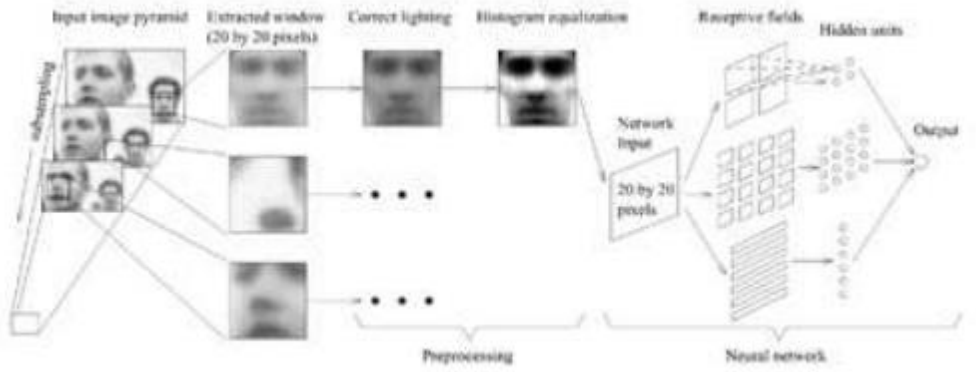

Fig.2.2.3.1 *(Neural Network-based face detection)*

The original image is decomposed in a pyramid of images (by simple sub-sampling) in order to detect faces larger than the basic detector size. The Receptive fields and Hidden units are shown in figure. There are three types of hidden units to represent local features that represent well faces. This first stage yields good detection rates (if the training set is particularly well chosen) but it remains still an insufficient false positive rate.

- **Arbitration and merging overlapping detections.**

  In order to improve this high false positive rate, two neural networks are trained with various initializations (in term of non-face training set, weight initialization and order presentation). These two networks are built by the methods of the first step.

  Even if the two networks have individually bad false positive rates, the false alarms may differ from one network to the other. Hence, an integration of the result using a simple arbitration strategy improve significantly the detection results. The most common of these strategy is called ANDing. A window if definitively classified as face only if the two neural networks have detected it.

This method using neural networks have good results in term of false positive rate and detection rate, but one limitation is that the quality of the detection depends highly on the coherence of the training sets and on the tuning of the neural networks which has lots of parameters.

2.2.3.2 Hidden Markov Model

The principle is to divide a face pattern into several regions such as forehead, eyes, nose, mouth and chin. A face pattern is then recognize if these features are recognize in an appropriate order. In other words, a face pattern is a sequence of observation vectors where each vector is a strip of pixels. An image is scanned in a precise order and an observation is taken by block of pixels. The boundaries between strips of pixels are represented by probabilistic transitions between states and the image data within a region is modeled by a multivariate Gaussian distribution. The output states correspond to the class to which the observation belong.

2.3 Geometrical- Based Detection

2.3.1 Introduction

The previous statistical methods are based on a learning to obtain a face model from one positive and one negative data set. They are not directly correlated to the particular geometrical features of a typical face. Some other methods are in such a point of view. They are called Geometrical- based or Feature-based. Many approaches have been taken is this large area of feature-based detection and we can distinguish:

- **The top-down approach**: One model is computed for one scale.
- **The bottom-up approach**: The faces are searched in an image by the presence of facial features.

The main advantage of this geometric approach is that the face detection is not restricted to frontal faces. In fact the main face features (eyes nose, skin color etc...) are present independently of the pose and the lighting conditions.

2.3.2 Top-down methods

This category includes all the methods that used a multi scale approach. The great majority of them use the skin color to find faces in images. The existing system use several segmentation algorithms to extract faces from the images.

The skin color is maybe one of the features the first noticed by the human visual system. Many methods use different color spaces. The main advantage of this approach is that the face detection is very fast. However, there is one important issue: lots of problems appears if the background contains faces of the skin color.

Yang and Ahuja [5] have built their system in this sense. Although the human skin color seems to change from one example to one other, the effective variation is more luminance than the color itself. The distribution is modeled by a Gaussian distribution. All the pixel are tested and we attribute them the skin color if their corresponding probability is greater than a given threshold. Finally, a region is declared as face if more than 70 percent of its pixels have the skin color.

2.3.3 Bottom- up methods

The principle is to find invariant features of faces. By invariant, we mean invariant by scaling, poses, lighting conditions and other variations. The common and natural features that are usually extracted are the eyes, the nose the mouth and the hair line. Any edge detector might be used to extract them. A bottom up method try to find this features in an original image and then they are grouped according to their geometrical relationships. The difference between the methods in this bottom-up approach resides in the way to choose the features and how to establish the links between them.

2.4 Evaluation Difficulties

As the definition of detecting faces in images is really simple: determine whether or not there are any faces in images and, if present, return the image location and extend for each face, we can think that it is easy to evaluate the performances of a face detector. However many parameters have to be taken in account to do this. How can we measure the goodness of a detection? How do we have to integrate the false alarms (how do we have to consider the false positive rate?) What about the detection speed? Several such questions make hard the face detection evaluation.

It would be interesting to compare the existing methods in face detection but the major problem is that every method is made in a particular context and today there are still no standards for face detection evaluation that will make easier the future research work about face detection.

The first step in detection evaluation is to use a common testing set which contains a large variety of situations. The most common used set is probably the CMU testing set which contains many faces manually labeled. Then we usually use the detection rate over false positive rate ratio to characterize the performances even if the number of

false alarms is directly related to the way how the images are scanned (more precisely the number of sub-windows scanned). The image based techniques are quite efficient regarding the frontal face detection. The detection rate reaches more than 90% with at most several tens of false alarms in a typical sized image but the main limitation of the image based methods is that the faces detected will slightly match with the training examples. Thus it is difficult for example to include in the training set faces at many different poses, with both rotations in and out of plane. The geometrical approaches are more robust in term of face pose, i.e. the face orientation in front of the acquisition system but they generally give worse detection results. Both the segmentation part and the feature extraction are critical points. The use of the color information needs is not really representative of faces because lots of objects in the background may have the human skin color.

The speed of the detector is without any doubt the parameter the most difficult to take into account. Each method has its own speed and it is difficult to determine the speed performances: it depends on the scanning method and on the way it is implemented.

So it has been shown that a lot of different approaches are available but face detection is still an open task. Many solutions are possible taking into account the results of the existing methods. The main promising approach seems to be combined approaches of image based and feature based methods.

Chapter 3

Fast Face Detection Using AdaBoost

3.1 Introduction

3.1.1 Choice of the Method

In this third chapter, we discuss about a face detection based on a boosting algorithm which yield good decision rates. This detector is highly inspired by the Robust Real-time Object Detection of Viola and Jones. We have chosen to build a model using a statistical learning given some positive and negative examples. A learning algorithm trains a classifier by selecting visual features, so we will discuss why this chosen algorithm is appropriate for face detection and explain how it works.

3.1.2 Context of the frontal face detection

Before going into details, we just remark that every face detection method is designed in a particular context that is why it is not always easy to compare the results between them. Some detectors have for only goal to have a detection rate as near as possible from 100% but our project is a little bit in a different context: even if we naturally want reach good detection rates, we want to build a real-time oriented detector. So the goal is to detect all the faces (or almost all of them) even if this means we have to accept a higher false positive rate (non-face images labeled as face by the detector). This choice is only in order to respect most of applications which need for example to detect all the people in front of a video camera. (Video surveillance for instance).

3.1.3 Why Boosting and Haar features?

The chapter 2 presented the main approaches available to build a face detection system. Now the context of our face detection is given, we can explain why we choose this approach using a boosting learning algorithm and simple Haar features. As we want to detect faces in various background and principally low resolution faces, it would be improper to use purely geometrical methods. In fact the main advantage to these geometrical methods is the geometric invariant properties. We are not interested by them because we have chosen to stay in a frontal face detection context. So it is quite naturally that we have oriented our choice towards learning algorithms. Boosting is a powerful iterative procedure that builds efficient classifiers by selecting and combining very simple classifiers. Good theoretical results have been demonstrated, so we have some theoretical guarantees for achieving good detection rates. This idea is interesting in the sense that a combination of simple classifiers may intuitively give a rapid detection without deteriorating the detection rates. So it seems to be one of the best compromise between efficiency in term of detection and speed.

3.1.4 Overview of the detection

This new method given by Viola [1] is a combined method of more traditional ones like geometrical and image based detection. It is a geometrical in the sense that it uses general features of human faces: position of particular features among which the eyes the nose and the mouth. We will not try to extract particular face features: It is only an a-posteriori observation in the sense that the selected Haar-like masks are effectively representing particular facial features but it is not our decision.

Viola [1] has developed this face detector in July 2001 and he was inspired by the work of Papageorgiou. It seemed to be the fastest and the most robust and it is still today. The speed of the detection is notably given by the simplicity of the features chosen and the good detection rates are obtained by the use of the fundamental boosting algorithm AdaBoost which selects the most representative feature in a large set.

Let us look at the main steps of the fast face detector that will be explored in the next sections.

The detector consists in scanning an image by a shifting window at different scales. Each sub-window is tested by a classifier made of several stages (notion of cascade). If the sub-window is clearly not a face, it will be rejected by one of the first steps in the cascade while more specific classifier (later in the cascade) will classify it if it is more difficult to discriminate.

The first contribution is the choice of the features that describe the faces. The principle of the detection is to apply successively simple classifiers to combine them in a final strong classifier. The choice of these features is fundamental for the performances of the detection. The difficulty is to find masks simple enough to permit a fast classification but characteristic enough to discriminate faces and non faces. A good compromise for that is obtained by the use of reminiscent of Haar Basis functions. In fact the feature response is nothing more than the difference of two, three or four rectangular regions at different scales and shapes. To improve the computation time of these features, we introduce a new image representation called Integral Image which permits to compute a rectangle area with only 4 elementary operations, i.e. additions and subtractions.

Then, as we have a large set of features at disposition, AdaBoost is used to select a small set of them to construct a strong final classifier. We want to keep only the features which separates the best positive and negative examples. At each selection step, a weak classifier (one feature) is selected so AdaBoost provides an effective learning algorithm and strong bounds on generalization performance. Finally, the third important contribution is the cascade implementation which focuses the detection on critical regions of interest. Thus, it first eliminates quickly regions where there are no positive examples and then, the more we go down in the cascade process, the more specific the classifiers are and so almost only faces are detected.

For example, the first stage of Viola's detector is a combination of only two features which rejects 60% of negative examples and it provides a false negative rate of 1% with only 20 simple operations.

3.2 Features and Integral Image

This section presents the features used in our statistical face detection. Human faces are objects particularly hard to model because of their significant variety in color and texture and there are no constraints on the background. In fact, if we want to build a model which is able to take in account this face variability without identifying cluttered backgrounds, it will not work to use such as maximum likelihood methods for example.

3.2.1 Overview of the existing face models.

Due to the context of our face detection, methods like maximum likelihood are particularly not efficient. We will thus focus on example based face models to train a significant classifier. Many descriptive features could be used to train a classifier by Boosting. The next subsections explain some of them that have been used recently. We can distinguish to main methods that seems to be the more efficient:

- Pixel- based models
- Haar- like features model

Let's define the Haar-based model:

3.2.2 Haar- like features

Comparing these face modeling methods and taking into account the specific needs of our application, we arrived to conclusion that a feature based method would be more appropriate rather than pixel based. There are many motivations for using features (some reminiscent of Haar Basis functions) than pixels directly as Pavlovic [5]. The most common reason is that features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. And as we will see, these features can operates much faster than pixel-based system.

3.2.2.1 Rectangular Haar Features

In our face detection system, very simple features are used. We use some reminiscent of Haar Basis. Recall that the wavelet function corresponding to Haar wavelet is:

$$\text{Ʉ}(X) = \begin{cases} 1 \text{ if } 0 \le x < ½ \\ -1 \text{ if } ½ \le x < 1 \\ 0 \text{ otherwise} \end{cases}$$

There are three kinds of Haar-like features. The value of a two-rectangle feature is the difference between the sum of the pixels within two rectangular regions. The regions have the same size and shape and are horizontally or vertically adjacent.(see figure 3.2). A three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a center rectangle. Finally, a four-rectangle feature computes the difference between diagonal pairs of rectangles.

Figure shows the different two- three- and four- rectangles prototypes used by our detector.

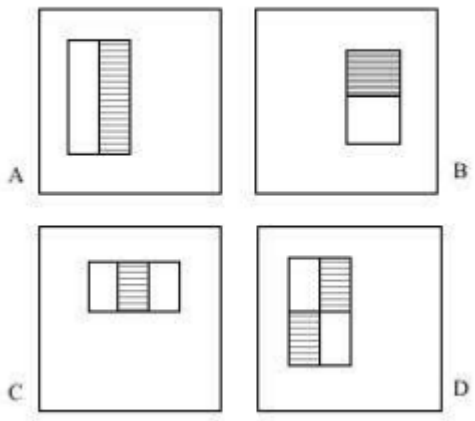

Fig. 3.2.2.1*(Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two- rectangle features are shown in (A) and (B). Figure (C) shows a three- rectangle features, and (D) a four- rectangle feature)*

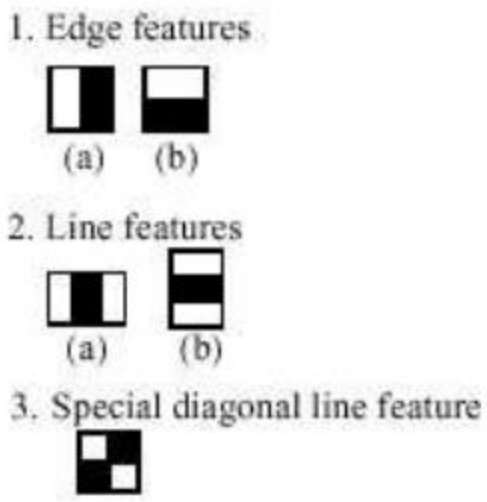

Fig. 3.2.2.2 *(Feature prototype of simple Haar- like. Black areas have negative and white areas positive weights.)*

Number of features:

The number of features derived from each prototype is quite large and differs from prototype to prototype and can be calculated as follows. Let H and W be the size of H x W pixels window and let *w* and *h* be the size of one prototype inside the window as shown in figure.

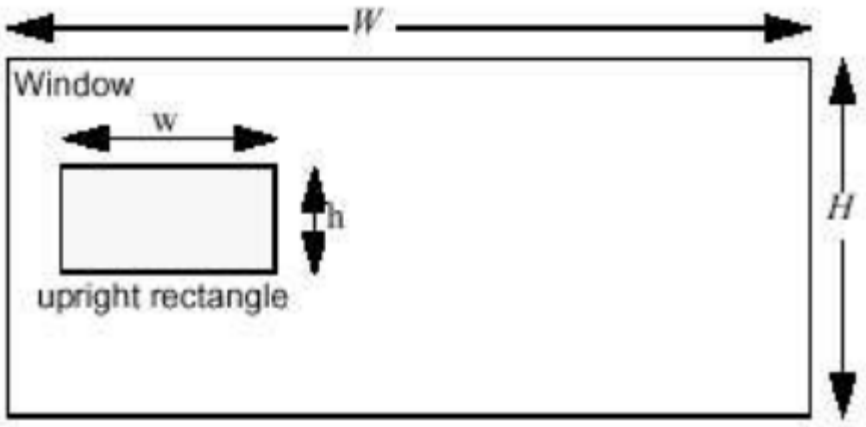

Fig. 3.2.2.3 *(Upright rectangle in a window)*

Let X = W/ w and Y = H/h be the maximum scaling factors in *x* and *y* direction. An upright feature of size *w x h* then generates features for an image of size W x H:

$$X.\ Y\ (W + 1 - w\ (X+1)/2)\ (H + 1 - h\ (Y + 1)/2)$$

3.2.3 Integral Image

We now know that we need Haar-like features to train the classifiers. The goal of this part is to introduce a new image representation called Integral Image which yields a fast feature computation.

The value of the Integral Image at the coordinates (x, y) is the sum of all the pixels above and to the left of (x, y), including this last point as shown in Figure.

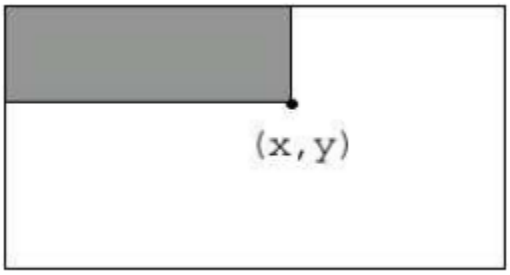

Fig. 3.2.3 *(The Integral Image representation. The Integral image value at the point (x, y) is the sum of all the pixels above and to the left of (x, y).)*

Let *ii* be the integral image of the initial image *i* and *ii(x, y)* the value of the integral image at the point (x, y).

We can define the integral image *ii* by:

$$\sum_{\substack{x' \le x, \\ y' \le y}} i(x', y')$$

The main advantage of using such a representation is that any rectangular sum in the original image can be computed in four array references in the integral image. The

difference between two rectangular sums can be computed in eight references. Therefore, computing a feature is only difference of two, three or four rectangular sums.

The two rectangle features are computed with six references because the two rectangles are adjacent. The three rectangle features need eight references and four rectangle array only nine.

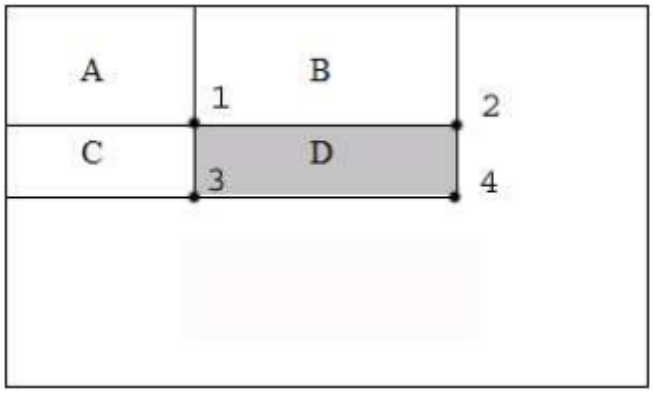

Fig. 3.2.3.1 *(The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is of the pixels in rectangle A. The value at location 2 is A+B, at location 3 is A+C, and at location 4 is A+B+C+D. The sum within D can be computed as 4+1-(2+3).)*

3.3 Learning with AdaBoost

3.3.1 Introduction

Considering a mono-stage classifier and given a set of features, we can build a face detector by applying all the masks at each image location (each shift and each scale). For this many different learning methods could be used.

Moreover, we have a complete set of features which is far larger than the number of pixels, so even if the features responses are very simple to compute (notably with the integral image representation), applying the all set of features would be two expensive in time. The next stage in the building of the face detector is thus to use a learning function which selects

a small set of these features: the ones which separates the best positive and negative examples. The resulting final classifier would be a simple linear combination of these few Haar-like features.

For this, we will discuss in this section about an algorithm called AdaBoost (Adaptive Boosting) which has two main goals:

- Selecting a few set of features which represents as well as possible faces.
- Train a strong classifier with a linear combination of these best features.

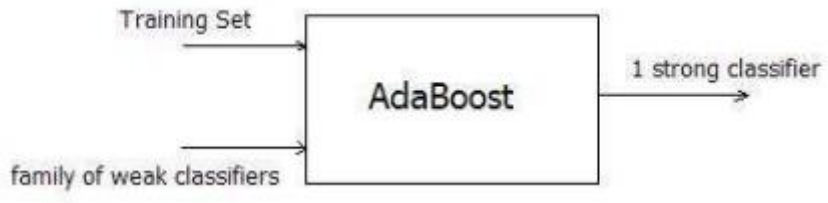

Fig. 3.3.1 *(Basic scheme of AdaBoost)*

In the following subsections, it is explained why we chosen this algorithm instead of more classical ones and then why AdaBoost is efficient and how it can be adopted to face detection.

Possible algorithms: Given a set of features and a training set of positive and negative examples, any machine learning approach could be used to learn a classification function. Here are the main possible approaches which could be used to train a classifier:

- Mixture of Gaussian model
- Simple image features and neural network
- Support Vector Machine
- Winnow learning procedure.

Why AdaBoost: AdaBoost is an efficient boosting algorithm which combine simple statistical learners while reducing significantly not only the training error but also the more elusive generalization error. As all the learning functions, it presents advantages and drawbacks which are exposed here:

Advantages:

- **No prior knowledge.** AdaBoost is an algorithm which only needs two inputs: a training dataset and a set of features (classification functions). There is no need to have a prior knowledge about face structure. The most representative features will automatically be selected during the learning.
- **Adaptive algorithm.** At each stage of the learning, the positive and negative examples are tested by the current classifier. If an example is misclassified, that means that it is hard to classify i.e. it cannot be clearly assigned in the good class. In order to increase the discriminant power of the classifier these misclassified examples are up-weighted for the next algorithm iterations. So the easily classified examples are detected in the first iterations and will have less weight in the learning if the next stages to focus on the harder examples.
- **The training error theoretically converge exponentially towards 0.** Given a finite set of positive and negative examples, the training error reaches 0 in a finite number of iterations.

Drawbacks:

- **The result depends on the data and weak classifiers.** The quality of the final detection depends highly on the consistence of the training set. Both the size of the sets and the interclass variability are important factors to take in account. Other way, the types of basic classifiers which are combined have some influence on the result. The only need for all the basic functions is to be better than random selection but if we want to achieve good detection rates in a cogent number of iterations, they have to be as well chosen as possible.

- **Quite slow training.** At each iteration step, the algorithm tests all the features on all the examples which requires a computation time directly proportional to the size of the features and examples sets. Imagine that the training set has many thousands of positive and negative examples and a complete set of features. However, the computation time is increased linearly with the size of the both sets.

Chapter 4

Introduction to Face Recognition

4.1 Face Recognition

Face is a complex multidimensional structure and needs good computing techniques for recognition. The face is our primary and first focus of attention in social life playing an important role in identity of individual. We can recognize a number of faces learned throughout our lifespan and identify that faces at a glance even after years. There may be variations in faces due to aging and distractions like beard, glasses or change of hairstyle.

Face recognition is an integral part of biometrics. In biometrics basic traits of human is matched to the existing data and depending on result of matching identification of a human being is traced. Facial features are extracted and implemented through algorithms which are efficient and some modifications are done to improve the existing algorithm models.

Computers that detect and recognize faces could be applied to a wide variety of practical applications including criminal identification, security systems, identity verification etc. Face detection and recognition is used in many places nowadays, in websites hosting images and social networking sites. Face recognition and detection can be achieved using technologies related to computer science.

Features extracted from a face are processed and compared with similarly processed faces present in the database. If a face is recognized it is known or the system may show a similar face existing in database else it is unknown. In surveillance system if an unknown face appears more than one time then it is stored in database for further recognition. These steps are very useful in criminal identification. In general, face recognition techniques can be divided into two groups based on the face representation they use appearance-based, which uses holistic texture features and is applied to either whole-face or specific regions in a face image and feature-based, which uses geometric facial features (mouth, eyes, brows, cheeks etc), and geometric relationships between them.

4.2 Literature Survey

4.2.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) was invented in 1901 by Karl Pearson. PCA is a variable reduction procedure and useful when obtained data have some redundancy. This will result into reduction of variables into smaller number of variables which are called Principal Components which will account for the most of the variance in the observed variable.

Problems arise when we wish to perform recognition in a high-dimensional space. Goal of PCA is to reduce the dimensionality of the data by retaining as much as variation possible in our original data set. On the other hand dimensionality reduction implies information loss. The best low-dimensional space can be determined by best principal components.

The major advantage of PCA is using it in eigenface approach which helps in reducing the size of the database for recognition of a test images. The images are stored as their feature vectors in the database which are found out projecting each and every trained image to the set of Eigen faces obtained. PCA is applied on Eigen face approach to reduce the dimensionality of a large data set.

4.2.2 Eigen Face Approach

It is adequate and efficient method to be used in face recognition due to its simplicity, speed and learning capability. Eigen faces are a set of Eigen vectors used in the Computer Vision problem of human face recognition. They refer to an appearance based approach to face recognition that seeks to capture the variation in a collection of face images and use this information to encode and compare images of individual faces in a holistic manner.

The Eigen faces are Principal Components of a distribution of faces, or equivalently, the Eigen vectors of the covariance matrix of the set of the face images, where an image with N by N pixels is considered a point in N 2 dimensional space. Previous work on face recognition ignored the issue of face stimulus, assuming that predefined measurement were relevant and sufficient. This suggests that coding and decoding of face images may give

information of face images emphasizing the significance of features. These features may or may not be related to facial features such as eyes, nose, lips and hairs. We want to extract the relevant information in a face image, encode it efficiently and compare one face encoding with a database of faces encoded similarly. A simple approach to extracting the information content in an image of a face is to somehow capture the variation in a collection of face images.

We wish to find Principal Components of the distribution of faces, or the Eigen vectors of the covariance matrix of the set of face images. Each image location contributes to each Eigen vector, so that we can display the Eigen vector as a sort of face. Each face image can be represented exactly in terms of linear combination of the Eigen faces. The number of possible Eigen faces is equal to the number of face image in the training set. The faces can also be approximated by using best Eigen face, those that have the largest Eigen values, and which therefore account for most variance between the set of face images. The primary reason for using fewer Eigen faces is computational efficiency.

4.2.2.1 Eigen Values and Eigen Vectors

In linear algebra, the eigenvectors of a linear operator are non-zero vectors which, when operated by the operator, result in a scalar multiple of them. Scalar is then called Eigen value (λ) associated with the eigenvector (X). Eigen vector is a vector that is scaled by linear transformation. It is a property of matrix. When a matrix acts on it, only the vector magnitude is changed not the direction.

4.2.2.2 Face Image Representation

Training set of (m) images of size NxN are represented by vectors of size N^2.

Each face is represented by $\Gamma_1, \Gamma_2, \Gamma_3, \ldots \Gamma_M$.

Feature vector of a face is stored in a N x N matrix. Now, this two dimensional vector is changed to one dimensional vector

For Example: - $\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix}$

Each face image is represented by the vector Γ_i.

$\Gamma 1 = \begin{pmatrix} 1 \\ -2 \\ 1 \\ -3 \end{pmatrix}$ $\Gamma 2 = \begin{pmatrix} 1 \\ 3 \\ -1 \\ 2 \end{pmatrix}$ $\Gamma 3 = \begin{pmatrix} 2 \\ 1 \\ -2 \\ 3 \end{pmatrix}$ ..... $\Gamma M = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix}$

4.2.2.3 Mean and Mean Centered Images

Average face image is calculated by

$\Psi = (1/M)\sum_{i=1}^{M} \Gamma i$

$$\begin{bmatrix}1\\-2\\1\\-3\end{bmatrix} + \begin{bmatrix}1\\3\\-1\\2\end{bmatrix} + \begin{bmatrix}2\\1\\-2\\3\end{bmatrix} + \ldots\ldots\ldots\ldots + \begin{bmatrix}1\\2\\2\\1\end{bmatrix} \text{ --> } \begin{bmatrix}-1\\-1\\2\\-3\end{bmatrix}$$

$\Psi = (\Gamma_1 + \Gamma_2 + \Gamma_3 + \ldots\ldots\ldots + \Gamma_M)/M$

Each face differs from the average by $\Phi_i = \Gamma_i - \Psi$ which is called mean centered image.

$$\Phi 1 = \begin{bmatrix}2\\-1\\-1\\0\end{bmatrix} \quad \Phi 2 = \begin{bmatrix}2\\4\\-3\\5\end{bmatrix} \quad \Phi 3 = \begin{bmatrix}3\\2\\-4\\6\end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots \quad \Phi M = \begin{bmatrix}2\\3\\0\\4\end{bmatrix}$$

4.2.2.4 Covariance Matrix

A covariance matrix is constructed as:

$C = AA^T$, where $A = [\Phi_1, \Phi_2 \ldots \Phi_M]$ of size $N^2 \times N^2$.

$$A = \begin{bmatrix}2 & 3\\-1 & -2\\-1 & 1\\0 & 2\end{bmatrix} \quad A^T = \begin{bmatrix}2 & -1 & -1 & 0\\3 & -2 & 1 & 2\end{bmatrix}$$

Size of covariance matrix will be N^2 x N^2 (4 x 4 in this case).

Eigen vectors corresponding to this covariance matrix is needed to be calculated, but that will be a tedious task therefore,

For simplicity we calculate A^TA which would be 2 x 2 matrix in this case.

$A^TA = \begin{pmatrix} 6 & 7 \\ 7 & 18 \end{pmatrix}$ size of this matrix is M x M

Consider the eigenvectors of v_i of A^TA such that

$$A^TAX_i = \lambda_i X_i$$

The eigenvectors v_i of A^TA are X_1 and X_2 which are 2 x 1. Now multiplying the above equation with A both sides we get:-

$$AA^TAX_i = A\lambda_i X_i$$

$$AA^T(AX_i) = \lambda_i(AX_i)$$

Eigen vectors corresponding to AA^T can now be easily calculated now with reduced dimensionality where AX_i is the Eigen vector and λ_i is the Eigen value.

4.2.2.5 Eigen Face Space

The Eigen vectors of the covariance matrix AA^T are AX^i which is denoted by U^i. U^i resembles facial images which look ghostly and are called Eigen faces. Eigen vectors correspond to each Eigen face in the face space and discard the faces for which Eigen values are zero thus reducing the Eigen face space to an extent. The Eigen faces are ranked according to their usefulness in characterizing the variation among the images.

A face image can be projected into this face space by

$\Omega_k = U^T(\Gamma_k - \Psi)$; k=1... M, where $(\Gamma_k\Psi)$ is the mean centered image.

Hence projection of each image can be obtained as Ω_1 for projection of *image*$_1$ and Ω_2 for projection of *image*$_2$ and hence forth.

4.2.3 Recognition Step

The test image, Γ, is projected into the face space to obtain a vector, Ω as

$\Omega = U^T (\Gamma - \Psi)$

The distance of Ω to each face is called Euclidean distance and defined by

$\epsilon_k^2 = ||\Omega - \Omega_k||^2$; k = 1… M where Ω_k is a vector describing the k^{th} face class.

A face is classified as belonging to class k when the minimum ϵ_k is below some chosen threshold Θ_c. otherwise the face is classified as unknown.

Θ_c, is half the largest distance between any two face images:

$\Theta c = (1/2)\ max_{j,k}\ ||\Omega j - \Omega k||$; j,k = 1,.....,M

We have to find the distance ϵ between the original test image Γ and its reconstructed image from the Eigen face Γ_f

$\epsilon^2 = ||\Gamma - \Gamma^f||^2$, where $\Gamma^f = U * \Omega + \Psi$

If $\epsilon < \Theta_c$ and $\epsilon_k < \Theta$ for all k then input images are the individual face image associated with the class vector Ω_k.

Chapter 5

Results and Discussion

5.1 Tutorial

Training

Step 1:

To do face detection, first we need a dataset of images of faces and non- faces. You need as many as you can get. Because you will do supervised training, which means you are already telling the computer if each picture is a face or a non- face, you need to get those images.

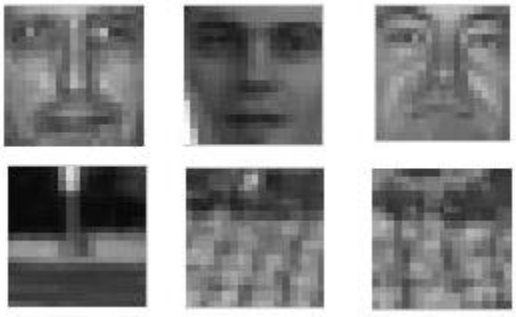

Fig. 5.1 (*Sample images of faces and non- faces*)

Once you have your own nice database, you need to understand the Haar features. They are rectangles that map over the faces of people and tell you if you are a face or non-face.

Why are there so many Haar features in a 19 x 19 image? SO here are what they look like:

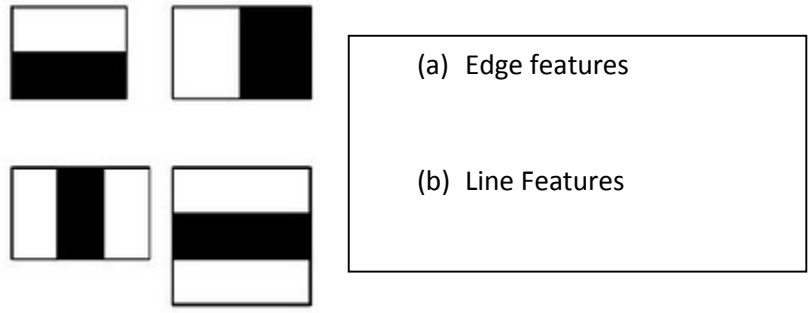

Fig. 5.1.1 (*Haar Features*)

There are so many more variations, but these are the only ones I used. So let's take the first one in the list here. A white rectangle above a black rectangle. If you see your image as a 19 x 19 matrix (that is from x1 to x19 and y1 to y19), and you start with this feature being a 1 x 2 in size and in position x1,y1, then you have your first feature:

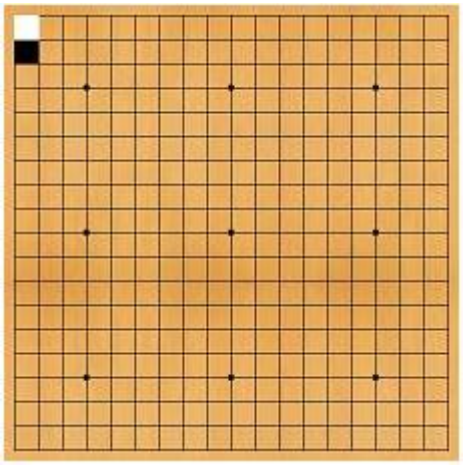

Fig. 5.1.2 (Feature *of 1 x 2 in image window*)

We would do all our calculations based on this classifier, then move on to the next size, which would be a 1 x 4 in position x1, y1, our second feature:

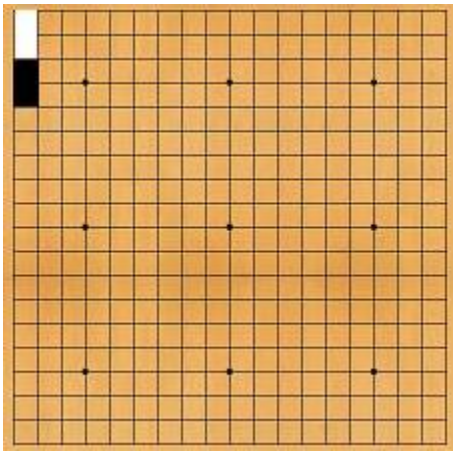

Fig. 5.1.3 (*Feature of 2 x 2 in image window*)

And in this manner, it will keep increasing. It would be 1 x 8, 1 x 16. Then the classifiers would start with a 2 x 2.

The code for this would be:

```
% There are 5 rectangles associated with haar features
feature = [1 2; 2 1; 1 3; 3 1; 2 2];
% a window of 19 x 19 containing a face or non face
frameSize = 19;
```

Fig. 5.1.4 (Code for feature array initialization)

The feature matrix explains the first sizes that each of the 5 classifiers can be.

```
% for each feature
for i = 1:5
    sizeX = feature(i,1);   % length
    sizeY = feature(i,2);   % width
    % for all pixels inside the boundaries of our feature
    for x=2:frameSize-sizeX
        for y=2:frameSize-sizeY
            % for each width and length possible in frameSize
            %for winLength = sizeX:sizeX:frameSize-x
             %for winWidth = sizeY:sizeY: frameSize-y;
            for winLength = sizeX:sizeX:frameSize-x
                for winWidth = sizeY:sizeY: frameSize-y;
                    disp('Send feature to calculate best threshold');
                    fprintf('x: %e\n',x);
                    fprintf('y: %e\n',y);
                    fprintf('width: %e\n',winWidth);
                    fprintf('length: %e\n',winLength);
                    fprintf('Classifier: %e\n',i)
                    CalcBestThresh(x,y,winWidth,winLength,i);
                end
            end
        end
    end
end
```

Fig. 5.1.5 (*The feature matrix*)

- Each feature (5 total), this is *i*.
- They all must start at 1 x 2, this is sizeX and sizeY.
- They cannot go over the size of 19 x 19, this is (x) and (y).
- winLength and winWidth are parameters that each feature will increase in size through the image. CalcBestThresh() is the function called that will see if the feature being tested is a good one or not.

Step 2:

Now that all the features to be used have been setup for each classifier, it is time to CalcBestThresh (calculate best threshold).

Each feature that is passed to CalcBestThresh has to be passed to every single image in the database.

```
function [] = CalcBestThresh(x,y,winWidth,winLength,classifier)
PosImgSize =     ;
NegImgSize =     ;
totalSize = PosImgSize + NegImgSize;
threshPos = zeros(1,PosImgSize);
threshNeg = zeros(1,NegImgSize);
% open a weights mat file that contains current weights of Ada boost for
% each image
weightsFile = 'weights.mat';
```

Fig. 5.1.6 (*Function for calculating Best Threshold*)

threshPos and threshNeg are the numbers received from calculating the Integral image. weightsFile is just a file in which I will keep the weight of each classifier (for Adaptive boosting).

Integral Image: The integral image is a summed area matrix that allows for fast computation of pixel values.

Step 2 continued:

Below weightsFile = 'weights.mat' is:

```
adaWeights = open(weightsFile);
adaWeights = adaWeights.adaWeights;
% for every face image, calculate the haar feature (over 51000 haar
% features for a 19 x 19 window)
disp('Evaluating images using CalcBestThresh');
for i=1:PosImgSize
    str = strcat('VarianceFaces\',int2str(i),'.pgm');
    eval('img = imread(str);');
    threshPos(i) = HaarFeatureCalc(img,x,y,winWidth,winLength,classifier);
end
% get mean and std of the face images to use as our classifier
posMean = mean(threshPos);
posStd = std(threshPos);
posMax = max(threshPos);
posMin = min(threshPos);
```

Fig. 5.1.7 (*Step 2 continued)*

The first two lines adaWeights is what opens the file so that it can be modified. What we want to focus on in this code is the "for" loop below adaWeights.

Haar feature calculation will happen for every single face image using the function HaarFeatureCalc.

HaarFeatureCalc:

All this is doing is parsing the information given to it (x,y,winWidth,winLength,classifier), in order to calculate which pixels need to be added for the Integral Image (with winWidth and winLength) and from which pixels starting (x and y) and which classifier.

Here is the whole code for HaarFeatureCalc:

```
function [thresh] = HaarFeatureCalc(img,x,y,winWidth,winLength,classifier)
% calculates integral image
% x,y gives coordinates
% winWidth, winLength gives size of window
% classifier gives the classifier ( only 5)
% [start_row start_column width length]
if (classifier==1)
    firstRec = [x y ((winWidth/2)-1) (winLength-1)];
    secondRec = [x (y+winWidth/2) ((winWidth/2)-1) (winLength-1)];
    % gets the integral image
    intWin = IntImg(img);
    thresh = CalcIntRec(intWin,firstRec) + CalcIntRec(intWin,secondRec);
elseif (classifier==2)
    firstRec = [x y (winWidth-1) ((winLength/2)-1)];
    secondRec = [(x + winLength/2) y (winWidth-1) ((winLength/2)-1)];
    % gets the integral image
    intWin = IntImg(img);
    thresh = CalcIntRec(intWin,firstRec) + CalcIntRec(intWin,secondRec);
elseif (classifier==3)
    firstRec = [x y ((winWidth/3)-1) (winLength-1)];
    secondRec = [x (y+(winWidth/3)) ((winWidth/3)-1) (winLength-1)];
    thirdRec =  [x (y+((2*winWidth)/3)) ((winWidth/3)-1) (winLength-1)];
    intWin = IntImg(img);
    thresh = CalcIntRec(intWin,firstRec) - CalcIntRec(intWin,secondRec)
    + CalcIntRec(intWin,thirdRec);
elseif (classifier==4)
    firstRec = [x y (winWidth-1) ((winLength/3) -1)];
    secondRec = [(x+(winLength/3)) y (winWidth-1) ((winLength/3) -1)];
    thirdRec = [(x+((2*winLength)/3)) y (winWidth-1) ((winLength/3) -1)];
    intWin = IntImg(img);
    thresh = CalcIntRec(intWin,firstRec) - CalcIntRec(intWin,secondRec)
    + CalcIntRec(intWin,thirdRec);
elseif (classifier==5)
    firstRec = [x y (winWidth/2-1) (winLength/2-1)];
    secondRec = [x (y + winWidth/2) (winWidth/2-1) (winLength/2-1)];
    thirdRec = [(x + winLength/2) y (winWidth/2-1) (winLength/2-1)];
    fourthRec = [(x + winLength/2) (y + winWidth/2) (winWidth/2-1)
        (winLength/2-1)];
    intWin = IntImg(img);
    thresh = CalcIntRec(intWin,firstRec) - CalcIntRec(intWin,secondRec)
    + CalcIntRec(intWin,thirdRec) - CalcIntRec(intWin,fourthRec);
end
```

Fig. 5.1.8 (*Function for HaarFeatureCalc*)

IntImg

```
function [A] = IntImg(img)
% calculate the integral image of a window
A = cumsum(cumsum(double(img)),2);
```

Fig. 5.1.9 (*IntImg function*)

This is matlab's command to sum up all the pixels within the 19 x 19 image and put them in a matrix of its sum from left to right, top to bottom.

CalcIntRec

```
function [ret] = CalcIntRec(img,fourpoints)
% given four reference points in the integral image
% to calculate the sum of pixels inside it.
row_val = fourpoints(1,1);
col_val = fourpoints(1,2);
img_width = fourpoints(1,3);
img_length = fourpoints(1,4);
one = img(row_val-1,col_val-1);
two = img(row_val-1,col_val+img_width);
three = img(row_val+img_length,col_val-1);
four =  img(row_val+img_length,col_val+img_width);
ret = four + one - (two + three);
```

Fig. 5.1.10 (*CalcIntRec function*)

The same should be calculated for non-face images:

```
% for every non face image, calculate the haar feature
for i=1:NegImgSize
    str = strcat('VarianceNonFaces\',int2str(i),'.pgm');
    eval('img = imread(str);');
    threshNeg(i) =  HaarFeatureCalc(img,x,y,winWidth,winLength,classifier);
end
```

Fig. 5.1.11 (*Funtion to repeat for non- face images also*)

Step 3:

```
% look for best threshold of points between mean and std of face images for
% each classifier
disp('Finished evaluating all images at specified classifier');
disp('Looking for best threshold, thresholds are:');
thresh = [threshPos threshNeg]
for j = 1:50
    % calculate false positives ( WE REALLY DONT WANT THESE)
    pos = 0;
    % calculate false positives (NOT SO IMPORTANT, but must be below 50%)
    neg = 0;
    C = ones(1,totalSize);
    % Increase the bandwitdh of our gaussian distribution to accept more
    % recognition outside the mean to get a probability with almost 100%
    % face detection with a low (65% or less) non face recognition and a
    % total error lower than 50%. THIS IS A WEAK CLASSIFIER
    for i = 1:PosImgSize
        if (thresh(i) <=posMean +(abs(posMax -posMean)/50)*j && thresh(i)
            >= posMean -(abs(posMean -posMin)/50)*j)
            pos = pos+1;
            C(i) = 0;
        end
    end
    for i = 1:NegImgSize
        if (thresh(i) <=posMean +(abs(posMax -posMean)/50)*j && thresh(i)
            >= posMean -(abs(posMean -posMin)/50)*j)
        else
            neg = neg+1;
            C(i+PosImgSize) = 0;
        end
    end
    i = 1:totalSize;
    totalErr = sum(adaWeights(i)*C(i)');
    posErr = sum(adaWeights(1:PosImgSize)*C(1:PosImgSize)');
    negErr = sum(adaWeights(PosImgSize+1:totalSize)*C(PosImgSize+1:totalSize)');
    fprintf('Total Error: %e\n',totalErr);
    fprintf('Positive Error: %e\n',posErr);
    fprintf('Negative Error: %e\n',negErr);
    if (posErr <=0.05)
        if (totalErr<0.5)
            disp('Found weak classifier');
            fprintf('Total error is: %e\n',totalErr);
            fprintf('False Negative error: %e\n',posErr);
            fprintf('False positive error: %e\n',negErr);
            % pass to adaboost to save our weak classifier and update our
            % weights
            AdaBoost(x,y,winWidth,winLength,classifier,posMean,
            posStd,j,totalErr,adaWeights,C,posMax,posMin,posErr,negErr);
            break;
        end
    end
end

fprintf('Mean: %e\n',posMean);
fprintf('Std: %e\n',posStd);
fprintf('Max: %e\n',posMax);
fprintf('Min: %e\n',posMin);
        else
            neg = neg+1;
            C(i+PosImgSize) = 0;
        end
    end
```

Fig. 5.1.12 (*Looking for best threshold of points*)

Evaluation

```
try
    % How many frames per call of this function
    while (vid.FramesAcquired<=1)
        trigger(vid);
        img = getdata(vid);
        img = rgb2gray(img);
        img3 = imresize(img,resizeImg);
        % variance normalize our image just like training data
        img2 = varNorm(img3);
        %imshow(img2);
        imshow(img);
        hold on
        sum = 0;
        for y =2:1:40
            for x = 2:1:y
                x
                y
                sum=sum+1
                subimg = img2(x:x+19-1,y:y+19-1);
                cascadeClass(subimg,x,y);
                %subimg = img2(y:y+19-1,x:x+19-1);
                %cascadeClass(subimg,x,y,Weights);
            end
            %toc
        end
        sum
    end
    stop(vid)
catch
    disp('Error');
    stop(vid)
end
```

Fig. 5.1.13 (Evaluation of the code)

OUTPUT (Face Detection):

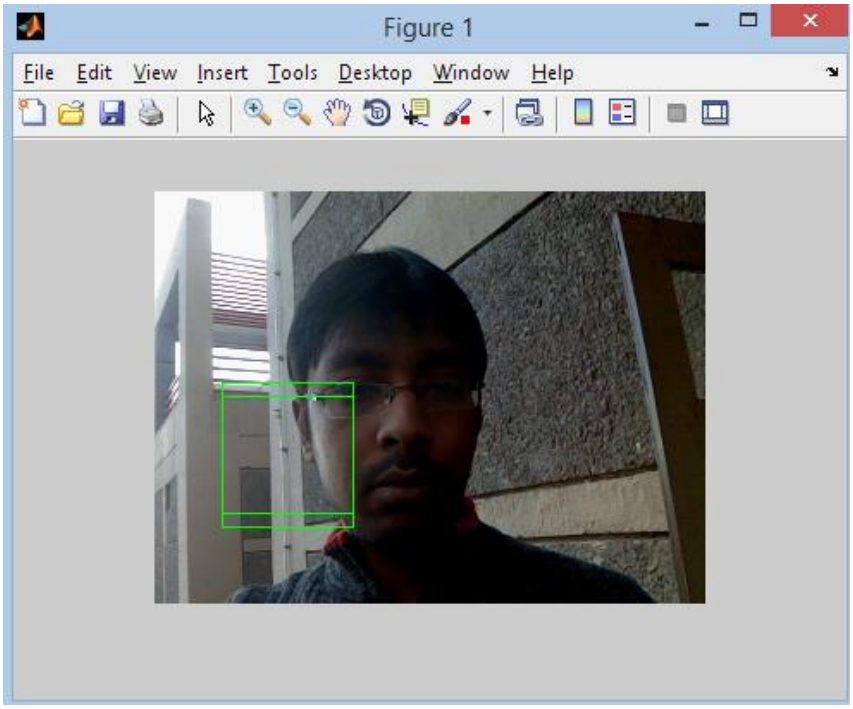

Fig. 5.1.14 (*Output Image 1)*

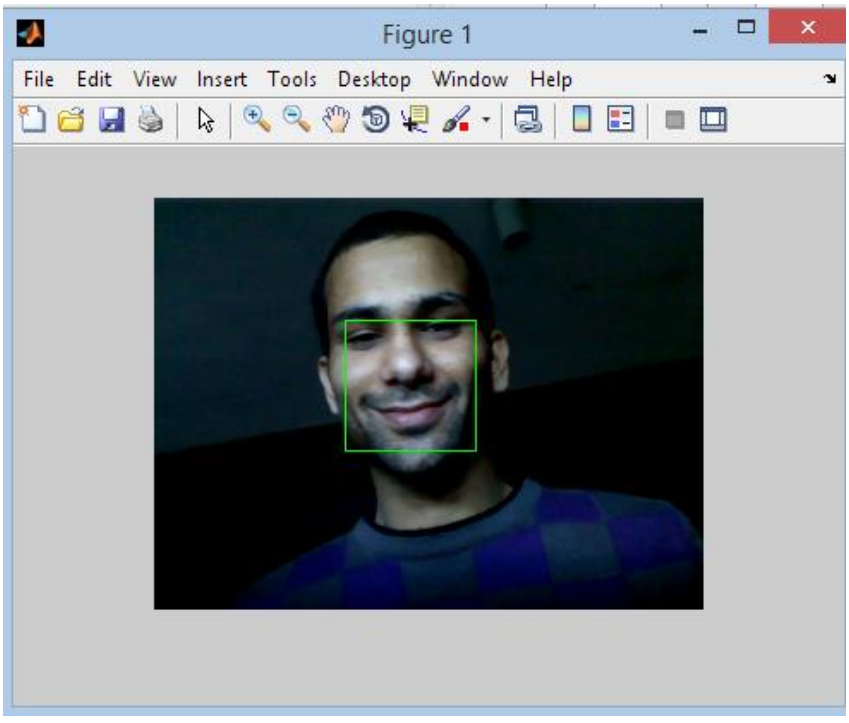

Fig. 5.1.15 (*Output Image 2)*

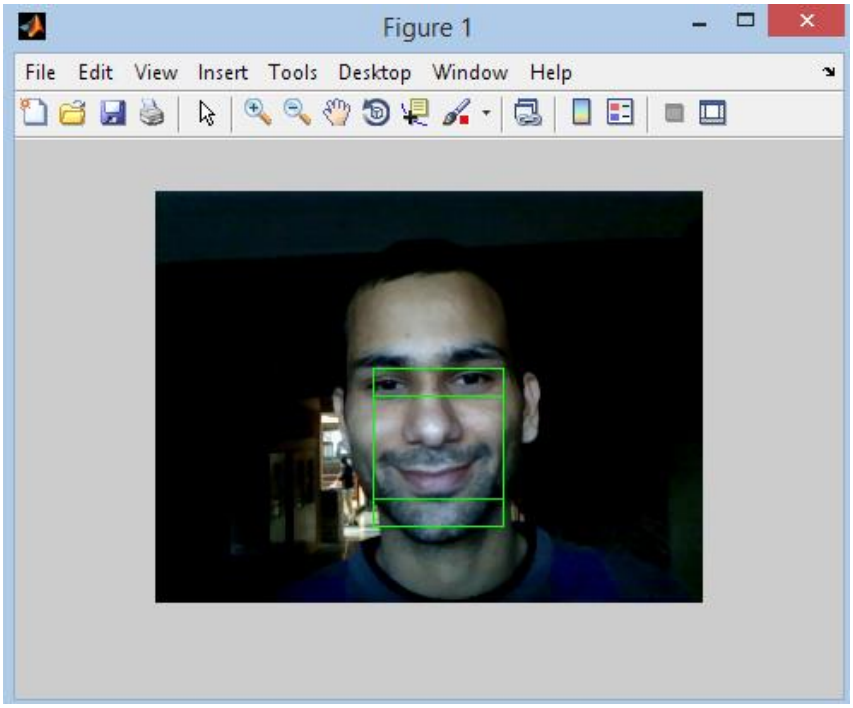

Fig. 5.1.16 (*Output Image 3)*

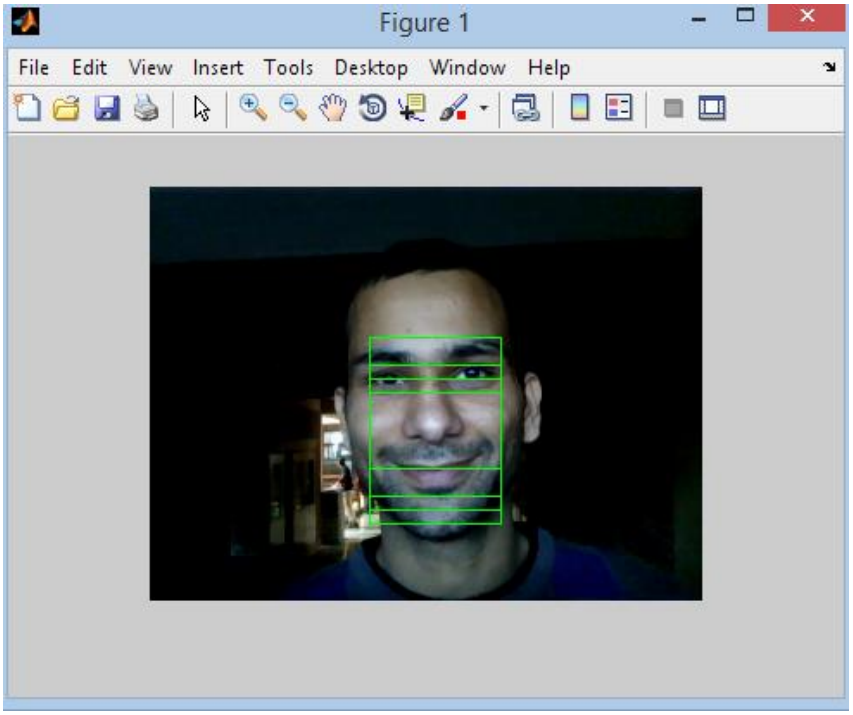

Fig. 5.1.17 (*Output Image 4)*

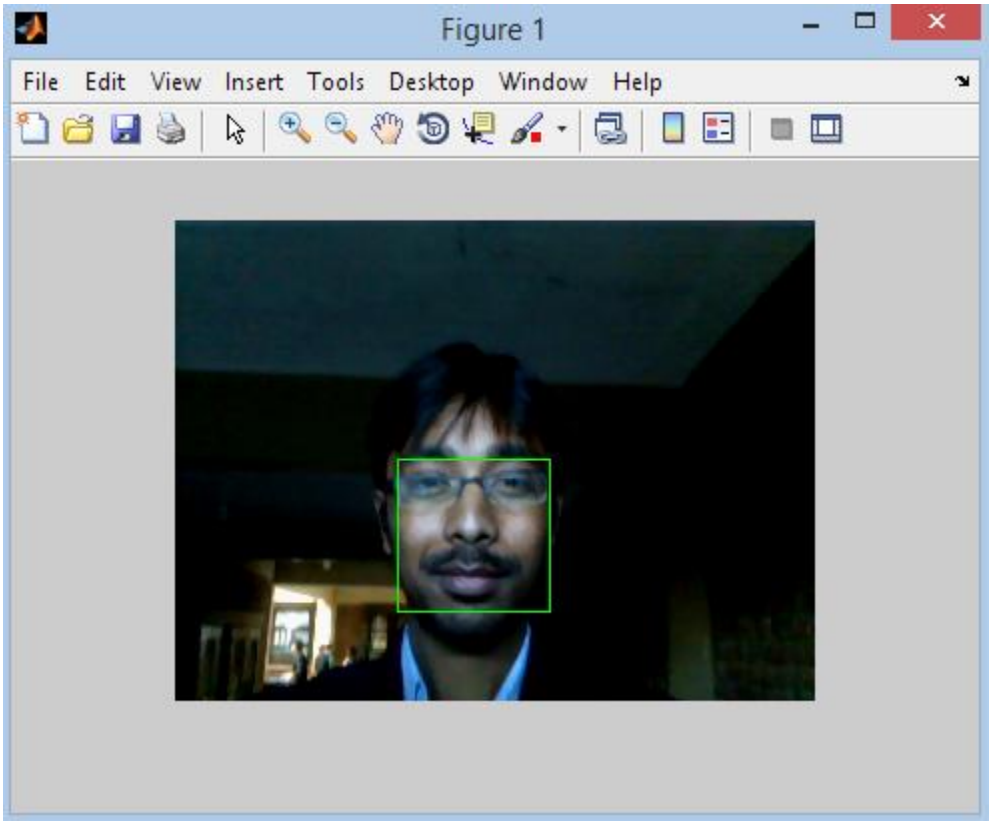

Fig. 5.1.18 (*Output Image 5)*

OUTPUT: (Face Recognition)

(**5.2.1)** This is the GUI which gets displayed when we run the main file of the project. It has the features of Face Detection in which a new webcam opens up and detects the face in a frame.

And the main feature is the Face Recognition which allows you to input an image and matches the face with the existing database of faces and displays the best matched image according to the system.

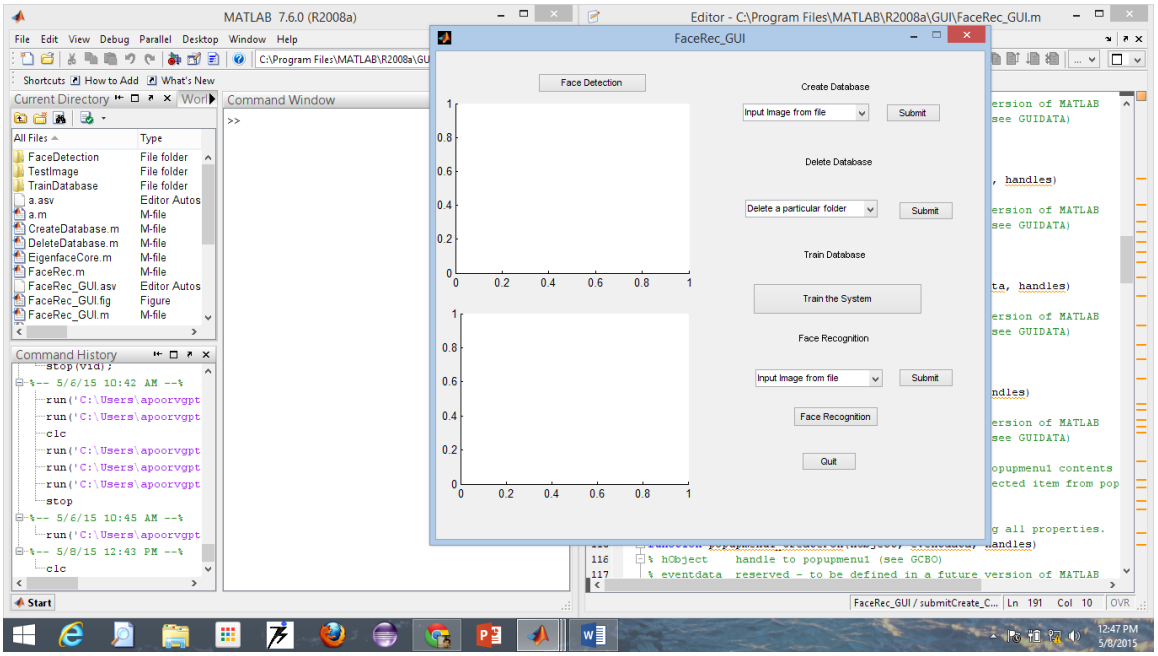

Fig. 5.2.1 (*Project GUI)*

(**5.2.2)** This is for creating a database. When the user selects the "Input Image from file" from the drop-down menu and click "Submit", a dialog box opens and ask for the image to be added to the database. After selecting the image, it displays the image in the GUI and opens another dialog box asking the user for the location where the image is to be saved.

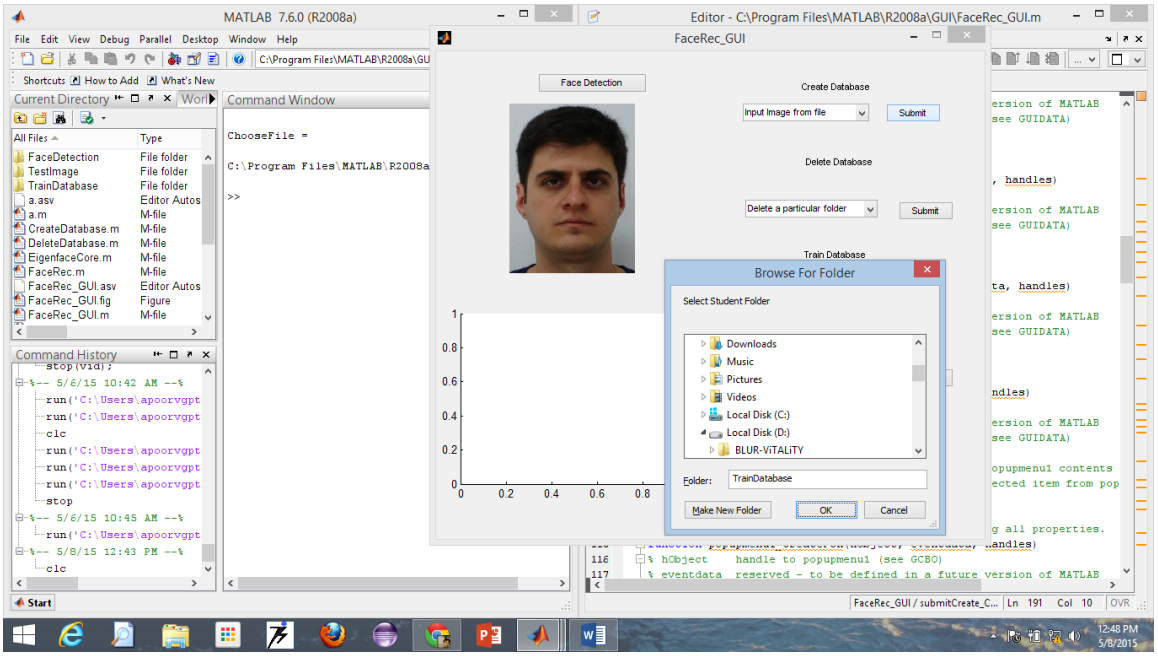

Fig. 5.2.2 (*Inputting the image to be added to the database and displaying the image)*

(**5.2.3)** This screenshot depicts the "Training the System" with the image database. After clicking the "Train the System" button, a dialog box opens up asking the user to select the folder containing the database of images.

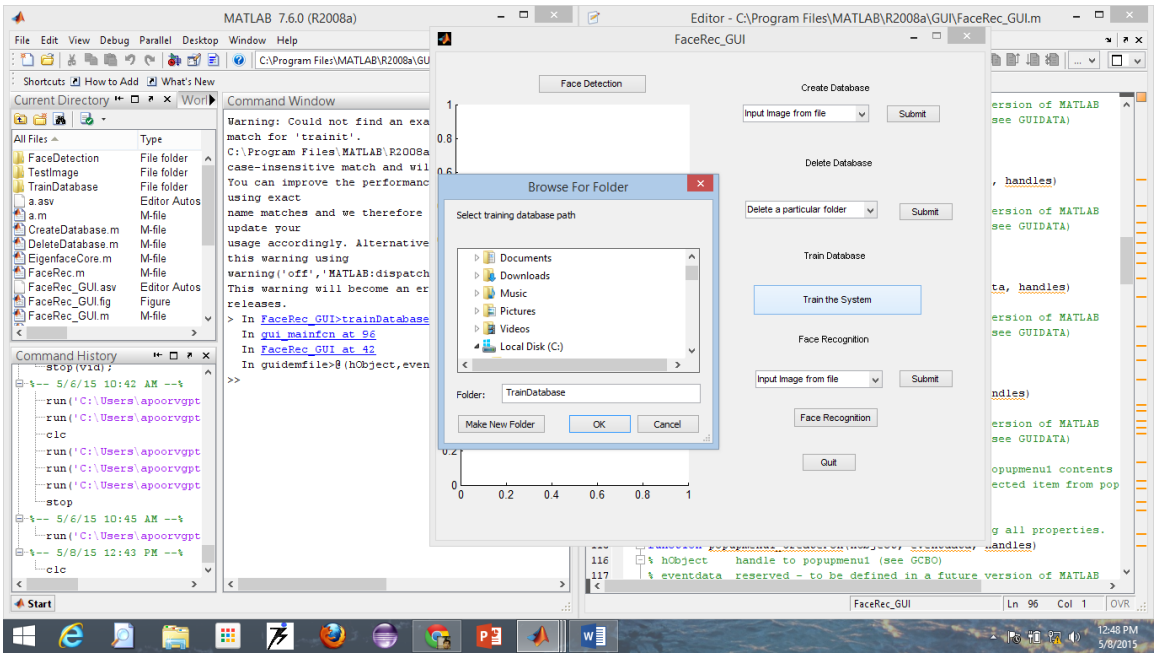

Fig. 5.2.3 (*Inputting the folder to be trained by the system)*

(**5.2.4)** After selecting the folder containing the database of images, it scans each folder for the images, and performs some calculation with the images such as creating eigen vectors and computing the mean image etc.)

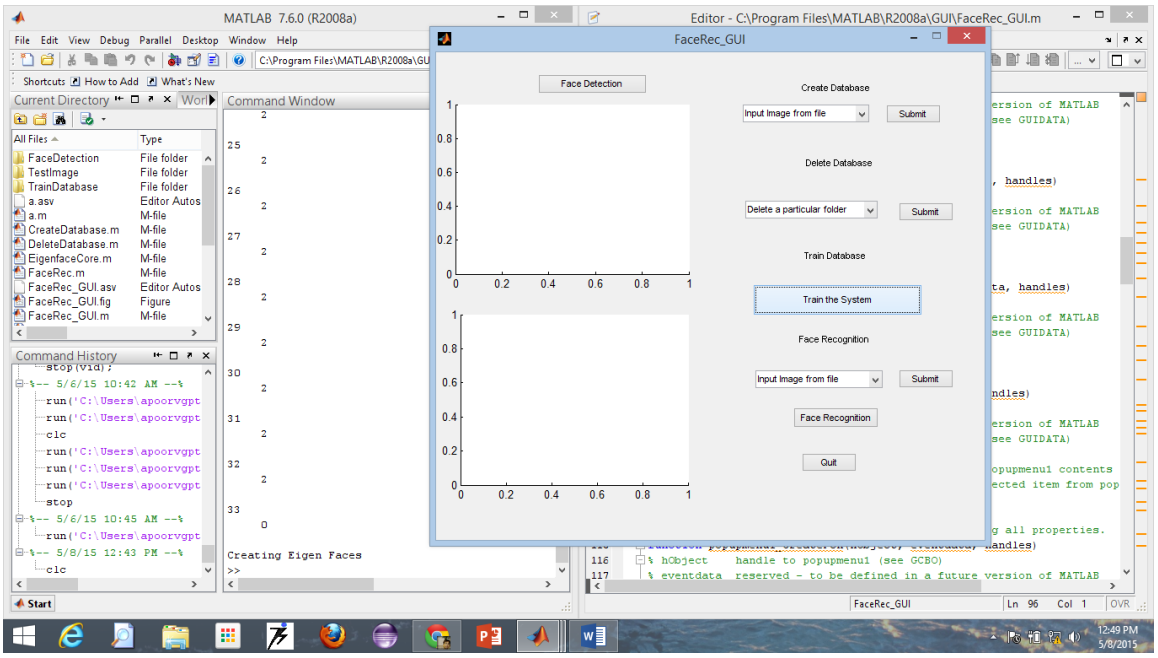

Fig. 5.2.4 (*Training the system)*

(**5.2.5)** After successfully training the system, it displays a message box prompting the user that the system has been successfully trained with the images.

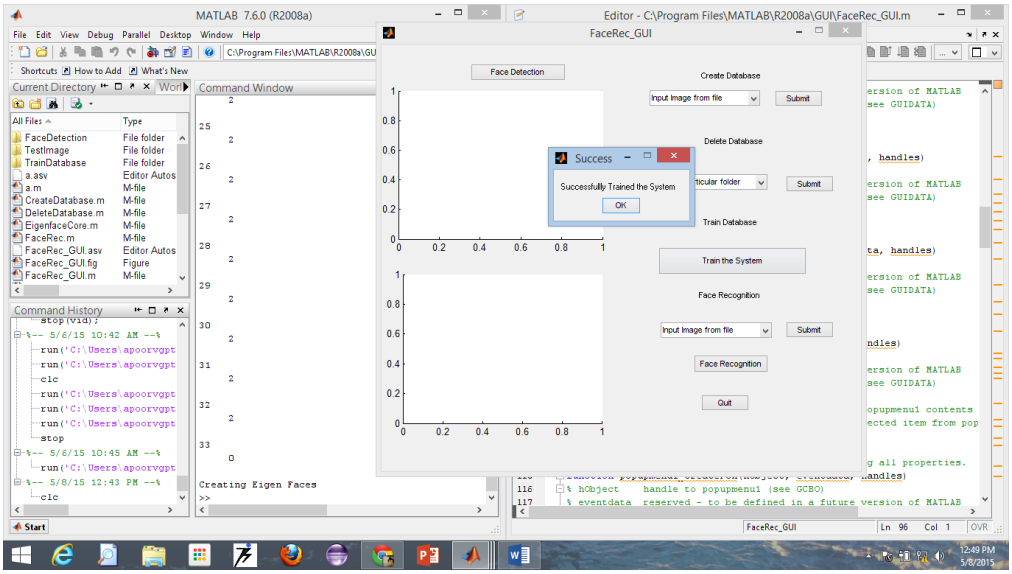

Fig. 5.2.5 (*Prompting the message after training the system)*

(5.2.6) Now, the main part is the Face Recognition. At first, after selecting an appropriate option("Input Image from File" in this case), a dialog box shows up asking the user for the Test Image to be entered to perform Face Recognition on!

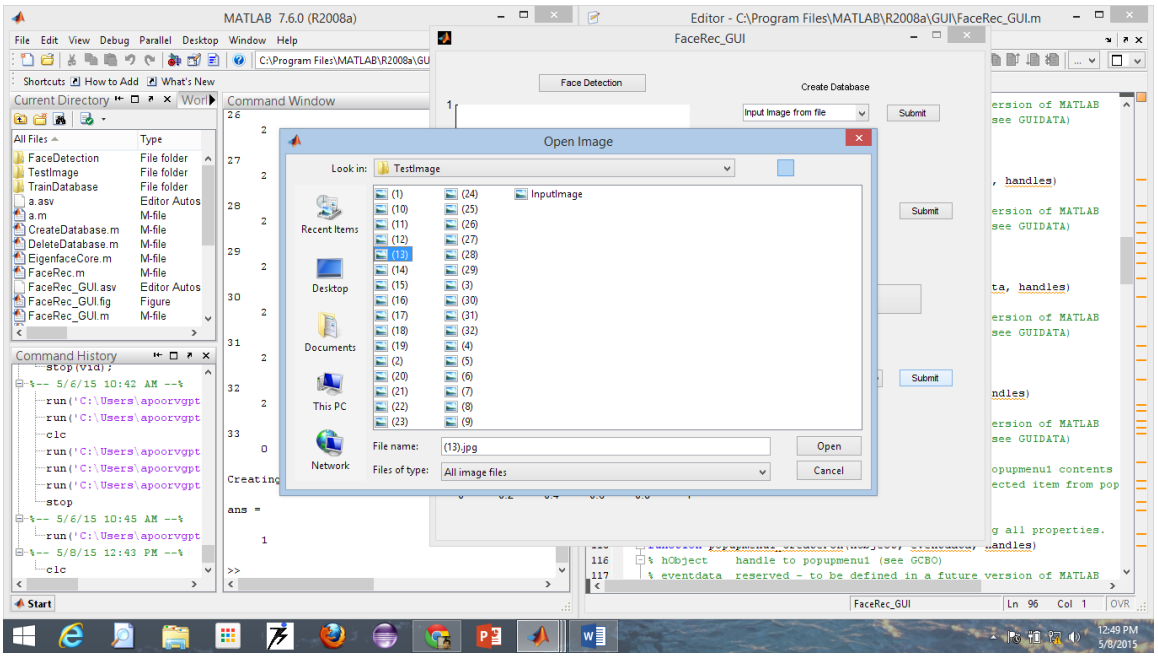

Fig. 5.2.6 (*Inputting the test image for face recognition)*

(5.2.7) After selecting the image from above, the image gets displayed in the GUI.

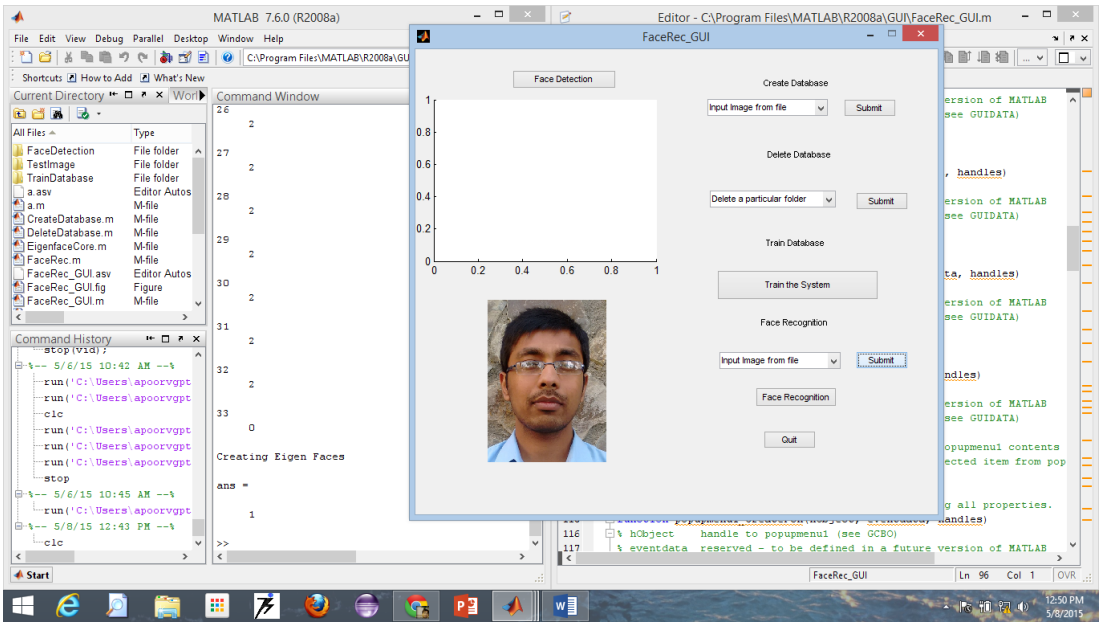

Fig. 5.2.7 (*Test Image is displayed in the GUI)*

(5.2.8) By clicking on the "Face Recognition" button, the computation is performed and the best matched image is displayed in the GUI along with our Test Image. Here, we can observe that our system has correctly displayed the Test Image with our images in the database.

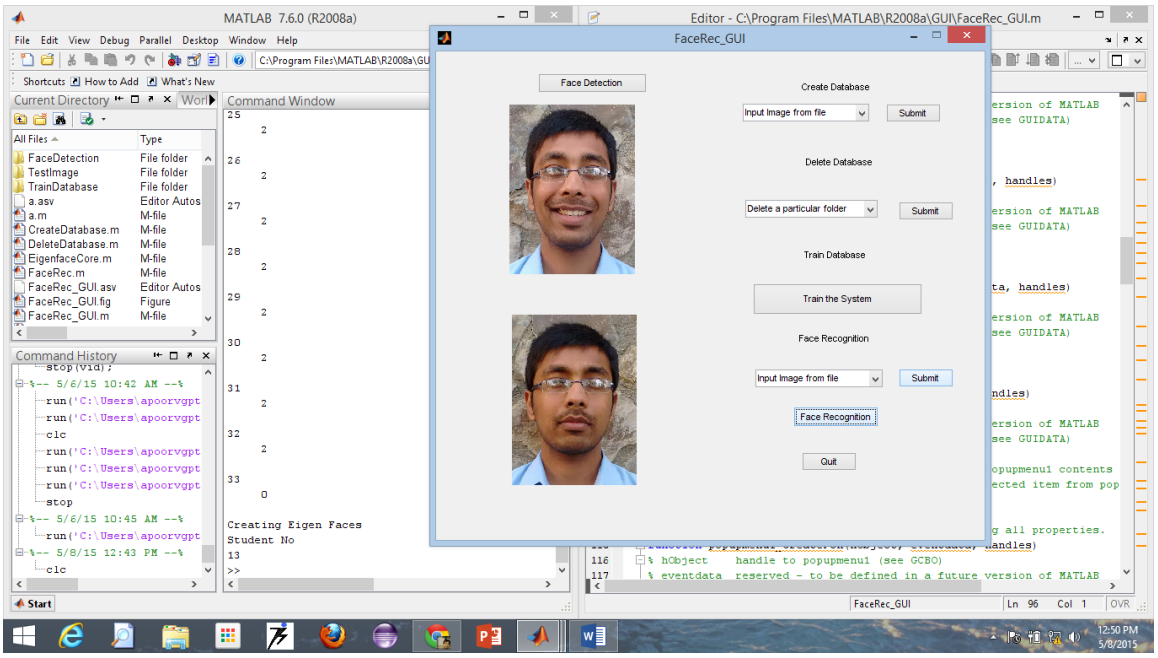

Fig. 5.2.8 (*Performing the face recognition and the best matched image gets displayed after recognition)*

Another Test Image:

Here's another case in which the image is not correctly identified from the existing database.

(**5.2.9)** Taking the Test Image as input from the dialog box.

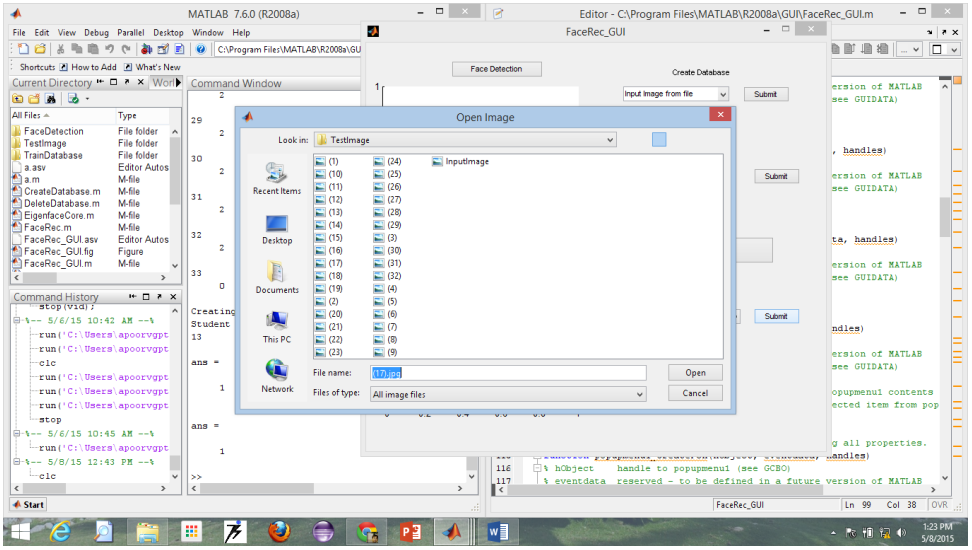

Fig. 5.2.9 (*Taking image as input)*

(**5.2.10)** Displaying the selected Test Image in the GUI and performing the Face Recognition.

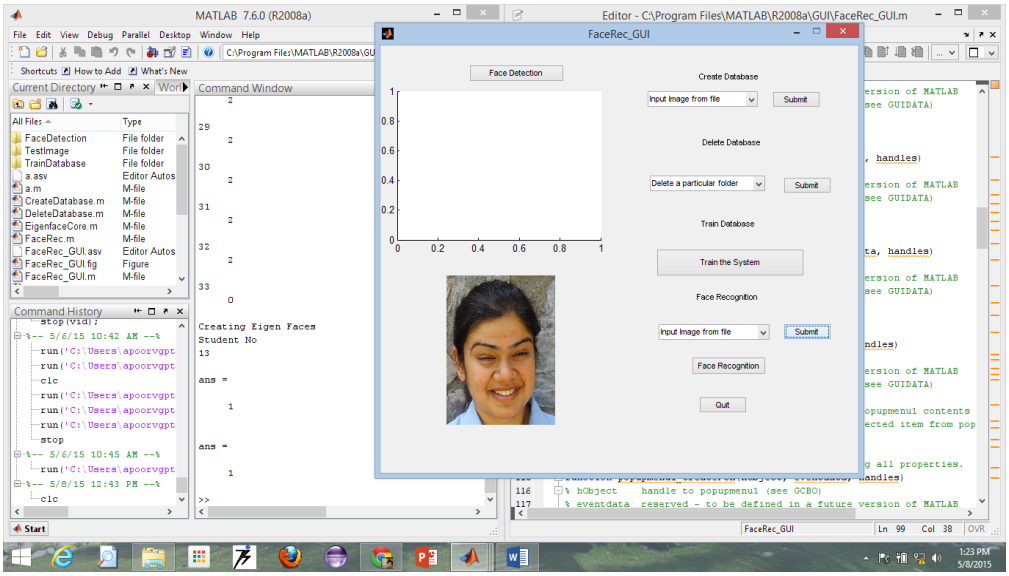

Fig. 5.2.10 (*Displaying the inputted image)*

(**5.2.11)** We can observe that the best matched image is not the same as our Test Image and our Test Image is not correctly identified.

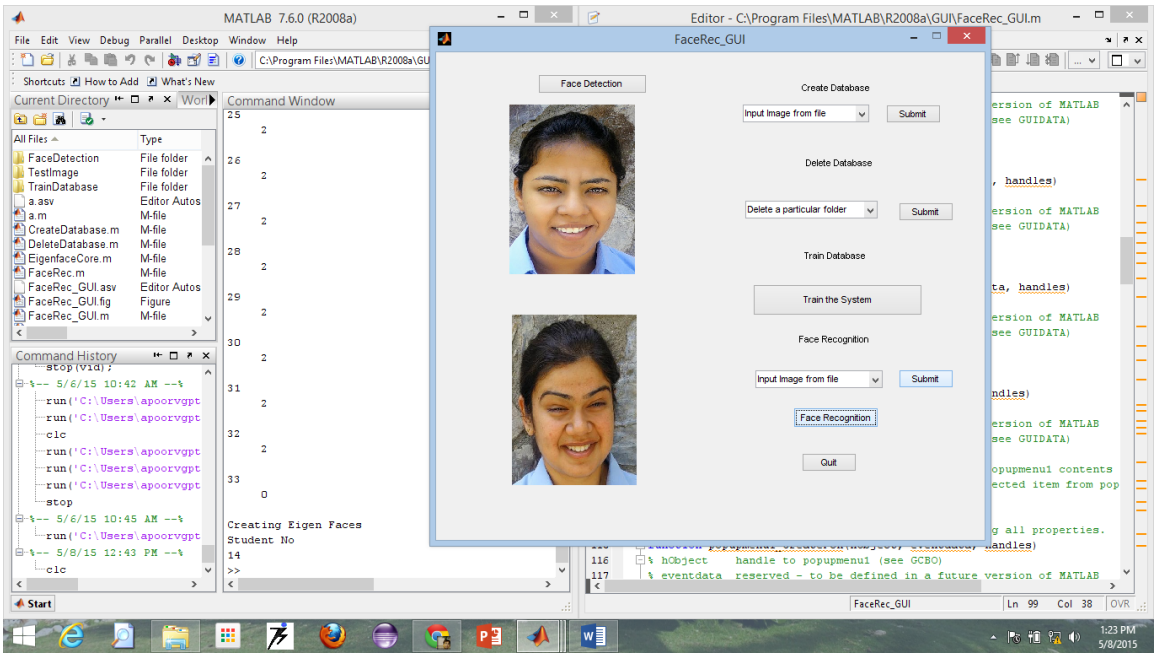

Fig. 5.2.11 (*Performing the face recognition process and displaying the best matched image)*

Chapter 6

Conclusions and Future Work

6.1 Conclusion

Early attempts at making computers recognize faces were limited by the use of impoverished face models and feature descriptions, assuming that a face is no more than the sum of its parts, the individual features. Recent attempts using parameterized feature models and multiscale matching look more promising, but still face severe problems before they are generally applicable.

The eigenface approach to face recognition was motivated by information theory, leading to the idea of basing face recognition on a small set of image features that best approximates the set of known face images, without requiring that they correspond to our intuitive notions of facial parts and features. Although it is not an elegant solution to the general recognition problem, the eigenface approach does provide a practical solution that is well fitted to the problem of face recognition. It is fast, relatively simple, and has been shown to work well in a constrained environment.

It is important to note that many applications of face recognition do not require perfect identification, although most require a low false-positive rate. In searching a large database of faces, for example, it may be preferable to find a small set of likely matches to present to the user. For applications such as security systems, the system will normally be able to "view" the subject for a few seconds or minutes, and thus will have a number of chances to recognize the person. Our experiments show that the eigenface technique can be made to perform at very high accuracy, although with a substantial "unknown" rejection rate, and thus is potentially well suited to these applications.

6.2 Future Work

The results of this face recognition system are good but it is clear that many improvements are possible.

Future work will include

- Combining the face detection and face recognition system to make a robust system in general.
- Improving both the systems to achieve even better accuracy.

List of references

[1] Paul Viola & Micheal Jones. Robust real- time object detection. Second International Workshop on Statistical Learning and Computational Theories of Vision Modeling, Learning, Computing and Sampling, July 2001.

[2] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *International Conference on Computer Vision,* 1998.

[3] K –K Sung and T. Poggio. Example- based learning for view- based human face detection. A.I Memo, 1521, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, December 1994.

[4] Rainer Lienheart and Jochen Maydt, An Extended Set of Haar- like Features for Rapid Object Detection, Intel Labs, Intel Corporation, Santa Clara, CA 95052, USA

[5] M. H. Yang, D. Kreigman, N. Ahuja, *Detecting Faces in Images*: A survey. IEEE Transactions on pattern analysis and machine intelligence, vol. 25 1, January 2002.

[6] H. Rowley, S. Baluja and T. Kanade, Neural network- based face detection. In IEEE Patt. Anal. Mach. Intell., volume 20, pages 22-38, 1998.

[7] V.N. Vapnik, *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag, 1982

[8] Pavlovic V. and Garg A. *Efficient Detection of Objects and Attributes using Boosting.* IEEE Conf. Computer Vision and Pattern Recognition, 2001.

[9] M. Turk and A. Pentland (1991). “Eigenfaces for recognition”. Proc. IEEE Conference on Computer Vision and Pattern Recognition. pp. 586- 591.

[10] A. Pentland, B. Moghaddam, T. Starner, O. Oliyide, and M. Turk. (1993). “View-based and modular Eigenspaces for face recognition”. Technical Report 245, M.I.T Media Lab