FACE DETECTION IN ACTIVE ROBOT VISION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MURAT ÖNDER

IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

AUGUST 2004

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan ÖZGEN

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Mübeccel DEMİREKLER

Head Of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Uğur HALICI

Supervisor

Examining Committee Members

Prof. Dr. Kemal LEBLEBİCİOĞLU	(METU,EE)	_____
Prof. Dr. Uğur HALICI	(METU,EE)	_____
Prof. Dr. Mete SEVERCAN	(METU,EE)	_____
Dr. Ece GÜRAN	(METU,EE)	_____
M.Sc. Alper ÜLKÜ	(ASELSAN)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Murat Önder

Signature :

ABSTRACT

FACE DETECTION IN ACTIVE ROBOT VISION

ÖNDER, Murat
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Uğur HALICI

June 2004, 109 pages

The main task in this thesis is to design a robot vision system with face detection and tracking capability. Hence there are two main works in the thesis: Firstly, the detection of the face on an image that is taken from the camera on the robot must be achieved. Hence this is a serious real time image processing task and time constraints are very important because of this reason. A processing rate of 1 frame/second is tried to be achieved and hence a fast face detection algorithm had to be used. The Eigenface method and the Subspace LDA (Linear Discriminant Analysis) method are implemented, tested and compared for face detection and Eigenface method proposed by Turk and Pentland is decided to be used. The images are first passed through a number of preprocessing algorithms to obtain better performance, like skin detection, histogram equalization etc. After this filtering process the face candidate regions are put through the face detection algorithm to understand whether there is a face or not in the image. Some modifications are applied to the eigenface algorithm to detect the faces better and faster.

Secondly, the robot must move towards the face in the image. This task includes robot motion. The robot to be used for this purpose is a Pioneer 2-DX8 Plus, which is a product of ActivMedia Robotics Inc. and only the interfaces to move the robot have been implemented in the thesis software. The robot is to detect the faces at different distances and arrange its position according to the distance of the human to the robot. Hence a scaling mechanism must be used either in the training images, or in the input image taken from the camera. Because of timing constraint and low camera resolution, a limited number of scaling is applied in the face detection process. With this reason faces of people who are very far or very close to the robot will not be detected. A background independent face detection system is tried to be designed. However the resultant algorithm is slightly dependent to the background. There is no any other constraints in the system.

Keywords: Face detection, eigenface, principal component analysis, subspace LDA, Fisher linear discriminant, robot motion.

ÖZ

AKTİF ROBOT GÖRMESİNDE YÜZ SAPTAMA

ÖNDER, Murat
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Uğur HALICI

Ağustos 2004, 109 sayfa

Bu tezdeki temel amaç, yüz saptama ve izleme yeteneğine sahip bir robot geliştirmektir. Bu yüzden tez iki temel çalışma içermektedir: İlk olarak robot üzerindeki kameradan alınan görüntü üzerindeki insan yüzü içeren noktaların belirlenmesi gerekmektedir. Buna gore yapılması gereken iş ciddi bir gerçek zamanlı görüntü işleme çalışması içermekte olup zaman sınırlamaları bu yüzden önem taşımaktadır. 1 çerçeve/saniye derecesinde hıza ulaşmak istenilen tezde bu hızı sağlamak amacıyla hızlı bir yüz saptama algoritması kullanılması gerekli görülmüştür. Yüz saptama için Özyüz metodu ve Altuzay Doğrusal Ayrışım Analizi metodu uygulanmış, test edilmiş ve karşılaştırılmış, Turk ve Pentland tarafından önerilen Özyüz yönteminin kullanılmasına karar verilmiştir. Performansı artırmak için görüntüler önce insan teni bulma, histogram eşitleme gibi önsüreçlerden geçirilmiştir. Bu filtreleme işlemlerinin ardından yüz adayı bölgeler yüz içerip içermediklerinin anlaşılması için yüz saptama algoritmasına sokulurlar. Yüzlerin daha doğru ve hızlı bulunabilmesi için Özyüz algoritması üzerinde birtakım değişiklikler yapılmıştır.

İkinci iş olarak da robotun bulunan bu yüze doğru ilerlemesi gerekmektedir. Bu iş robot görmesi içermektedir. Bu amaçla kullanılan robot ActivMedia Robotics firmasının ürünü olan Pioneer 2-DX8 Plus olup tez yazılımında yalnızca robotun hareket arayüzlerine uygun kod yazılmıştır. Robot, farklı uzaklıklardaki insan yüzlerini tespit edebilmeli ve pozisyonunu yüzün kendine olan uzaklığına gore ayarlamalıdır. Bu işlem de eğitim görüntülerinde veya kameradan alınan görüntüde bir ölçeklendirme yapılmasını zorunlu kılmaktadır. Zaman kısıtlamaları ve düşük kamera çözünürlüğü nedeniyle sınırlı sayıda ölçekleme kullanılmıştır. Bu yüzden robota çok yakındaki ve çok uzaktaki insanların yüzleri tespit edilemeyecektir. Arka plandan bağımsız bir yüz bulma sistemi elde edilmeye çalışılmışsa da sonuçta oluşan sistemin arka plana bir miktar bağımlı olduğu tespit edilmiştir. Sistemde başka sınırlamalar yoktur.

Anahtar Kelimeler : Yüz saptama, özyüz, ana bileşen analizi, altuzay doğrusal ayrışım analizi, Fisher doğrusal ayrışımı, robot hareketi.

To My Mother

ACKNOWLEDGEMENTS

I would like to thank Prof. Dr. Uğur Halıcı for her valuable supervision and support throughout the development and improvement of this thesis. This thesis would not have been completed without her guidance.

I thank all the members of the METU Computer Vision and Intelligent Systems Research Laboratory for their invaluable support, and for sharing the facilities of the laboratory generously.

I give my special thanks to my friend Özkan Gönder for his generous support at each phase of the thesis.

I am grateful to my parents and my colleagues for their continuous encouragement and support.

This thesis is partially supported under project BAP–2002–03–01–05, 2002-2004 Active Robot Vision and BAP–2002–07–04–04, 2002-2004 Face Recognition and Modelling.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1. Overview of Face Detection

Images containing faces are essential to intelligent vision-based human computer interaction, and research efforts in face processing include face tracking, face recognition, pose estimation, and expression recognition. However, many reported methods assume that the faces in an image or an image sequence have been identified and localized. To build fully automated systems that analyze the information contained in face images, robust and efficient face detection algorithms are required. Given a single image, the goal of face detection is to identify all image regions which contain a face regardless of its three-dimensional position, orientation, and lighting conditions. Such a problem is challenging because faces are nonrigid and have a high degree of variability in size, shape, color, and texture. Numerous techniques have been developed to detect faces in a single image. The starting point for any face detection technique, is detecting faces in still images. After obtaining an algorithm which works successfully in still images, the video streams taken from a camera or other source, can be evaluated.

Face detection, especially together with face recognition has become an important issue in many applications such as security systems, credit card verification and criminal identification. For example, the ability to model a particular face and distinguish it from a large number of stored face models would make it possible to

vastly improve criminal identification. Even the ability to merely detect faces, as opposed to recognizing them, can be important. Detecting faces in photographs can be very useful in automatic color film development, since the effect of many enhancement and noise reduction techniques depends on the image content.

Face detection is difficult for three main reasons. First, although most faces are similarly structured with the facial features arranged in roughly the same spatial configuration, there could be a large component of non-rigidity and textural differences among faces. For the most part, these elements of variability are due to the basic differences in "facial appearance". As a result, traditional fixed template matching techniques and geometric model-based object recognition approaches that work well for rigid and articulate objects tend to perform inadequately for detecting faces. Second, face detection is also made difficult because certain common but significant features, such as glasses or a moustache, can either be present or totally absent from a face. All this adds more variability to the range of permissible face patterns that a comprehensive face detection system must handle. Third, face detection can be further complicated by unpredictable imaging conditions in an unconstrained environment. Because faces are essentially three-dimensional structures, a change in light source distribution can cast or remove significant shadows from a particular face, hence bringing even more variability to 2D facial patterns.

1.2. Overview of Robot Vision

Bridging the gap between the world of the computer and that of its user has always been one of the chief goals of computer science research. Graphical interfaces, input devices, speech generators, and handwriting recognition systems are just a few examples of how we have made computers more accessible. Machine vision is a more recent thrust in this direction, and represents a major step in bringing the computer into our world.

Jain et.al. [30] defines the goal of a robot vision system simply as creating a model of the real world from images. A robot vision system recovers useful information about a scene from its two-dimensional projections. Since images are two-dimensional projections of the three-dimensional world, the information is not directly available and must be recovered. This recovery requires the inversion of a many-to-one mapping. To recover the information, knowledge about the objects in the scene and projection geometry is required.

Humans rely upon vision more than any of our other senses, and the human brain has evolved in a way that indicates the necessity and complexity of observing our world through sight. The primary visual, striate, and visual association cortexes occupy over 25% of the neocortex, the region of the cerebral cortex which developed latest in our evolution [27]. Given that our brains seem to devote such resources to the process of sight, it is not a surprise that robot vision is a very compelling, though challenging, field of research in computer science. This thesis focuses entirely on two specific challenges in robot vision: the tasks of detecting human faces in real time, and moving the robot accordingly.

1.3. Basic Terminology

- *Face Detection:* The face detection problem can be stated as follows: given an arbitrary image, which could be a digitized video signal or a scanned photograph, determine whether or not there are any human faces in the images, and if there are, return an encoding of the location and spatial extent of each human face in the image.
- *Face Recognition:* The business of determining the identity of the face among the given individuals.

1.4. Object and Outline of the Thesis

The objective of this thesis is to design a system which includes a robot that dynamically follows the faces of individuals. Hence, to implement a real-time face detection software running together with the functions necessary to move the robot and camera, is the main task for the project. The content of the thesis is organized as follows:

The first chapter is an introduction chapter in which some introductory materials are given together with the objective and outline of the thesis, for the reader to better understand the next chapters of the thesis report.

The second chapter gives general information about face detection. In this chapter the most common techniques of face detection are given without detail. Some skin detection methods are also explained in this chapter. Using skin detection, the probable human skin areas are determined and then only those skin areas are explored to understand if there is a human face in the area or not.

The two face detection algorithms, which are implemented in this thesis, are explained in the third chapter. The algorithms are given in detail according to their original implementations without giving the modifications or further developments.

In Chapter 4, a flow diagram is given, which shows all the phases of the whole system. Then every part in this flow diagram is explained in detail, together with the modifications and developments of the methods, algorithms used. The experimental results are also given in this chapter.

Finally, in the last chapter, conclusions are summarized, and the results of our project giving directions for further studies are discussed.

CHAPTER 2

OVERVIEW OF SKIN AND FACE DETECTION

In this chapter, general concepts of skin and face detection are discussed. The most common methods used for both skin detection and face detection are examined.

2.1. Skin Detection

Obviously, every human face is a part of the human skin, which means that we can use skin detection as a preprocess for face detection. The reason for using skin detection is that it works much faster than face detection. The areas which are labeled as non-skin are not applied to face detection algorithm, which makes the whole program work faster.

Detection of skin in video is an important component of systems for detecting, recognizing, and tracking faces and hands. It can be used to begin the process of face recognition or facial expression extraction, and can provide an initial estimate or follow-up verification for face and hand tracking algorithms. Being able to do these types of detection based merely on skin color would eliminate the need for cumbersome tracking devices or artificially placed color keys. A video conference can be a good example for this, in which the speakers can move freely, while cameras automatically track their positions, always keeping them centered in the frame.

One of the primary problems in skin detection is color constancy. Ambient light, bright lights, and shadows change the apparent color of an image. Different cameras affect the color values as well. Movement of an object can cause blurring of colors. Finally, skin tones vary dramatically within and across individuals.

The skin detection methods use off-line training rather than incremental run-time training, and are tested on images representing a wide variety of people, environments, cameras, and lighting conditions. These methods usually use different color spaces for skin detection, hence it would be a benefit for the reader to learn about color spaces first, which are explained in the next section.

2.1.1. Color Spaces

The color is understood to be made of two components: *Chrominance* and *Luminanc*e. The Chrominance is property of the object itself that identifies the coloring of that object. On the other hand, the Luminance is a property of the lighting environment surrounding the object. Each colored image contains values of both luminance and chrominance. It is obvious that colored objects appear black on the darkness and white under extremely bright light. On the other hand, glass is always transparent under all lighting conditions. It is recognized that the luminance is a major problem in detecting human skin color since human skin color varies widely by changing of lighting conditions.

Luminance and chrominance make human skin colors cover a wide area of the color spectrum; this makes it hard to detect the human skin color based on the RGB color values themselves. Hence, the need for different color spaces that reduce the variance of human skin colors arises.

Color space, is a method of representing color information obtained for an image. Human skin color tends to cluster in different color spaces. Different color spaces tend to enhance some characteristics of images on the expense of others. Some of the color spaces are discussed in this section.

The colored images are stored in the computer files as two-dimensional arrays of three distinct color values, usually Red, Green and Blue. In this RGB color space, great change in color values might not be detected by human observer. Moreover, variations in lighting highly change the RGB values of the image. Therefore, this color space is not very suitable for human skin detection.

2.1.1.1. Normalized Color Space

Normalized rg-color space is a good solution for the problem of varying magnitude. Normalized values of r, g are calculated from RGB values of the RGB-color space as follows:

$$r = R / (R + G + B) \qquad (2.1)$$

$$g = G / (R + G + B)$$

This color space reduces the complexity of processing, since it reduces the 3-D RGB-color space into a 2-D rg-color space. This makes the calculations and manipulations easier. Moreover, this color space eliminates the effect of color intensity on human skin color. It was observed by [21] that the human skin color tend to cluster on three areas of the rg-color space, one cluster for Caucasians, one for Asians and another for Dark people. This gives another complexity of searching for a skin sample in three areas, which not only introduces more error and complexity, but also covers almost all of the rg-color space.

2.1.1.2. HSV Color Space

The Hue, Saturation, Value (or HSV) model defines a color space in terms of three constituent components: *Hue*, H is the color type such as red, blue, or yellow and ranges from 0-360 (but normalized to 0-100% in some applications). *Saturation*, S is the "vibrancy" of the color and ranges from 0-100%. It is sometimes called the "purity" and the lower the saturation of a color, the more "grayness" is present and the more faded the color will appear, thus it is useful to define *desaturation* as the qualitative inverse of saturation. *Value*, V is the brightness of the color and ranges from 0-100%

As discussed in [21], the human skin colors cover almost all the S values for a limited H range. This is due to the different skin colors of different human races. This leaves the S range useless in detecting the human skin, and makes it difficult to detect the skin color based on the H by itself. In addition, the shape of the human skin color cluster in HSV-space is difficult to be represented mathematically.

2.1.1.3. YUV Color Space

YUV, also known as Y'CbCr and YPbPr, is a color space in which the Y stands for the luminance component (the brightness) and U and V are chrominance (color) components. It is commonly used in video applications, where it is also referred to as component video.

YUV signals are created from an original RGB (red, green and blue) source. The weighted values of R, G and B are added together to produce a single Y signal, representing the overall brightness, or luminance, of that spot. The U signal is then created by subtracting the Y from the blue signal of the original RGB, and V by subtracting the Y from the red.

2.1.1.4. YIQ Color Space

YIQ, like YUV, is a color space used in television signals. The Y component, like in YUV, is used to encode luminance information, and is the only component used by black-and-white television receivers. The I and Q signals contain the actual color information. The YIQ color space is actually exactly the same as YUV, except the I-Q plane differs from the U-V plane by a simple 33-degree rotation. This rotation puts one of the two color axes in the orange region of the color space, which is where flesh tones are found. This axis is known as the "I" axis, which stands for in-phase, while the other color axis is known as "Q", meaning quadrature. Because of this feature of “I” axis we used this color space in the thesis for skin detection. The details are given in Sec. 4.1.1, while the formula for RGB to YIQ conversion is given here:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.229 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.320 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.2}$$

2.1.2. Skin Detection Methods

A number of existing systems employ a skin detection algorithm. Face detectors and face trackers make up the majority of these systems. In real-time trackers, a skin detector does not do the actual tracking, but instead, does the initial location of the face and acts as a reality check. Face recognition and facial expression analysis systems often use skin detection as an initial step. Hand trackers, while not as common as face trackers, also can make use of skin detectors to aid in tracking. A majority of the skin detection algorithms use color histograms for segmentation, either directly or for Maximum Likelihood (ML) estimation [22]; others perform pixel classification based on predefined ranges in color space [23]. Differences among existing skin detection systems occur primarily in the following areas: color space used, incremental run-time training versus off-line

training, and the techniques used for sorting and identifying colors corresponding to skin. Individual color spaces used in prior skin detection methods include HSV [23, 24], normalized RGB [22], simple RGB [25], YUV [26]. These methods simply convert the RGB pixel values into corresponding color space components and checks all or some of these components, if they are between some boundaries then that pixel is assumed to be a skin pixel, otherwise it is not. These simple methods are not explained here, except the one that is used in the thesis (YIQ Color Space method), which is explained in more detail in Sec. 4.1.1.

2.1.2.1. Lookup Table Method

The first algorithm presented here uses a color histogram-based approach for segmenting the skin pixels from the remainder of the image. This approach relies on the assumption that skin colors form a cluster in some color measurement space [25]. A two-dimensional histogram is used to represent the skin tones. By using the two parameters of a color system which do not correspond to intensity or illumination, the histogram should be more stable with respect to differences in illumination and local variations caused by shadows [24]. The two-dimensional histogram used here is referred to as the lookup table (LT). Each cell in the LT represents the number of pixels with a particular range of color value pairs. A set of training images is used to construct the LT as follows: Each image, having been previously segmented by hand, undergoes a color space transformation. Then, for each pixel marked as skin, the appropriate cell in the LT is incremented. After all the images have been processed, the values in the LT are divided by the largest value present. The normalized values in the LT cells reflect the likelihood that the corresponding colors will correspond to skin. To perform skin detection, an image is first transformed into the color space. For each pixel in the image, the color values index the normalized value in the LT. If this value is greater than a threshold, the pixel is identified as skin. Otherwise, the pixel is considered to be non-skin.

2.1.2.2. Bayesian Method

The second method presented here uses Bayes' Theorem to choose the most likely hypothesis, given the value of a feature. Here, the mutually exclusive classes are skin (*s*) and non-skin (-*s*). The feature is the two-dimensional color value $\bar{x}$ of a pixel. In contrast with the lookup table method, the Bayesian method uses two color histograms, one for skin and one for non-skin pixels. When constructing the probabilities for Bayesian decision making, there are two possible assumptions. In the first case, the probability that a pixel is skin is assumed to be the same as the probability that a pixel is non-skin *(P(s) = P(-s))*. This corresponds to maximum likelihood (ML) estimation. For any pixel, if the ratio found from (2.3) is greater than one, then the pixel can be classified as skin. Otherwise, the pixel can be classified as non-skin:

$$\frac{P(s \mid \bar{x})}{P(-s \mid \bar{x})} = \frac{P(\bar{x} \mid s)}{P(\bar{x} \mid -s)} \tag{2.3}$$

In the second case, the values of probabilities *P(s)* and *P(-s)* are estimated from the training data. This corresponds to maximum a posteriori (MAP) estimation. If the ratio in (2.4) is greater than 1, then the pixel can be classified as skin. Otherwise, the pixel can be classified as non-skin:

$$\frac{P(s \mid \bar{x})}{P(-s \mid \bar{x})} = \frac{P(\bar{x} \mid s)P(s)}{P(\bar{x} \mid -s)P(-s)} \tag{2.4}$$

2.2. Face Detection

In this section, we review existing techniques to detect faces on a still, single intensity or color image. There are hundreds of different techniques for detection of faces in still images, but only some of the most common ones are explained here.

Although there are some overlaps, according to [2], single face detection methods are classified into four categories:

1. Knowledge-based methods
2. Feature invariant methods
3. Template matching methods
4. Appearance-based methods

Below, the motivation and general approach of each category are given in more detail.

2.2.1. Knowledge-Based Methods

In this approach, face detection methods are developed based on the rules derived from the researcher's knowledge of human faces. It is easy to come up with simple rules to describe the features of a face and their relationships. For example, a face often appears in an image with two eyes that are symmetric to each other, a nose, and a mouth. The relationships between features can be represented by their relative distances and positions. Facial features in an input image are extracted first, and face candidates are identified based on the coded rules. A verification process is usually applied to reduce false detections.

One problem with this approach is the difficulty in translating human knowledge into well-defined rules. If the rules are detailed (i.e., strict), they may fail to detect faces that do not pass all the rules. If the rules are too general, they may give many false positives. Moreover, it is difficult to extend this approach to detect faces in different poses since it is challenging to enumerate all possible cases. On the other hand, heuristics about faces work well in detecting frontal faces in uncluttered scenes.

Yang and Huang used a hierarchical knowledge-based method to detect faces [5]. Their system consists of three levels of rules. At the highest level, all possible face

candidates are found by scanning a window over the input image and applying a set of rules at each location. The rules at a higher level are general descriptions of what a face looks like while the rules at lower levels rely on details of facial features. A multiresolution hierarchy of images is created by averaging and subsampling. An example is shown in Fig. 2.1, here in (a) n = 1 (i.e. original image), in (b) n = 4, in (c) n = 8 and in (d) n = 16. Each square cell consists of (nxn) pixels in which the intensity of each pixel is replaced by the average intensity of the pixels in that cell.

(a) (b) (c) (d)

Figure 2.1. Original and corresponding low resolution images for different n values [5].

Rules are coded based on human knowledge about the characteristics (e.g., intensity distribution and difference) of the facial regions. Examples of the coded rules used to locate face candidates in the lowest resolution include: “the center part of the face (the dark shaded parts in Fig. 2.2) has four cells with a basically uniform intensity,” “the upper round part of a face (the light shaded parts in Fig. 2.2) has a basically uniform intensity,” and “the difference between the average gray values of the center part and the upper round part is significant.”

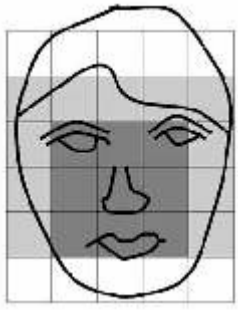


Figure 2.2. A typical face used in knowledge-based top-down methods [5]

The lowest resolution (Level 1) image is searched for face candidates and these are further processed at finer resolutions. At Level 2, local histogram equalization is performed on the face candidates received from Level 2, followed by edge detection. Surviving candidate regions are then examined at Level 3 with another set of rules that respond to facial features such as the eyes and mouth. One attractive feature of this method is that a coarse-to-fine or focus-of-attention strategy is used to reduce the required computation. Although it does not result in a high detection rate, the ideas of using a multiresolution hierarchy and rules to guide searches have been used in later face detection works, such as [6] at which Kotropoulos and Pitas presented a rule-based localization method which is similar to [5].

2.2.2. Feature Invariant Methods

In contrast to the knowledge-based top-down approach, researchers have been trying to find invariant features of faces for detection. The underlying assumption is based on the observation that humans can effortlessly detect faces and objects in different poses and lighting conditions and, so, there must exist properties or features which are invariant over these variabilities. Numerous methods have been proposed to first detect facial features and then to infer the presence of a face. Facial features such as eyebrows, eyes, nose, mouth, and hair-line are commonly

extracted using edge detectors. Based on the extracted features, a statistical model is built to describe their relationships and to verify the existence of a face. One problem with these feature-based algorithms is that the image features can be severely corrupted due to illumination, noise, and occlusion. Feature boundaries can be weakened for faces, while shadows can cause numerous strong edges which together render perceptual grouping algorithms useless.

2.2.2.1. Facial Features

Graf et al. developed a method to locate facial features and faces in gray scale images [8]. After band pass filtering, morphological operations are applied to enhance regions with high intensity that have certain shapes (e.g., eyes). The histogram of the processed image typically exhibits a prominent peak. Based on the peak value and its width, adaptive threshold values are selected in order to generate two binarized images. Connected components are identified in both binarized images to identify the areas of candidate facial features. Combinations of such areas are then evaluated with classifiers, to determine whether and where a face is present. Their method has been tested with head-shoulder images of 40 individuals and with five video sequences where each sequence consists of 100 to 200 frames. However, it is not clear how morphological operations are performed and how the candidate facial features are combined to locate a face.

Takacs and Wechsler described a biologically motivated face localization method based on a model of retinal feature extraction and small oscillatory eye movements [9]. Their algorithm operates on the conspicuity map or region of interest, with a retina lattice modeled after the magnocellular ganglion cells in the human vision system. The first phase computes a coarse scan of the image to estimate the location of the face, based on the filter responses of receptive fields. Each receptive field consists of a number of neurons which are implemented with Gaussian filters tuned to specific orientations. The second phase refines the

conspicuity map by scanning the image area at a finer resolution to localize the face. The error rate on a test set of 426 images (200 subjects from the FERET, Facial Recognition Technology Database [37]) is 4.69 percent. Han et al. developed a morphology-based technique to extract what they call eye-analogue segments for face detection [10]. They argue that eyes and eyebrows are the most salient and stable features of human face and, thus, useful for detection. They define eye-analogue segments as edges on the contours of eyes. First, morphological operations such as closing, clipped difference, and thresholding are applied to extract pixels at which the intensity values change significantly. These pixels become the eye-analogue pixels in their approach. Then, a labeling process is performed to generate the eye-analogue segments. These segments are used to guide the search for potential face regions with a geometrical combination of eyes, nose, eyebrows and mouth. The candidate face regions are further verified by a neural network similar to [11]. Their experiments demonstrate a 94 percent accuracy rate using a test set of 122 images with 130 faces.

2.2.2.2. Skin Color

Human skin color has been used and proven to be an effective feature in many applications from face detection to hand tracking. Although different people have different skin color, several studies have shown that the major difference lies largely between their intensity rather than their chrominance. Several color spaces have been utilized to label pixels as skin including RGB, normalized RGB, HSV (or HSI), YCrCb, YIQ, YES, CIE XYZ, and CIE LUV.

Since skin detection has been used in the thesis as a preprocess, more detailed information is given in the thesis about skin detection which is located in part 2.1.

2.2.3. Template Matching Methods

In template matching, a standard face pattern (usually frontal) is manually predefined or parameterized by a function. Given an input image, the correlation values with the standard patterns are computed for the face contour, eyes, nose, and mouth independently. The existence of a face is determined based on the correlation values. This approach has the advantage of being simple to implement. However, it has proven to be inadequate for face detection since it cannot effectively deal with variation in scale, pose, and shape. Multiresolution, multiscale, subtemplates, and deformable templates have subsequently been proposed to achieve scale and shape invariance.

2.2.3.1. Predefined Templates

Craw et al. presented a localization method based on a shape template of a frontal-view face (i.e., the outline shape of a face) [12]. A Sobel filter is first used to extract edges. These edges are grouped together to search for the template of a face based on several constraints. After the head contour has been located, the same process is repeated at different scales to locate features such as eyes, eyebrows, and lips. Later, Craw et al. describe a localization method using a set of 40 templates to search for facial features and a control strategy to guide and assess the results from the template-based feature detectors [13].

Sinha used a small set of spatial image invariants to describe the space of face patterns, [14]. His key insight for designing the invariant is that, while variations in illumination change the individual brightness of different parts of faces (such as eyes, cheeks, and forehead), the relative brightness of these parts remain largely unchanged. Determining pairwise ratios of the brightness of a few such regions and retaining just the “directions” of these ratios (i.e., Is one region brighter or darker than the other?) provides a robust invariant. Thus, observed brightness

regularities are encoded as a ratio template which is a coarse spatial template of a face with a few appropriately chosen subregions that roughly correspond to key facial features such as the eyes, cheeks, and forehead. The brightness constraints between facial parts are captured by an appropriate set of pairwise brighter-darker relationships between subregions. A face is located if an image satisfies all the pairwise brighter-darker constraints. Fig. 2.3 shows the enhanced template which is composed of 16 regions (the gray boxes) and 23 relations (shown by arrows). These defined relations are furthered classified into 11 essential relations (solid arrows) and 12 confirming relations (dashed arrows). Each arrow in the figure indicates a relation, with the head of the arrow denoting the second region (i.e., the denominator of the ratio). A relation is satisfied for face temple if the ratio between two regions exceeds a threshold and a face is localized if the number of essential and confirming relations exceeds a threshold.

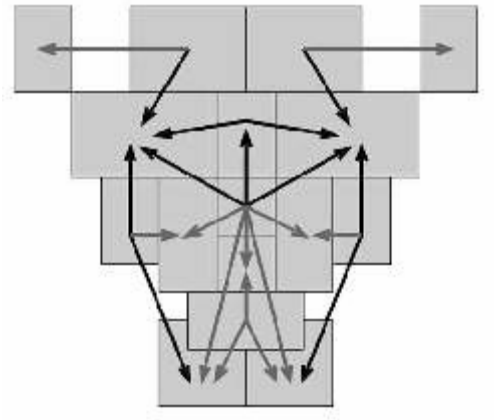


Figure 2.3. A 14x16 pixel ratio template for face localization based on Sinha method [14].

2.2.3.2. Deformable Templates

Yuille et al. used deformable templates to model facial features that fit an a priori elastic model to facial features (e.g., eyes) [15]. In this approach, facial features are described by parameterized templates. An energy function is defined to link edges, peaks, and valleys in the input image to corresponding parameters in the template. The best fit of the elastic model is found by minimizing an energy function of the parameters. Although their experimental results demonstrate good performance in tracking nonrigid features, one drawback of this approach is that the deformable template must be initialized in the proximity of the object of interest.

2.2.4. Appearance-Based Methods

Contrasted to the template matching methods where templates are predefined by experts, the "templates" in appearance-based methods are learned from examples in images. In general, appearance-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and nonface images. The learned characteristics are in the form of distribution models or discriminant functions that are consequently used for face detection. Meanwhile, dimensionality reduction is usually carried out for the sake of computation efficiency and detection efficacy.

Many appearance-based methods can be considered in a probabilistic framework. An image or feature vector derived from an image is viewed as a random variable x, and this random variable is characterized for faces and nonfaces by the class-conditional density functions $p(x|face)$ and $p(x|nonface)$. Bayesian classification or maximum likelihood can be used to classify a candidate image location as face or nonface. Unfortunately, a straightforward implementation of Bayesian classification is infeasible because of the high dimensionality of x, because

p(x|face) and *p(x|nonface)* are multimodal, and because it is not yet understood if there are natural parameterized forms for *p(x|face)* and *p(x|nonface)*. Hence, much of the work in an appearance-based method concerns empirically validated parametric and nonparametric approximations to *p(x|face)* and *p(x|nonface)*.

2.2.4.1. Eigenfaces

An early example of employing eigenvectors in face recognition was done by Kohonen [16] in which a simple neural network is demonstrated to perform face recognition for aligned and normalized face images. The neural network computes a face description by approximating the eigenvectors of the image's autocorrelation matrix. These eigenvectors are later known as Eigenfaces.

One of the face detection methods that is used in the thesis is the Eigenface method, and the details about it are explained in Sec. 3.1.

2.2.4.2. Distribution-Based Methods

A probabilistic visual learning method based on density estimation in a high-dimensional space using an eigenspace decomposition was developed by Moghaddam and Pentland [17]. Principal component analysis (PCA) is used to define the subspace best representing a set of face patterns. These principal components preserve the major linear correlations in the data and discard the minor ones. This method decomposes the vector space into two mutually exclusive and complementary subspaces: the principal subspace (or feature space) and its orthogonal complement. Therefore, the target density is decomposed into two components: the density in the principal subspace (spanned by the principal components) and its orthogonal complement (which is discarded in standard PCA). This is shown in Fig. 2.4 for a face image for an arbitrary density. Every data point *x* is decomposed into two components: distance in feature space (DIFS)

and distance from feature space (DFFS). A multivariate Gaussian and a mixture of Gaussians are used to learn the statistics of the local features of a face. These probability densities are then used for object detection based on maximum likelihood estimation. The proposed method has been applied to face localization, coding, and recognition.

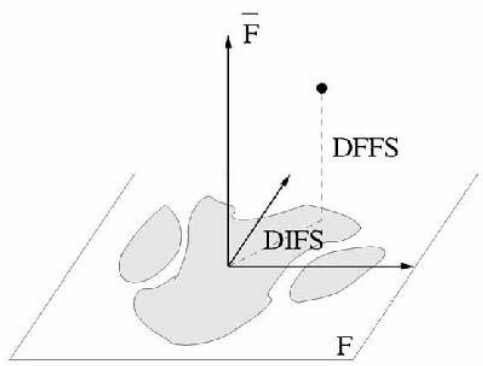


Figure 2.4. Decomposition of a face image space into the principal subspace F and its orthogonal complement F for an arbitrary density. [17].

Another distribution-based method is the Fisher Linear Discriminant [18], which is explained in Chapter 3. This method is recently used to outperform the classical eigenface method.

2.2.4.3. Neural Networks

Neural networks have been applied successfully in many pattern recognition problems, such as optical character recognition, object recognition, and autonomous robot driving. Since face detection can be treated as a two class pattern recognition problem, various neural network architectures have been proposed. The advantage of using neural networks for face detection is the feasibility of training a system to capture the complex class conditional density of

face patterns. However, one drawback is that the network architecture has to be extensively tuned (number of layers, number of nodes, learning rates, etc.) to get exceptional performance.

Among all the face detection methods that used neural networks, the most significant work is arguably done by Rowley et al. [19]. A multilayer neural network is used to learn the face and nonface patterns from face/nonface images (i.e., the intensities and spatial relationships of pixels). They also used multiple neural networks and several arbitration methods to improve performance. There are two major components: multiple neural networks (to detect face patterns) and a decision-making module (to render the final decision from multiple detection results). As shown in Fig. 2.5, the first component of this method is a neural network that receives a 20x20 pixel region of an image and outputs a score ranging from -1 to 1. Given a test pattern, the output of the trained neural network indicates the evidence for a nonface (close to -1) or face pattern (close to 1). To detect faces anywhere in an image, the neural network is applied at all image locations. To detect faces larger than 20x20 pixels, the input image is repeatedly subsampled, and the network is applied at each scale. Nearly 1,050 face samples of various sizes, orientations, positions, and intensities are used to train the network. In each training image, the eyes, tip of the nose, corners, and center of the mouth are labeled manually and used to normalize the face to the same scale, orientation, and position. The second component of this method is to merge overlapping detection and arbitrate between the outputs of multiple networks. Simple arbitration schemes such as logic operators (AND/OR) and voting are used to improve performance.

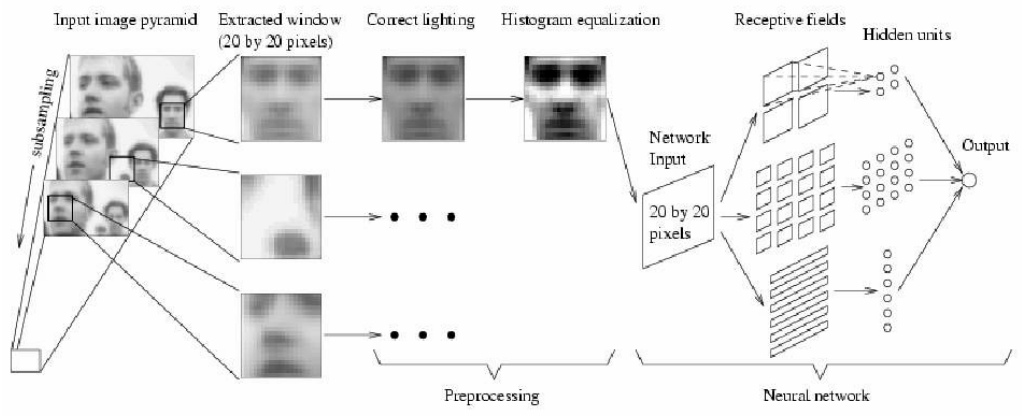


Figure 2.5. System diagram of Rowley's method [19].

2.2.4.4. Hidden Markov Model

Intuitively, a face pattern can be divided into several regions such as the forehead, eyes, nose, mouth, and chin. A face pattern can then be recognized by a process in which these regions are observed in an appropriate order (from top to bottom and left to right). Instead of relying on accurate alignment as in template matching or appearance-based methods (where facial features such as eyes and noses need to be aligned well with respect to a reference point), this approach aims to associate facial regions with the states of a continuous density Hidden Markov Model. HMM-based methods usually treat a face pattern as a sequence of observation vectors where each vector is a strip of pixels, as shown in Fig. 2.6 (a). During training and testing, an image is scanned in some order (usually from top to bottom) and an observation is taken as a block of pixels, as shown in Fig. 2.6 (a). For face patterns, the boundaries between strips of pixels are represented by probabilistic transitions between states, as shown in Fig. 2.6 (b), and the image data within a region is modeled by a multivariate Gaussian distribution. An observation sequence consists of all intensity values from each block. The output states correspond to the classes to which the observations belong. After the HMM has been trained, the output probability of an observation determines the class to

which it belongs. HMMs have been applied to both face recognition and localization. Samaria [20] showed that the states of the HMM he trained corresponds to facial regions, as shown in Fig. 2.6 (b). In other words, one state is responsible for characterizing the observation vectors of human foreheads, and another state is responsible for characterizing the observation vectors of human eyes. For face localization, an HMM is trained for a generic model of human faces from a large collection of face images. If the face likelihood obtained for each rectangular pattern in the image is above a threshold, a face is located.

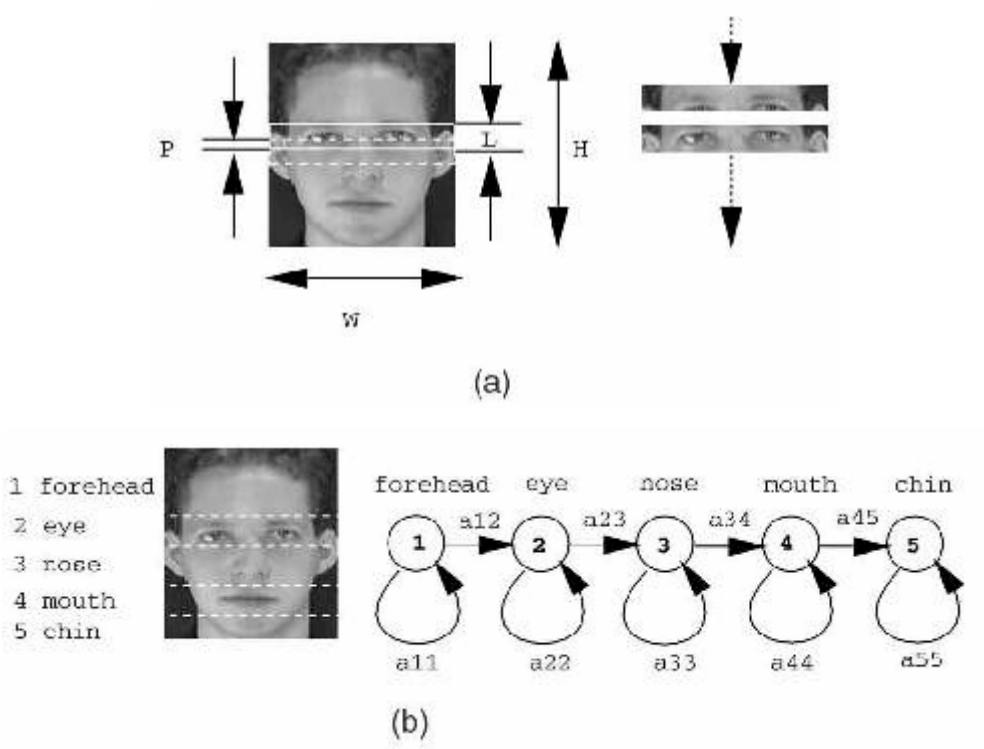


Figure 2.6. Hidden Markov model for face localization. (a) Observation vectors, (b) Hidden states [20].

CHAPTER 3

IMPLEMENTED FACE DETECTION METHODS

As it is explained in Chapter 2 there are a lot of face detection methods which have been used for many applications for many years. Among these, the appearance based methods (see Sec. 2.2.4) are the most commonly used techniques nowadays since the detection performance of the other methods are usually worse. In our case the face detection process is a real time application which makes its process time as critical as its performance. After our survey on face detection techniques we had to eliminate very popular methods, like neural networks and hidden Markow model because of longer process time. Subspace LDA method and Eigenface method, which are similar from some aspects, were our surviving candidates when comparing "computational cost – detection performance" efficiency. Subspace LDA method is a little more complex hence its performance is expected to be better, but of course computational cost is high. Both methods have been implemented and tested during the research process and the eigenface method has been decided to be used.

3.1. Eigenface Method

Eigenface method has been used for face detection for many years. The popularity of this method comes from its simplicity. Face detection performance of this method is not as good as many others, but the algorithm is easy to implement and there are many resources, which makes it the first method that comes to mind

when the subject is face detection. Among the many resources on face detection using eigenfaces the paper of Turk et.al. [1] is accepted as the leading source for long years. With this reason we implemented the algorithm according to their published resources.

Turk et.al. [1], actually developed their system for face recognition but it can be used also for face detection purposes. According to their approach, face recognition problem is a two-dimensional problem to be solved, which also means that faces can be described by a small set of 2-D characteristic views. The system works by projecting face images onto a feature space that spans the significant variations among known face images. The significant features are named as "eigenfaces", because they are the eigenvectors (principal components) of the set of faces; they do not necessarily correspond to features such as eyes, ears and noses, which are thought as the critical sections of the face and usually used for face detection and recognition. The projection operation characterizes an individual face by a weighted sum of the eigenface features and so to recognize a particular face it is necessary only to compare these weights with those of known individuals. One particular advantage of this approach is that it provides the ability to learn and later recognize new faces in an unsupervised manner, while one other is that it is easy to implement.

Looking at this face detection solution from mathematical side, the principal components of the distribution of faces, which are actually the eigenvectors of the covariance matrix of the set of face images, treating an image as point (or vector) in a very high dimensional space is sought. The eigenvalues are ordered, because the greater an eigenvalue, the more effect corresponding eigenvector has, since this means that eigenvector has a larger amount of variation among the face images.

These eigenvectors can be thought of as a set of features that together characterize the variation between face images. Each image location contributes more or less to each eigenvector, so that it is possible to display each of these eigenvectors as a sort of ghostly face image which is called an "eigenface". Some of the face images, belonging to METU Vision Face Database (see App. B.1) that we studied during thesis, are illustrated in Fig. 3.1, and some of the eigenfaces belonging to this dataset are given in Fig. 3.3.

This approach, proposed by [1], involves the following steps for training phase:
1. Get the training images used for training purposes
2. Calculate the eigenfaces from the training set, keeping only the M' eigenfaces that correspond to the highest eigenvalues. These M' images define the face space.
3. Calculate the corresponding distribution in M'-dimensional weight space for each known individual, by projecting their face images onto the face space.

After this initialization process, following steps are applied to detect a face:
1. Calculate a set of weights based on the input image and the M' eigenfaces by projecting the input image onto each of the eigenfaces.
2. Determine if the image is a face or not, by checking to see if the image is sufficiently close to face space.

3.1.1. Calculating Eigenfaces

A face image of height *n* and width *m* has obviously *nxm* pixels and usually stored in a two-dimensional array of 8-bit intensity values. This virgin input face is shown with *I(x,y)* whose size is *nxm*. Converting this two-dimensional array to a vector means putting columns one after the other starting from the first column. This gives a vector of size *Px1*, where $P = nxm$, and we show this vector with letter Γ. According to this, an image of size 76x106 turns into a vector of

dimension 8056, or equivalently a point in 8056-dimensional space. As a result, a bundle of images maps to a collection of points in this huge space.

Since the face images should be similar to each other little or much, these collection of points will not be randomly distributed in this huge image space, but will be relatively “near” to each other. This means that they can be described by a lower dimensional subspace. Turk et.al. used principal component analysis for this purpose, which seeks the lower dimensional vectors that best account for the distribution of face images within the entire image space. These vectors define the subspace of face images, which we call "face space". Each vector is of length *nxm*, describes an *nxm* image, and is a linear combination of the original face images. These vectors are the eigenvectors of the covariance matrix corresponding to the original face images, and they are face-like in appearance, so we refer to them as "eigenfaces". Some examples of eigenfaces are shown in Fig. 3.3.

In the following, the process of extracting eigenfaces will be explained after giving some insight on the terms that are going to be used.

A *PxP* matrix A is said to have an eigenvector X, and corresponding eigenvalue λ if

$$AX = \lambda X \tag{3.1}$$

Evidently, Eq. (3.1) can hold only if

$$Det\,|A - \lambda I| = 0 \tag{3.2}$$

which, if expanded out, is a P^{th} degree polynomial in λ whose roots are the eigenvalues. This proves that there are always P (not necessarily distinct) eigenvalues. Equal eigenvalues coming from multiple roots are called "degenerate".

The training set of face images converted to vectors are denoted as $\Gamma_1, \Gamma_2, ..., \Gamma_M$ then the average of the set is defined by

$$\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n \tag{3.3}$$

Each face differs from the average by the vector

$$\Phi_i = \Gamma_i - \Psi \tag{3.4}$$

An example training set is shown in Fig. 3.1. The average face, Ψ of these images is seen in Fig. 3.2 (a), when histogram equalization is applied to the training faces, recalculated average face is shown in Fig. 3.2 (b).

Figure 3.1. Some of the faces used as training images, chosen from METU Vision Face Database

(a) (b)

Figure 3.2. Average of the training faces (a) without histogram equalization (b) with histogram equalization

This set of very large vectors is then subject to principal component analysis, which seeks a set of M orthonormal vectors, u_n, which best describes the distribution of the data. The k^{th} vector, u_k , is chosen such that

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(u_k^T \Phi_n)^2 \tag{3.5}$$

is a maximum, subject to

$$u_l^T u_k = \delta_{lk} = \begin{cases} 1, \textit{ if } l = k \\ 0, \textit{ otherwise} \end{cases} \tag{3.6}$$

Using the derivation in [33], it is understood that the vectors u_k and scalars λ_k are the eigenvectors and eigenvalues, respectively of the covariance matrix

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n \Phi_n^T \tag{3.7}$$

Collecting the mean subtracted input images in matrix A gives an array of dimension PxM, where $P=nxm$.

$$A = [\Phi_1, \Phi_2, \ldots, \Phi_M] \tag{3.8}$$

Hence the covariance can be written in a more compact form as

$$C = AA^T \tag{3.9}$$

The covariance matrix C, however is a PxP real symmetric matrix, and calculating the P eigenvectors and eigenvalues is a very difficult task for typical image sizes. So a computationally feasible method is used to find these eigenvectors.

In almost all cases the number of data points in the image space is less than the dimension of the space ($M < P$). In this circumstance there will be only $M-1$ meaningful eigenvectors, rather than P. The eigenvalues of the remaining eigenvectors will be equal to zero. Fortunately we can solve for the P dimensional eigenvectors in this case by first solving the eigenvectors of an MxM matrix such as solving 24 x 24 matrix rather than a 8056 x 8056 matrix and then, taking appropriate linear combinations of the face images Φ_i.

Let us consider the eigenvectors v_i of A^TA such that

$$A^TA\ v_i = \mu_i\ v_i \tag{3.10}$$

Premultiplying both sides by A, gives

$$AA^TA\ v_i = \mu_i\ Av_i \tag{3.11}$$

from which we see that $A\ v_i$ are the eigenvectors of $C = AA^T$.

Following these analysis, we construct the MxM matrix

$$L = A^TA \tag{3.12}$$

where $L_{mn} = \Phi^T_m\ \Phi_n$, and find the M eigenvectors, v_l , of L. These vectors determine linear combinations of the M training set face images to form the eigenfaces u_l.

$$u_l = \sum_{k=1}^{M} v_{lk}\Phi_k, \quad l = 1, \ldots, M \tag{3.13}$$

As a result we could obtain the eigenfaces using much less computational effort, reducing the order of calculations from the number of pixels in the images (P) to the number of images in the training set (M). In practice, the training set of face images will be relatively small ($M << P$), and the calculations become quite manageable.

Since we calculated the eigenvectors of the covariance matrix of the mean subtracted images, the eigenvalues with greater values gives us more information about the variation of a face image from the other images, other faces. Hence the associated eigenvalues allow us to rank the eigenvectors according to their usefulness in characterizing the variation among the images. Fig. 3.3 shows some of the eigenfaces derived from the 24 of the training images of METU Vision face database. The number below each image indicates the principal component number, ordered according to eigenvalues.

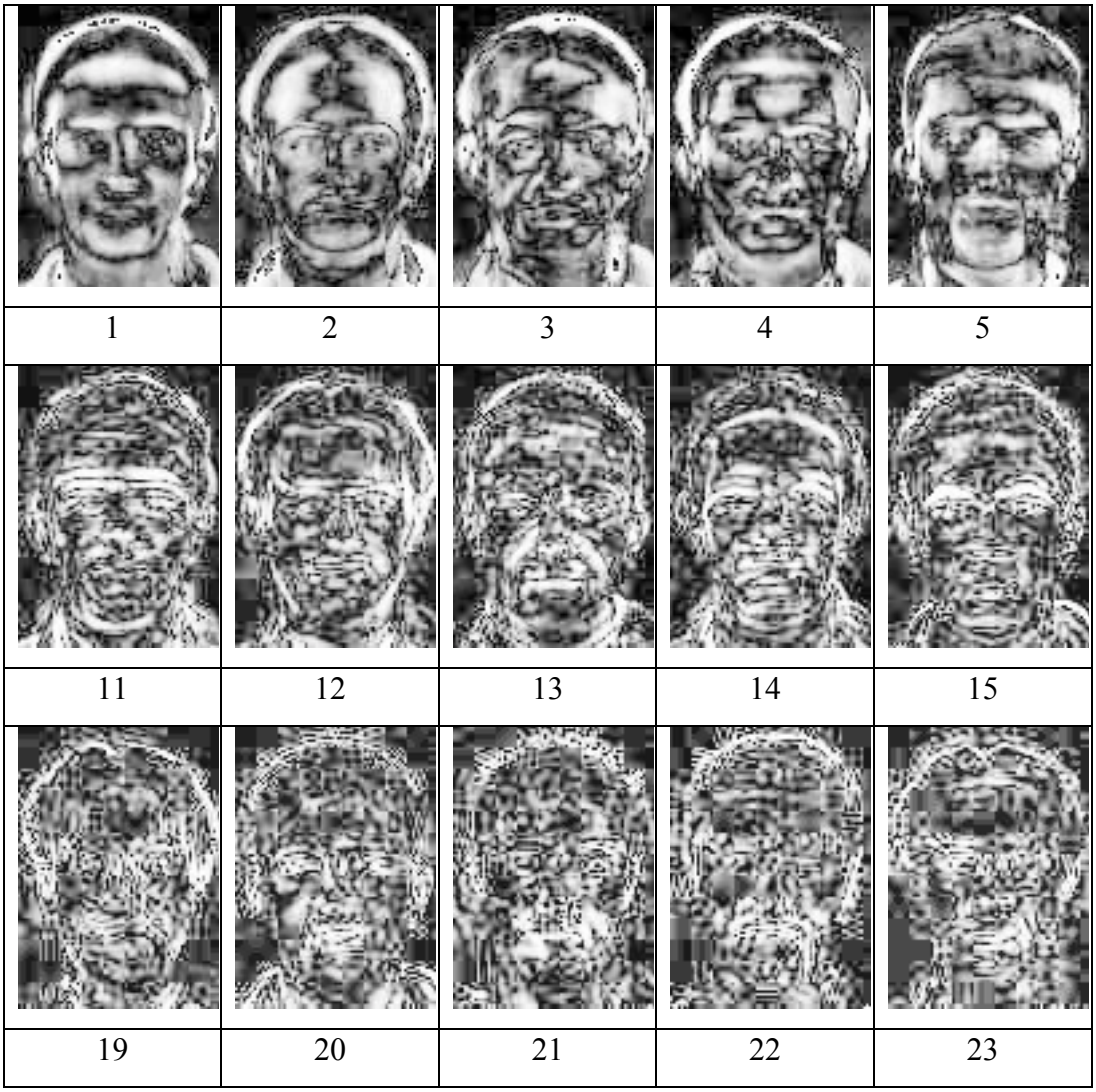


Figure 3.3. Some eigenfaces computed from METU Vision Face database.

3.1.2. Construction of Weight Space

The eigenface images calculated from the eigenvectors of L, span a basis set with which to describe face images. Sirovich et.al. [29] evaluated a limited version of this framework on an ensemble of M = 115 images of Caucasian males digitized in a controlled manner, and found that 40 eigenfaces were sufficient for a very

good description of face images. With M' = 40 eigenfaces, RMS pixel by pixel errors in representing cropped versions of face images were about 2%.

In practice, a smaller M' can be sufficient for identification, since accurate reconstruction of the image is not a requirement. Based on this idea, the proposed face recognition system lets the user specify the number of eigenfaces (M') that is going to be used in the detection. For maximum accuracy, the number of eigenfaces should be equal to the number of images in the training set. But, looking at Fig. 3.4, which shows the eigenvalues of a sixteen-image training set, one can see that after 7 eigenvalues the effect of the next eigenvalue is about 10% of the effect of first eigenvalue. But the best value for the number of eigenvectors to be used, can be decided after experiments, which are simulated in Sec. 4.5.

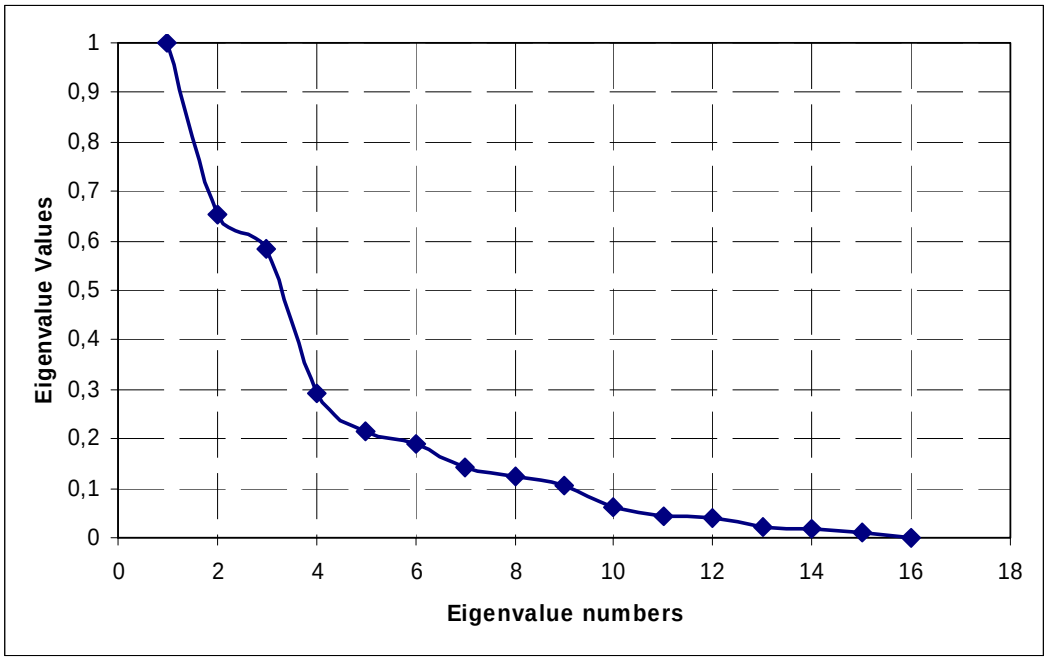


Figure 3.4. Eigenvalues of a sixteen-image dataset

The weight space is constructed by applying the following formula for each of the training images, Γ_i,

$$w_k = u_k^T(\Gamma_i - \Psi) \tag{3.14}$$

for k = 1, ... ,M'. This describes a set of point by point image multiplications and summations, and gives a scalar number for each k value. As a result a vector of length M' in the form

$$\Omega_i = [w_1, w_2, \ldots, w_{M'}]^T \tag{3.15}$$

is obtained for each training image representing that training image in the face space. The weight space is then composed of these Ω_i vectors and has dimensions $M'xM$.

3.1.3. Detection of Faces (Projection and Reconstruction)

When a new image (Γ), is wanted to be tested whether or not it contains a face, it is firstly projected onto the face space with the same operation seen in Eq. 3.14.

$$w_k = u_k^T(\Gamma - \Psi) \tag{3.16}$$

for k = 1, … , M'. And constructs a weight vector of the form $\Omega = [w_1, w_2, \ldots, w_{M'}]^T$. If face is to be recognized than this weight vector is compared with the weight vectors of the training images to understand which face class it belongs to. However, face recognition is not the topic in this thesis hence will not be given in more detail.

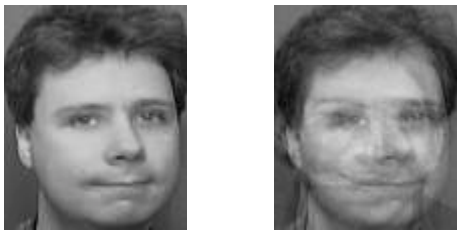
The weight vector of this new image is used in the detection process. The projection of the mean subtracted original image is now reconstructed back to see if it is similar to the input image. A face image which is reconstructed after projection into the face space, looks like a face, while the reconstruction of a non-face image differs from the input face. This property is used to detect the face, since the distance ϵ between the original image and the reconstructed image is simply the squared distance between the mean-adjusted input image;

$$\Phi = \Gamma - \Psi \tag{3.17}$$

and its reconstruction;

$$\Phi_f = \sum_{j=1}^{M'} w_j u_j \tag{3.18}$$

An example of an input image, Γ, and its reconstruction (mean-added) $\Phi_f + \Psi$, are seen in Fig. 3.5 (a) and (b), respectively.



(a) (b)

Figure 3.5. (a) An input image and (b) its reconstruction

The reconstruction error can then be calculated as follows

$$\varepsilon^2 = \left|\Phi - \Phi_f\right| \tag{3.19}$$

If this reconstruction error is below some threshold, which is determined experimentally, then it is classified as a "face", otherwise it is declared as non-face.

It is possible to relate weights to different parts of the image during the calculation of Eq. 3.19, since some parts of the face, such as eyes, mouth etc. are more effective in deciding whether or not the input image is a face. So in Eq. 3.19, greater weights are usually used for pixels near critical parts, and smaller weights are used for more "usual" parts, like forehead, cheek, chin.

3.2. Subspace LDA (Linear Discriminant Analysis) Method

A successful face detection or recognition methodology depends heavily on the particular choice of the features used by the classifier [31]. In a face recognition system these classifiers are used to distinguish a face from other faces, while in face detection, task of classifiers is to distinguish between face and non-face. Feature selection in pattern recognition involves the derivation of salient features from the raw input data in order to reduce the amount of data used for classification and simultaneously provide enhanced discriminatory power. Two popular techniques for selecting a subset of features are Principal Component Analysis (PCA) (See Sec. 3.1.1 Calculating Eigenfaces) and the Fisher Linear Discriminant (FLD). PCA is a standard decorrelation technique and following its application one derives an orthogonal projection basis which directly leads to dimensionality reduction, and possibly to feature selection. FLD is a method following PCA and operating then in a compressed subspace, seek for discriminatory features by taking into account within- and between-class scatter as the relevant information for pattern classification. While PCA maximizes for all the scatter as appropriate for signal representation, FLD differentiates between the within- and between-class scatter as appropriate for pattern classification.

FLD has been applied to solve face detection problems by applying first PCA for dimensionality reduction and then FLD for discriminant analysis. Belhumire, Hespanha, and Kriegman [32] developed an approach called Fisherfaces for face recognition and applied the method in this way. Zhao et.al. [34] proposed a similar method combining PCA and Linear Discriminant Analysis (LDA) and calling the method Subspace LDA.

3.2.1. Dimensionality Reduction

The training images are constructed in a different way in subspace LDA method than in eigenface. In the eigenface method all the classes that are used in the training process are the face classes of different individuals. But in subspace LDA, face and non-face classes are used. In the result of the algorithm the input image will be classified as belonging to one of the classes, and if this class is a face class then the input is labeled as face, on the other hand if it belongs to a non-face class it will be labeled as non-face. c_1 face and c_2 non-face classes are used, choosing c_1 and c_2 is critical and will be discussed later.

The procedure for finding the low dimensional eigenface space using PCA is same as in Sec. 3.1, hence will not be repeated here. What should be remembered is that, after applying PCA the image vectors of length P pixels, are represented by M' dimensional (M' < P) feature vectors here in the following form:

$$\Omega_k = [w_1, w_2, \ldots, w_{M'}]^T \tag{3.20}$$

3.2.2. Linear Discriminant Analysis

LDA training is carried out via scatter matrix analysis [35]. For a c-class problem, the within and between scatter matrices S_w, S_b are calculated as follows

$$S_w = \sum_{i-1}^{c} \sum_{x \in X_i} (x - m_i)(x - m_i)^T \tag{3.21}$$

$$S_b = \sum_{i=1}^{c} n_i (m_i - m_0)(m_i - m_0)^T \tag{3.22}$$

where m_i is the mean of class X_i, m_o is the mean of all the samples in all classes and n_i is the number of samples in class X_i. Here S_w is the within-class scatter matrix showing the average scatter of the sample vectors x of different classes X_i around their respective means m_i.

Similarly S_b is the between-class scatter matrix, representing the scatter of the conditional mean vectors m_i around the overall mean vector m_o.

Various measures are available quantifying the discriminatory power, a commonly used one being the ratio of the determinant of the between-class scatter matrix of the projected samples to the within-class scatter matrix of the projected samples:

$$J(T) = \frac{\left| T^T \cdot S_b \cdot T \right|}{\left| T^T \cdot S_w \cdot T \right|} \tag{3.23}$$

Maximizing this function increases intercluster distances by maximizing the between cluster matrix and decreases the intracluster distances by minimizing the within class scatter matrix.

Let us denote the optimal projection matrix which maximizes *J(T)* by *W*; then *W* can be obtained via solving the generalized eigenvalue problem [36]:

$$S_b w_i = S_w w_i \lambda_i \tag{3.24}$$

This generalized eigenvalue problem can be solved by first computing the eigenvalues as the roots of the following characteristic polynomial

$$| S_b - \lambda_i S_w | = 0 \tag{3.25}$$

and then solving the following, to find the corresponding eigenvectors, which are the columns of *W* matrix

$$(S_b - \lambda_i S_w) w_i = 0 \tag{3.26}$$

Finally the linear discriminant function for LDA is

$$g(\Omega_i) = W^T \cdot \Omega_i \tag{3.27}$$

for i=1,2,…,M. The *W* matrix includes *M'* eigenvectors, but only c-1 of the corresponding eigenvalues will have non-zero values. Hence the resultant *W* matrix will have the dimensions of (*M' x c*-1). Finally, all of the eigenface space projections, Ω_i, are now projected to this new FLD space and from now on every sample will be represented with $g(\Omega_i)$, which is of size (c-1 x 1) and i=1,2,…,M.

3.2.3. Detection of Faces

The detection phase in Subspace LDA starts with projecting the input image, Γ into the eigenface space first to find out the eigenface projection, Ω using Eqn. 3.16. Like the samples in the training process, the input image is then represented using *M'*x1 matrix, Ω. This vector is then projected into the FLD space using the equation

$$g(\Omega) = W^T \cdot \Omega \tag{3.28}$$

The dimension is then reduced again to (c-1 x 1). The projected version of the input image is now ready to be classified. The classification is done by simply calculating the distances from the projected input image to each of the projected sample images in the training set, i.e.

$$d_i = \| g(\Omega) - g(\Omega_i) \| \quad \text{i} = 1,2,\ldots,\text{M} \tag{3.29}$$

The image belongs to the class that includes the sample for which the distance, d_i is minimum. If this class is a face class then the input is labeled as face, on the other hand if it belongs to a non-face class it will be labeled as non-face.

CHAPTER 4

FACE DETECTION IN ACTIVE ROBOT VISION

In this chapter the algorithms that are used in the thesis are explained in detail together with preprocessing steps such as histogram equalization, skin detection.

The thesis is composed of two complementary parts, face detection and robot vision, and these parts should work collectively and correctly to have a working face tracking robot system. The aim is to have a robot which finds out whether there is a face or not in the image that it sees, and move through it to arrange its position so as to stay in a reasonable distance to the face (human). Hence the program should find out not only the existence of the face but also the distance of the face to the camera.

Without any scaling in the input image, algorithm normally detects the faces at a predetermined distance. If the input image is up-sampled, the algorithm then detects the faces at greater distance, while it detects faces at smaller distance if input image is down-sampled. The approximate distance of the face to the camera can be found out in this way and robot is moved accordingly. But this method has a big disadvantage; rescaling of every new image inputting into the algorithm makes it work slower, hence another method is applied here, which will be explained later in this chapter. Each part of the software is summarized in Fig. 4.1. and explained in order of application starting from the next section.

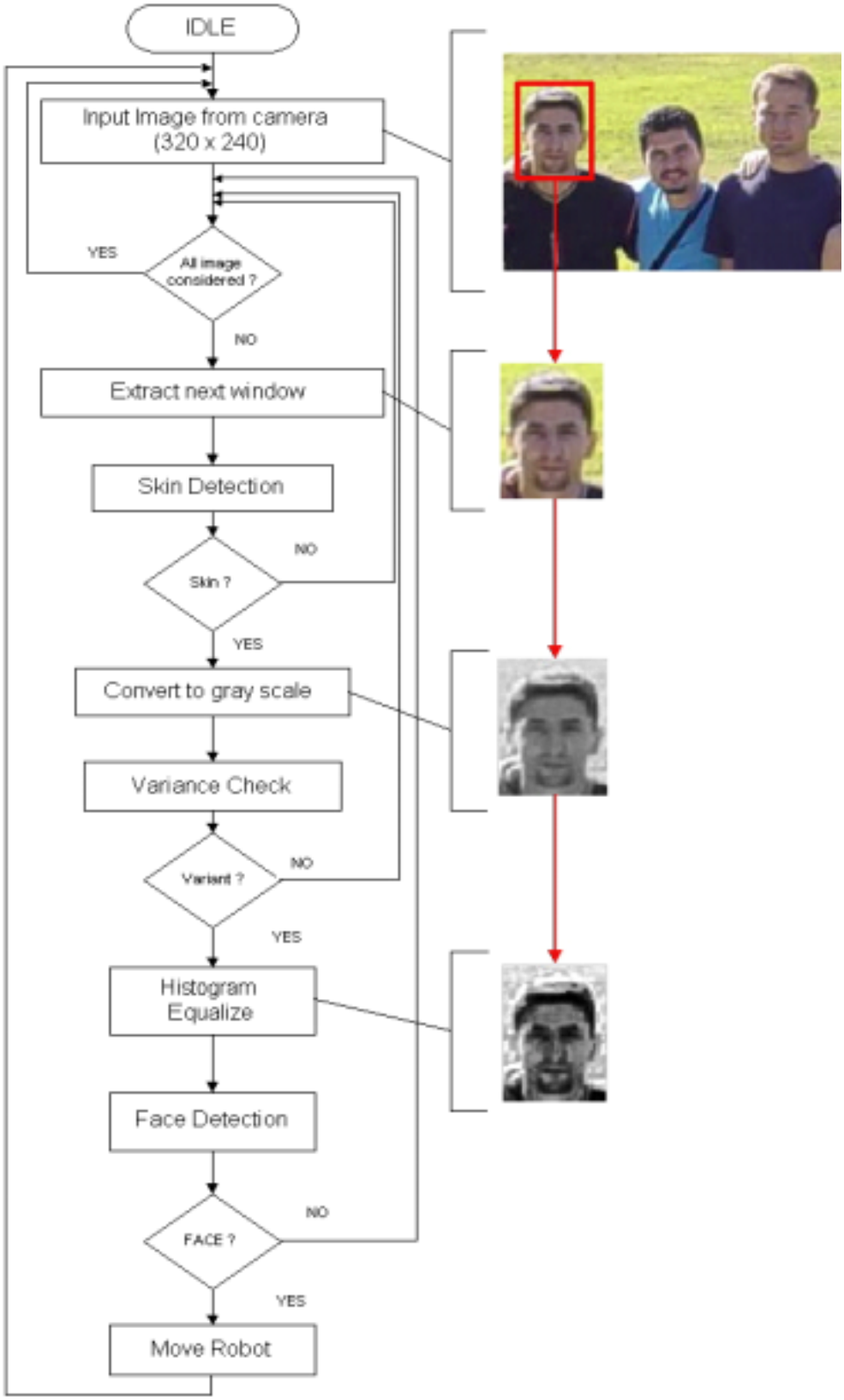


Figure 4.1. Flowchart of the complete program

4.1. Preprocessing

In the resultant application, input image is taken from a digital camera, hence it must be applied to some preprocessing before being input to the face detection algorithm.

The input image, taken from the camera, has dimensions of, say MxN (M=320 and N=240 in our case). But images must be applied to the algorithm with dimensions of the training images, (m x n). Hence, input image is cropped in training image dimensions and then put through the algorithm. This process is started from the top left of the input image and goes on until all the image is evaluated. At each time, the window of dimensions (m x n) is evaluated and is then shifted a few pixels.

The preprocessing occurs in the following order:

- Skin detection filtering
- Converting to gray level
- Variance check
- Histogram equalization

After these steps; if the image is still a face candidate then it is applied to face detection algorithm.

4.1.1. Skin Detection

Input image is taken from a color camera hence it has the color information which allows using skin detection.

For using the color information input image is first passed through a skin detection filter to understand whether or not it includes human skin. To speed up the program only some part of the input image is taken into consideration in skin detection process, and this part is chosen as the region of the face where there are cheeks. If the number of skin pixels in the selected region, is over a threshold the region is accepted as a skin, that is, a face candidate, and algorithm goes on with remaining preprocesses.

In the thesis YIQ color space is used in the skin detection process (see Sec. 2.1.1.4).

The YIQ values for the corresponding pixels in the image are calculated. After a lot of trials the following rule is decided to be applied to understand whether the pixel is a skin or not.

$$(44 < Y < 223) \text{ AND } (0 < I < 64) \quad (4.1)$$

Q value is not used in the process.

Figure 4.2 (a) shows an example of a color image and 4.2 (b) shows the same image where detected skin parts given in white.

(a) (b)

Figure 4.2. (a) An image and (b) its skin detected state

4.1.2. Color Image to Gray Level Conversion

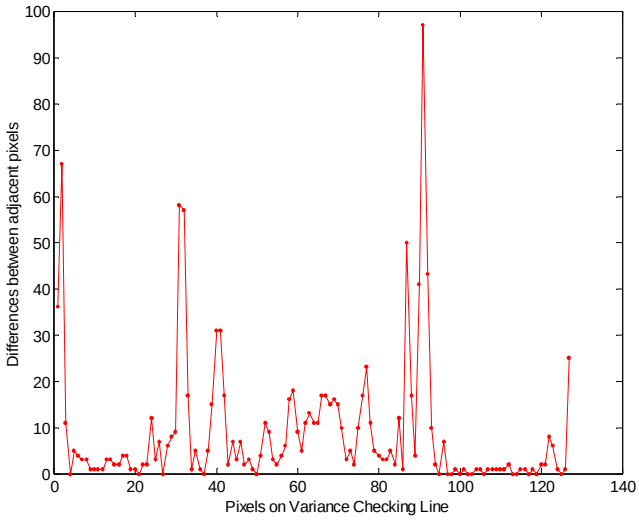
The color information is used only for skin detection and after this process it is no longer needed, hence the image is then converted to gray scale to be ready for the next processes. The method used here for this conversion is to take the arithmetic average of the R(red), G(green) and B(blue) values of each pixel, as given in (4.2).

$$Gray\ Value = \frac{R + G + B}{3} \tag{4.2}$$

4.1.3. Variance Check

During the development phase of the thesis we observed that, our algorithm sometimes gives false positive detections in the plain parts of the image, hence variance check process is used to deal with this "plain figure false positive" problem. The position of the eyes on a human face is approximately known and it is obvious that following a horizontal line on this part of the face, the pixel values should not be constant but change. The best horizontal line on the image for this process is found out after some tests, we name this line "Variance Check Line" (VCL). The differences between the adjacent pixel values on this horizontal line are calculated. If the number of 'edges' in this line is greater than a threshold value, it is decided that this line is a part of a plain figure (absolutely not a face) and vice versa (may or may not be face).

In Fig. 4.3 and Fig. 4.4 face and non-face pictures are seen, respectively. The VCLs are drawn onto the images. On the right of these images seen the graphs of the differences between the corresponding pixels on the variance checking line. These graphs and experimental results show that the threshold value for labeling two adjacent pixels as edge can be chosen as 20, and the threshold value for the number of edges on the VCL can be chosen as 4.



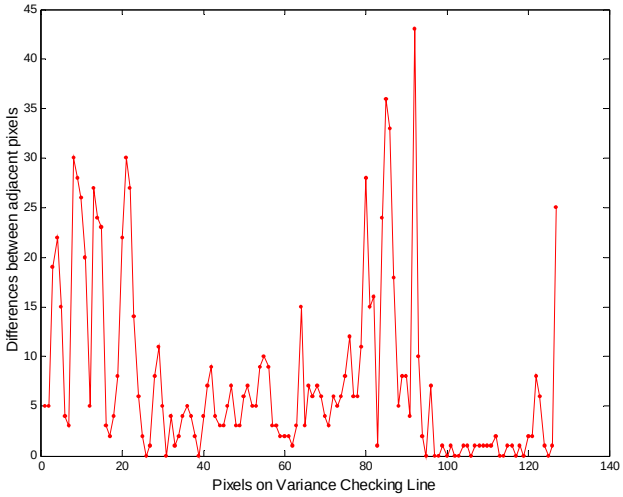


Figure 4.3. Input face images with VCLs in black and graphs of differences between adjacent pixels

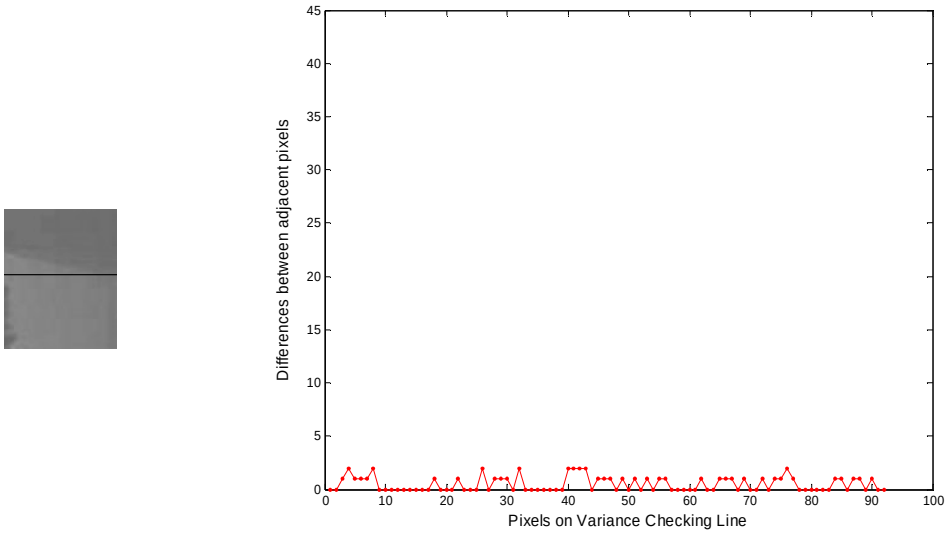


Figure 4.4. Input plain image with VCL in black and graph of differences between adjacent pixels

According to our training face images if there is a face just in the middle of the image and almost covering the input image, the position of the eye from the top is about 47.17%. i.e.

$$Position\ of\ \ Eye\ Center\ \ from\ top = 0.4717 * Height \qquad (4.3)$$

This means somewhere on face quite near to the middle, but just a bit higher part of the face. We extract this line and make our test for variance checking, if it passes this test, the face detection algorithm is applied, if it appears as plain figure, program eliminates this current window.

4.1.4. Histogram Equalization

The histogram of an image records the frequency distribution of grey levels in that image. We can use the histogram of an image to define a non-linear mapping of

grey levels, specific to that image, that will yield an optimal improvement in contrast. This technique, known as histogram equalization, redistributes grey levels in an attempt to flatten the frequency distribution. Less grey levels are allocated where there are most pixels, more grey levels where there are fewer pixels. This tends to increase contrast in the most heavily populated regions of the histogram, and often reveals previously hidden detail, which results in a better seen picture.

In face detection it is very important to have a border between face and the background. Obviously, the sharper this border is, the easier to detect the face, and histogram equalization makes a great job on this. More details and algorithms on histogram equalization can be found in [28].

Fig. 4.5 shows an image and the resultant figure after histogram equalization.

Figure 4.5. An image and its histogram equalized version.

Histogram equalization is applied to both the training images and to the test images in the thesis. Since the project would possibly work indoor most of the time the image is not very qualified and hence the algorithm gives better results after histogram equalization. These results are discussed in Sec. 4.5.

4.2. Face Detection

The eigenface method and the Subspace LDA method are compared in the thesis. The parameters for both methods are found out after the tests, whose results are given in Sec. 4.5. The windows that pass from the previous tests, preprocessing algorithms are applied to this face detection algorithm.

The eigenface method that is proposed by [1], and Subspace LDA method proposed by [34] are implemented in our thesis. The Subspace LDA method is implemented without any significant difference from the original implementation.

Only difference from the original implementation in Eigenface method, is in the calculation of the reconstruction error. Looking at the training images we see that the regions near the edges do not include a face part. Hence when calculating the reconstruction error these edge regions are not taken into account. We also understood during the experiments that the eyes are more critical than other parts of the face, and must have a greater weight than other parts. So a weighting mechanism was obviously needed, and we used the following coefficients for the corresponding parts of the image seen in Fig. 4.6.

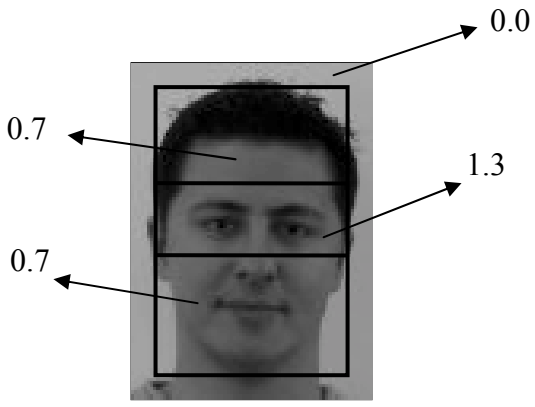


Figure 4.6. Weighing coefficients used in reconstruction error calculation

These coefficients were found out after trial and errors.

To detect faces in different distances a scaling mechanism is used, which is explained in Sec. 4.3.

4.3. Scaling Process

Normally in a system, if you need to make a robot to follow an object, the robot must detect that object at different distances. The best way for this process seems to be scaling of the input picture, but it obviously takes time to re-scale and re-process the input picture. Hence we developed another method which does the same thing in a shorter time, which we call; training image scaling. According to this, we prepared the scaled versions of the training images that are used in face detection. This is done before the program runs and enough to be done only ones, hence it will not cost time while the program is running.

Table 4.1. Training image sizes and corresponding face detection distances

	Training Image Size (pixels)	Used to detect faces at distance: (cms)
I	76 x 106	80 – 115
II	59 x 79	115 – 155
III	38 x 53	155 - 225

Three different scaled versions of every training image, are saved before. The sizes of these three groups of images are seen for METU Vision Face database in Table 4.1 with captions I, II and III. The ideal distance from the robot to the face is decided to be about 1,5 meters. The robot first uses the eigenfaces of the (38x53)

sized training images and aims to find the faces at long distance, after that it uses (59x79) sized images and then (76x106) sized ones. If it finds a face using (I) then it turns to get the detected face to the center of the screen, and moves back to reach to the ideal distance from the face. If it finds a face using (II) this means that it is at about ideal distance from the face, hence only turns to make the face in the center of the screen. If it finds a face using (III) then it turns again in the same way, and moves forward, through the face to reach to the ideal distance from the face.

4.4. Robot Navigation

The face tracking robot system which is developed in the thesis, is actually composed of a robot and a camera mounted onto it. The algorithm part of the thesis has in fact a theoritical basis, and what helps us to put our work into real life, is the robot. After detecting the position of the face in the scene, robot needs to move to the best place to track the face. If the face is far then robot approaches it, if it is near then robot moves back from it. If the face is in suitable distance then robot only rotates not translates to a new position.

Rotation is performed so as to make the detected face bring into the center of the camera view. Fig. 4.7. gives the general robot vision system schematic.

In Fig. 4.7. assume that the robot detects a face at distance x from the center of the camera view. In this case it should rotate β degrees clockwise, hence β must be calculated. The angle β is equal to

$$\beta = \tan^{-1}(x/y) \tag{4.4}$$

The distance of the face to the camera view center is returned from the face detection algorithm, but in pixels. Let us call the pixel correspondings of x and w as, x' and w', respectively. It is trivial that real distances and corresponding pixel values are proportional to each other, i.e.

$$\frac{x}{x'} = \frac{w}{w'} \tag{4.5}$$

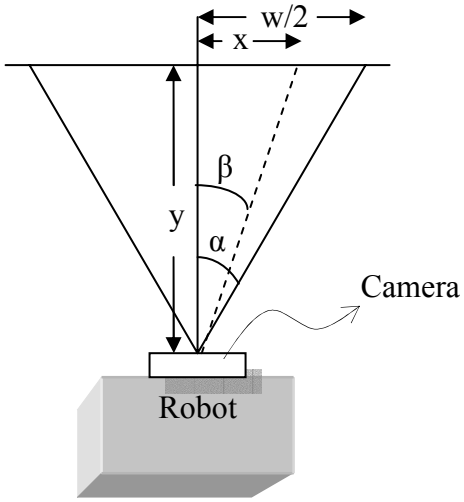


Figure 4.7. Robot navigation system schematic

Looking at Fig. 4.7. one can see that

$$\tan(\alpha) = \frac{w}{2y} \tag{4.6}$$

The angle, α is the view angle of the camera and is calculated using the distances *w/2* and *y*. This view angle is measured as about 22 degrees. Noting that *x*' is returned from the face detection algorithm, *w*' is the width of the resolution of the camera, and the angle α is known, the angle β is calculated combining the equations above:

$$\beta = \tan^{-1}\left(\frac{x'w/w'}{w/(2\tan(\alpha))}\right) \tag{4.7}$$

which can be further simplified as follows:

$$\beta = \tan^{-1}\left(\frac{2x'}{w'}\tan(\alpha)\right) \tag{4.8}$$

After rotating the robot β degrees it is translated simply back or forth for some predetermined distance. More information about the translation process is given in Sec. 4.3.

4.5. Experimental Results

In this section the parameters for the best performance of the system will be determined for both face detection methods, Eigenface and Subspace LDA. Then the performances of these methods will be compared to see which method we should use in the final system. Then the evaluation test results will be given for chosen method and parameters.

A critical point in face detection systems, is that the detection rate has no important meaning on its own. It should be accompanied with the number or rate of the false detections and (if important for the system) the computation time. The parameters should be chosen according to the system needs, hence different systems who has got the same experimental results will probably choose different parameters. For our system all the three parameters counted above has about equal importance. Because the system is expected to detect the faces with minimum number of false detections and in a short time period.

The algorithms have been tested on a lot of images and the logs are kept for further development of the project. During the development of the thesis the following training sets are used:

1. ORL training set: This is the face set of Olivetti Research Laboratory and includes the pictures of 40 people, 10 pictures per individual with different orientations. The original face images have dimensions 92x112. (See App. B.2)
2. METU Vision face database: This set is introduced in this thesis first and created by METU Computer Vision and Intelligent Systems Research Laboratory.

It includes pictures of 15 people, 10 pictures per individual with different poses, orientations and lightining conditions. The image dimensions in this database are 76x106. (See App. B.1)

The algorithm is tested using the following test sets:

1. The Combined MIT/CMU test set: This set is introduced by the Carnegie Mellon University and is a combined set of MIT and CMU test sets. It includes 65 image files in its frontal database part and there are a total of 186 labeled faces in these images. All the images are in greyscale.
2. The CMU II test set of profile faces: This set is also introduced by the Carnegie Mellon University, and is composed of 208 images with a total of 368 labeled faces. All images are in greyscale and almost all of the faces in the images are non-frontal. Hence this test set is used in the rotation tests.
3. The METU Vision test set: This set is introduced by METU Computer Vision and Intelligent Systems Research Laboratory. It includes 17 image files, all of which are color images and contain a total of 72 labeled faces.

Testing the algorithms on the MIT/CMU test set actually have only limited meaning for our case. Since the input image is actually taken from the camera on the robot and the quality and lightening conditions are actually quite far different in real life than that of those in that test set. But this test set is useful in understanding the performance of the original face detection algorithms and obtaining the algorithm parameters according to the performance. Hence the first test set is used to see the performance of the pure face detection algorithms, while the third one is used to see the performance of the resultant program which will be running on the robot. The parameters are all arranged for obtaining the best performance on our real-time application which captures image from the camera and applies face detection on this captured image.

4.5.1. Eigenface Method Test Results

The detailed theoritical information on Eigenface face detection method is given in Sec. 3.1. According to the information given in that section, there are two important parameters in Eigenface method, which are the *number of training images*, *M* and the *number of eigenfaces, M'*. Hence the experiments are performed to obtain the best values of these parameters. Tests are done using two different training image sets, ORL and METU Vision. The combined MIT/CMU test set is used in the experiments and programs are run on MATLAB. Sec. 4.5.1.1 gives the test results for the number of training images, and Sec. 4.5.1.2 gives test results for the number of eigenfaces.

4.5.1.1. Obtaining Number of Training Images

The number of training images used in the training process is an important parameter in Eigenface method, like almost all face detection methods. The experiments are performed to obtain the best value for the number of training images, for our real-time face detection system.

During the tests all the parameters are kept constant except the number of training images. The number of eigenfaces are taken as equal to the number of training images for every different value. Results are shown in Figures 4.8, 4.9, 4.10 and 4.11 and exact numbers are given in Table 4.2.

According to Fig. 4.8 the ideal number for training image seems to be between 10 and 20, and Fig. 4.9 shows that the number of false detections increases exponentionally after 16 training images. Higher number of training images also increases the computation time, which is also very important for our system. Hence, after investigating Figures 4.8, 4.9 and 4.10 we choose the number of training images as M=16.

Looking at the figures one may think that the number of false detections is to be decreased with increasing training image number. However this is not the case. As mentioned before, a face detection algorithm should never be evaluated checking only one of those figures. The correct detection and false detection rates must be examined together to obtain a reliable result. Hence changing a parameter in a face detection system usually results in a similar effect on correct detection and false detection rates. But one of them usually changes more, and this gives us information about the effect of that changing parameter on the system. And in such a case it is important to find out the point where face detection rate is satisfactory and false detection rate is tolerable. The ROC curve can be used for this purpose which is the graph of face detection rate versus the number of false detections. The saturation point of a ROC curve means that after this point the face detection rate increases very little while the number of false detections increases rapidly. The ROC curve for changing training image number is given in Fig. 4.11 and according to this curve it is confirmed that the value of 16 (check Table 4.2 for corresponding detection rate and number of false positives) is a good choice for the number of training images.

We see that the face detection performance is lower when the METU Vision face database is used as the training set. This is because the ORL dataset is a more suitable face database especially when applying the face detection to the MIT/CMU test database, from the aspects of lightening conditions, background characteristics etc. The METU Vision face database is expected to give better results when testing the METU test set, which will be examined later in this chapter. The image dimension in the METU Vision face database is lower than the image dimensions in ORL database, this results in less computation time when using the METU Vision database, since a smaller-dimension-image means a smaller vector, which the computer handles using less time.

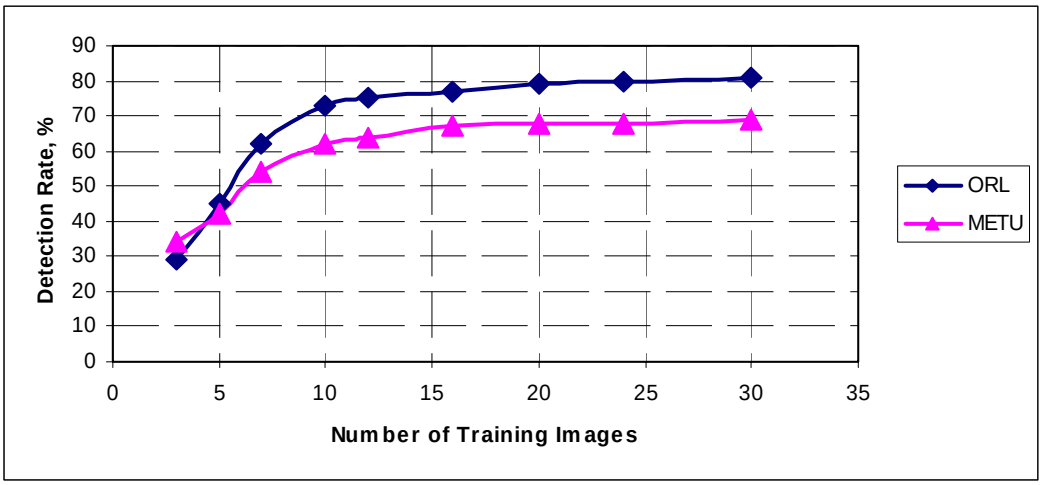


Figure 4.8. Detection rate versus number of training images

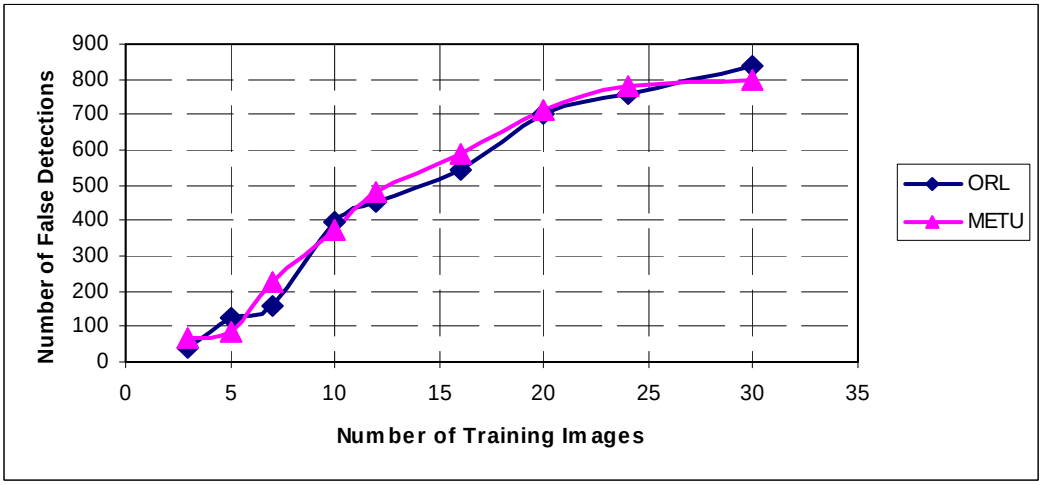


Figure 4.9. Number of false detections versus number of training images

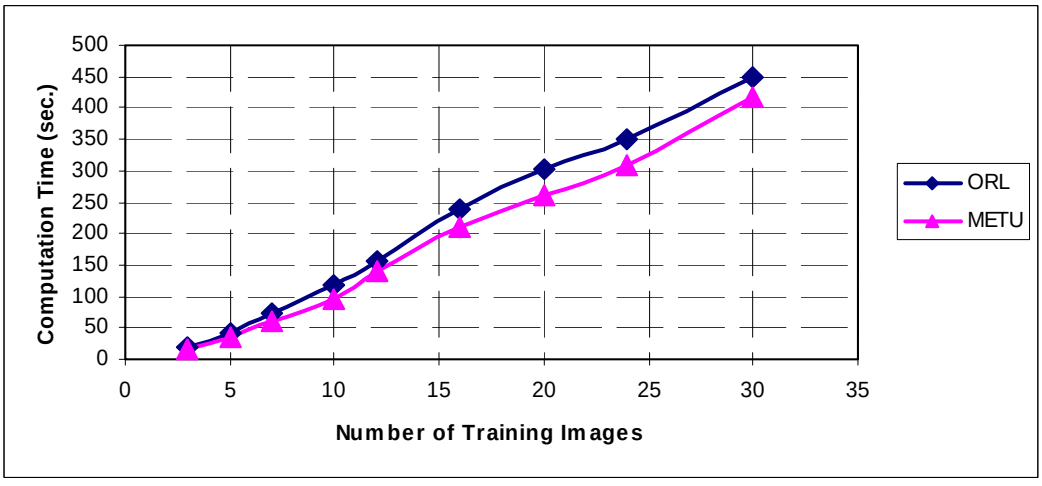


Figure 4.10. Computation time versus number of training images

Table 4.2. Face detection performance of Eigenface method for changing number of training images on ORL and METU Vision databases

Training Database	Number of Training Images	Detection Rate %	Number of False Detections	Computation Time (sec.)
ORL	3	29	42	18
ORL	5	45	123	41
ORL	7	62	161	72
ORL	10	73	396	117
ORL	12	75	453	157
ORL	16	77	544	238
ORL	20	79	701	301
ORL	24	80	761	351
ORL	30	81	838	450
METU Vision	3	34	67	15
METU Vision	5	42	87	35
METU Vision	7	54	225	61
METU Vision	10	62	371	96
METU Vision	12	64	481	141
METU Vision	16	67	589	211
METU Vision	20	68	714	260
METU Vision	24	68	782	309
METU Vision	30	69	799	417

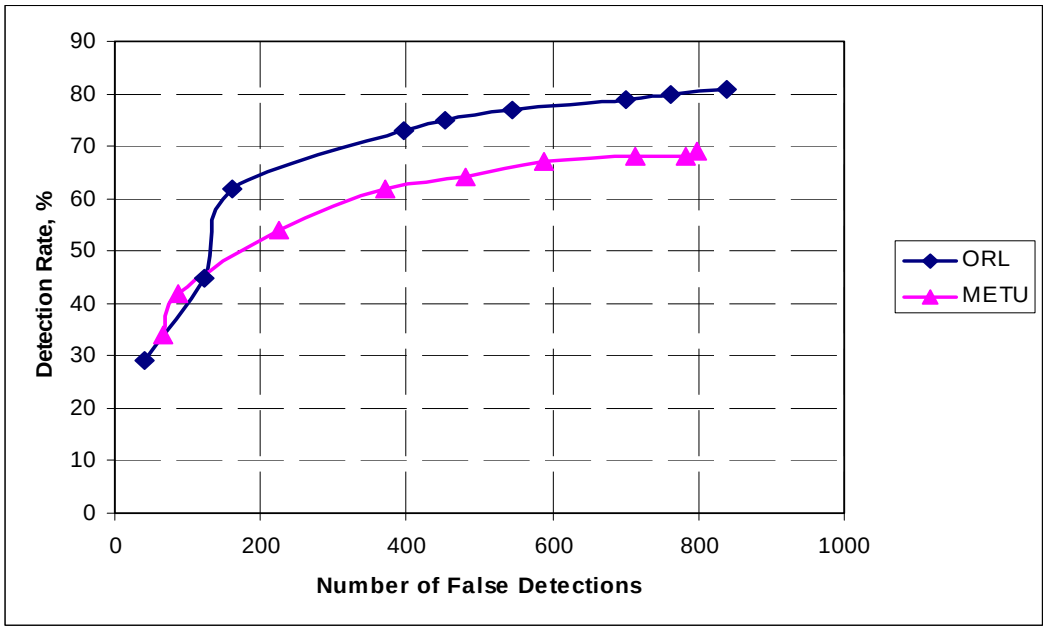


Figure 4.11. The ROC curve, detection rate versus number of false detections

4.5.1.2. Obtaining Number of Eigenfaces

The best performance in face detection using eigenfaces is obviously achieved using the maximum number of eigenfaces, which is equal to one less than the training image number, *M*-1. However, the small eigenvalues have little effect on the system hence they are usually omitted to save time. The eigenvalues of a training set of sixteen images are shown in Fig. 3.4. This test is performed to find out the best number of eigenfaces used to construct the face space. Results are given graphically in Figures 4.12, 4.13, 4.14 and with exact numbers on Table 4.3.

Inspecting those figures and also the ROC curve given in Fig. 4.15, we see that after 7 eigenfaces the number of false detections increases rapidly. The values of the detection rate and number of false detections for both the ORL database tests and the METU Vision face database tests, corresponding to 7 eigenfaces can be seen in the ROC curve. It is understood that this point corresponds to about the saturation point of the ROC curve. Hence the number of eigenvalues used in the system is chosen to be M'=7.

Looking at Fig. 4.12 we see that face detection performance is still worse when using the METU Vision face database as the training set. This is what we expect, because the method used is still the same and only the number of eigenfaces is the changing.

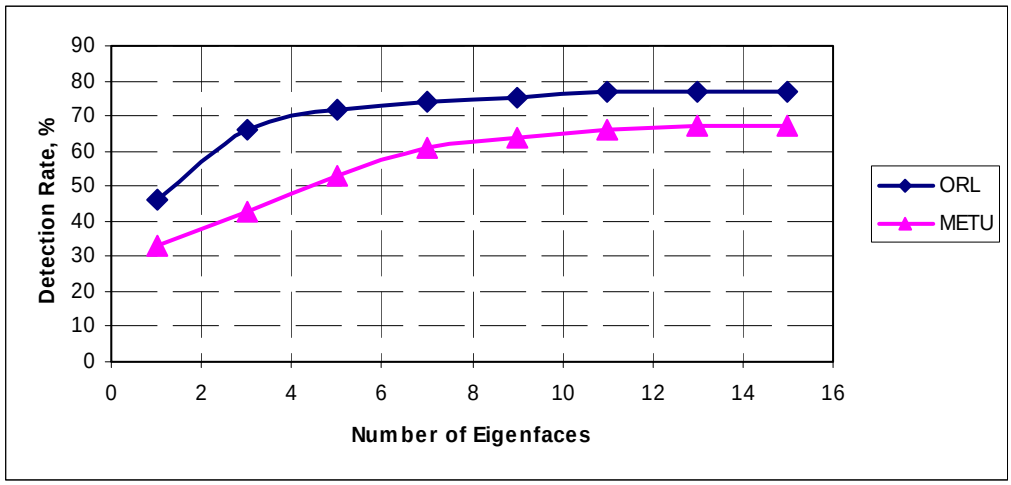


Figure 4.12. Detection rate versus number of eigenfaces

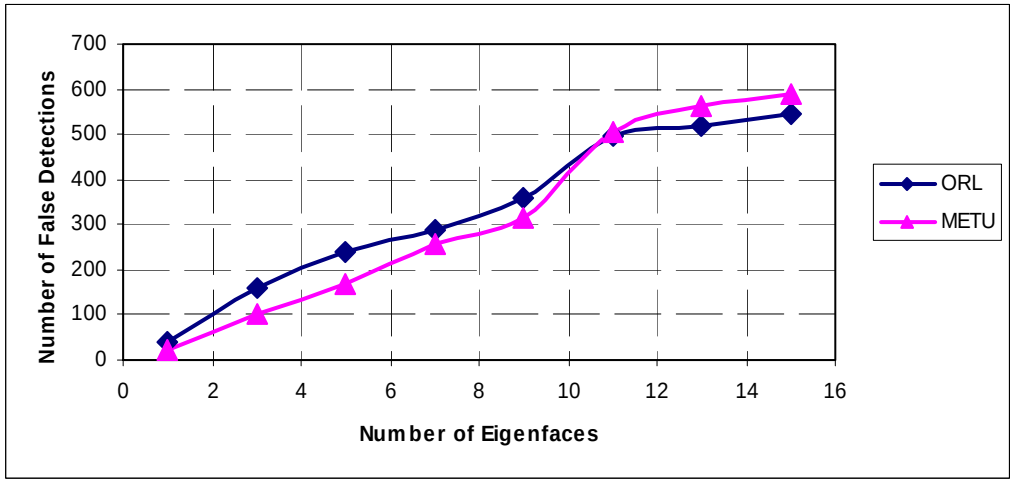


Figure 4.13. Number of false detections versus number of eigenfaces

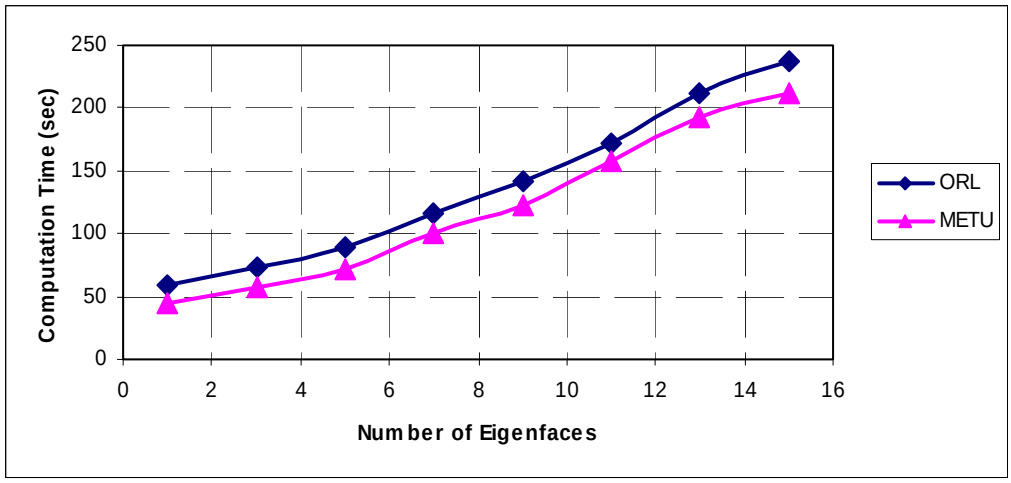


Figure 4.14. Computation time versus number of eigenfaces

Table 4.3. Face detection performance of Eigenface method for changing number of eigenfaces on ORL and METU Vision databases

Training Database	Number of Eigenfaces	Detection Rate %	Number of False Detections	Computation Time (sec.)
ORL	1	46	40	59
ORL	3	66	158	73
ORL	5	72	238	89
ORL	7	74	287	116
ORL	9	75	360	142
ORL	11	77	495	172
ORL	13	77	518	211
ORL	15	77	544	238
METU Vision	1	33	22	45
METU Vision	3	43	101	57
METU Vision	5	53	168	72
METU Vision	7	61	255	101
METU Vision	9	64	314	122
METU Vision	11	66	504	158
METU Vision	13	67	562	192
METU Vision	15	67	589	211

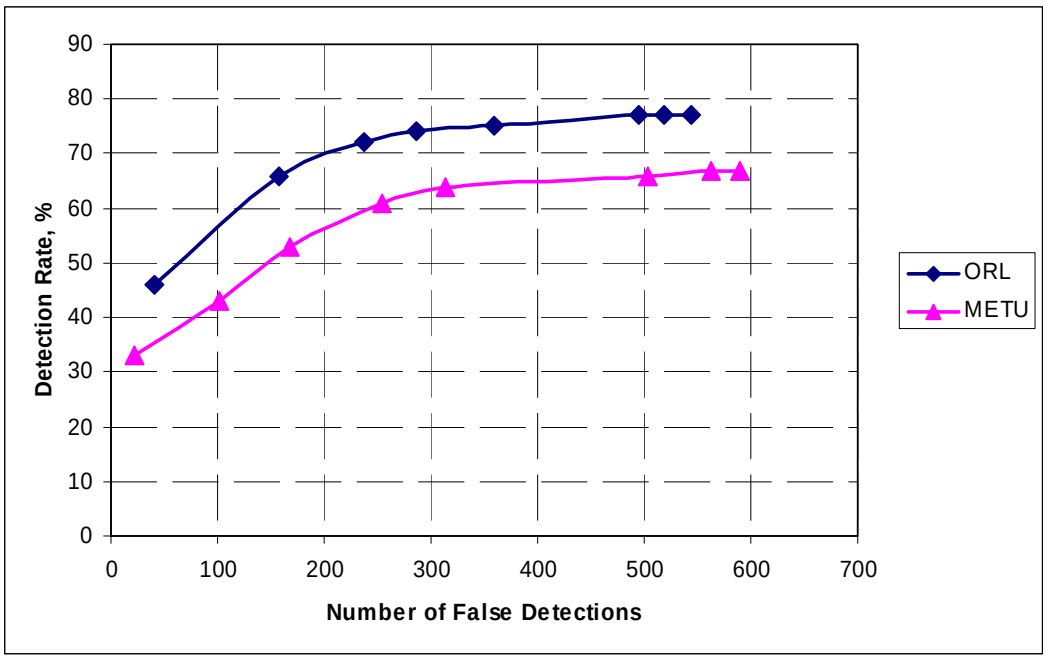


Figure 4.15. The ROC curve, detection rate versus number of false detections

4.5.2. Subspace LDA Method Test Results

Sec. 3.2 gives the theoritical information on the subspace LDA method. There are again two important parameters which effect the face detection performance in this method, which are the number of classes and the number of features. A similar way will be tracked as the one for the Eigenface method. The best value for the number of classes will be found out first and then the number of features will be searched for the best number of classes.

Tests are done using two different training image sets, ORL and METU Vision face databases. The MIT/CMU test set is used in the experiments and codes are run on MATLAB. Sec. 4.5.2.1 gives the test results for the number of classes, and Sec. 4.5.2.2 gives test results for the number of features.

4.5.2.1. Obtaining Number of Classes

When using the subspace LDA method the number of classes is the most important parameter especially when using this method for face detection. Because unlike the eigenface method, in subspace LDA both face and non-face image classes are used in the training phase.

The class probabilities for face and non-face classes are assumed to be same in our system, and the number of face and non-face classes are kept equal to each other. That is if there are 20 classes in the system for example this means there are 10 face and 10 non-face classes.

During the tests for the number of classes the number of features are kept equal to the maximum possible number which is the number of classes minus one.

Table 4.4 displays the test results and Figures 4.16, 4.17, 4.18 and 4.19 show the results graphically. This time results seem different than the one in Eigenface method. Figures 4.17 shows that the number of false detections increases very slowly after some point but the detection rate also increases slowly. Figures 4.16 and 4.17 together show that when the number of classes is chosen as 40 then the detection rate is high enough and the number of false detections are reasonable. However we must investigate Fig. 4.18 before ending up with a decision. Fig. 4.18 is the plot of computation time versus number of classes and it is seen that the computation time is much higher in this method than the Eigenface method. Hence we need to find a point where the computation time is tolerable. This can be the point for number of classes is equal to 10 or 20. The difference between the detection rates for ORL database is %12, which is not a small value, hence the ideal value for number of classes in subspace LDA method is chosen as 20.

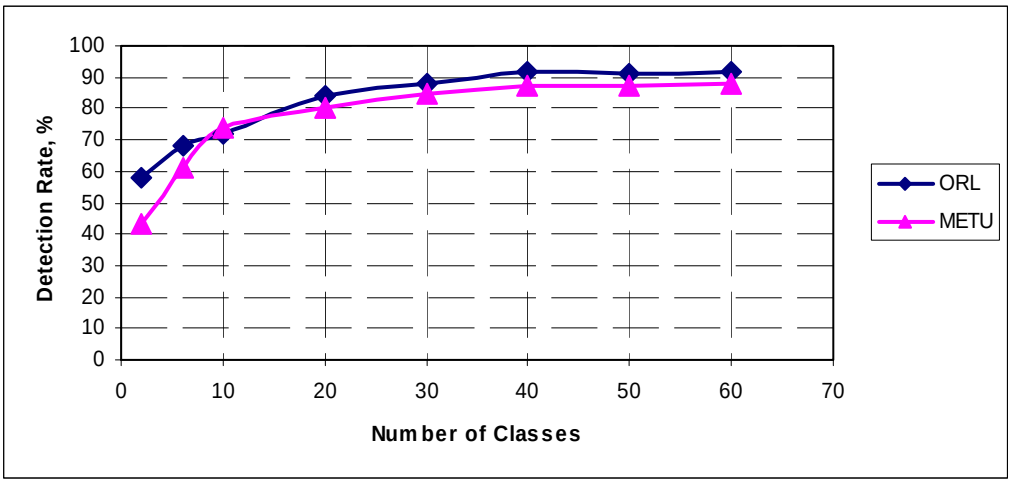


Figure 4.16. Detection rate versus number of classes

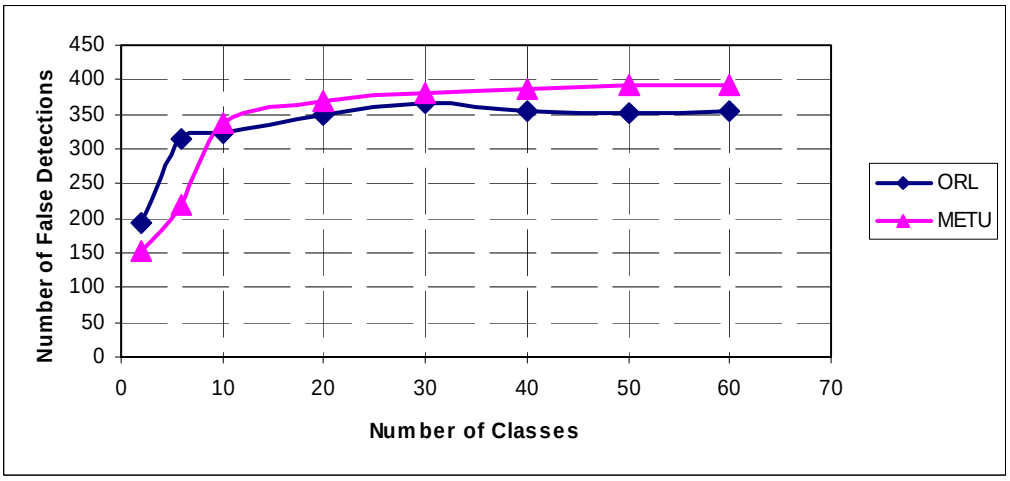


Figure 4.17. Number of false detections versus number of classes

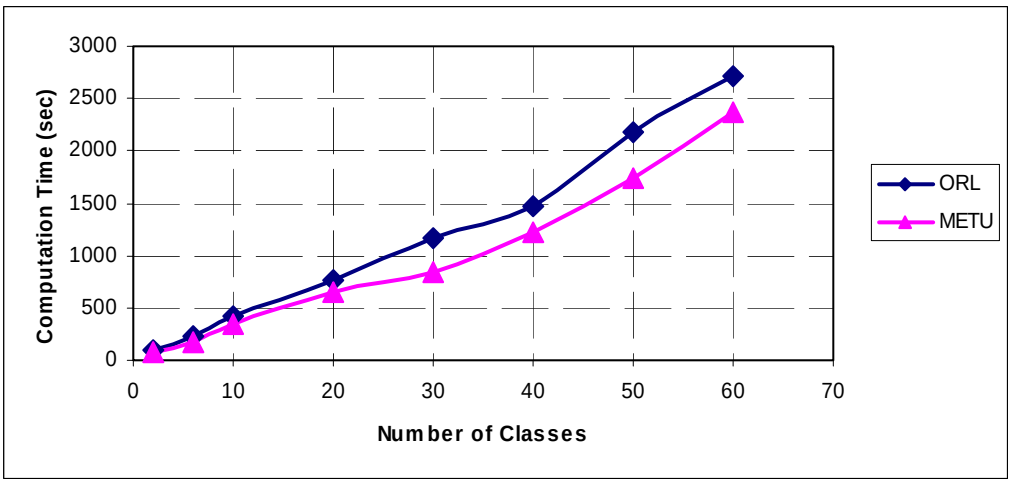


Figure 4.18. Computation time versus number of classes

Table 4.4. Face detection performance of subspace LDA method for changing number of classes on ORL and METU Vision databases

Training Database	Number of Classes	Detection Rate %	Number of False Detections	Computation Time (sec.)
ORL	2	58	193	101
ORL	6	68	314	237
ORL	10	72	322	411
ORL	20	84	349	770
ORL	30	88	366	1163
ORL	40	92	355	1479
ORL	50	91	353	2187
ORL	60	92	354	2711
METU Vision	2	43	153	81
METU Vision	6	61	220	181
METU Vision	10	74	337	351
METU Vision	20	80	369	650
METU Vision	30	85	382	843
METU Vision	40	87	387	1224
METU Vision	50	87	393	1743
METU Vision	60	88	392	2367

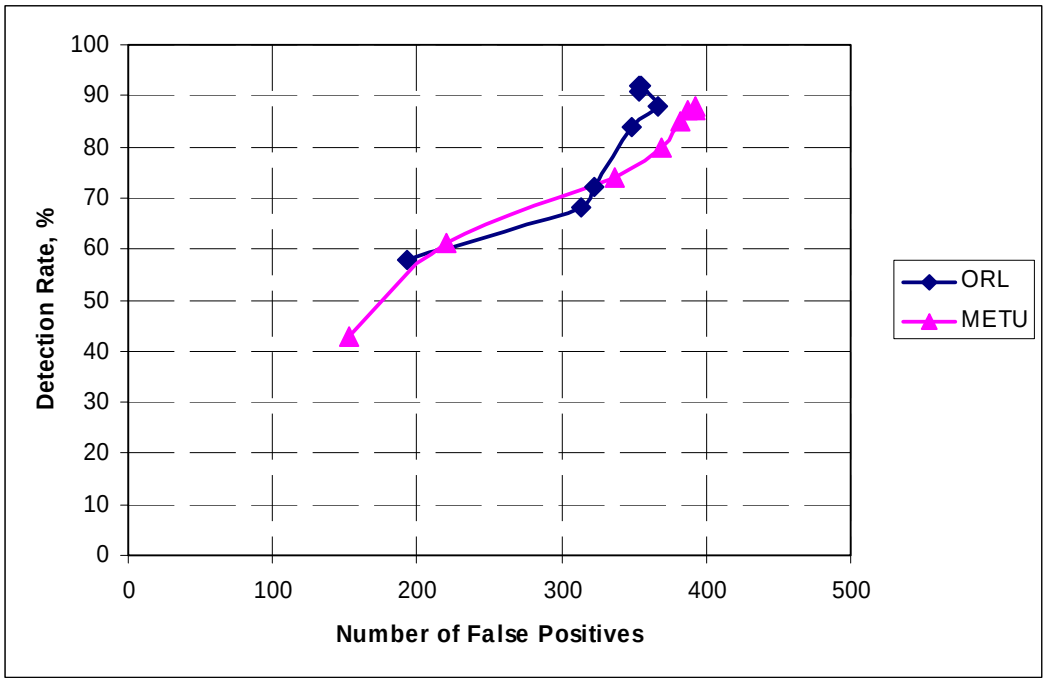


Figure 4.19. The ROC curve, detection rate versus number of false detections

4.5.2.2. Obtaining Number of Features

To obtain a system with the best performance the number of features must be equal to the number of classes minus one. However the face detection systems usually use a smaller number of features to gain time, since using less number of features a processor can make the comparisons faster.

In this test the number of classes is kept constant to the best value, 20, obtained in the previous section and experiments are performed with different number of features.

The experimental results for this section are given in Table 4.5 numerically, and in Figures 4.20, 4.21, 4.22 and 4.23 graphically. Using less number of features means having less number of parameters to compare, which results in having lots of false detections.

Fig. 4.22 shows that, increase in the number of features actually does not increase the computation time too much. Hence we do not need to choose a very small value for number of features. Since time constraints are not tight this time, the ROC curve can be very useful. Looking at Fig. 4.23 we immediately see that the ideal value for the number of features is the one corresponding to a detection rate of just below 90%, and number of false detections just below 500. Checking Table 4.5, it can be seen that these values correspond to the number of features value of 9, hence we choose the number of features as 9.

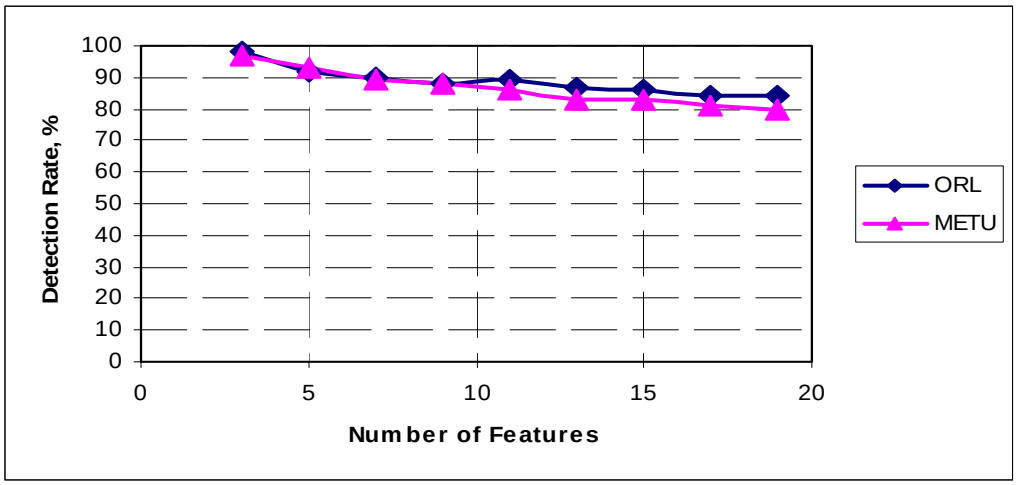


Figure 4.20. Detection rate versus number of features

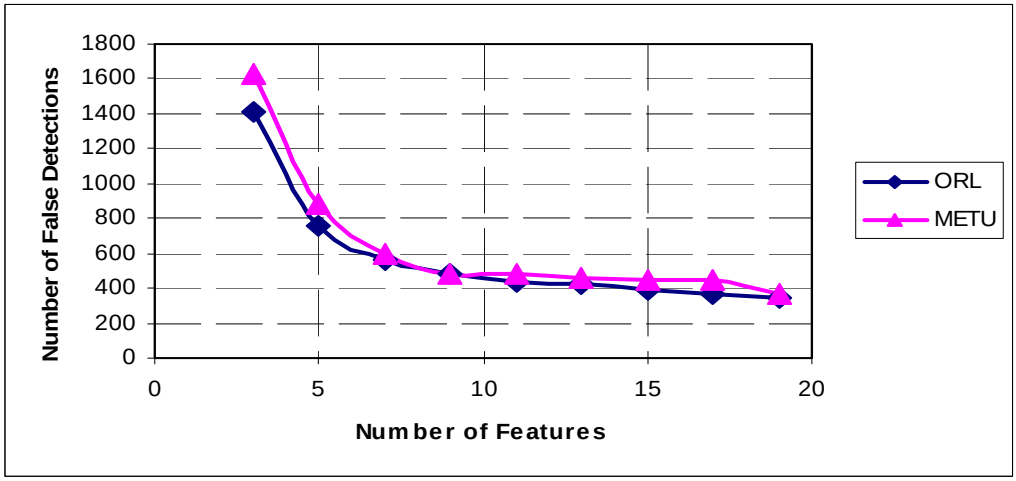


Figure 4.21. Number of false detections versus number of features

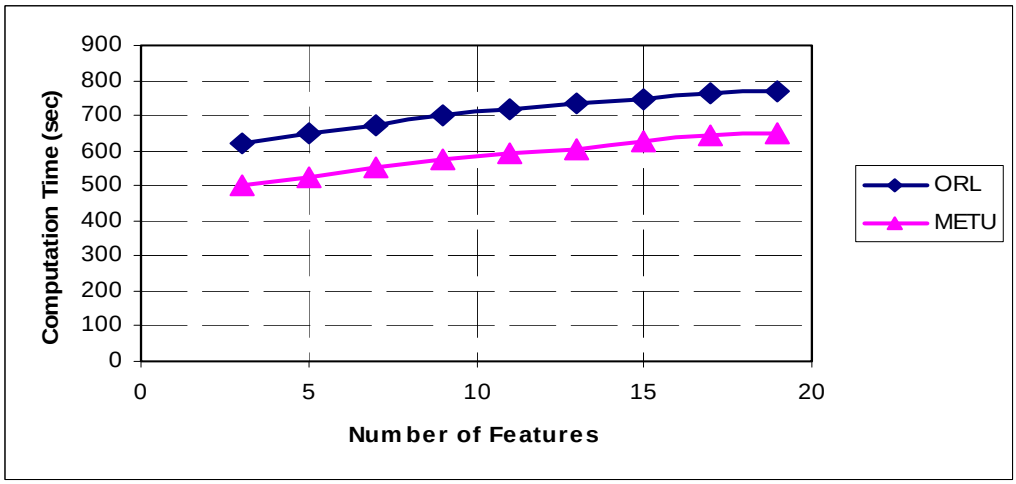


Figure 4.22. Computation time versus number of features

Table 4.5. Face detection performance of subspace LDA method for changing number of features on ORL and METU Vision databases

Training Database	Number of Features	Detection Rate %	Number of False Detections	Computation Time (sec.)
ORL	3	98	1412	621
ORL	5	92	751	647
ORL	7	90	557	671
ORL	9	88	481	698
ORL	11	89	440	717
ORL	13	87	421	732
ORL	15	86	388	749
ORL	17	84	362	761
ORL	19	84	349	770
METU Vision	3	97	1624	501
METU Vision	5	93	883	525
METU Vision	7	89	601	552
METU Vision	9	88	483	577
METU Vision	11	86	476	591
METU Vision	13	83	455	604
METU Vision	15	83	443	628
METU Vision	17	81	442	642
METU Vision	19	80	369	650

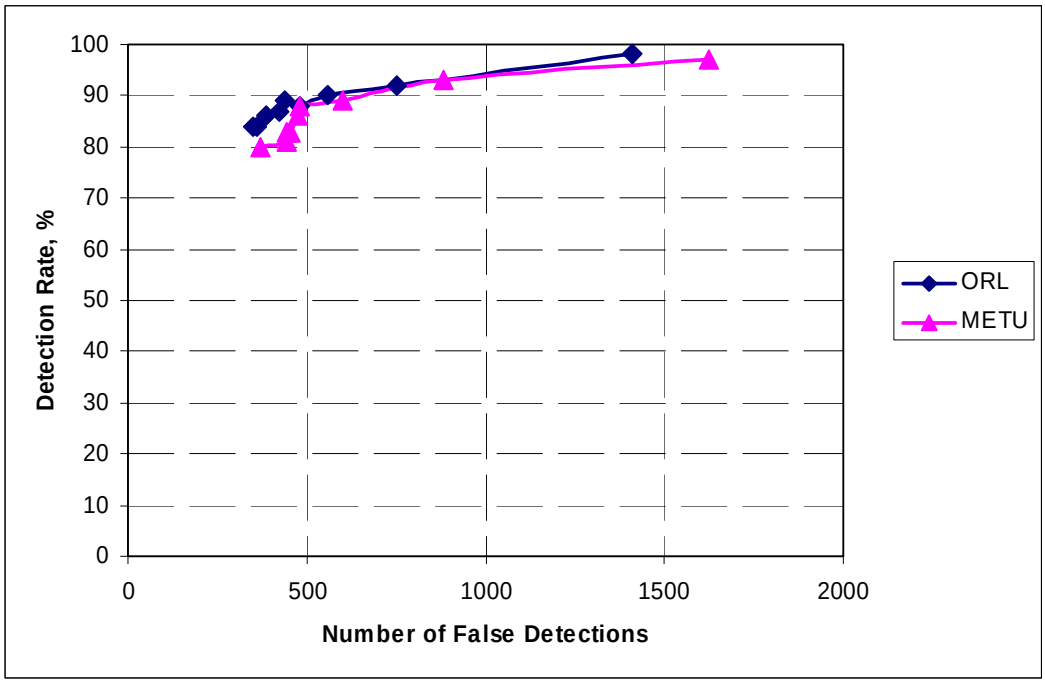


Figure 4.23. The ROC curve, detection rate versus number of false detections

4.5.3. Evaluation Tests

After performing the experiments, the best values are found out for both methods. Table 4.6 shows the face detection performances of the two methods, Eigenface and subspace LDA for the best values of the parameters:

Table 4.6. Performance comparison of face detection methods

Face Detection Method	Training Database	Detection Rate %	Number of False Detections	Computation Time (sec.)
Eigenface	ORL	74	287	116
Eigenface	METU Vision	61	255	101
Subspace LDA	ORL	88	481	698
Subspace LDA	METU Vision	88	483	577

According to Table 4.6, the Eigenface method has a lower detection rate than that of the subspace LDA method. But in the same time the subspace LDA method gives too many false detections, which may or may not be tolerable for a system. For our system the time constraints are very important and the computation time for the subspace LDA method is much over the tolerable limit. As a result considering the last three columns together, we see that Eigenface method is more suitable for our real-time face tracking system.

After performing the tests and determining to use the Eigenface method in our face tracking system, the experimental setup is prepared for evaluation tests. In the remaining tests the Eigenface method is used as face detection method, and all tests are performed on Java programming language. Both training databases, ORL and METU Vision are used for training purposes, a color image test set, METU test set is used to test the overall algorithm. Table 4.7 gives the results of the test which is performed to see the effect of the preprocessing operations on the system performance.

Table 4.7. Overall algorithm results using ORL / METU Vision face databases

Preprocessing	Detection Rate %	Number of False Detections	Computation Time (sec.)
No Preprocess	87.5 / 95.8	164 / 118	47 / 41
Skin det.	80.5 / 91.7	96 / 72	12 / 9
Skin det.+ Var.chk.	76.4 / 86.1	67 / 43	7 / 6
Skin det.+ Var.chk.+ Hist. eq.	79.2 / 87.5	36 / 32	9 / 8

As expected, the algorithm gives better results when the METU Vision face database is used as the training set. Table 4.7 shows that the preprocess algorithms not only decreases the number of false detections but also decreases the computation time dramaticaly. However as a result of decrease in number of false detections the detection rate of the system decreases as well.

The METU Vision test set includes both frontal and profile faces, hence this test is also a rotation test. But still one more test is applied with the resultant parameters and preprocessing algorithms using the CMU II test set of profile images. 10 images with a total of 22 faces, from this database are used in the experiment. Since the test set is in greyscale, skin detection part is omitted. The results are seen on Table 4.8.

Table 4.8. Rotation test results on final system

Training Set	Detection Rate %	Number of False Detections	Computation Time (sec.)
ORL	77.3	64	28
METU Vision	68.2	47	27

As expected, the face detection performance of the system is worse when the test set includes rotated faces, which is an important problem to be solved in face

detection systems. The training databases must contain different poses and orientations of the individuals, this makes the system more rotation invariant. Our training sets obey this rule, hence the results are still acceptable for a database of profile images.

Finally a brightness test is applied to our final system with chosen parameters and preprocessing algorithms. The METU Vision face database is used as the training set and experiment is performed on METU test set, on Java environment. Results are collected on Table 4.9.

Table 4.9. Brightness test results on final system

Brightness Change %	Detection Rate %	Number of False Detections	Computation Time (sec.)
+30	20.8	11	6
+20	48.6	13	7
+10	72.2	28	11
0	87.5	32	8
-10	83.3	35	9
-20	76.4	63	6
-30	34.7	135	4

Inspecting Table 4.9 shows that our final face detection system is highly dependent to the lightening conditions. More than ±10 percentage of change in illumination effects the system quite bad. To obtain a better performance the preprocessing algorithms can be improved against the brightness changes in the input image. Another cure may be using a training set in which the images are taken under different lightening conditions.

4.5.4. Algorithm Outputs, Comments and Discussions

Up to this part the results are given in the graphs and tables, however in this section some of the outputs of the algorithms are given. Some outputs will be given together with comments and discussions for better performance. Many of the input images given here are taken from the test databases, MIT/CMU, CMU II, and METU Vision, while some of them are not part of a test set but given here to give information on an important point. However one should note that the output of a face detection algorithm on one input image does not give a scientific information about the performance of that algorithm. Hence the results given here are not aimed to prove the performances of the algorithms but only aimed to give some information using visual outputs. Since the Eigenface method is used in the final program most of the results given here are Eigenface algorithm outputs. Some subspace LDA method outputs are also given.

As it is explained in part 3.1.3, in Eigenface method, the decision process, of whether or not the image includes a face, is done by comparing the original image and its reconstruction. If the reconstructed face and the original face are similar so that the difference between them is below a threshold, then it is accepted as face. Here are some images and their reconstructions under different conditions.

Input Image	Reconstructed Image	Info
		Image Size: 92x112 Num.of Trn.Img, *M* : 24 Num.of Eigenfaces, *M'* : 9 Reconstruction Error: 1608
		Image Size: 92x112 Num.of Trn.Img, *M* : 24 Num.of Eigenfaces,*M'*: 23 Reconstruction Error: 0

Figure 4.24. Test output for different number of eigenfaces

The test, whose results are seen in Fig. 4.24, is done using ORL dataset with the same input image applied to the eigenface algorithm in both cases. The input image is chosen from the training set, which means that if *M'* is chosen such that *M'*=*M*-1 the reconstruction error must normally be equal to zero. The changing parameter here is the *M'* value. In the first try *M'* is chosen as 9, hence it is expected that the reconstructed image will not be the same as the input image, and a reconstruction error of greater than zero will be calculated. Choosing *M'* bigger makes the algorithm work better, the greater the number of eigenvalues used, the smaller reconstruction error is obtained. If the number is increased to the number of training images than it should give zero reconstruction error. Fig. 4.24 shows that our algorithm works in the same way and this is also a proof that algorithm works correctly.

Input Image	Reconstructed Image	Info
		(A) Image Size: 92x112 M : 16 M' : 10 Reconstruction Error: I: 5467, II: 2798
		(B) Image Size: 92x112 M : 16 M' : 10 Reconstruction Error: I: 4680, II: 3036
		(C) Image Size: 92x112 M : 16 M' : 10 Reconstruction Error: I: 2952, II: 1360

Figure 4.25. Test outputs for different inputs

ORL dataset is used in the test above (See Fig. 4.25) as the training database and the varying parameter is the input image. The *Reconstruction Error* values given with caption, 'I', means that all the image is used in the reconstruction error calculation, while 'II' means that 20 pixels from each edge is not taken into consideration to eliminate the background (See Sec. 4.2). Comparing (A) and (B) we obviously say that the reconstruction error of (A) must be smaller than the one of (B). But if we calculated the reconstruction error using the whole image then it is smaller in (A), which means it is wrong. Looking at ORL database we see that the background color is darker than the face, however in the input pictures given above the faces are darker than background. This explains the reason of the mistake, and the solution is, not taking into consideration some edge part in

reconstruction error calculation. Doing this, given with caption ‘II’ gives the result that we expect.

What we see in (C), was one of the most difficult problems that we faced in our research. When a plain, or nearly plain figure appears whose dominant color is similar to the color of the faces in the training set, obviously the difference between the original and the reconstructed face becomes small which results in wrong detections. A solution for this problem is to use weighting coefficients and increasing the importance of the region of eyes, nose and mouth, however doing this still does not always solve the problem, or results in some other problems. Hence a variance check method (See Sec. 4.1.3) is used in the thesis to eliminate such plain figure inputs, which not only prevents such wrong detections, but also makes the code work faster.

After examining some of the reconstructed faces, the input faces with dimensions higher than the training images will be examined from now on. In this part, an input image is entered and the output is obtained so that the algorithm is applied window by window starting from the top left and shifting until all the image is evaluated. If a face is detected it is given as surrounded by a rectangle.

Figure 4.26. Test outputs for shift interval and variance check

Fig. 4.26 shows the experimental result obtained using the Eigenface method. First two outputs are obtained without any preprocess and third one is obtained using variance check. Hence we see that the number of wrong detections in the first two pictures is very high. All the conditions, like reconstruction error calculations and thresholds are same in the two pictures except that the *shift interval* is different. In the first picture, shift interval is 5 pixels, and in the second picture it is equal to 10. For this reason the probability of finding a face in the upper picture is higher than the lower one. Increasing the *shift interval* makes the program work faster but decreases the number of correct detections. First two figures show the plain figure problem, with lots of false detections in the plain figures. The result of applying variance check is shown in the last figure, where all the false detections are eliminated.

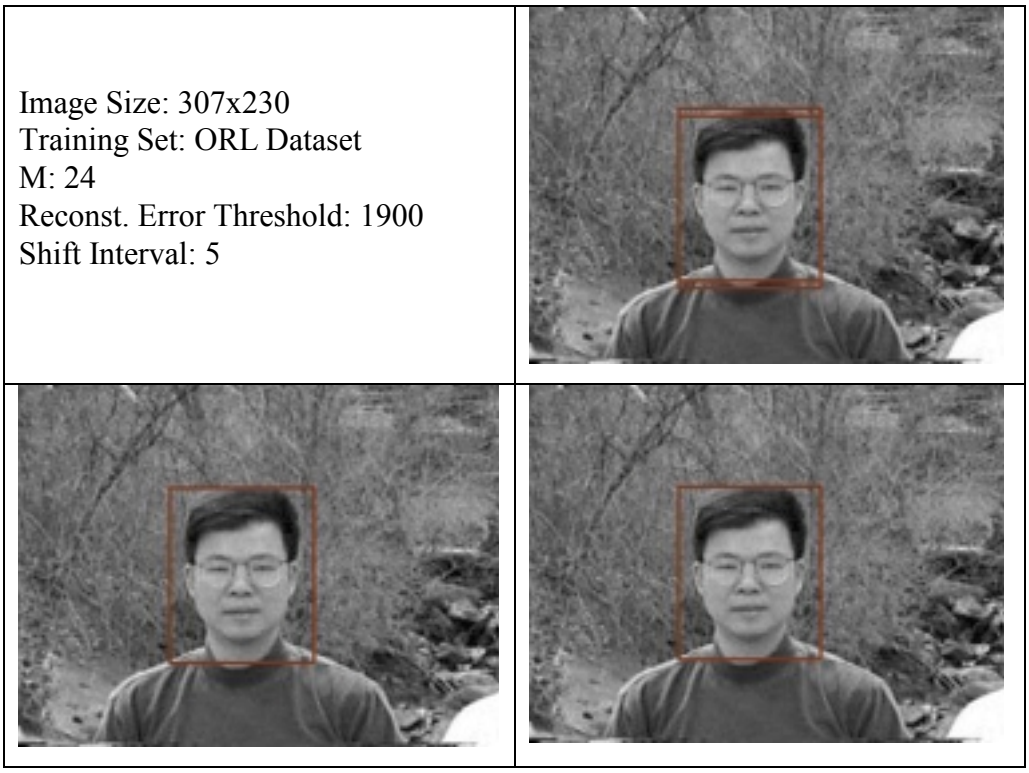


Figure 4.27. Test outputs for different number of eigenfaces

To see the effect of using a smaller *M'* in the Eigenface method, we made the test, whose results are shown in Fig. 4.27. Decreasing this number absolutely effects the success of the algorithm badly, however this decrease makes the program work faster, hence the aim is to find the optimum value, which makes tolerable effect on performance of the algorithm. None of the preprocessing algorithms are used here and only varying parameter is *M'*. In the first test its value is 16, in the second one it is equal to 14, and in the last one it is 10. Decreasing *M'* effected the value of the reconstruction error of the face in rectangle, and made it greater. Decreasing *M'* more resulted in not finding any face, and hence 10 seemed an optimal value for *M'* for *M*=24.

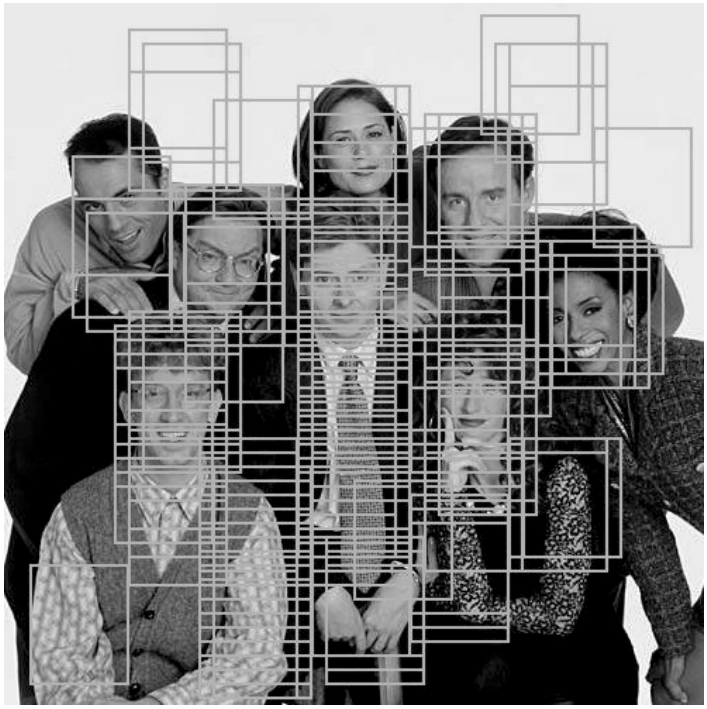
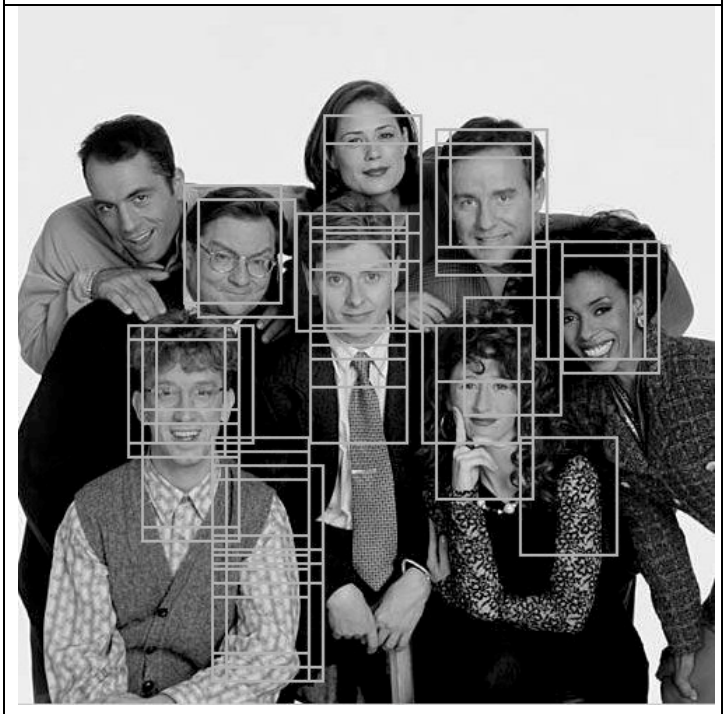
Image Size: 500x500 Training Set: ORL face database Number of classes, *c*: 20	 

Figure 4.28. Test output for different number of features

A test output is seen in Fig. 4.28 which is performed to show the effect of changing the number of features used in the Subspace LDA method. According to Figures 4.20 and 4.21, increasing the number of features decreases both the number of false detections and the detection rate. But detection rate decreases with a smaller slope which means that the system performance increases. The first picture in Fig. 4.28 shows the output for the number of features equal to 5, while the second picture shows it for the number of features equal to 9.

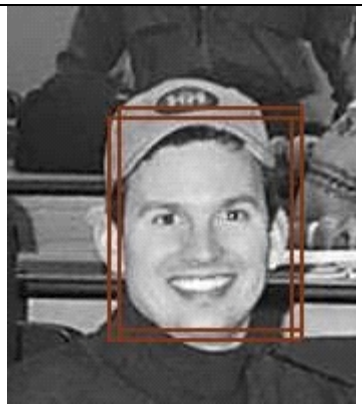
Image Size: 175x199 Training Set: METU Vision face database M: 16; M': 7 Reconst. Error Threshold: 2200 Shift Interval: 5	

Figure 4.29. Test output of a perfect result

Fig. 4.29 shows another figure with face detected output. Here, the Eigenface algorithm is used without any preprocessing algorithms. Investigating the figure can give an idea why it is successful without any preprocesses. The edges around the face are quite sharp, and the figure is very smooth, hence the histogram equalization is not needed here. Another reason is, there is no any plain region on the image, so variance check is not required. As a result the program gives a very successful result only using the eigenface algorithm here.

Image Size: 375x255
Training Set: METU Vision face database
M: 16; M': 7

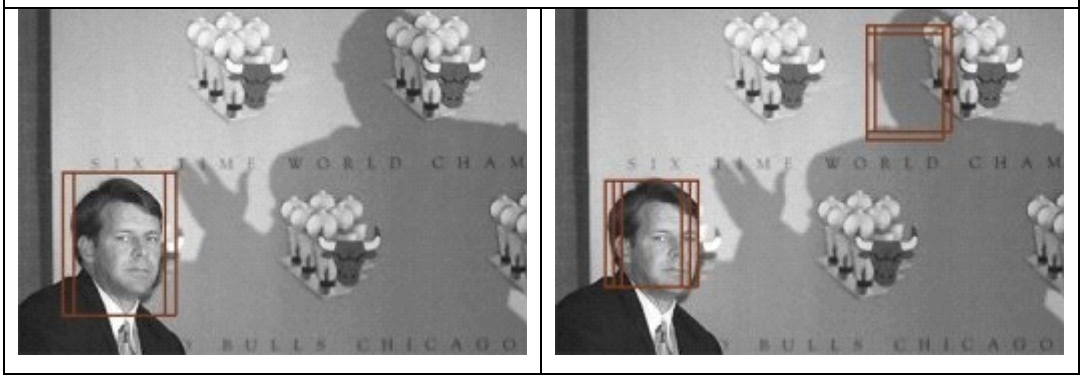


Figure 4.30. Test output for scale problem

The image shown in Fig. 4.30, is taken from the CMU II test set. The program detected the same face in two different scales, which is actually not a problem for a usual face detection algorithm while it is a problem in our case. This is because the robot needs to learn the distance of the face and arranges its position according to this distance. Robot guesses the difference according to the scale at which the face is detected. Reason of the problem is possibly the dimensions of the face whose ideal dimensions are between the two scales which is not implemented because of our timing constraints.

Image Size: 370x254
Number of classes, *c*: 20; Number of features: 10

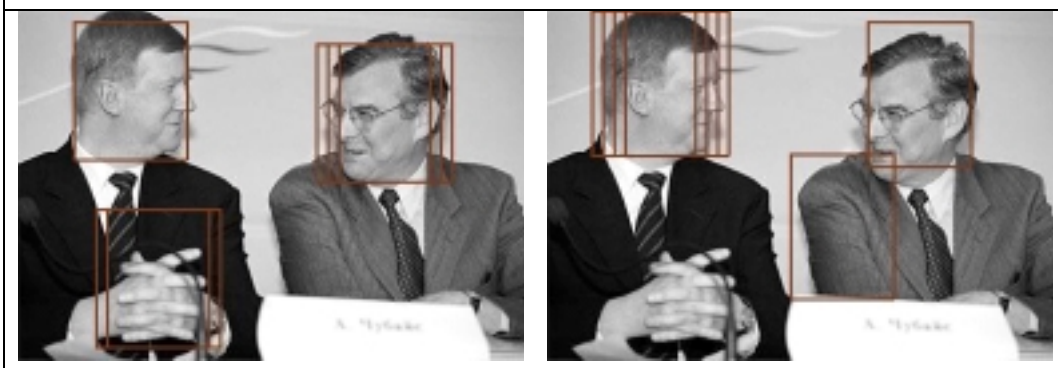


Figure 4.31. Test output for different training databases

The input image for the test on Fig. 4.31 is taken from the CMU II test database, and the experiment is performed to show the difference between using different training sets using the Subspace LDA method. In the first test the ORL training dataset is used, while in the second one the METU Vision face database is used. It is understood that the face detection algorithm is dependent slightly to the training image set, and that the results are quite similar.

Next figures show the outputs of the final system classified according to the test set used. The skin detection part is omitted for the greyscale images.

Figure 4.32. Test outputs of the final system for combined MIT/CMU test set (#1)

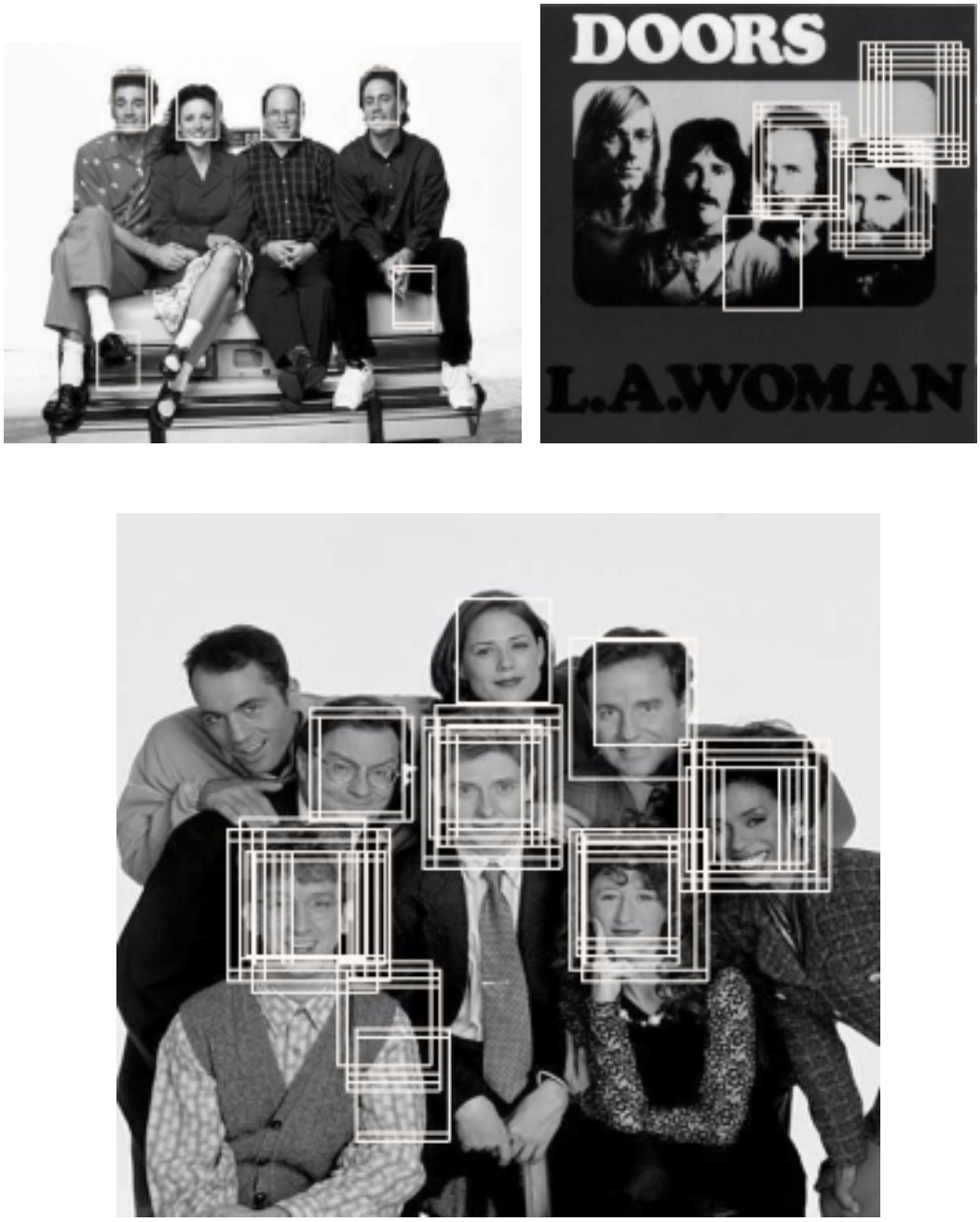


Figure 4.33. Test outputs of the final system for combined MIT/CMU test set (#2)

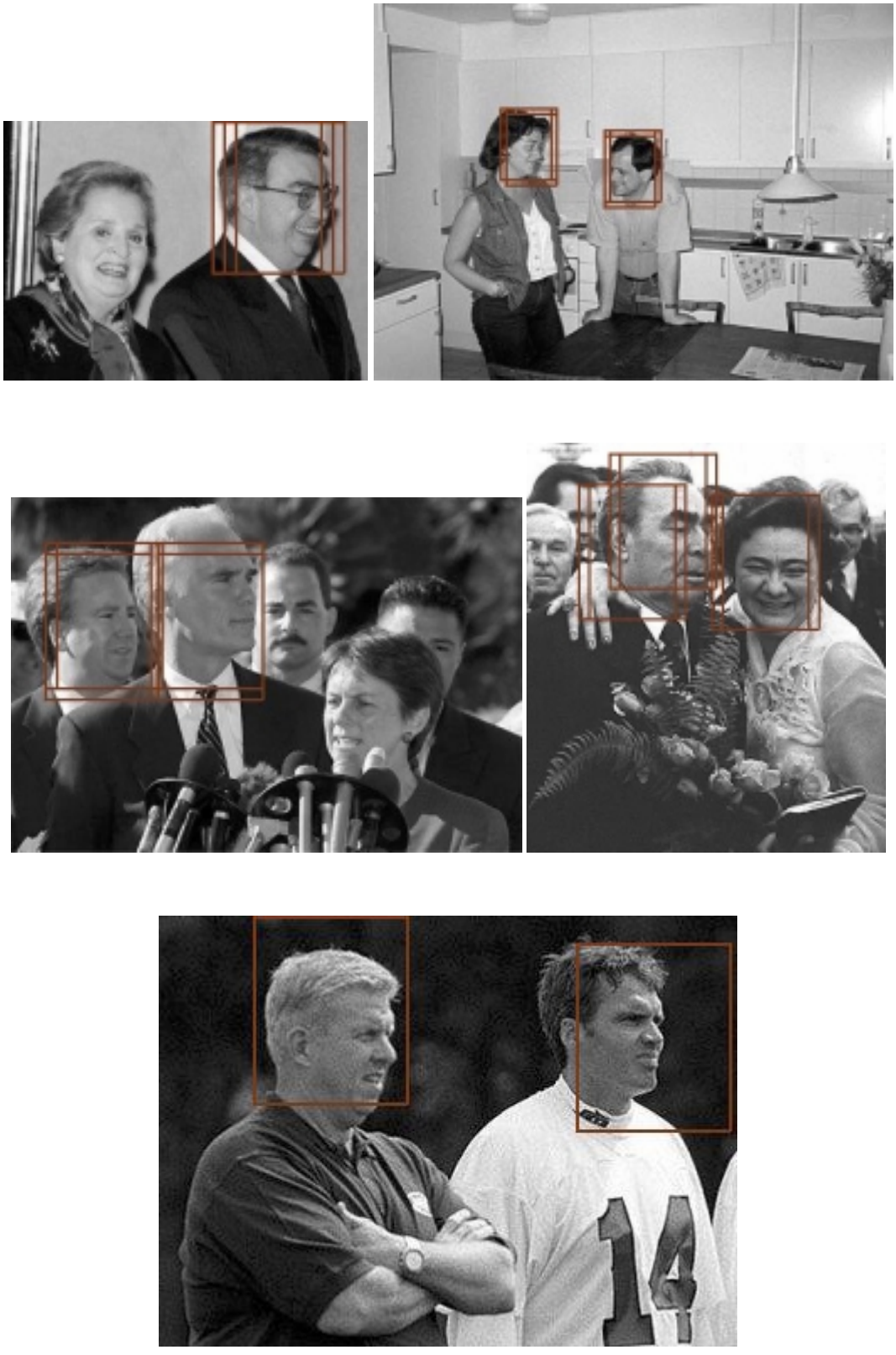


Figure 4.34. Test outputs of the final system for CMU II test set

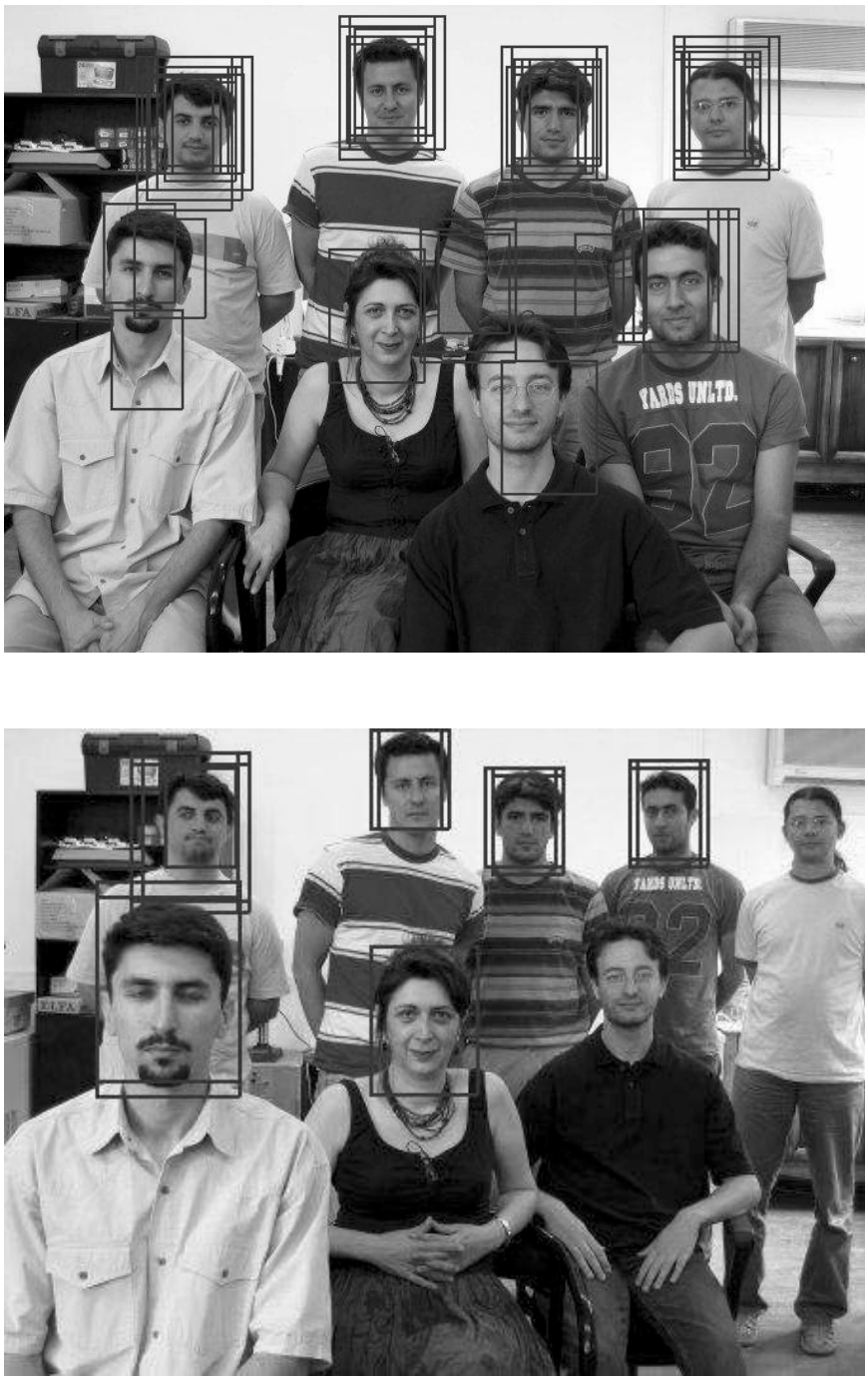


Figure 4.35. Test outputs of the final system for METU Vision test set (#1)

Figure 4.36. Test outputs of the final system for METU Vision test set (#2)

Figure 4.37. Test outputs of the final system for METU Vision test set (#3)

Figure 4.38. Test outputs of the final system for METU Vision test set (#4)

An active robot vision experiment is given in Fig. 4.39. These images are saved when the camera is on the robot. Since there are lots of images saved, only some of them can be given here with the hope of representing the active robot vision process. Because of the code delay of about 1 sec. the images does not seem like flowing, so it is difficult to obtain an idea on the robot vision performance of the

system just by looking at these pictures. The video files which are the collections of these images give better idea about system.

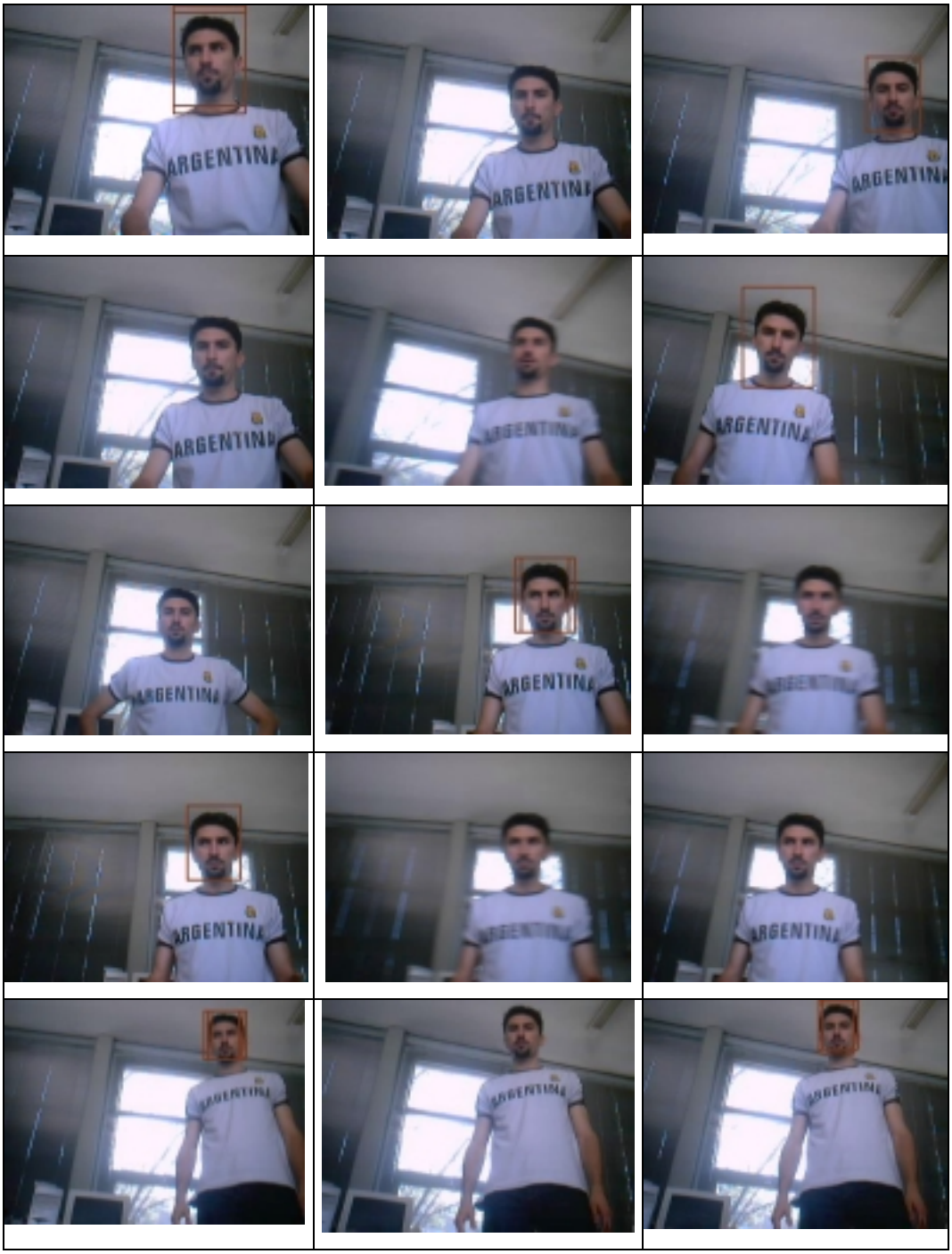


Figure 4.39. Real time robot vision system outputs

CHAPTER 5

CONCLUSIONS

In this thesis, a robot with face tracking capability is studied. In the final system the robot should get the image of the scene from the camera. Then the robot should arrange its position according to the position of the face in the image, if detected. After evaluating an image it arranges its position, captures a new image from the camera, puts the input image to the algorithm and so on.

The main topic in the thesis is obviously face detection. Two different methods are studied for face detection, which are Eigenface method and Subspace LDA method. These two methods are implemented, tested and compared. The combined CMU/MIT test set is used in these experiments. The best values for some parameters are found out first before starting the evaluation tests. The values for these parameters are investigated using the test results for the detection rate, number of false detections and the computation time. The ROC curve, which is the plot of the detection rate versus the number of false detections, has been very useful in this decision.

For the Eigenface method these parameters are *number of training* images and *number of eigenfaces*. Test results showed that the system performance increases with the increasing number of training images and/or the number of eigenfaces. However, since our system has strict time constraints we had to choose the values considering also the computation time. The parameters are chosen with this criteria as 16 training images and 7 eigenfaces. Using these values the system has

a detection rate of 74% with 287 false detections in the CMU/MIT test set when the training image set is ORL database. Using METU Vision face database as the training set, decreases the system performance so that the detection rate decreases to 61% with 255 false detections. This is not an unexpected result though, because this database is created to achieve best performance for our robot vision system when tested in the laboratory conditions.

For the Subspace LDA method the critical parameters were *number of classes* and *number of features*. According to the test results system performance increases with the increase in these values. Considering again the performance plots and the computational time plot the best values for the number of classes is obtained as 20 and the number of features is obtained as 9. With these values the CMU/MIT test set results for the Subspace LDA method show that it has 88% face detection rate for both training image sets with 481 and 483 false detections for the ORL database and the METU Vision face database, respectively.

The results showed that the Subspace LDA method has a better face detection performance than the Eigenface method, however the high computation time of the Subspace LDA method was a very important drawback for our system. Hence the Eigenface method is used in the final system whose computational cost is much lower than the one of Subspace LDA method.

The input image which will be used in face detection is first applied to a number of preprocessing operations, which are used not only to decrease the number of false detections, but also to make the algorithm work faster. As usual there is a trade-off between using preprocessing and not using them. Using preprocessing usually prevents ending in false detections, but surely it also causes some real faces to be eliminated.

The skin detection is used in a different way here. To make it faster not all the input image but only 18 pixels are used taken from the both cheeks and if the number of skin pixels is greater than a threshold it is assumed to be a skin rectangle. The YIQ color space is used for skin detection with threshold values adapted to the thesis after trial and errors.

One of the most challenging problems that we faced during our work, was ending up with false detections on a plain figure, when it is almost the same color through the image and this base color similar to a face color. In a face, eyes and mouth are the only parts that differ in color from the other parts of the face, but these parts do not cover a big percentage on the face. So we decided to use weights in the calculation of the reconstruction error and applied greater weights on the area of eyes. This did not be enough to eliminate such errors though. Hence we applied a variance check algorithm and tried to eliminate the windows quickly where there is a plain figure. This method eliminated many of the false detections and also made the code run faster.

If the input image passes the skin detection and variance check tests, it is applied to histogram equalization and is than given to the Eigenface algorithm for face detection. As usual, there may be more than one image in the camera scene, when the robot detects more than one image it takes into account only the one on the upper left part of the camera and moves accordingly.

The overall system performance is tested using the METU Vision color test set. According to this test, the detection rate of the final system is found out as 87.5% with 32 false detections which is tolerable. The rotation test results show that the system performance decreases a little when the test set is composed of profile faces, but still has a satisfactory performance. Finally the brightness test is applied to our system and it showed that the system is highly dependent to the brightness.

There are several materials which can be further studied for better performance. One of them should absolutely be creating a larger database as the training set and as the test set. This must be the starting point for the future studies. Another point for further study can be obtaining a faster face detection system. This can be either by using another method or by improving the Eigenface method.

REFERENCES

[1] M. Turk, A. Pentland, “Eigenfaces for Recognition”, Journal of Cognitive Neuroscience Volume 3, Number 1, pp. 71-86, 1991.

[2] M. H. Yang, N. Ahuja, “Detecting Faces in Images: A Survey”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1, 2002.

[3] M. A. Turk, A.P. Pentland, “Face Recognition Using Eigenfaces”, IEEE, 1991.

[4] İ. Atalay, “Face Recognition Using Eigenfaces”, Ms. Thesis, Institute of Science and Technology, Istanbul Technical University, 1996.

[5] G. Yang, T. S. Huang, “Human Face Detection in Complex Background”, Pattern Recognition, vol. 27, no. 1, pp. 53-63, 1994.

[6] C. Kotropoulos, I. Pitas, “Rule-Based Face Detection in Frontal Views”, Proc. Int’l Conf. Acoustics, Speech and Signal Processing, vol. 4, pp. 2537-2540, 1997.

[7] T. Kanade, “Picture Processing by Computer Complex and Recognition of Human Faces”, PhD thesis, Kyoto Univ., 1973.

[8] H.P. Graf, T. Chen, E. Petajan, E. Cosatto, “Locating Faces and Facial Parts”, Proc. First Int’l Workshop Automatic Face and Gesture Recognition, pp. 41-46, 1995.

[9] B. Takacs, H. Wechsler, “Face Location Using a Dynamic Model of Retinal Feature Extraction”, Proc. First Int’l Workshop Automatic Face and Gesture Recognition, pp. 243-247, 1995.

[10] C.C. Han, H.Y.M. Liao, K.C. Yu, and L.H. Chen, “Fast Face Detection via Morphology-Based Pre-Processing”, Proc. Ninth Int’l Conf. Image Analysis and Processing, pp. 469-476, 1998.

[11] H. Rowley, S. Baluja, T. Kanade, “Neural Network-Based Face Detection”, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 203-208, 1996.

[12] I. Craw, H. Ellis, J. Lishman, “Automatic Extraction of Face Features”, Pattern Recognition Letters, vol. 5, pp. 183-187, 1987.

[13] I. Craw, D. Tock, A. Bennett, “Finding Face Features”, Proc. Second European Conf. Computer Vision, pp. 92-96, 1992.

[14] P. Sinha, “Processing and Recognizing 3D Forms”, PhD thesis, Massachusetts Inst. of Technology, 1995.

[15] A. Yuille, P. Hallinan, D. Cohen, “Feature Extraction from Faces Using Deformable Templates”, Int’l J. Computer Vision, vol. 8, no. 2, pp. 99-111, 1992.

[16] T. Kohonen, “Self-Organization and Associative Memory”, Springer, 1980

[17] B. Moghaddam, A. Pentland, “Probabilistic Visual Learning for Object Recognition”, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 696-710, 1997.

[18] M.-H. Yang, N. Ahuja, D. Kriegman, “Mixtures of Linear Subspaces for Face Detection”, Proc. Fourth Int’l Conf. Automatic Face and Gesture Recognition, pp. 70-76, 2000.

[19] H. Rowley, S. Baluja, T. Kanade, “Neural Network-Based Face Detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 23-38, 1998.

[20] F.S. Samaria, “Face Recognition Using Hidden Markov Models”, Ph.D. Thesis, Univ. of Cambridge, 1994.

[21] J.C. Terrillon, M. David, S. Akamatsu, “Automatic Detection of Human Faces in Natural Scene Images by use of a Skin Color Model and of Invariant Moments”, Proc. Int. Conf. on Automatic Face and Gesture Recognition, pp. 112-117, 1998.

[22] J.Yang, W. Lu, A. Waibel, “Skin-Recognition color modeling and adaptation”, Proc. of ACCU ‘98, vol. 2, pp. 687-694, 1998.

[23] K. Sobottka, I. Pitas, “Segmentation and tracking of faces in color images”, Second Int. Conf. Automatic Face and Gesture Recognition, pp. 236-241, 1996.

[24] D. Saxe, R. Foulds, “Toward robust skin identification in video images”, 2nd Int. Face and Gesture Recognition Conf., 1996.

[25] F. K. H. Quek, T. Mysliwiec, M. Zhao, “Fingermouse: A freehand pointing interface”, Proc. Int. Workshop on Automatic Face and Gesture Recognition, pp. 372-377, 1995.

[26] M. Collobert, R. Feraud, G. Le Tourneur, D. Bernier, J. E. Vaiallet, Y. Mahieux, D. Collobert, “Listen: A system for locating and tracking individual speakers”, Int. Conf. Automatic Face and Gesture Recognition, pp. 283-288, 1996.

[27] H.A. Drury, “Meta-Analysis of Functional Organization Using the Visible Man Surface-Based Cerebral Atlas”, BrainMap: Spatial Normalization And Registration, 1996.

[28] N. Efford, “Digital Image Processing, A Practical Introduction Using Java”, pp. 118-130, Addison Wesley, 2000.

[29] L. Sirovich, M. Kirby, “Low-dimensional procedure for the characterization of human faces”, Journal of the Optical Society of America A, pp. 519-524, 1987.

[30] R. Jain, R. Kasturi, B. G. Schunck, “Machine Vision”, pp. 1-5, Co-published by the MIT Press and McGraw-Hill, Inc., 1995.

[31] C. Liu, H. Wechsler, “Enhanced Fisher Linear Discriminant Models for Face Recognition”, 14th International Conference on Pattern Recognition, ICPR’98, 1998.

[32] P. Belhumeur, J. Hespanha, D. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection”, IEEE Trans. Pattern Analysis and Machine Intelligence, pp. 711–720, 1997.

[33] R. O. Duda, P. E. Hart, D.G. Stork, “Pattern Classification”, 2nd Edition, John Wiley and Sons Inc., pp. 114-121, 2001.

[34] W. Zhao, R. Chellappa, P.J. Phillips, “Subspace Linear Discriminant Analysis for Face Recognition”, EDICS: Image Analysis (2-ANAL), 1999.

[35] K. Fukunaga, “Statistical Pattern Recognition”, Academic Press, 1989.

[36] S.S. Wilks, “Mathmetical Statistics”, Wiley, 1962.

[37] J. P. Phillips, H. Wechsler, J. Huang, P. Rauss. “The FERET database and evaluation procedure for face recognition algoritms”. Image Vision Computing, 16(5), 1998.

APPENDIX A

PIONEER-2 ROBOT

The Pioneer-2 DX8 robot, which is a product of ActiveMedia Robotics Company, is used in the thesis. General features of this robot is given in this part.

A.1. Robot Hardware Properties

Pioneer is a family of mobile robots, both two-wheel and four-wheel drive, including many different featured robots. These small, research and development platforms share a common architecture and foundation software with all other ActivMedia robots and all employ a client-server robotics control architecture.

The P2-DX8 robot comes complete with a sturdy aluminum body, balanced drive system (two-wheel differential with caster), reversible DC motors, motor-control and drive electronics, high-resolution motion encoders, and long-life, hot-swappable battery power, all managed by an onboard microcontroller and mobile-robot server software. The components of P2-DX8 are seen in Fig. A.1.

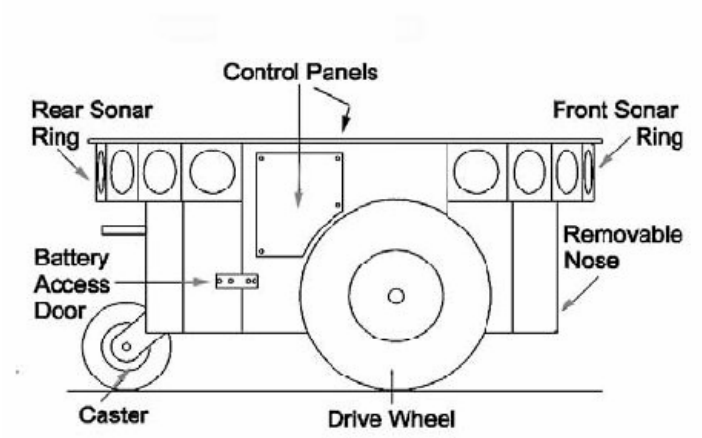


Figure A.1. Components of Pioneer 2-DX8.

The robot supports many accessories, which come with it, including built-in hardware support for sonar and bump sensors and lift/gripper effectors, as well as serial-port and server software support for a number of sensors, effectors, and control accessories, like an onboard PC system, 5-DOF arm, robotic pan-tilt cameras etc.

The Pioneer 2-DX8 robot has two interfaces with the outer world, first is through a microcontroller, second is through an on-board computer, which are given in detail in next sections.

A.1.1. Hitachi H8S-Based Microcontroller

The H8S-based ActivMedia robot has a variety of expansion power and I/O ports for attachment and close integration of a client PC, sensors, and a variety of accessories – all accessible through a common application interface to the robot server software, AROS. Features include:

- 18 MHz Hitachi 8HS/2357 with 32K RAM and 128K FLASH
- Optional 512K FLASH or SRAM expansion

- 3 RS-232 serial ports (4 connectors) configurable from 9.6 to 115.2 kbaud
- 4 sonar arrays of 8 sonar each
- 2 8-bit bumpers / digital input connectors
- 1 P2 Gripper/User I/O connector with 8-bits digital I/O and 1 analog input
- 1 Expansion / bus connector containing
- 5 analog input
- 2 analog output
- 8-bit I/O bus with r/w and 4 chip-selects
- 2-axes, 2-button joystick port
- User Control Panel
- Controller host serial connector
- Main power and bi-color LED battery level indicators
- RESET and MOTORS pushbutton controls
- Piezo buzzer
- Motor/Power Board (drive system) interface with PWM (Pulse Width Modulation) and motor-direction control lines and 8-bits of digital input

The developer can connect to the P2-DX8 via the serial port, using ARIA (See Sec. A.2.1), or can run his program on the onboard computer. With the onboard PC option, the P2-DX8 robot becomes an autonomous agent. Fig. A.2 shows an example of connections.

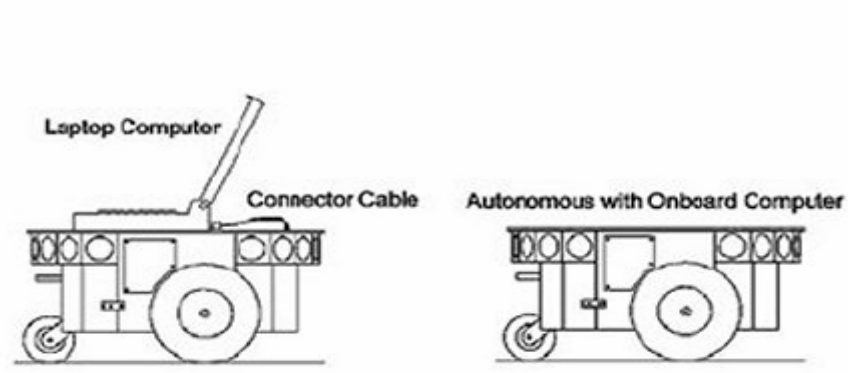


Figure A.2. Connection styles to P2-DX8

The “User Control Panel” (See Fig. A.3) is where one have access to the AROS-based onboard microcontroller. Found on the left-side panel of the DX8, it consists of control buttons and indicators, and an RS232-compatible serial port with a 9-pin DSUB connector.

The red PWR LED is lit whenever main power is applied to the robot. The green STAT LED state depends on the operating mode and other conditions. It flashes slowly when the controller is awaiting a connection with a client and flashes quickly when connected with a client and the motors are engaged. It also flashes moderately fast when in maintenance mode.

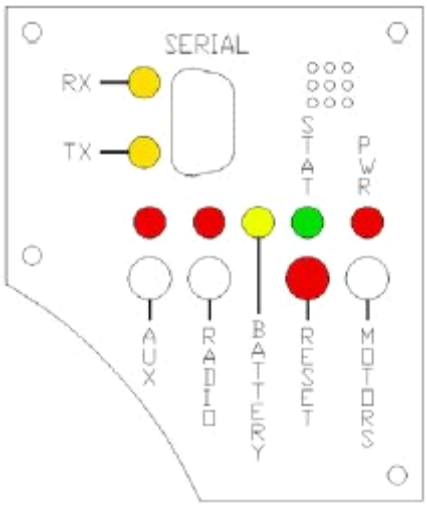


Figure A.3. P2-DX8 User Control Panel

The BATTERY LED’s apparent color depends on the robot’s battery voltage: green when fully charged (>12.5 volts) through orange, and finally red when the voltage is below 11.5 V. When in maintenance mode, however, the BATTERY LED glows bright red only, regardless of battery charge.

A built-in piezo buzzer (audible through the holes just above the STAT and PWR LEDs) provides audible clues to the robot's state, such as upon successful startup of the controller and a client connection.

The SERIAL connector, with incoming and outgoing data indicator LEDs (RX and TX , respectively), is through where you may interact with the H8S microcontroller from an offboard computer for tethered client-server control and for AROS system maintenance. The port is shared internally by the HOST serial port, to which one connects the onboard computer or radio modem. Digital switching circuitry disables the internal HOST serial port if the computer or radio modem is OFF. However, serial port interference will be a problem if the HOST and User Control SERIAL ports are both occupied and engaged.

RADIO and AUX are pushbutton switches which engage or disengage power to the respective devices on the Motor/Power Interface board. Respective red LEDs indicate when power is ON. The red RESET pushbutton acts to unconditionally reset the H8S controller, disabling any active connections or controller-attached devices, including the motors.

The white MOTORS pushbutton's actions depend on the state of the controller. When connected with a client, it is pushed to manually enable and disable the motors, as its label implies. When not connected, press the pushbutton once to enable joydrive mode, and again to enable the motors self-test.

To engage AROS maintenance mode, one should press and hold the white MOTORS button, press and release the red RESET button, then release MOTORS.

A.1.2. Onboard Computer

The Pioneer 2 robot line has provisions for an onboard, internally integrated PC. Mounted just behind the nose of the robot, the PC is a common EBX form-factor that comes with up to four serial ports, 10/100Base-T Ethernet, monitor, keyboard, and mouse ports, two USB ports, and support for floppy, as well as IDE hard-disk drives. For additional functionality, such as for sound, video framegrabbing, firewire or PCMCIA bus, and wireless Ethernet, the onboard PC accepts PC104 and PC104-plus (PCI bus-enabled) interface cards that stack on the motherboard. Necessary 5 VDC power comes from a dedicated DC-DC converter, mounted nearby. A hard-disk drive is specially shock-mounted to the robot's nose, in between a cooling fan and computer speaker.

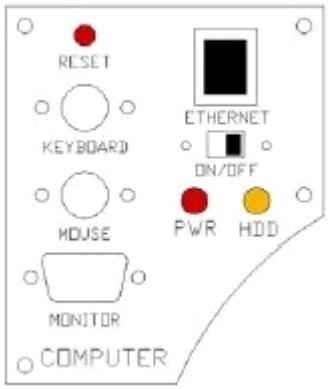


Figure A.4. Pioneer 2-DX8 computer control sidepanel

The developer can choose any of the connection types, either connecting from an external PC, running the software on this external PC and communicating using the serial port or connecting from the onboard PC running software on this onboard PC.

A.2. Robot Software Properties

The P2-DX8 robot is the server in a client-server environment: It handles the low-level details of mobile robotics, including maintaining the platform's drive speed and heading over uneven terrain, acquiring sensor readings, such as the sonar, and managing attached accessories like the P2-Gripper. To complete the client-server architecture, P2-DX8 robot requires a client connection: software running on a computer connected with the robot's controller via the HOST serial link and which usually provides the high-level, intelligent robot controls, including obstacle avoidance, path planning, features recognition, localization, gradient navigation, and so on.

An important benefit of P2-DX8's client-server architecture is that different robot servers can be run using the same high-level client. For example, a robot simulator is provided that runs on the host machine that can look and act just like the real robot. With the Simulator, one can conveniently perfect his application software, then run it without modification on real P2-DX8 robot. Several clients also may share responsibility for controlling a single mobile server, which permits experimentation in distributed communication, planning, and control.

Besides the open-systems ActivMedia Robotics Operating System (AROS) software onboard the robot controller, the P2-DX8 robot also comes with a host of advanced robot-control client software applications and application-development environments. Software development includes ActivMedia Robotics Interface for Applications (ARIA), released under the GNU Public License, and complete with C++ libraries and source code.

A.2.1. AROS

All ActivMedia robots use a client-server mobile robot-control architecture. In the model, the robot's controller servers work to manage all the low-level details of the mobile robot's systems. These include operating the motors, firing the sonar, collecting sonar and wheel encoder data, and so on - all on command from and reporting to a separate client application, such as ARIA.

With this client/server architecture, robotics applications developers do not need to know many details about a particular robot server, because the client insulates them from this lowest level of control. Some of them, however, may want to write their own robotics control and reactive planning programs, or just would like to have a closer programming relationship with the robot. The ActivMedia Robotics Operating System (AROS) client-server interface lets the developer to communicate with and control the ActivMedia robot. The same AROS functions and commands are supported in the various client-programming environments.

All the communications, commands/responds between the computer and the robot are accomplished using ARIA during the thesis, hence the other protocol implementation details are not given here.

A.2.2. ARIA

The ActivMedia Robotics Interface for Applications (ARIA) is a C++-based open-source development environment that provides a robust client-side interface to a variety of intelligent robotics systems, including P2-DX8 robot's controller and accessory systems.

ARIA is the ideal platform for integration of developer's own robot-control software, since it handles the lowest-level details of client-server interactions,

including serial communications, command and server-information packet processing, cycle timing, and multithreading, as well as a variety of accessory controls, such as for the PTZ robotic camera, the P2-Gripper, scanning laser-range finder, motion gyros, among many others.

Without going into too much detail we give a simple code here written in Java which helps to connect to the robot simulator. This code must be run after installing the ARIA software to the computer and running the robot simulator. This code makes the robot simulator start running, move forward 1 m (1000 mm), turn 90 degrees around itself.

```
public class simple {
  static {
    try {
        System.loadLibrary("AriaJava");
    } catch (UnsatisfiedLinkError e) {
      System.err.println("Native code library failed to load." +
          e);
      System.exit(1);
    }
  }
  public static void main(String argv[]) {
    Aria.init(0, true);
    ArRobot robot = new ArRobot("robot1", true, true, true);
    ArTcpConnection conn = new ArTcpConnection();
    conn.setPort("localhost", 8101);
    robot.setDeviceConnection(conn);
    if (!robot.blockingConnect()) {
      System.err.println("Could not connect to robot,
          exiting.\n");
      System.exit(1);
    }
    robot.runAsync(true);
    robot.lock();
    robot.move(1000);
    robot.unlock();
    ArUtil.sleep(5000);
    robot.lock();
    robot.setHeading(90);
    robot.unlock();
    ArUtil.sleep(5000);
    System.out.println("Robot coords (" + robot.getX() + ", " +
       robot.getY() + ", " + robot.getTh() + ")");
    robot.lock();
    robot.stopRunning(true);
    robot.disconnect();
    Aria.shutdown();
  }
}
```

APPENDIX B

TRAINING IMAGE SETS

B.1. METU Vision Face Database

Size of each image: 76x106

Number of individuals: 15

Number of pictures per individual: 10

Figure B.1. METU Vision Face Database, Part I

Figure B.2. METU Vision Face Database, Part II

B.2. ORL Database

Size of each image: 92x112

Number of individuals: 40

Number of pictures per individual: 10

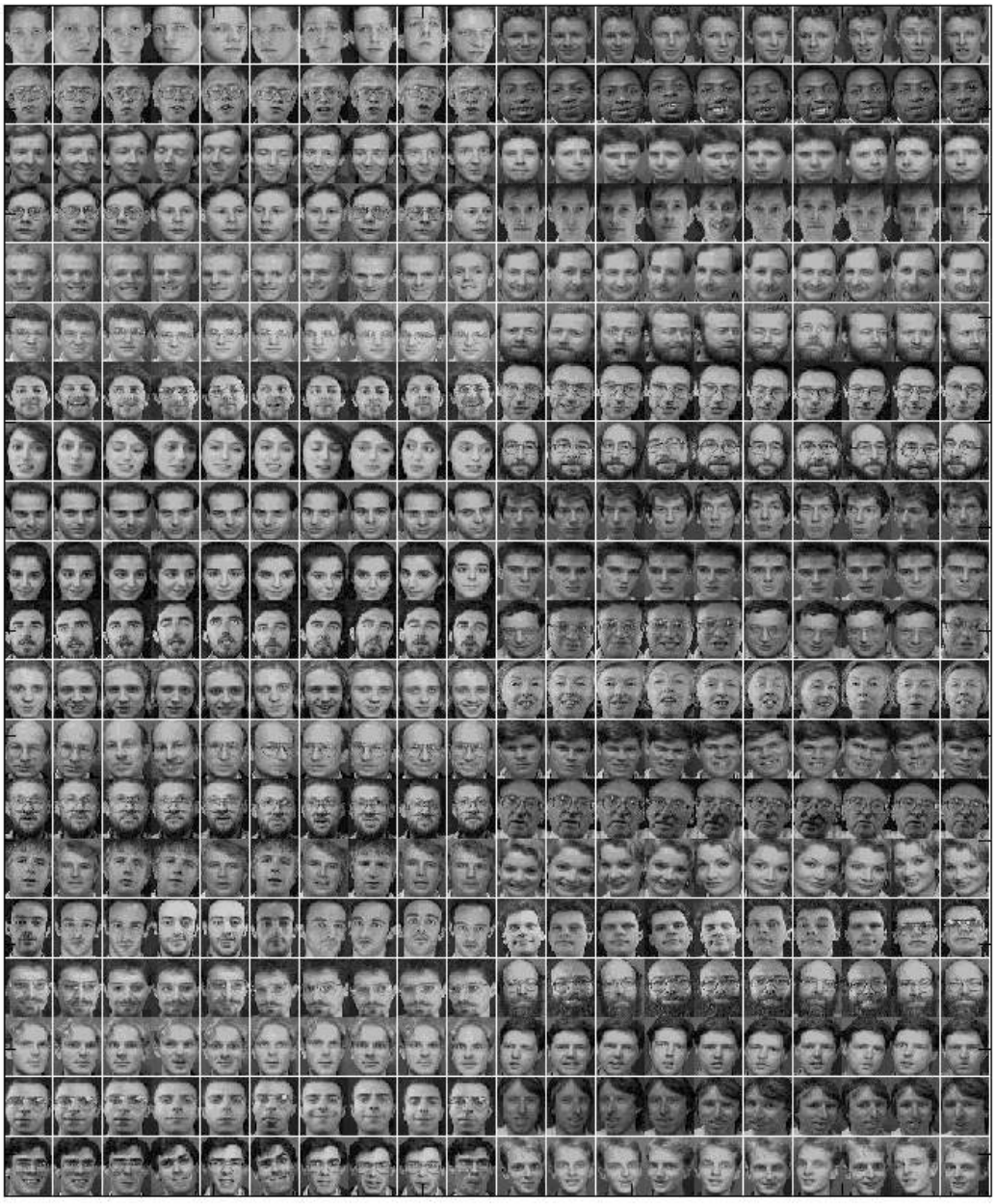


Figure B.3. ORL database