REAL-TIME DETECTION AND TRACKING OF HUMAN EYES IN VIDEO SEQUENCES

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

BY

ZAFER SAVAS

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

AUGUST 2005

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. Ismet ERKMEN
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. Ugur HALICI
Supervisor

Examining Committee Members

Prof. Dr. Kemal LEBLEBICIOGLU	(METU,EE)	_____
Prof. Dr. Ugur HALICI	(METU,EE)	_____
Assist. Prof. Dr. Ilkay ULUSOY	(METU,EE)	_____
Assist. Prof. Dr. Ilhan KONUKSEVEN	(METU,ME)	_____
Assist. Prof. Dr. Kürsat ÇAGILTAY	(METU,CEIT)	_____

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 the original work.

Name, Last name : Zafer SAVAS

Signature :

ABSTRACT

REAL-TIME DETECTION AND TRACKING OF HUMAN EYES IN VIDEO SEQUENCES

SAVAS, Zafer

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Ugur HALICI

August 2005, 98 Pages

Robust, non-intrusive human eye detection problem has been a fundamental and challenging problem for computer vision area. Not only it is a problem of its own, it can be used to ease the problem of finding the locations of other facial features for recognition tasks and human-computer interaction purposes as well. Many previous works have the capability of determining the locations of the human eyes but the main task in this thesis is not only a vision system with eye detection capability; Our aim is to design a real-time, robust, scale-invariant eye tracker system with human eye movement indication property using the movements of eye pupil. Our eye tracker algorithm is implemented using the Continuously Adaptive Mean-Shift (CAMSHIFT) algorithm proposed by Bradski and the EigenFace method proposed by Turk & Pentland. Previous works for scale invariant object detection using Eigenface method are mostly dependent on limited number of user predefined scales which causes speed problems; so in order to avoid this problem an adaptive eigenface method using the information extracted from CAMSHIFT algorithm is implemented to have a fast and scale invariant eye tracking.

First of all; human face in the input image captured by the camera is detected using the CAMSHIFT algorithm which tracks the outline of an irregular shaped object that

may change size and shape during the tracking process based on the color of the object. Face area is passed through a number of preprocessing steps such as color space conversion and thresholding to obtain better results during the eye search process. After these preprocessing steps, search areas for left and right eyes are determined using the geometrical properties of the human face and in order to locate each eye indivually the training images are resized by the width information supplied by the CAMSHIFT algortihm. Search regions for left and right eyes are individually passed to the eye detection algortihm to determine the exact locations of each eye. After the detection of eyes, eye areas are individually passed to the pupil detection and eye area detection algorithms which are based on the Active Contours method to indicate the pupil and eye area. Finally, by comparing the geometrical locations of pupil with the eye area, human gaze information is extracted.

As a result of this thesis a software named “TrackEye” with an user interface having indicators for the location of eye areas and pupils, various output screens for human-computer interaction and controls for allowing to test the effects of color space conversions and thresholding types during object tracking has been built.

Keywords : Eye Tracking, Continuously Adaptive Mean-Shift Algorithm, EigenFace method, Principal Component Analysis, Active Contours, Intel Open Source Computer Vision Library (OpenCV)

ÖZ

VIDEO GÖRÜNTÜLERINDE GERÇEK ZAMANLI INSAN GÖZÜ SAPTAMA VE TAKIBI

SAVAS, Zafer

Yüksek Lisans , Elektrik-Elektronik Mühendisligi Bölümü

Tez Yöneticisi: Prof. Dr. Ugur HALICI

Agustos 2005, 98 sayfa

Bilgisayarla görme alani için güçlü, çevre etmenlerinden etkilenmeyen insan gözünün saptanmasi problemi temel ve ugrastirici bir problem olmustur. Bu problem, sadece kendi alani ile sinirli kalmayan ayni zamanda diger yüz nitelerinin bulunmasi ve insan? makine etkilesimi problemlerinde de yardimci amaçli kullanilabilir. Geçmis birçok çalisma, insan gözünün yerininin saptanmasi yetenegine sahiptir fakat bu tezdeki ana amaç, sadece gözün saptanmasi degil; gerçek-zamanli, güçlü, ölçeklendirmeden bagimsiz göz takip eden ve göz hareketlerini göz bebegi hareketlerini takip ederek saptayan bir sistem gelistirilmesidir. Tasarladigimiz göz takip edici sistem Bradski tarafindan gelistirilen Sürekli Uyarlanan Mean-Shift ve Turk & Pentland tarafindan gelistirilen Özyüz metotlari kullanilarak gelistirilmistir. Özyüzleri kullanarak ölçekten bagimsiz cisim saptama gerçeklestiren geçmis çalismalar, genellikle hiz problemine sebep olan sinirli sayida ve önceden belirlenen ölçekteki cisimleri saptamaya dayanmaktadir; dolayisiyla bunun üstesinden gelebilmek için CAMSHIFT algoritmasi sonucu elde edilen bilgilerin kullanildigi uyarlanabilen özyüz yöntemi gelistirilerek hizli ve ölçekten bagimsiz bir göz takibi gelistirilmistir.

Ilk olarak; kameradan alinan görüntüdeki insan yüzü önceden belirlenen renk degerlerinin, burada insan ten rengidir, tespit edilmesine dayali bir algortima olan CAMSHIFT algoritmasi kullanilarak tespit edilmektedir. Tespit edilen yüz alani göz arama süreci sirasinda daha iyi sonuçlar elde edebilmek amaciyla renk uzayi çevirimi ve esikleme gibi bir dizi adimlardan geçirilmektedir. Bu önsüreçleme adimlarindan sonra insan yüzünün geometrik özellikleri kullanilarak sol ve sag gözler için arama bölgeleri belirlenir ve gözlerin ayri ayri saptanabilmesi için kullanilacak egitim görüntüleri CAMSHIFT algoritmasi tarafindan elde edilen en bilgisine göre tekrar boyutlandirilir. Gözlerin tespit edilmesinden sonra göz bölgeleri, temeli aktif dis hatlar metoduna dayanan göz bebegi ve göz alani saptama algoritmalarina ayri ayri geçirilerek göz bebegi ve göz alanlari isaretlenir. Son olarak göz bebeginin göz alanina göre konumunun degerlendirilmesi sonucu insan bakis yönü bilgisi çikarilir.
Bu tez çalismasinin sonucu olarak kullanici arayüzünde göz alani ve göz bebeginin isaretlendigi, insan-makine etkilesiminde kullanilabilecek çesitli çiktilarin yer aldigi, renk uzayi çevrimlerinin ve esikleme tiplerinin cisim takibindeki etkilerinin test edilmesine olanak saglayan “TrackEye” yazilimi hazirlanmistir.

Anahtar Kelimeler : Göz Takibi, Sürekli Uyarlanan Mean-Shift algoritmasi, Özyüz Yöntemi, Ana Bilesen Analizi, Aktif Dis Hatlar, Intel Açik Kod Bilgisayarla Görme Kütüphanesi (OpenCV)

to my family & dolores

ACKNOWLEDGEMENTS

I would like to thank Prof. Dr. Ugur HALICI for her valuable support throughout the development of this thesis. This thesis really would not have been reach to an end without her guidance and patience.

I would like to thank to Ilkay ULUSOY for her guidance at the beginning of this work.

I also would like to thank to ASELSAN A.S. especially to my department Test Engineering for letting me to involve in this thesis work.

Finally I am grateful to my parents and my colleagues at work for their continuous encouragement.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

LIST OF ABBREVIATIONS

2D	: Two dimensional
CAMSHIFT	: Continuously Adaptive Mean Shift
HSV	: Hue Saturation Value
MPEG-7	: Standard for Multimedia Content Description Interface
OPENCV	: Open Source Computer Vision Library
PC	: Personel Computer
PCA	: Principal Component Analysis
RGB	: Red Green Blue
YCbCr	: Luminance Chrominance

CHAPTER 1

INTRODUCTION

1.1 Background and Rationale of the Study

Eye tracking and eye movement-based interaction using computer vision techniques have the potential to become an important component in future perceptual user interfaces. So by this motivation designing a real-time eye tracking software compatible with a standard PC environment is the main aim of this thesis.

In general, the term *"eye detection"* is widely used when static face images are of concern and the main aim is to find the face region which contains both eyes, and *eye tracking* term is used referring to the process of continuously detecting eyes in video sequences which contains only face images. In this thesis the term *eye tracking* means real-time, continuously detection of human eyes individually and extraction of eye features with scale invariance property and without making the assumption that the image sequences contain only face images.

The most accurate, but least user-friendly technology uses physical attachment to the front of the eye. A non-slipping contact lens is ground to fit precisely over the corneal bulge. Another popular common technologhy is based on non-contacting, special equipment aided vision techniques such as illuminating the eye with a barely-visible infrared light source. More detailed summary of the eye tracking methods is discussed in Chapter 3. These methods are obviously practical only for laboratory studies, as they are very awkward, uncomfortable for practical approaches. In this thesis a more practical real-time approach for simultaneously tracking and feature extraction of individual eyes is implemented using a video camera based vision technique without using the special equipment given above.

The eye tracker sits in a several meters range to the camera and head motion is restricted only to the extent necessary to keep the face, eye region and pupil of eye within view of the camera. The eye tracker provides data about the location of the face and eye regions, nature of human eye movements, which can be used to design effective interaction techniques. The *x* and *y* coordinates data of both eyeball areas, outline of the visible sclera and pupil area are detected using image processing techniques. Also the position of the pupil with respect to eyeball region is measured for *visual line of gaze* information.

The developed technique is aimed to be a fast and easy to operate real-time method, thus, suitable for ordinary user settings outside the laboratory environment although it is not thought to be as accurate as equipment based techniques given above.

1.2 Approach

In this thesis, several approaches are combined for developing an eye tracker system. As our approach does not make the assumption that the image sequences contain only face images; the study has to start with dealing with the problem of face detection in video sequences.

The starting problem in most of the cases of facial feature extraction, in this study eyes are the features of concern, is face detection problem, which deals with locating faces in images. Numerous techniques have been developed to detect faces in images. The starting point for any face detection technique, is detecting faces in a single image. After developing an algorithm which works succesfully in still images, the video streams taken from a camera or other source, can be evaluated. In our case Continuously Adaptive Mean Shift Algorithm (CAMSHIFT) is used for face detection.

After detection of the face region, an adaptive Principal Component Analysis method for scale invariant detection is developed to locate the eye regions. The method aims multi scale eye detection using eigeneyes with the particular information gathered during face detection step.

Active contours (Snakes), which can be considered as generic deformable templates based on parametric curves, are used for extracting the contours of eye features such as outline of the visible eyeball area and boundary between the sclera (white portion of the front eye) and iris (colored portion).

The three milestones of the thesis can be outlined as;

- Scale invariant face detection in whole image
- Eye detection in the face region
- Extraction of visible eye features such as the eye region and pupil

Raw algorithms without making image preprocessings do not give accurate results so developed algorithms must be strongly supported wih basic image preprocessing techniques. The developed software is tested using a setup with different preprocessing parameters during each step and different results have been obtained.

1.3 Road Map

The outline of the thesis is organized as follows;

This 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 chapter of the thesis report.

Chapter 2 provides necessary theoretical background that is necessary to understand this thesis and gives information about general fundamentals such as human eye structure, Principle Component Analysis and low-level image processing operations such as color spaces, thresholding/adaptive thresholding, edge detection. Then, active contours (snakes) are discussed as an important approach related to object shape description.

Chapter 3 presents some of the important previous studies on object recognition and tracking that we used during this thesis; first of all the chapter gives a brief literature survey on eye detection and several techniques for eye detection in images are

discussed in brief detail. After that, two previous studies that form the basis of the developed technique to track human eyes in this thesis; Continuously Adaptive MeanShift algortihm and EigenFace for Recognition are explained in detail.

Chapter 4 provides detailed information explaining the design and implementation of the developed eye tracking and eye feature extraction system, which tracks faces, eyes and extracts position and detailed shape information of both eyes. A flow chart showing all the phases of the whole system is also given in this chapter. Each step is explained step-by-step in detail explaining the algortihms used and modifications made to improve the accuracy.

Chapter 5 gives some outputs of the developed system and discusses the results under the effects of different parameters used during the running of the system.

Finally in Chapter 6 the study is concluded by summarizing the overall system and results obtained during the tests evaluated. Ideas that can be used for future works are also proposed in this section.

CHAPTER 2

GENERAL CONCEPTS AND IMAGE PROCESSING BACKGROUND

Many solutions to the computer vision problems involve image manipulation and image understanding. So in this chapter we will briefly cover some of the general techniques that are used to accomplish image understanding and some special concepts especially those that apply to the particular problems presented in this thesis.

2.1 Structure of Human Eye

The eye has been referred to as the most complex organ in the human body. Briefly; the very back of the eye is lined with a layer called the retina which acts very much like the film of the camera. The retina is a membrane containing photoreceptor nerve cells that lines the inside back wall of the eye. The photoreceptor nerve cells of the retina change the light rays entering through the pupil into electrical impulses and transmit them through the optic nerve to the visual part of brain where an image is perceived.

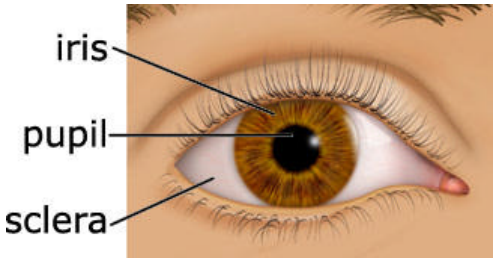

Figure 2.1 Frontal view of human eye

The position of eye over the head decides the field of vision. There are two types of fields of vision; the field of view of an individual eye, and overlapped portion of field of view (binocular field). When the eyes are on either side of the head, view is almost panoramic, but there is a loss of stereoscopic depth perception. Humans have a total field of view between 160 to 208 degrees, about 140 degrees or so, for each eye and by regular field of 120 ? 180 degrees where as a dog has a total field of view of about 280 degrees with 180 ? 190 degrees for each eye and 90 degrees of binocular field.

The most important property of our eye movement system is that it can move the eye from one point of gaze to a new location of gaze very rapidly. These *saccadic* eye movements are among the fastest movements the body can make; the eyes can rotate at over 500 deg/sec, and over one hundred thousand of these *saccades* are done during the day. These rapid eye movements are accomplished by a set of six muscles attached to the outside of each eye. They are arranged in three pairs of agonist-antagonist pairs; one pair rotates the eye horizontally (left - right), the second rotates the eye vertically (up - down), the third allows 'cyclotorsion,' or rotation about the line of sight.

Eye tracking system developed in this thesis is interested in indicating the position of iris and pupil area of eye during

- saccadic eye movements while face is stationary
- eyes are stationary but the face is moving
- face is moving and eyes are making saccadic movements

in video images which do not only contain human face images with scale and rotation invarience property.

2.2 Principal Components Analysis

Principal Component Analysis is a useful statistical technique that has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension. To understand the process of Principal Component Analysis, one should know the basic mathematical statistics concepts given in section 2.2.1;

2.2.1 Basic mathematical statistics concepts

Mean : Mean is the average value of a given data set and is symbolized as Y where Y is the data set.

$$\hat{Y} = \frac{\sum_{i=1}^{n} Y_i}{n} \tag{2.1}$$

The mean does not give a lot of information about the data, except the middle point. Two data sets [1 2 3] and [-10 6 +10] have the same mean value but they are quite different.

Standard Deviation : The standard deviation of a data set is a measure of how spread out the data is and calculated using the formula given below;

$$d = \sqrt{\frac{\sum_{i=1}^{n} (Y_i - \hat{Y})^2}{(n-1)}} \tag{2.2}$$

For example for the data sets given above the dataset [1 2 3] has a standard deviation of 1, whereas the data set [-10 6 10] has a standard deviation of 10.58 showing that the second data set more spreaded out than the first data set.

Variance : Variance is also another measure of how spread the data is and simply it is simply square of standard deviation.

$$s = \frac{\sum_{i=1}^{n} (Y_i - \hat{Y})^2}{(n-1)} \; or \; s = d^2 \tag{2.3}$$

Covariance : Standard deviation and variance are 1-dimensional measures about the data sets. However many data sets are multi-dimensional and it is needed to measure the relationship between the dimensions such as the relationship between the hours studied and grades got for a class of students. Covariance is a useful measure to find out how much the dimensions vary from each other with respect to each other.

$$\mathrm{cov}(x, y) = \frac{\sum_{i=1}^{n} (X_i - X_{mean})(Y_i - Y_{mean})}{(n-1)} \tag{2.4}$$

As it can be seen from the above formula variance is a special form of covariance; *cov(X,X)* where the similarity is measured according to the itself. The result of the

covariance doesn't give a meaningful idea except the sign; A positive sign indicates that the two dimensions increase/decrease together, a negative sign indicates that as one dimension increases the other dimension decreases and a zero means that the two dimensions are independent of each other.

Covariance Matrix : Covariance is always measured between two 2 dimensions. For multi-dimensional data sets there is more than one covariance value so a better way to represent all covariance values is to calculate all of the possible covariance values and place them in matrix. For an *n-dimensional* data set covariance matrix is a *nxn* matrix. For example; for a 3-dimensional data set covariance matrix is represented as follows;

$$C = \begin{bmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{bmatrix} \tag{2.5}$$

Eigenvectors and Eigenvalues : For a transformation matrix *t*, a vector e_i that provides the following property is called an Eigenvector of matrix *t*.

$$\begin{bmatrix} t_{11} & .. & t_{1n} \\ .. & .. & .. \\ t_{n1} & .. & t_{nn} \end{bmatrix} \times \begin{bmatrix} e_{i1} \\ .. \\ e_{in} \end{bmatrix} = l \begin{bmatrix} e_{i1} \\ .. \\ e_{in} \end{bmatrix} \tag{2.6}$$

Some properties of Eigenvectors are;

- Eigenvectors can only be found for *square* matrices.
- All the Eigenvectors of a matrix are linearly independent of each other.
- Eigenvectors are usually represented in unit length form.
- There are n possible different Eigenvectors for a nxn matrix.

Eigenvalue, ?, is the amount by which the Eigenvector is scaled after multiplication by the transformation matrix. Every Eigenvector is associated with an Eigenvalue so Eigenvectors and Eigenvalues always come in pairs.

2.2.2 Principal Components Analysis Method

As mentioned above PCA is a way of identifying patterns in data i.e expressing the data in such a way to highlight their similarities and differences. For datas of high dimensions, such as images, PCA is a strong tool for analysing the data. The steps of PCA are described below;

1) Get some data
2) Subtract the mean
3) Calculate the covariance matrix *C*
4) Calculate the Eigenvectors & Eigenvalues of the covariance matrix *C*
5) Choose Eigenvectors to use and form the transformation matrix *U*
6) Derive the new data set

Steps 1-4 are explained above. Reduced dimensionality is obtained in steps 5 and 6; to reduce the dimensionality of the data set for further processing some Eigenvectors that were obtained at the end of step 4 are ignored. Ignoring some of the Eigenvectors results in losing some information in final data set but if the Eigenvectors with low Eigenvalues are ignored then much information is not lost.

In step 5 Eigenvectors with high Eigenvalues (components) are chosen to form the space transformation *U* which is

$$U = \lfloor e_1 \quad e_2 \quad .. \quad e_{p-1} \quad e_p \rfloor \tag{2.7}$$

In step 6 final data is obtained by

$$FinalData = U^T \times RawDataAdjust^T \tag{2.8}$$

where RawDataAdjust is the matrix whose colums are formed by each data set used to calculate the space transformation matrix.

As a result; a data set having n of m-dimensional samples form a covariance matrix of size *nxn*. The matrix *U* with all the Eigenvectors has the size *nxn* but to reduce the dimensionality assume that only p Eigenvectors are chosen where p=m than our transformation matrix has dimenson of *nxp*. RawDataAdjust has dimension of *mxn*.

In step 6 we obtain FinalData of dimension *pxm* after the multiplication of two matrixes of dimensions *pxn* and *nxm*.

The following transformation is used to get the original data back;

$$OriginaData = (U \times FinalData)^T \times OriginalMean \quad (2.9)$$

Here *U* is of dimension *nxp*, FinalData has dimension of *pxm* and OriginalMean which is the mean calculated for each data set and has dimension of *mxn*. As a result OriginalData will have the dimension of *mxn*.

To visualize the Principle Components Analysis steps two data sets are chosen as;

Table 2.1 Data Sets

Data Set#1	Data Set#2
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

If only one Eigenvector is chosen to form the transformation matrix than the reconstructed data will have the form given on the left part of the Figure 2.2.

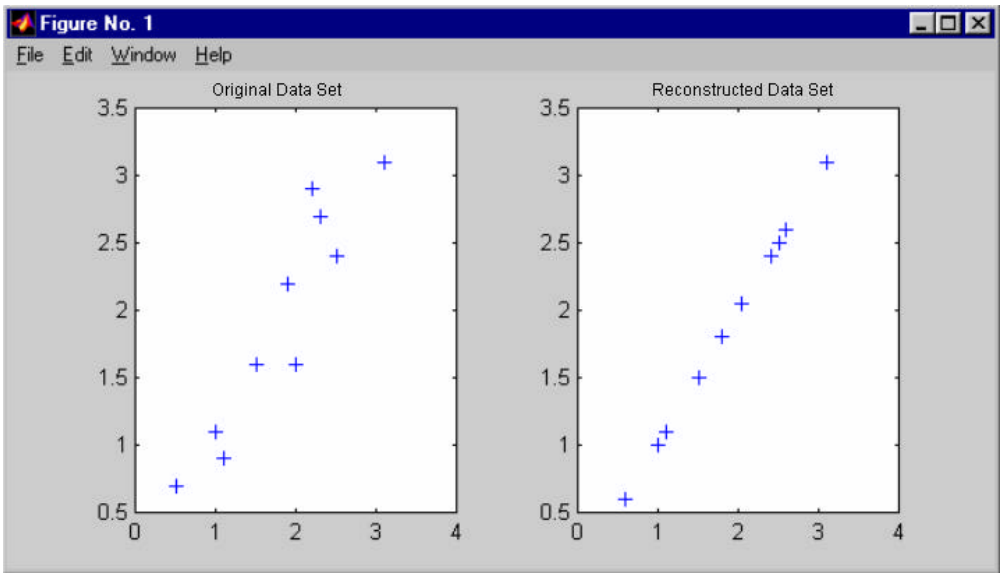

Figure 2.2 Effect of reducing dimensions using PCA

2.2.3 Application to Computer Vision Area

Using Principal Components Analysis method we can express large amount of data with lower dimensional patterns that describe the relationship between the data. It can be used in various areas such as pattern recognition and image compression. In this thesis Principal Components Analysis method is used for eye detection in face areas.

2.3 Color Spaces

Color is accepted to be made of two components: *Chrominance* and *Luminance*. The Chrominance value of an object identifies the coloring property of that object whereas luminance is a property of the environment where the object is. For a color image we can talk about both the chrominance and limunance values.

A color space is a mathematical model describing the way color can be represented as numbers, typically as three or four components. First step in this thesis for eye detection was to detect human faces in images and faces are detected using the skin color property of the faces. Detecting an object using its color property may cause problems with changing ambient light as this change also effects the apparent color of the object. Color images are usually represented in RGB color space that is; the

red, green and blue components reside in three seperate intensity matrixes, each having the same dimensions as the original RGB image and the intensities of corresponding pixels from each matrix combine to create the actual pixel color at a given location. But variation in ambient light also changes highly the RGB values of image pixels. So different color spaces which are more robust to varying ambient light shall be used.

In this thesis work; HSV color space is used for skin detection and for eye detection. Also different components of several color spaces are used during eigenvalue decomposition and their effects are discussed. So before proceding to the next chapters that describes the algorithm for eye detection, a summary information about different color spaces is provided in the following sections.

2.3.1 GRAY Color Space

A grayscale image is simply one in which the only colors are shades of gray. The reason for converting color images to grayscale is that less information needs to be provided for each pixel. In fact a gray color is one in which the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, whereas three intensities needed to specify each pixel in a full color image. One common conversion formula from RGB to Gray is;

$$[Gray] = [0.212 \quad 0.715 \quad 0.072] \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.10}$$

Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive graylevels is significantly better than the graylevel resolving power of the human eye.

Grayscale images are still very common in todays technology and image processing algortihms; they are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process color images. For example; for object

detection/recognition, such as face, using eigenvalue decomposition images must be represented as single 2-D matrixes so usually gray scale images are used however in this thesis we observed the effect of using single components of color spaces which in effect a kind of gray scale with different conversion constants than given in formula 2.10.

2.3.2 HSV Color Space

The HSV coordinate system is cylindrical, and the model is defined as a cone; Vertical position defines brightness, angular position - hue, and radial position - saturation. HUE can range between 0 and 360 but in some applications it is normalized to 0-100%, saturation range is 0 to 100%, and specifies relative position from the vertical axis to the side of the cone. Value is the brightness of the color and its range is 0 to 100%.

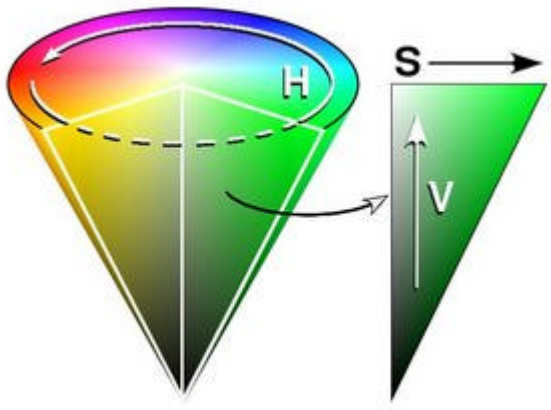

Figure 2.3 Visualization of HSV color space

Artists usually prefer to use HSV color model alternative to the classical color spaces such as RGB because it is similar to the way humans tend to recognize color. For example; we might want to change the color of a bright yellow car moving down a road to blue, but we want to leave the rest of the scene, including highlights and shadows on the car, unaffected. This would be a difficult task in RGB, but it is relatively simple in HSV. Because the yellow pixels of the car have a specific range

of hue, regardless of intensity or saturation, those pixels can be isolated easily and their hue component modified, thus giving a different-colored car. Since most of the digital image processing systems operate on RGB images, the operation described above would be performed in three steps;

1) Convert the original RGB image to HSV
2) Modify the hue value
3) Finally, convert the modified image back to RGB

Many face detection algorithms implemented so far is based on detection of skin color first. A common misconception is that different color models are required for different human races. In fact this is not true because almost all humans have nearly the same hue [4]. But human skin color almost may cover all the S values for a limited range of hue value as discussed in [3]. As a result hue value of images can be used in face detection algorithms.

2.3.3 XYZ Color Space

The XYZ space allows colors to be expressed as a mixture of the three tristimulus values X, Y, and Z. The term *tristimulus* comes from the fact that color perception results from the retina of the eye responding to three types of stimuli. After experimentation, the CIE set up a hypothetical set of primaries, XYZ, that correspond to the way the eye's retina behaves.

The CIE defined the primaries so that all visible light maps into a positive mixture of X, Y, and Z, and so that Y correlates approximately to the apparent lightness of a color. Generally, the mixtures of X, Y, and Z components used to describe a color are expressed as percentages ranging from 0 percent up to, in some cases, just over 100 percent.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412411 & 0.357585 & 0.180454 \\ 0.212649 & 0.715169 & 0.072182 \\ 0.019332 & 0.119195 & 0.950390 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.11}$$

During the experiments we have performed, we have observed that using Z component of XYZ color space for a face image, eigenvalue decomposition for eye

detection, gives better results instead of its GRAY scale version. Further details are discussed in later chapters.

2.3.4 Lab Color Space

Lab, sometimes referred as CIELAB, is the most complete color model used conventionally to describe all the colors visible to the human eye. Three parameters in the model are; *L* is the luminancy, *a* is the position between red and green, *b* is the position between yellow and blue. Conversion formula from RGB color space to Lab color space is;

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.433910 & 0.376220 & 0.189860 \\ 0.212649 & 0.715169 & 0.072182 \\ 0.017756 & 0.109478 & 0.872915 \end{bmatrix} \times \begin{bmatrix} R/255 \\ G/255 \\ B/255 \end{bmatrix}$$

$$L = 116 \times Y^{1/3} \ for \ Y > 0.008856$$

$$L = 903.3 \times Y \ for \ Y <= 0.008856$$

$$a = 500 \times (f(x) - f(y))$$

$$b = 200 \times (f(y) - f(z))$$

where

$$f(t) = t^{1/3} \ for \ t > 0.008856$$

$$f(t) = 7.787 \times t + 16/116 \ for \ t <= 0.08856 \qquad (2.12)$$

2.3.5 YCbCr Color Space

YCbCr color space is based on luminance and chrominance. It is derived from RGB representation using

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & 0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (2.13)$$

where Y is the luminance or brigthness component and Cb and Cr are components of blue and red respectively.

2.4 Image Thresholding

In many vision applications, it is useful to be able to separate out the regions of the image corresponding to objects in which we are interested, from the regions of the image that correspond to background. Thresholding often provides an easy and

convenient way to perform this segmentation on the basis of the different intensities in the foreground and background regions of an image. Generally thresholding can be divided into two categories; Global Thresholding and Adaptive Thresholding.

2.4.1 Global Thresholding

Global thresholding is a method to convert a gray scale image into a binary image, which is a special type of gray scale having only two pixel values; black and white, using threshold value(s) for the whole image. Single or multiple threshold levels can be determined for the image to be segmented; for single value thresholding each pixel in the image is compared with this threshold and if the pixel's intensity is higher than the threshold, the pixel is set to white (or black) in the output similarly if it is less than the threshold, it is set to black (or white). For multiple thresholding values there are bands of intensities to be set to white while the image regions out of these bands are set to black. General global thresholding with multiple levels scheme may be represented as;

$$O(i,j)=\begin{cases}0 \; if \;\; I(i,j)\in Z \\ 1 \quad otherwise\end{cases} \tag{2.14}$$

where Z is a set of intensity values.

Figure 2.4 Original image and global thresholding it with 80

2.4.2 Adaptive Thresholding

Global thresholding uses a fixed threshold for all pixels in the image and therefore works only if the intensity histogram of the input image contains neatly separated peaks corresponding to the desired subject(s) and background(s). Hence, it cannot deal with images containing, for example, a strong illumination gradient. Local

adaptive thresholding, on the other hand, selects an individual threshold for each pixel based on the range of intensity values in its local neighborhood. This allows for thresholding of an image whose global intensity histogram doesn't contain distinctive peaks and as a result can overcome the problem of changing lighting conditions in the image.

There are different approaches for calculating the local threshold value;

$$Threshold = mean \;\; or \;\; Threshold = median \;\; or \;\; Threshold = \frac{\min + \max}{2} \tag{2.15}$$

The critical point in adaptive thresholding is choosing the right size of neighborhood for the pixel. The size of the neighborhood has to be large enough to cover sufficient foreground and background pixels, otherwise a poor threshold is chosen. On the other hand, choosing regions which are too large can violate the assumption of approximately uniform illumination. Also in some situations adaptive threshold value can be improved by subracting a constant value from the mean i.e *mean-C*. Adaptive thresholding considering the intensity histogram values is out of the scope our work.

Figure 2.5 Original image and adaptive thresholding it with a 7x7 window, C=4

In this thesis we have used both global and adaptive thresholding to filter the input to the other operations such as circle detection for pupil detection, edge detection or outline detection using active contours and we have observed that without thresholding, these operations give wrong results.

2.5 Active Contour Models (Snakes)

Active Contour Models are proposed in [5] and since then have been successfully applied in a variety of problems in computer vision and image analysis, such as edge and subjective contours detection, motion tracking and segmentation. An active contour is an energy minimizing spline. The energy function of a snake involves terms to attract the snake to the desired features on the image. Snakes deform continuously for minimizing their energy function, so they can be considered as a special case of a general technique for matching a deformable mode to an image [6].

Simple snake structure consists of a set of control points which are connected by straight lines and form a close loop. Each control point has a position given by *(X,Y)* coordinates in the image. Adjustments to the snake are made by moving these control points according to snakes energy function. The energy function for a snake is composed of two parts; the *internal energy* and *external energy* [7].

$$E_{snake} = E_{\text{int}\,ernal} + E_{external} \tag{2.16}$$

Internal Energy **:** The internal energy is the part that depends on intrinsic properties of the snake such as its length or curvature.

External Energy **:** The external energy depends on factors such as image sturcture, human operations and initialization procedures.

2.5.1 Internal Energy

The internal energy of a snake may explained simply as follows;

$$E_{\text{int}\,ernal\,energy} = E_{elastic\,energy} + E_{bending\,energy} \tag{2.17}$$

Elastic Energy **:** Elastic energy is the energy that increases with the total length of the snake. The snake will be have minimum energy if the control points are equally spaced rather than being bunched up. Also we can increase the energy by adding a penalty for two control points that are further apart than the average. So a convenient form for an energy function with this property is the sum of the squares of the distances between adjasent control points. So the energy can be written as;

$$E_{Elastic} = K_1 \times \sum_{i=1}^{N} \left| P_i - P_{i+1} \right|^2 \tag{2.18}$$

where N is the number of points, K_1 isan arbitrary constant and P_i is the i^{th} control point.

So how will be the snake minimization due to elastic energy? For point P_i that has coordinates X_i, Y_i the elastic force on the point can be represented as;

$$\begin{aligned} F_{elasticXi} &= 2 \times K_1 \times \left((x_{i-1} - x_i) + (x_{i+1} - x_i) \right) \\ F_{elasticYi} &= 2 \times K_1 \times \left((y_{i-1} - y_i) + (y_{i+1} - y_i) \right) \end{aligned} \tag{2.19}$$

The total force will be the combination of the *Fx* and *Fy* components of the force. And once we have determined the force acting on a point we can also determine its moving strategy. At each time step we simply move each control point by an amount directly propotional to the force acting on it. Therefore the updating formula may be;

$$X_i + K_2 \times F_{elasticX_i} = X_i' \tag{2.20}$$

where X_i' is the new X coordinate for control point Pi. After calculation of the new coordinates for all points, control point coordinates can be updated i.e. old value of X_i is used to calculate the shift in X_{i+1}. Here K_2 is another constant defining how fast the point can move. So; elastic force has an argument $2xK_1xK_2$ to control how much the point will move in each iteration. Usually multiplication of these constants is called *alpha* which should be normally much less than 1.

It is obvious that such a force applied to every control point will pull the snake inwards and will pull the control points into line with one another, adjusting the position by smoothing the snake.

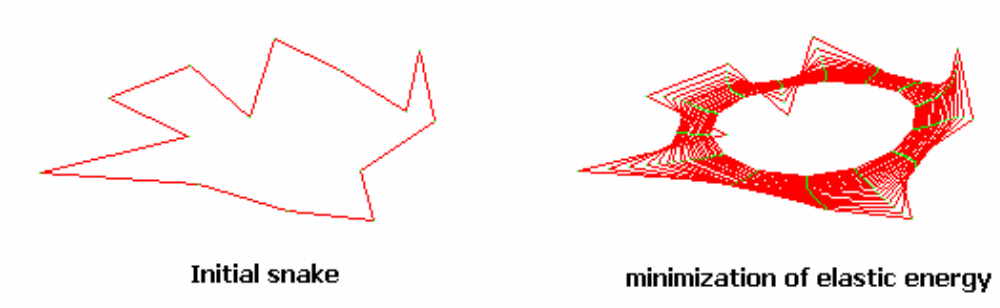

Figure 2.6 Minimization of elastic energy

Bending Energy **:** It is sometimes useful for a snake to behave like a thin metal strip rather than an elastic band. That is, it should try to be a smooth curve or straight line, but should not contract. This can be done by defining the bending energy function.

The energy function in this case is the sum of the squared curvatures of the snake measured at the control points; the sharper the angle made by the nearest neighbours at the control point, the bigger the curvature. In practice, a simple approximation to the curvature is used. This energy translates into forces at each control point that depend on its 4 nearest neighbours, not just the nearest two as for the elastic force, and which tend to straighten out the snake. Also as in elastic forces a constant parameter *beta* is used to control the size of the bending energy which is typically chosen as 0.05.

As it can be seen in the figure below; the snake shrinks slightly but the dominant effect is smoothing rather than shrinkage.

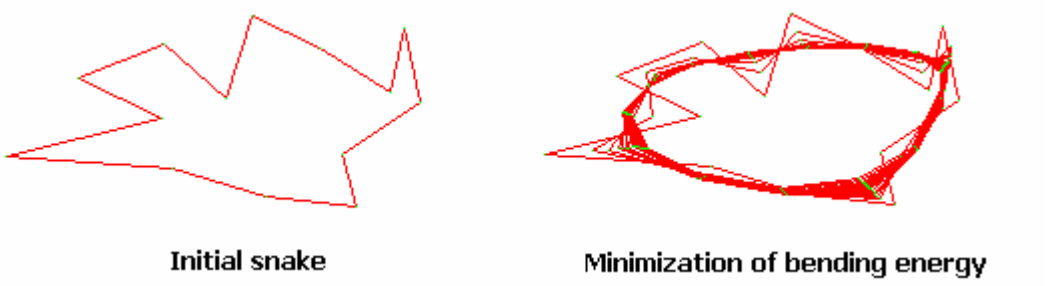

Figure 2.7 Minimization of bending energy

2.5.2 External Energy

External energy terms reflect the relation of snake with image or constraints coming from human operators such as external forces or initialization procedures. As the main purpose our work is to build an automatic detection system, we are not interested in external forces by human factors.

Image Gradient Energy **:** This kind of energy produces a force that can be employed to drive snakes towards features of interest such as bright structures or edges. A very simple energy function for moving the snake to bright structures may be minus the sum of the grey levels of the pixels the snake is on top of. Reducing this energy means moving the control points towards brigther parts of the image.

$$E_{image} = -K_3 \times \sum_{i=1}^{N} image(x_i, y_i) \tag{2.21}$$

where $image(x_i,y_i)$ is the grey level of the control point pixel at (x,y) coordinate in the image. And similarly the force present on the control point can be expressed as;

$$F_{image\_Xi} = \frac{K_3}{2} \times (image(x_i + 1, y_i) - image(x_i - 1, y_i))$$
$$F_{image\_Yi} = \frac{K_3}{2} \times (image(x_i, y_i + 1) - image(x_i, y_i - 1)) \tag{2.22}$$

That is, if the pixel in the direction of increasing X is brighter than the pixel in the direction of decreasing X, then the control point is pulled in the positive X direction, and likewise for Y. In short, the force on the control point is in the direction of the grey-level gradient. The rate of the motion due to external force is given by a control constant called *gamma*.

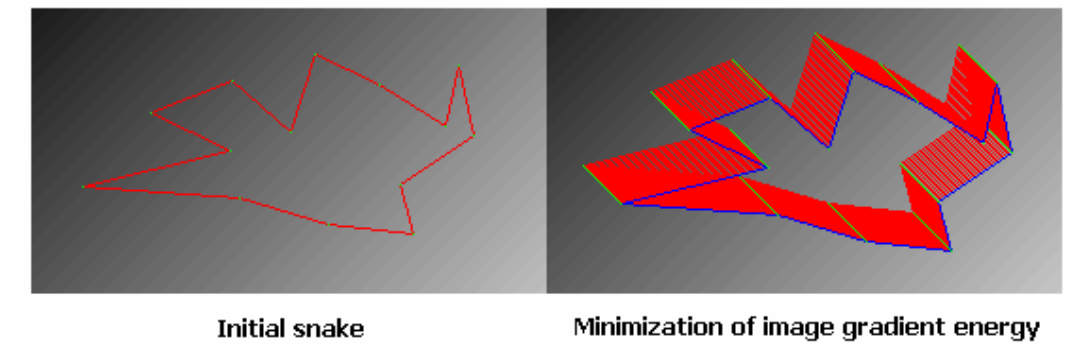

Figure 2.8 Minimization of image gradient energy

Using the image structure that is within the 1 pixel of the control points may result in erroneous cases. To avoid the effects of very local structure and give the snake more

of a chance to find sructures that are not right next to it, more sophisticated estimate of the grey-level gradient, which averages over more pixels may be used.

A snake being used for image analysis attempts to minimize its total energy, which is the sum of the internal and external energies given above. When energies are added their forces add too. In this thesis work snakes were used to outline the visible eyeball regions in the eye images which have been found by a different method. Infact they were used for edge detection which is a special kind of gradient energy minimization where the energy decreases if the snake is lying on a region of high image gradient. Also to outline the eyeball region with an ellipse, coordinates of the control points on the snake with minimized energies were used for ellipse fitting to get a better indication.

Different approaches have been proposed for snake energy minimization problem so far. Our approach is based on the method proposed by [5]. A more efficient method has been proposed in [8], which uses greedy algorithm. But generally the methods employed for the minimization problem of snake energy, suffer from oscillations. A number of methods also have been proposed to solve this problem. For Example method proposed by [9], starts minimizing snake energy with highly smoothed version of the image and them gradually reduces the image blurring making the snake fit more accurately on the features of interest.

CHAPTER 3

PREVIOUS WORKS ON FACE AND EYE TRACKING

3.1 Introduction

Eyes are the most important features of human face. So effective usage of eye movements as a communication technique in user-to-computer interfaces can find place in various application areas. As the first step of designing interaction techniques using natural eye movements, our aim is to track eyes and recognize its features such as pupil movements, location of pupil with respect to outline of the visible eyeball area in images which do not only contain face images. Our cost effective, robust, fast and real-time tracking solution to this problem depends on simple computer vision techniques with easy to implement setup and has the property of tracking with scale and rotation invariance properties.

So far there has been a lot of work on eye detection area such as; eye pupil movement detection, eye feature extraction, eye state detection, eye gaze detection using different techniques both in still images and in video sequences for real-time applications. This chapter will first present a brief literature survey on applications about eye movement as eyeball movements are the most valuable features of the human eye and then important studies related to our approach will be detailed; face detection is a prerequisite in our approach so different face detection techniques will be presented, previous work on detection of eye features in face images will be detailed. Finally two main topics; “Continuously Adaptive Meanshift Algorithm” and “Eigenfaces for Detection and Recognition” will be described in details as they form the main idea of “Adaptive Eigeneye Method for Eye Detection” which is proposed in this thesis work.

3.2 Methods For Measuring Eye Movements

Eye tracking is a technique used in cognitive science, psychology, human-computer interaction, advertising, medical research, and other areas. One simple method is; a camera focused on one or both eyes recording their movements as the viewer looks at some kind of stimulus and then these records are processed by an image processing software. Most modern eye-trackers use infrared beams to create a corneal reflection, from which the angle of movement can be calculated. However eye tracking setups vary greatly; some are head-mounted, some require the head to be stable (for example, with a chin rest), and some automatically track the head as well. Most of them use a frequency of at least 30Hz in order to capture the details of the very rapid eye movements. Using necessary equipment which will give a signal of some sort which is proportional to the position of the eye in the orbit, and changes when the position of the eye changes in some direct way is an accurate way of measuring eye position but it is unpractical and expensive compared to the image based methods. The following describes various methods that are currently in use;

3.2.1 Electrooculography

Because there is a permanent potential difference between the cornea and the fundus of approximately 1mV, small voltages can be recorded from the region around the eyes which vary as the eye position varies. By carefully placing electrodes it is possible to separately record horizontal and vertical movements. However, the signal can change when there is no eye movement. It is dependent on the state of dark adaption (used clinically to calculate the Arden ratio as a measure of retinal health), and is affected by metabolic changes in the eye. It is prone to drift and giving spurious signals, the state of the contact between the electrodes and the skin produces and other source of variability. There have been reports that the velocity of the eye as it moves may itself contribute an extra component to the EOG. It is not a reliable method for quantitative measurement, particularly of medium and large saccades. However, it is a cheap, easy and non-invasive method of recording large eye movements, and is still frequently used by clinicians.

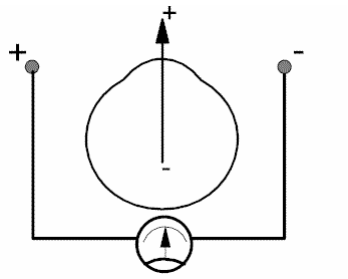

Figure 3.1 Electrooculography method

3.2.2 Infra-Red Oculography

If a fixed light source is directed at the eye, the amount of light reflected back to a fixed detector will vary with the eye's position. This principle has been exploited in a number of commercially available eye trackers. Infra-red light is used as this is "invisible" to the eye, and doesn't serve as a distraction to the subject. As infra-red detectors are not influenced to any great extent by other light sources, the ambient lighting level does not affect measurements. Spatial resolution (the size of the smallest movement that can reliably be detected) is good for this technique, it is of the order of 0.1° , and temporal resolutions of 1ms can be achieved. It is better for measuring horizontal than vertical eye movements. Blinks can be a problem, as not only do the lids cover the surface of the eye, but the eye retracts slightly, altering the amount of light reflected for a short time after the blink.

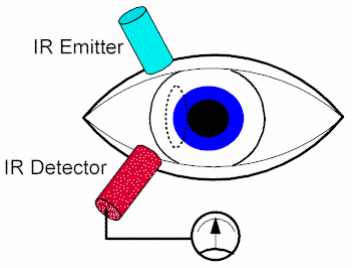

Figure 3.2 Infra-Red Oculography method

3.2.3 Scleral search coils

When a coil of wire moves in a magnetic field, the field induces a voltage in the coil. If the coil is attached to the eye, then a signal of eye position will be produced. In order to measure human eye movements, small coils of wire are embedded in a modified contact lens or anulus. This is inserted into the eye after local anaesthetic has been introduced. A wire from the coil leaves the eye at the temporal canthus. The field is generated by two field coils placed either side of the head. This allows horizontal eye movement to be recorded. If it is necessary to also monitor vertical eye movements, then a second set of field coils, usually set orthogonally to the first set, it used. The two signals (one for horizontal, one for vertical eye movement) generated in the eye coil can then be disentangled using appropriate electronics. If the eye coil is of an appropriate design, then torsional movements can also be recorded. In experiments on eye movements in animals, the eye coils are frequently implanted surgically. The advantage of this method is that it has a very high temporal and spatial resolution allowing even the smalled types of eye movements (eg microsaccades) to be studied. It's disadvantage is that it is an invasive method, requiring something to be placed into the eye. This method is rarely used clinically, but is an invaluable research tool.

3.2.4 Image based methods

With the development of video and image analysis technology, various methods of automatically extracting the eye position from images of the eye have been developed. Tracking the relative movements of these images gives an eye position signal. More commonly a video image is combined with computer software to calculate the position of the pupil and its centre. This allows vertical and horizontal eye movements to be measured. However, image based methods tend to have temporal resolutions lower than achieved with IR techniques. Spatial resolution can also be limited. As technology improves, the resolutions these systems can supply will also improve. Image based methods are discussed in detail in section 3.3

3.3 Face Detection In Computer Vision Area

In survey [10] two terms face detection and face localization are explained as follows;

Face Detection **:** Given an arbitrary image, the goal of face detection is to determine whether or not there are any faces in the image and if present, return the image location and extent of each image.

Face Localization **:** It aims to determine the image position of a single face; this is a simplified detection problem with the assumption that an input image contains only one face [11], [12].

We are interested in eye detection of a single person in video seqeunces. So; although the term *detection* is used for finding the locations of multiple objects in images, in this document we will use the term *face detection* for finding the face of the single person in image sequences.

First problem for our automatic eye tracking system is detection of human face in image sequences. After detection of face in the image, both eyes can be searched and features of them can be extracted. As our main purpose of face detection is finding a search area for eye detection, we need a fast face detection method independendent of the structural components of face such as beard, mustache. Although simple methods could be chosen without making detailed research about previous work on face detection we prefered to search surveys and categorize face detection methods in the literature.

Human vision system can easily detect and recognize faces in images. The performance of the human vision system is so high that it can detect not only a single face but multiple faces in the same scene having different pose, facial expression, lightening conditions, scales, orientation etc. Also faces do not have to be complete that is; a partial view of a face is enough for humans to detect them in images. Unfortunately in todays computer vision technology no system can achieve that performance. Their operations depend on controlled conditions.

There are about 150 different techniques for face detection in images although they share some common methods through their ways. A detailed survey about various face detection methods is given in Yang's survey [10] and classified into four categories. Also in [9], another survey about face detection methods is given. Using these two surveys, details about previous face detection methods are explained below;

3.3.1 Knowledge-Based Methods

These rule-based methods encode human knowledge of what constitutes a typical face. Usually, the rules capture the relationships between the facial features. These methods are designed mainly for face localization.

In this approach, methods use simple rules to describe features of a face such as; a face often appears in an image with two eyes that are symetric to each other, a nose and a mouth. The relationships between features can be respresented by their relative distances and positions.

Problem with this approach is; it is difficult to translate human knowledge into well-defined rules. If these rules are strict than they may fail to detect faces that do not pass all the rules. But on the other hand if the rules are too general than there may be many false detections.

One popular work about this approach was performed by Yang and Huang [13]. They used a hierarchical knowledge-based method to detect faces. Their systems 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 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 hierarcy of images is created by averaging and subsampling.

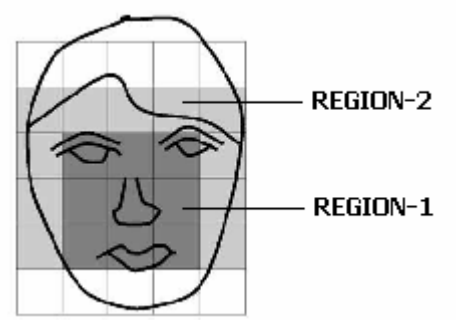

Figure 3.3 A typical face used in knowledge-based top-down method [13]

Examples of the coded rules, which are based on the characteristics of human face, used to locate the face candidates in the lowest resolution include: “the center part of the face (Region-1) has a four cells with basically uniform intensity”, “the upper round part of a face (Region-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”. 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 1, 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.

Yangs method does not result in a high detection rate but later work [14] used the idea of multiresolution images and rules based searching for face detection in frontal views.

3.3.2 Feature-Invariant Approaches

These algorithms aim to find structural features that exist even when the pose, viewpoint, or lightening conditions vary, and then use these to locate faces. These methods are designed mainly for face localization.

The underlying assumption is based on the observation that human can easily detect faces and objects in different poses and lighting conditions 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 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. A common problem for feature-based methods is ambient conditions such as lighting, shadows which may corrupt facial features.

3.3.2.1 Facial Features

Sirohey proposed a method that uses Canny Edge Detector [15] to get an edge map [16]. Then it groups the edges so that only the ones on the face contour are preserved. An ellipse is fit to the boundary between head and background using head contour. Graf et al. developed a method to locate faces in gray scale images [17]. 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.

3.3.2.2 Skin Color

Human skin color is an effective feature in face detection. Several different color spaces have been utilized to label pixels as skin including RGB [18], normalized RGB [19], HSV [4], [20], [21], [22], YCbCr [23], YIQ [24], XYZ [25].

In this thesis we have used to track faces using skin color probability distribution in a given image. However detecting pixels in skin region alone is not sufficient to localize the face. For detecting the outline of the face boundary, algorithms such as proposed in [4] must be used. More detailed information will be given section 3.5.

3.3.3 Template Matching Methods

In template matching, a standard face pattern 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 can not 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.

Predefined template matching have been used early by Sakai et al [26]. They have used line segmented subtemplates for eyes, nose, mouth and face contour to model a face. After matching the face contour with candidate image regions other subtemplates were used to determine the existence of the face. Similar work was conducted by Craw et al. [27] which performed face contour matching after filtering the image with a sobel filter for edge detection. Deformable templates for facial feature extraction proposed by A. Yuille [28] was researhed for eye detection in face images for our work. Details about deformable templates will be given in section 3.4.

3.3.4 Appearance-Based Methods

In contrast to template matching, the models (or templates) are learned from a set of training images which should capture the representative variability of facial appearance. These learned models are then used for detection. These methods are designed mainly for face detection. 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.

Many apperarence-based methods can be understood 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 a face or nonface. Unfortunately because high dimensinality of *x*, implementation of Bayesian classification is infeasible. Another approach in appearance based methods is to find a discriminant function (i.e., decision surface, seperating hyperplane, threshold function) between face and nonface classes. Conventionally, image patterns are projected to a lower dimensional space and then a discriminant function is formed (usually based on distance metrics) for classification [29].

3.3.4.1 Eigenfaces

Eigenfaces approach [29] is a PCA (Principle Component Analysis) based approach for detecting faces. In this approach, faces are represented as points in a space called eigenspace. Face detection is performed by computing the distance of image windows to the space of faces. Assuming raw images as points in a high dimensional space (for an NxM image, the dimension of such a space is NxM) is impractical for two reasons; Firstly, working in very high dimensional spaces is computationally too expensive, and secondly, raw images contain statistically irrelevant data which degrades the performance of the system. Principal component analysis reduces the dimensionality of a feature space by restricting the attention to those directions along which the scatter is greatest [30]. Thus, principal component analysis is used for projecting the raw face images onto the eigenspace of a representative set of normalized face images, for eliminating redundant and irrelevant information.

Step by step explanation of how eigenvectors are used for face detection is given in details in section 3.6. In this thesis we have adapted the eigenfaces method for detection of eyes using *eigeneyes* in face images. Infact eigenface method is not vary suitable for real-time applications but as the object to be detected and search areas are small in size, good results can be obtained.

3.3.4.2 Hidden Markov Model

Intuitively, a face pattern can be divided into several regions such as 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 Figure 3.4 (a). During training and testing, an image scanned in some order (usually top to bottom) and an observation is taken as a block of pixels are represented by probabilistic transitions between states, as shown in Figure 3.4 (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 [31] showed that the states of the HMM that he trained corresponds to facial regions, as shown in figure 3.4 (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 for each rectangular pattern in the image is above a threshold, then a face is localized.

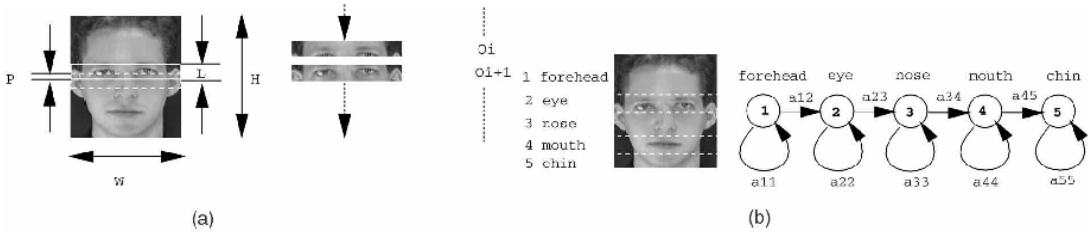

Figure 3.4 Hidden Markov Model for face localization. a) Observation vectors, b) Hidden states [31]

3.4 Eye Detection In Computer Vision Area

There has been much work in eye detection and tracking. First of all studies about eye can be classified in two main categories; 1) Detection of eyes 2) Detailed feature extraction of eye.

Detection of eye : Given an arbitrary face image, the goal of eye detection is to determine the location of the eyes. Simply in eye detection, the area where both eyes are located are found or two eyes individually localized. As a result of the process ususally eye areas are indicated by a rectangle.

Detailed feature extraction : On the other hand the goal of this category is to give detailed information such as the contour of the visible eyeball region, circular area formed by iris and pupil, location of pupil in the visible eye area, state of the eye (e.g., blink/not blink). This type of work is more difficult in computer vision area as detection or real-time tracking of small details are highly effected from varying ambient conditions and result may easily fail.

Although we have given details about many techniques of measuring eye movements in section 3.2 only a few of them are in the scope of computer vision area. The existing work for detailed eye feature extraction can be broadly classified into two categories: the active IR based approaches and traditional image based passive approaches.

3.4.1 IR Based Approaches

The first approach exploits the spectral properties of pupil under near IR illumination. Eye tracking is accomplished by tracking the bright (dark) pupils. There has been a lot of work using this technique and there are some commercial eye-tracking systems such as produced by ISCAN Incorporated, LC Technologies and Applied Science Laboratories (ASL). IR based eye detection is out of the scope our study.

3.4.2 Image Based Approaches

Kothari and Mitchell [32] use spatial and temporal information to detect the location of the eyes. Their process starts by selecting a pool of candidates using gradient

fields. The gradient over the iris/sclera boundary always point outward the center (dark pupil), thus by accumulating along these lines, the center of the iris can be estimated by selecting the bin with highest count. Heuristic rules and a large temporal support are used to filter erroneous pupil candidates.

The leading study on facial feature detection using deformable templates was presented in [33] and [34].

3.4.2.1 The Eye Template

Deformable templates propose solution to the limitations of classical template matching, by allowing deformation of the template geometry and providing relative invariance to lightening condition. A deformable template consists of three basic elements [28]:

1) Geometrical Model : This parametric geometrical model defines the geometry for the template and introduces constraints on the deformation of the geometry.
2) Imaging Model : The imaging model specifies how a deformable template of a specific geometry is related to specific intensity values in a given image.
3) Matching Algorithm : An algorithm using the geometrical and imaging measures of fitness to match the template to the image. It defines which optimization algortihm will be used for energy minimization.

Peak and valley potentials are used in [33] in the design of their eye template. The geometry of the eye template is specified by eleven parameters $g = (x_t, x_c, r, a, b, c, ?, p1, p2)$. x_t : the center of the whole template(x_t,y_t); x_c : the center of the iris, modelled as a circle (x_c, y_c); r : the radius of the iris; a, b, c : the parameters of the parabolas which bound the eye template; *?* : the orientation of the template; *p1* and *p2* : the parameters used to locate the position of the centers of the peaks in the left and right side of the iris.

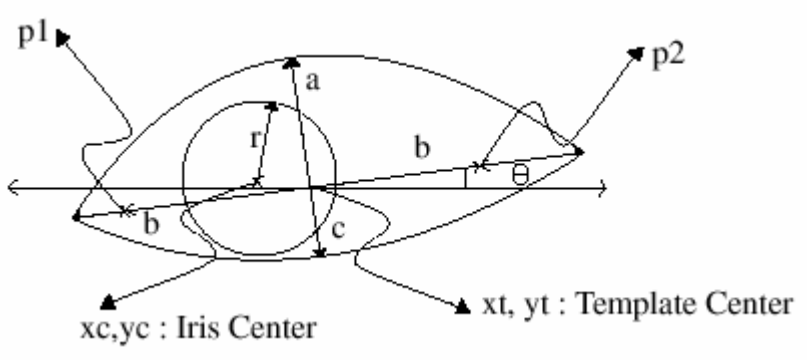

Figure 3.5 Eye template [33]

In order to specify the prior probabilities, an energy term, E_{prior}, is defined. This energy term imposes relations on the template parameters, such as the center of the eye is close to the center of the iris. The energy term used for specifying prior probabilities is given as follows:

$$Eprior = \frac{k1}{2}\|x_t - x_c\|^2 + \frac{k2}{2}\left(p1 - p2 - (r+b)^2\right) + \frac{k3}{2}(b-2r)^2 + k4(2c-a)^2 + (b-2a)^2 \quad (3.1)$$

where *k1, k2, k3* and *k4* are the coefficients used to combine the energy terms.

The imaging model of the template uses the following assumptions:

- The iris corresponds to a valley in the image
- The whites of the eye correspond to peaks in the image intensity
- The boundaries of the eye and the iris correspond to edges in the image
- The iris is a dark (low intensity values) region in the image
- The whites of the eye are bright (high intensity values) regions in the image

Thus in order to utilize this imaging model, peak, valley and edges of an image are extracted. In [33], peak, valley and edges are computed by using morphological filters and then smoothed by concolving with a gaussian filter, given the edge, valley and peak fields ? e(x,y), ? v(x,y), ? p(x,y). The algorithm used for matching the template to the image defines an energy function, *E*, which makes use of the geometrical model and the imaging model. The energy function is minimized by changing the parameters of the template using gradient descent algorithm. *E* has contributions from the valley (the iris), peaks (the whites of the eyes) and edges (boundaries of the iris and the eyes).

$$E = E_v + E_p + E_e + E_i + E_{prior} + E_{inertia} \tag{3.2}$$

where

E_v : The part of the energy function which considers valleys

E_p : The part of the energy function which considers peaks

E_e : The part of the energy function which considers edges

E_i : The part of the energy function which considers intensities

E_{prior} : The part of the energy function which considers the prior probabilities as defined above

$E_{inertia}$: The part of the energy function which is used for fixing the iris parameters

Detailed formula about the total energy function is given in [33]. The algorithm tries to minimize the energy function, *E*, by using a search strategy based on steepest descent. It first locates the iris by using the valley potentials, the peaks are used to orient the template and then the intensity values are used for fine tuning etc.

3.5 Adaptive Mean Shift Algorithm

3.5.1 Introduction

Computer vision algorithms that are intented to form a part of a perceptual user interface must be fast and efficient. As human face tracking is the first step in our eye tracking problem, face tracking part must be able track in real-time and not absorb a major share of computational sources so we were focused on color-based tracking i.ei, tracking human faces using the skin color. Previous works [35], [36] were based on color-based tracking but they were too computationaly complex due to their use of color correlation, blob and region growing, Kalman filter smoothing, prediction and contour considerations.

Adaptive Mean Shift algorithm [4] is used for tracking human faces and is based on robust non-parametric technique for climbing density gradients to find the mode (peak) of probability distributions called the mean shift algorithm given in [37]. As faces are tracked in video sequences, mean shift algorithm is modified to deal with

the problem of dynamically changing color probability distributions. The block diagram of the algorithm is given in Figure 3.6.

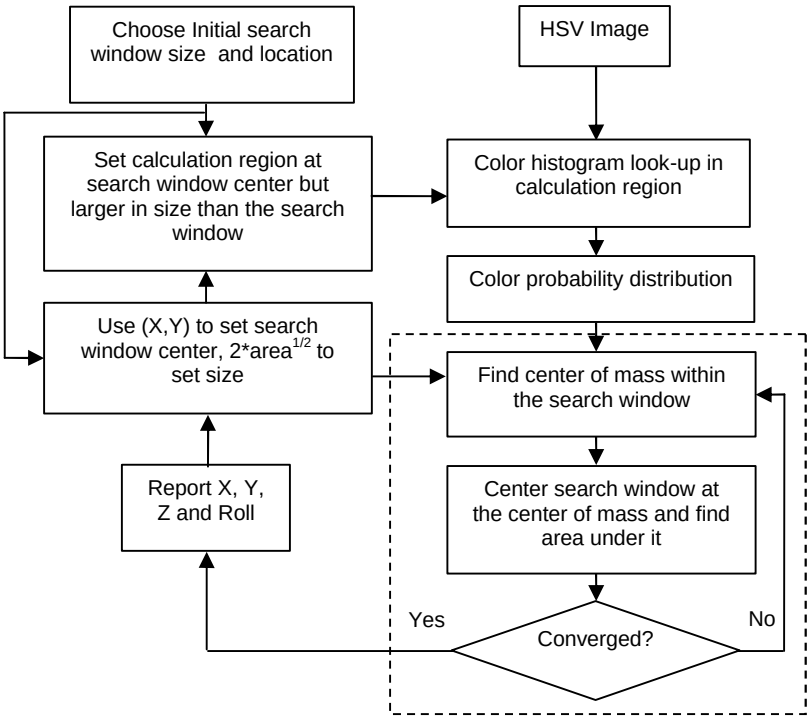

Figure 3.6 Block diagram of color object tracking [4]

3.5.2 Obtaining Color Probability Distributions

Mean shift algorithm operates on probability distributions. In order to use CAMSHIFT to track colored objects in a video scene, a probability distribution image of the desired color (flesh color in the case of face tracking) in the video scene must be created. In order to do this, firstly a model of the desired hue using color histogram is created by using HSV color space. HSV color space seperates out the hue (color) from saturation (how concentrated the color is) and from brightness. More information about HSV color space is given in section 2.3.2. So color models can be created by taking the 1D histograms of the hue channel.

For face tracking via a flesh color model, flesh areas from the user are sampled by prompting users to center their face in an onscreen box during the initialization. The hues derived from flesh pixels in the image are sampled from the H channel and

binned into an 1D histogram. When sampling is complete, the histogram is saved for future use. During operation, the stored flesh color histogram is used as a model, or lookup table, to convert incoming video pixels to a corresponding probability of flesh image as can be seen in Figure 3.7. This is done for each video frame. Using this method, probabilities range in discrete steps from zero (probability zero) to the maximum probability pixel value (probability 1.0). For 8-bit hues, this range is between 0 and 255. Tracking is performed using CAMSHIFT on this probability of flesh image.

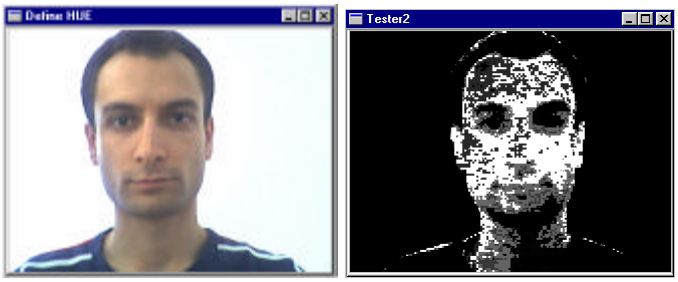

Figure 3.7 A video image and its flesh probability distribution

For HSV color space, a problem occurs when brigthness is low. Because if brigthness is low than saturation is also low then hue becomes quite noisy. To overcome this problem hue pixels that have very low brigthness values are ignored. Similarly at low saturation, hue pixels are also ignored.

3.5.3 CAMSHIFT Derivation

The main idea behing CAMSHIFT algorithm is mean shift algorithm. The mean shift algorithm is used for climbing the gradient of a probability distribution to find the nearest dominant peak. Mean shift algorithm can be summarized as follows;

1) Choose a search windows size
2) Choose the initial location of the search window
3) Compute the mean location in the search window
4) Center the search window at the mean location computed in step 3
5) Repeat steps 3 and 4 until convergence (or until the mean location moves less than a threshold)

Mean location (centroid) within the search window can be calculated as follows;

1) Find the zeroth moment

$$M_{00} = \sum_x \sum_y I(x, y) \tag{3.3}$$

2) Find the first moment for *x* and *y*

$$M_{10} = \sum_x \sum_y xI(x, y); \; M_{01} = \sum_x \sum_y yI(x, y) \tag{3.4}$$

3) Then the mean search window location (the centroid) is

$$x_c = \frac{M_{10}}{M_{00}} \; and \; y_c = \frac{M_{01}}{M_{00}} \tag{3.5}$$

where the *I(x,y)* is the pixel (probability) value at position *(x,y)* in the image and *x* and *y* range over the search window.

Mean shift algorithm can not be used for tracking alone because a window size that works at one distribution scale is not suitable for another scale as the color object moves towards and away from the camera. Small fixed size windows may get lost entirely for large object translation in the scene. Large fixed sized windows may include distractors (other people or hands) and too much noise.

So instead of fixed search window size CAMSHIFT uses a continuously adaptive search window relying on the zeroth order moment information. Zeroth momenth is the area found under the search window. Thus window height and width is set to a function of the zeroth moment found during the search. Also initial search window size is not a problem for CAMSHIFT as the window size will be adapted according to the probability distribution. Summary of the algorithm is given below;

1) Choose the initial location of the search window
2) Mean shift as above (can be one or many iterations); store the zeroth moment
3) Set the search window size equal to a function of the zeroth moment found in step 2
4) Repeat steps 2 and 3 until convergence (mean location moves less than a threshold)

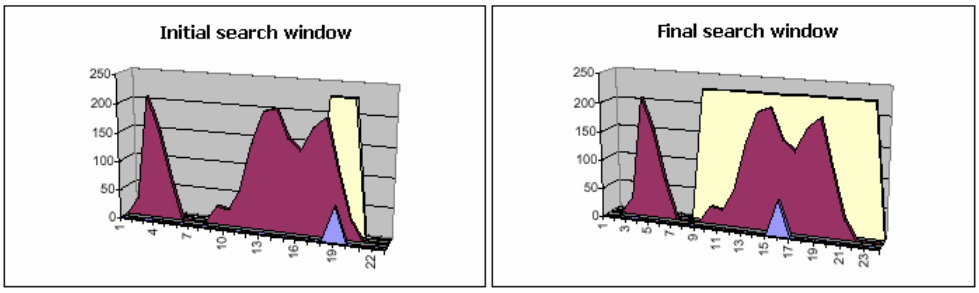

Figure 3.8 CAMSHIFT search window [4]

One modification to the CAMSHIFT algorithm can be reducing the calculation search window size during tracking i.e, we needn't calculate the color probability distribution over the whole image, but can instead restrict the calculation of the distribution to a smaller image region surrounding the current CAMSHIFT window. This will result in large computational savings when the area of the flesh color is small in size compared to the whole image.

1) First set the calculation region of the probability distribution to the whole image
2) Choose the initial location of the 2D mean shift search window
3) Calculate the color probability distribution in the 2D region centered at the search window location in an area sligthly larger than the mean shift window size
4) Mean shift to converge or for a number of iterations. Store the zeroth moment (area) and mean location
5) For the next video frame; center the search window at the mean location stored in step 4 and set the window size to a function of the zeroth moment found there. Go to step 3.

As a result; CAMSHIFT is a strong algorithm in face tracking; CAMSHIFT continuously re-scales itself in a way that naturally fits the structure of the data. When the color objects are close to the camera they can move rapidly but as their probability distribution occupies a large area and the search window is also large so can catch large movements. Similarly when the objects are distant from the camera

they can not move very rapidly and the color distribution is a small area so that small search window size is enough.

One major reason we have chosen CAMSHIFT as face tracker is; it is robust against distractors in the scene. Once it is locked onto the peak of a color distribution it will ignore the other non-connected peaks of color distributions. So when CAMSHIFT is tracking a face, the presence of other faces or hand movements in the scene will not cause to loose the original face unless they overlap onto the face area.

3.6 Detection and Recognition Using PCA

This section is dedicated to the explanation of a principal components analysis method that is used in object detection and recognition. Many previous works used this appearance-based method for face recognition so that the method is known as “eigenface method”. However we adapted the eigenfaces method for eye detection.

In this thesis, we have followed the method which was proposed by M. Turk and A. Pentland [29] in order to develop an eye detection system. Their work was motivated by the work of Sirovich and Kirby (1987, 1990) for representing images using PCA. However with the combination of our face detection algorithm we have proposed an efficient way called “adaptive eigeneyes for eye detection” to detect eyes. Details about our algorithm will be given in Chapter 4.

The approach [29] extracts the information contained in an image of face to capture the variation in a collection of face images, it is independent of features and uses this information to encode and compare individual face images. In mathematical terms, aim is to find the principal components of the distribution of faces or the eigenvectors of a set of face images, treating an image as a vector in very high dimensional space. Eigenvectors of covariance matrix can be thought as a set of features that together characterize the variation between face images.

The overal process can be divided into two phases;

3.6.1 Construction of Face Space

Let a face image I(x,y) be a N by N array. This image can be represented as a vector of dimension N^2. Let {G$_i$ | I=1…M} be the training set of face images i.e., our database. The average face of these *M* images is given by

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \tag{3.6}$$

Each face differs from the average face by the vector F i = Gi-? .

A covariance matrix of the training images can be constructed as follows;

$$C = AA^T \quad where \quad A = [\Phi 1, \quad ... \quad \Phi m] \tag{3.7}$$

It is exactly the same covariance matrix for multi dimensional data set given in section 2.2.1 with formula 2.5. The size of *A* is N^2xM so the size of the covariance matrix *C* is N^2xN^2 which means there are N^2 eigenvectors.

Finding the eigenvectors of the N^2xN^2 matrix C is an intractable task for typical images (262144x 262144 C matrix for 512x512 images!). Hence, a simplified way of calculation has to be used. As we defined in section 2.2.1 an NxN matrix *A* is said to have an eigenvector *X* and corresponding eigenvalue *?* if *AX = ?X*. So consider the eigenvectors of $L = A^TA$ such that;

$$A^T A \cdot V_i = m \cdot V_i \ \left(i = 1...M\right) \tag{3.8}$$

After multiplying both sides by *A* we get

$$AA^T AV_i = mAV_i \tag{3.9}$$

from which we can understand that *AV* are the eigenvectors of *C*. As a result we obtain;

$$U = [u_1 \quad ... \quad u_m] = [\Phi_1 \quad ... \quad \Phi_m] \times [V_1, \quad ... \quad V_m] = AV \tag{3.10}$$

where *U* is a matrix of size N^2xM. In practice a smaller set of eigenfaces is sufficient for recognition processes. So we can choose *M'<M*, where *M'* is the number of selected eigenvectors with high eigenvalues.

3.6.2 Face Recognition Using Eigenfaces

The eigenface-based recognition procedure is composed of two stages; training stage and a recognition stage :

In the training stage, the face of each known individual, G_k is projected into the face space and a *M'x1* sized vector O_k is obtained;

$$\Omega_k = U^T(\Gamma k - \Psi) \quad where \quad k = 1...M \ (m\,is\,the\,number\,of\ \ faces) \tag{3.11}$$

so O_k has size *M'x1*.

In recognition stage; a new image *G* is projected into the face space to obtain a vector *O*.

$$\Omega = U^T(\Gamma - \Psi) \tag{3.12}$$

The distance to each face class is defined by;

$$e_k^{\,2} = \left\|\Omega - \Omega_k\right\|^2 \quad where \quad k = 1...M \tag{3.13}$$

For the purpose of discriminating between face images and nonface images, the distance *e,* between the original image *G* and its reconstructed image from the eigenface space G_R is also computed.

$$e^2 = \left\|\Gamma - \Gamma_R\right\|^2 \quad where\ \Gamma_R = U.\Omega + \Psi \tag{3.14}$$

Let $?_c$ be the maximum allowable distance from face space;

If	$e = ?_c$	then the input image is not a face
If	$e < ?_c$	and $e_k = ?_c$ *(for k=1...M)* then input image is an unknown face
If	$e < ?_c$	and $e_k = min(e_k s) < ?_c$ then input image contains the face of k^{th} person

In this thesis for deciding whether the input area is an eye or not; we have calculated the distance between original image and the reconstructed image. The distance is then compared with the $?_c$ value which is the maximum allowable distance from eye space.

As a result we listed the advantages and disadvantages of eigenface method before choosing it as an eye detector;

Table 3.1 Advantages and Disadvantages of eigenface method

	Advantages	Disadvantages
1	Easy to implement	Not so suitable for real-time applications (400ms for detection [29])
2	Processes large size of images using reduced amount of data	Algorithm is scale variant, you must use images with different scales.
3	High speed if small sized database images are used.	Detection of small sized images may fail

The main problem with eigenface method is the scale variance problem. To overcome this problem previous works [29], [38], [39] used different scaled images in their databases which results detection of limited number of scales. For example in [29], 3 different scaled versions of face images were used which means ˜3 times slower for each detection. However we chose eigenface method as eye detector and decided to develop an adaptive eigenface method to solve this scaling problem which is given in detail in Chapter 4.

CHAPTER 4

IMPLEMENTATION

In this chapter the implementation of the eye tracking system named “TRACKEye” that we have designed, will be presented in details. The main capabilities of the system are;

- Face tracking
- Tracking both eyes individually
- Extracting features of eyes such as outline of the visible eyeball area, finding the color area formed by the iris and pupil
- Tracking eye movements and measuring the position of eye pupil in the outline area of the eye which is a kind of indication where the human is looking at

Also we have implemented features such as

- Blink detection
- 2D trajectory extraction of color objects such as face or hand in video sequences

which may provide usefull information for the perceptual user interfaces that will be implemented in the future.

The system is based on three components;

1) Face detection component
2) Eye detection component
3) Eye feature extraction component

The performance of each step is highly dependent on the steps before itself; because the results obtained in each component is passed as an input to the next component.

For example; Face area, face parameters (width of the face) obtained during face detection are used for defining search areas for eye localization and the scaling factor to be used in adaptive eigenface method in eye detection component. Also the eye locations obtained in eye detection component are used to extract the edge maps for pupil detection and determines the initial points of the snake control points in eye feature extraction component. As a result all components should work collectively and correctly to have a working eye tracking system.

The structure of the complete system may be outlined as Figure 4.1.

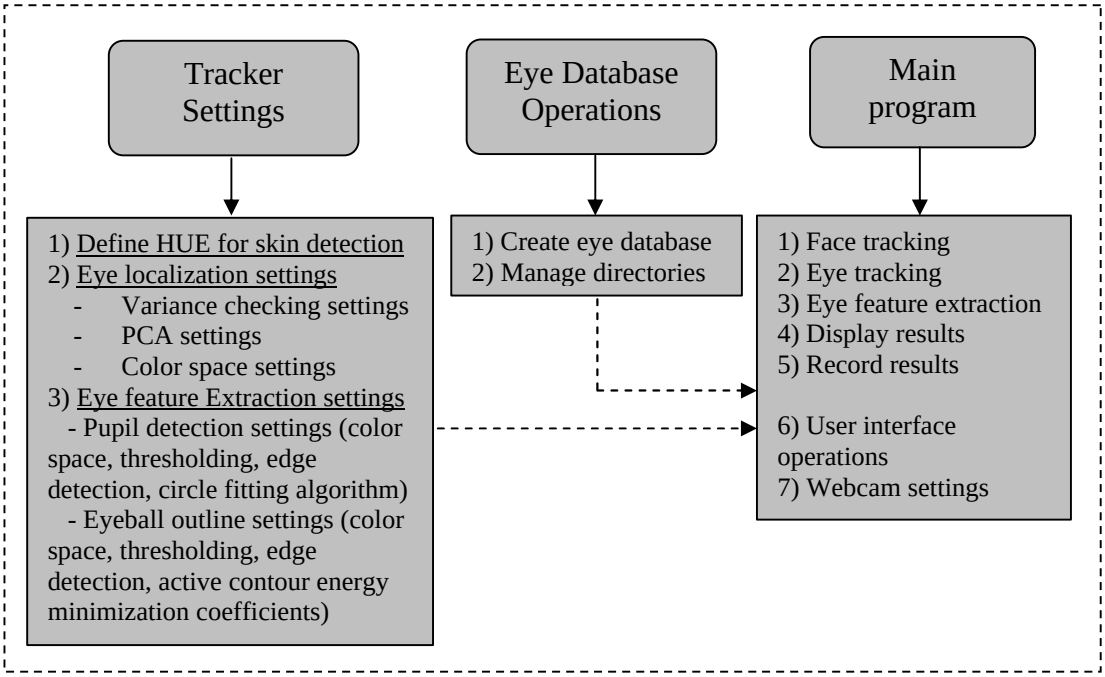

Figure 4.1 Structure of the system

The system performance is dependent on the settings defined by the user although default parameters stated may give good results. So there are mainly two interactions for the settings with the user; 1) Settings for detections 2) Defining eye database to use during PCA for eye detection.

The flowchart explaining the eye detection and tracking process is given in Figure 4.2. Every step in the flowchart is detailed through Sections 4.1 – 4.3.

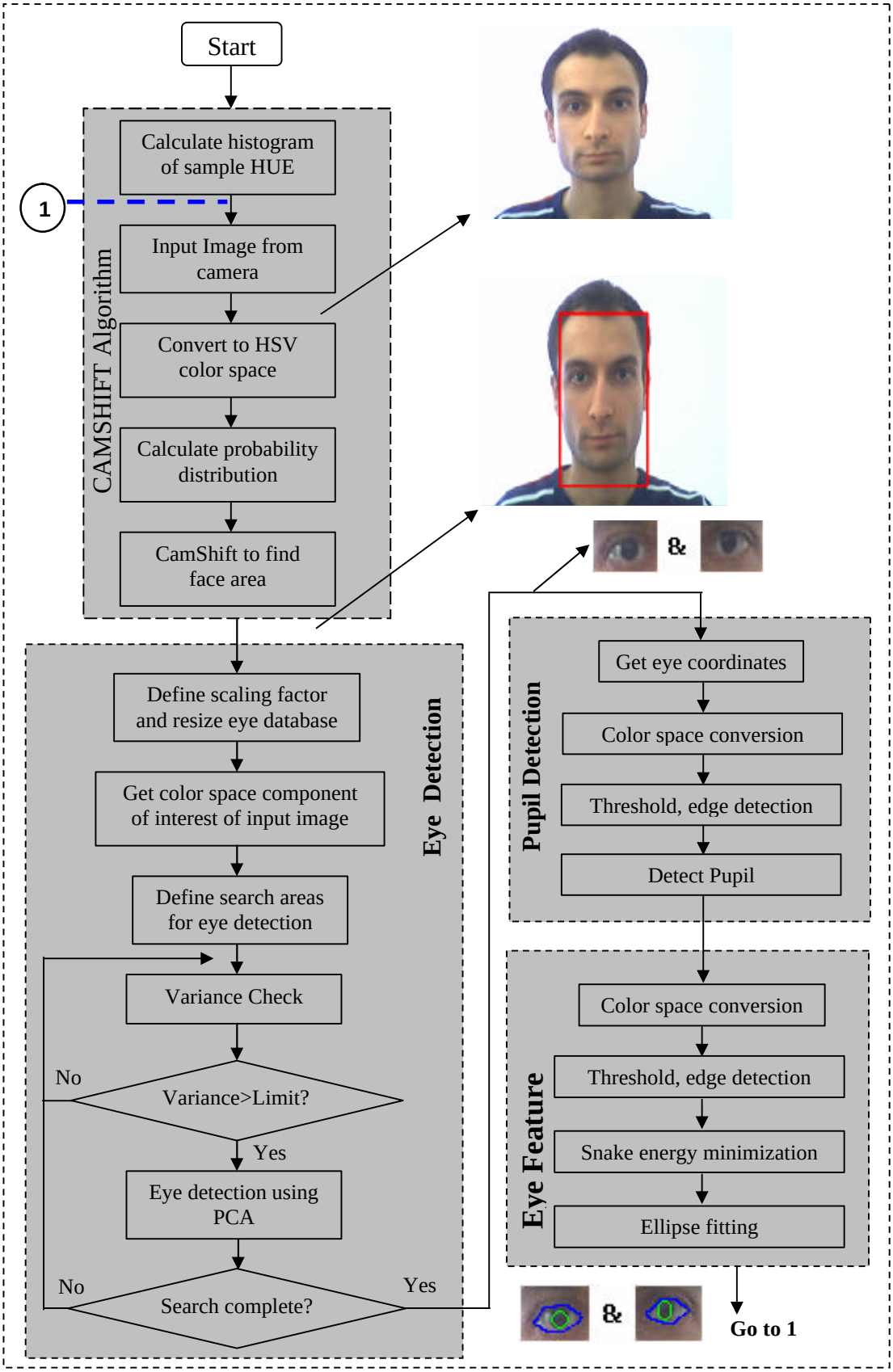

Figure 4.2 Flowchart of the tracking system

The first step in eye tracking system has to be localization of face because the eyes occupy a small area compared to the whole image and trying to detect eyes without localizing face first, probably will result in many false alarms. In this study a feature-based face detection algorithm is implemented for this purpose.

4.1 Face Detection

The face detection algorithm implemented in this study is a feature-based face detection method based on skin color detection and uses the Continuously Adaptive Mean Shift algorithm [4] which was previously discussed in detail in section 3.5.

As discussed earlier CAMSHIFT algorithm is a robust non-parametric method for finding the peak of probability distribution with an adaptive search window size. CAMSHIFT algorithm deals efficiently with image problems; irregular object motion, image noise, distractors and facial occulusions [4]. One disadvantage of CAMSHIFT algorithm is that it performs tracking using the color feature of the object so any overlapping with the object may lead the algortihm to fail. But as our aim is eye tracking, one person sitting in front of the camera without any distractors is assumed.

Implementation details of the algorithm is as follows;

1) In order to CAMSHIT algorithm to work, a sample HUE histogram to compare with images must be used. So at the beginning of the tracking process the user is prompt to place a part of his face which contains only the skin color in the box which is created on the screen. Infact; it does not have to be a part with only the skin color but using skin regions give better results.

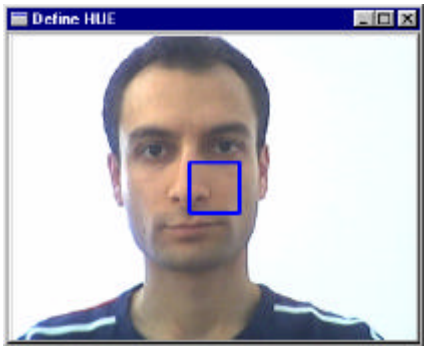

Figure 4.3 Example hue definition for probability distribution calculation

2) After getting the sample image, it is converted from RGB color space to HSV color space. Hue is the only component of interest in CAMSHIFT. So HUE component is extracted and stored as a 1 dimensional histogram for further use in building probability distributions.

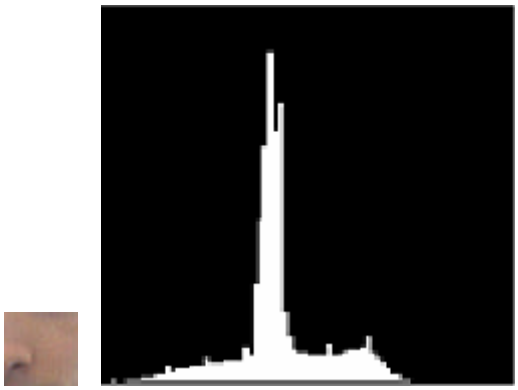

Figure 4.4 a) Sample hue region, b) Histogram of the hue component

3) A new image in which the face is going to be located is grabbed from the camera and converted to the HSV color space.
4) Initialize the calculation window as the whole image
5) Probability distribution and back projection image which is a kind of indication of probability distribution as an gray-scale image of the input image is formed by comparing the each pixel of the source hue image with the hue histogram constructed in step 2.

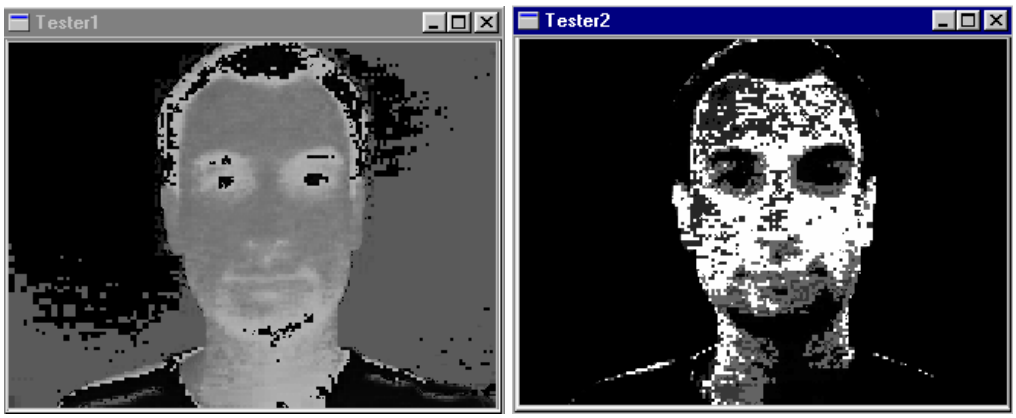

Figure 4.5 a) Hue component of the input image b) Probability distribution of the hue component

6) Mode of the probability distribution and search window size which indicates the face area is calculated.

Figure 4.6 Output of the face tracker

7) Calculation window for the next frame is set to the current search window size instead of the whole image for computational savings.

In this thesis aim of the face tracker is not just tracking face visually. It serves as an important information source for the eye detection component which is based on principal components analysis. It provides two important information;

1) Location of the face area
2) Center of the face image according to the whole image, thus we can calculate the width information of the face.

Face width, height, center, size information of the detected face is calculated by using the zeroth and first order moments of the back projection image. Width information of the face localized and coordinates of the face area play an important role in the performance of the eye detection component. Wrong results certainly will make the eye detection component fail.

Although the CAMSHIFT algorithm for face tracking is easy to implement because of its non-parametric structure, we have observed some disadvantages of it such as ambient lighting effect and low image quality with the use of standard cameras.

Detailed results under different conditions, advantages, disadvantages we have observed during the tests and comments about the CAMSHIFT algortihm will be presented in section 5.2.

4.2 Eye Detection

The eye detection algorithm implemented in this study is based on the eigenfaces approach [29] that was given in section 3.6. As we are using eigenfaces method for eye detection in face areas we will use the term “eigeneyes”. Eigeneyes approach uses principal component analysis technique on a representative set of eye images in order to lower dimensional eye space. Although the main purpose of the eigenface method is to find the best match in a database for an input we will use this method to classify an input region as an eye or noneye region by calculating the distance to the eye space as given in section 3.6.2.

One of the major problem of eigenface method is the scaling problem. One common solution to this problem is to use many databases which are scaled versions of the original database (typical databases contains three versions [29], [38]). Another solution is to resize the input image by gaussian pyramid [39]. But as all the scales can not be used during detecting because of speed problems, object at some distances will not be detected. So scaling can not totally be solved and will continue to remain as a problem for the eigenface problem.

4.2.1 Eye Database for Training

Figure 4.7 Sample eyedatabase used in eigeneye detection

The training database must contain different poses and orientations of both eyes. A typical eye database contains eye images for left and right eyes when the person is staring at different directions and when the head has a negative/positive roll angle. Eigeneye method is dependent on the lightening conditions and illumination effects the system negatively. To obtain a better performance; training eye images may be taken under different lightening conditions. One advantage of using eye images as training database is, complex databases such as the ones used in face detection is not a requirement as the view of the eye does not change so much from person to person. Like the way the user was prompt to define the sample hue image before the CAMSHIFT starts, the user can easily form the eye database.

Also the size of the eye database images plays an important role in adaptive eigeneye method; Given a face image in which the eyes are going to be detected, using the human face model in Figure 4.8 and the scaling factor calculated in formula 4.1, eye database images are rescaled. So the eye images shall characterize the properties of the eye area with the geometry as given in figure 4.8.

The number of eye images to be used plays an important role during detection phase. Performance of the system with different number of training images is illustrated in section 5.3.

4.2.2 Adaptive Eigeneye Method

Eigenface approach is accepted to be a nearly real-time method because of its speed problems. In this thesis we have implemented a new method to overcome the problem of speed. Using the information supplied by the face tracker we can use eigeneye method with only one database, having the property of detecting eyes without scale problems. Before the eye is searched over the face area, eyedatabase is resized with the scale factor which is computed according to the formula given below;

$$scale\ factor = \frac{face\ width \times ideal\left(\frac{eye\ width}{face\ width}\right) ratio}{eye\ width\ of\ the\ database} \tag{4.1}$$

Scale factor : Indicates how much the width and height of the eye images in the database will change before calculating eigenvectors.

Face width : Width of the face that was localized using CAMSHIFT face tracker in section 4.1. Heigth of the face is not a good idea to use as the heigth may vary with the neck or top of the head region between people.

Ideal (eye width/face width) ratio : Considering the face model given in MPEG-4 version 1 standard, this ratio is the ratio of the eye width to the total face width and is ˜ 0.35.

Eye width of the database : Width of the eye images in the database

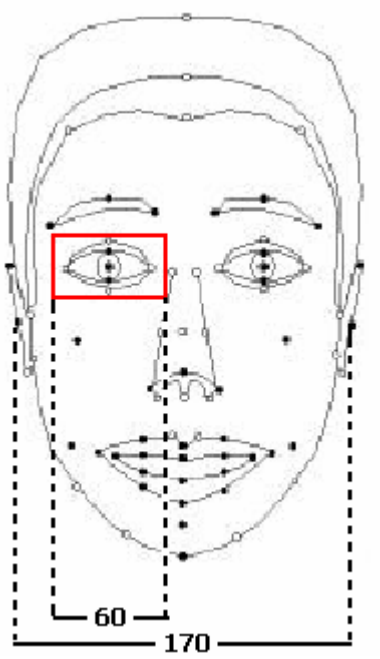

Figure 4.8 The face model used in eye detection

Eye images in the database are resized by the scale factor using interpolation i.e., the method interpolates between pixel values within a neighborhood by taking a statistical sample (such as the mean) of the local intensity values.

$$\begin{aligned} newWidth &= scale\ \ factor \times database\ eye\ width \\ newHeight &= scale\ \ factor \times database\ eye\ height \end{aligned} \tag{4.2}$$

The block diagram of the algorithm is given in Figure 4.9.

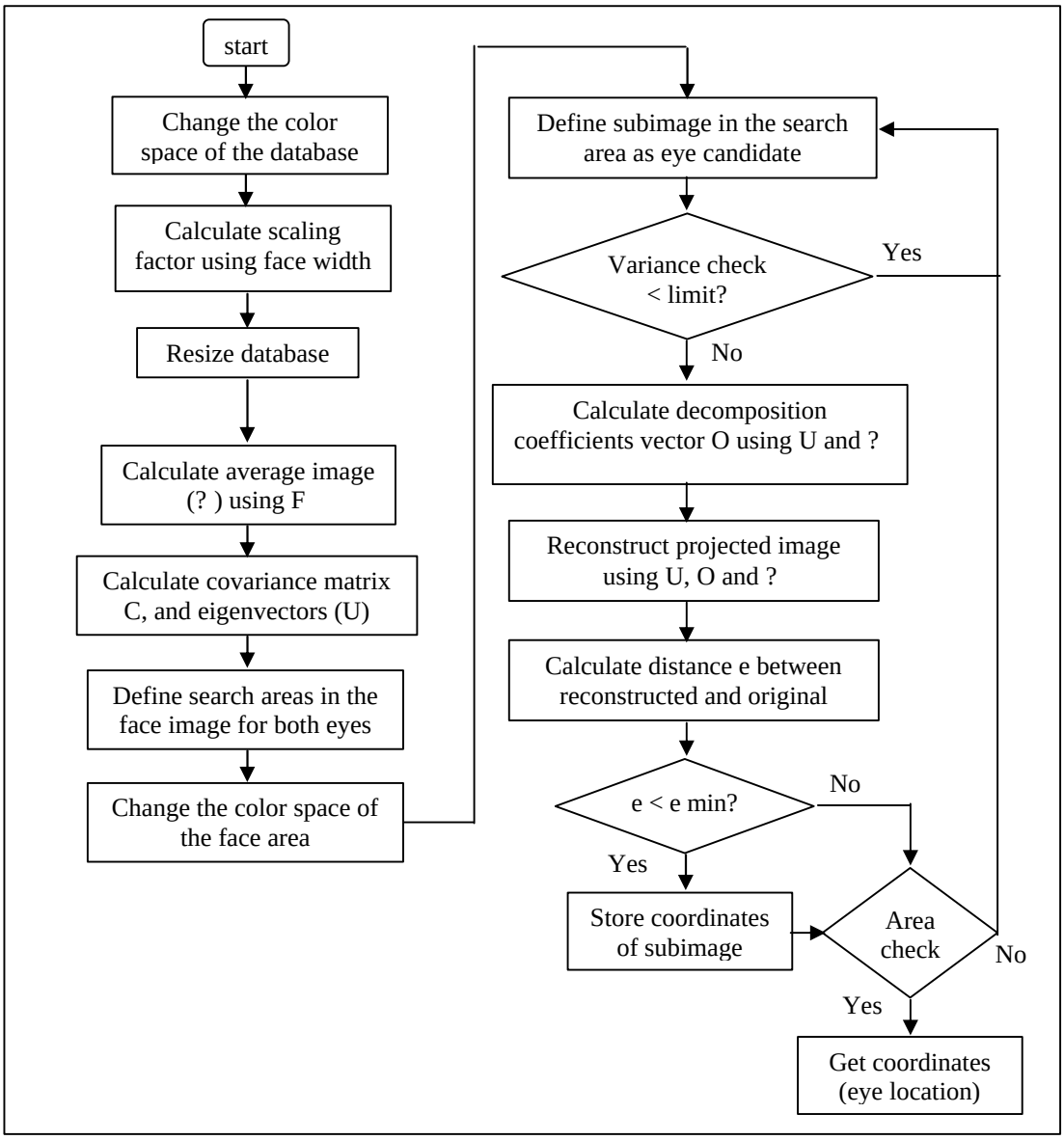

Figure 4.9 Adaptive eye detection using eigeneyes

Details of each step is explained as follows;

1) For eigeneye method, 2D image arrays must be represented as 1D data sets. However database images are color images so they must be converted into 2D forms as a first step. In this thesis, instead of representing color images as gray-scale images for eigeneye method, we have tried different components of different color spaces such as CIE XYZ, YCbCr, HSV, CIE Lab and obtained better results than using gray-scale.

2) After eye images are formed as 2D image arrays they are rescaled as explained above with the formulas 4.1 and 4.2.
3) Average image *?* is obtained using the formula 3.6 and 1D forms of the eye images, which are normalized to zero by removing the DC component, are used to construct the covariance matrix. Finally eigenvectors are calculated and eigenvectors with high eigenvalues are chosen to form the U eigenvector matrix *U*. Choosing correct number of training images and eigenvectors is a critical point in eigeneye detection. The experiments performed and result obtained about choosing these parameters are given in section 5.3.
4) Eye area candidates will be input to the eigendecomposition however for computational savings by using the geometircal model again we can reduce the area i.e., the number of candidates. Search areas for left end rigth eyes are defined as follow;

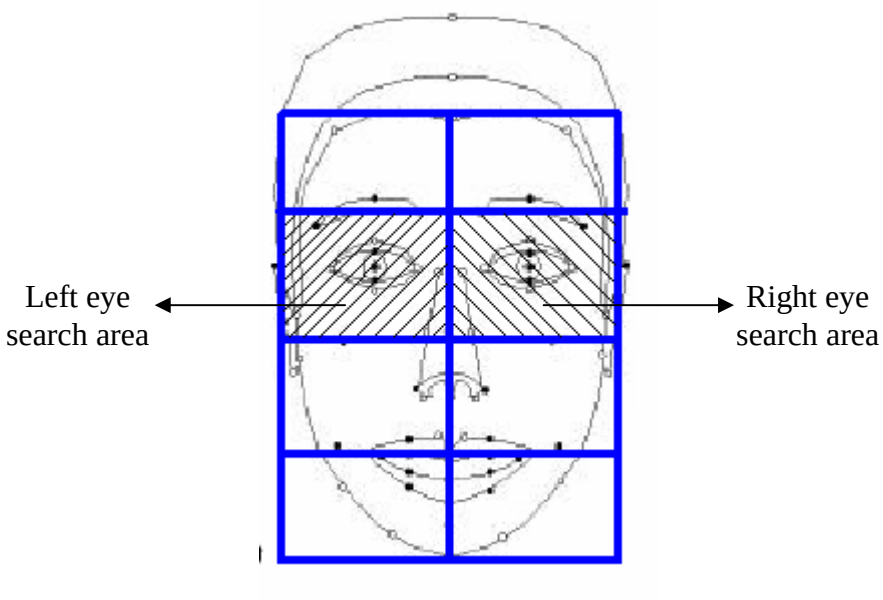

Figure 4.10 Search areas for left and right eyes

$$leftEyeX_{\min} = 0 \;\; and \;\; leftEyeX_{\max} = \frac{face\,area\,width}{2}$$

$$leftEyeY_{\min} = \frac{face\,area\,height}{2} \;\; and \;\; leftEyeY_{\max} = \frac{face\,area\,height \times 3}{4}$$

$$rightEyeX_{\min} = \frac{face\,area\,width}{2} \;\; and \;\; rightEyeX_{\max} = face\,area\,width$$

$$rightEyeY_{\min} = \frac{face\,area\,height}{2} \;\; and \;\; rightEyeY_{\max} = \frac{face\,area\,height \times 3}{4} \qquad (4.3)$$

5) During the development phase of our system in spite of reducing the eye search areas, sometimes we observed that plain areas were selected as eye locations. Using the fact that skin regions have a low variance value whereas areas with gradients such as eyes have higher variance value we decided to check the varaince of the area before performing eigeneye detection. This reduced the number of false detections and also we observed a significant computational saving.
6) Each candidate area with a high variance value is projected into the eye space formed by the eye database. Classification of the input area as an eye or noneye area is performed by calculating the distance to the eye space as given in section 3.6.2.

4.3 Eye Feature Extraction

Before the development phase of the thesis, our aim was decided to be as “To design a machine vision system capable of tracking the eye areas and the features of both eyes individually in real-time”.

A region in an image is a group of connected pixels with similar properties. Regions are important for the interpretation of an image because they correspond to areas of interest in a scene [40]. So considering an eye image, the eye area can be partitioned into two simple overlapping regions which may be accepted to be the features of a human eye for machine vision problems;

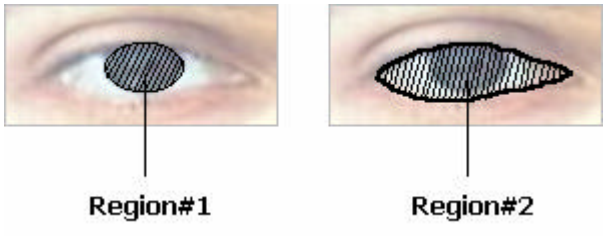

Figure 4.11 Human eye regions

Region#1 **:** The area occupied by iris and pupil parts of the eye.
Region#2 **:** The visible eyeball area. Region#2 contains region#1.

There are two common approaches for partitioning an image into regions [40];

1) Region-based segmentation
2) Boundary estimation using edge detection

The second approach which is simply finding the pixels that lie on a region boundary is used for indicating region#1 and region#2 in images in this thesis.

However, during our study we have observed that; When the eye area is found by the adaptive eigeneye method, often the simple edge detection methods, such as canny edge detection and sobel operator, are unable to use it directly for region#1 and region#2 partitioning. So several image enhancement methods for eliminating undesirable results were used in the early stages of feature extraction of eyes. Sections 4.3.1 and 4.3.2 will cover the details of boundary estimation using edge detection steps which were implemented during the development phase of the thesis.

4.3.1 Region#1 Detection and Measuring Eye Movements

As mentioned before, the region formed by eye pupil and iris can be obtained by an approach based on edge detection. This area has a significant change in the image intensity when compared to the surrounding area formed by the whites of the eye and skin. Due to the noise and other factors such as ambient lightening, some preprocessing steps shall be applied to the eye area before edge detection.

The part of the program related with the detection of region#1 is shown in Figure 4.12;

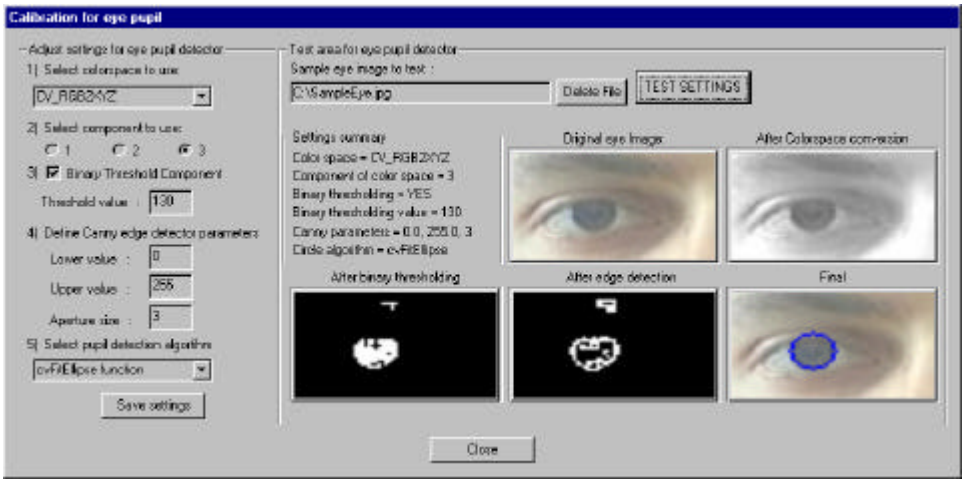

Figure 4.12 Extraction of eye features – 1 : Region#1

Algorithm for region#1 detection contains the following steps;

1) **Color Space Conversion :** In order to facilitate the detection of edges, it is essential to determine changes in intensity in the neighborhood of a point. This is achieved by converting color eye images into gray-scale images. During our study, we have observed that; using different components of different color spaces, such as Z component of XYZ color space, instead of RGB to gray-scale conversion results in better edge strengthened images.

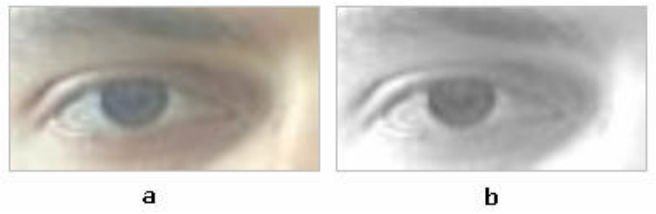

Figure 4.13 a) Original eye area, b) Z component of XYZ color space

2) **Thresholding :** There is a strong edge content in the area between region#1 and its outside area. However there are still many points with lower intensity values than region#1 and which will be marked as edges during edge detection. Therefore; the color space converted eye area image is binary thresholded to obtain a binary image eliminating edges formed by lower intensity areas.

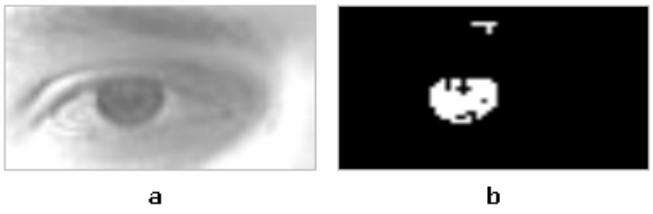

Figure 4.14 a) Color space converted eye, b) Binary image with threshold set to 130

3) **Edge Detection :** It is expected that the steps up to now will produce a segmented image having only two regions which are object of interest and remaining area. However, due to several factors there are some regions marked as region#1 as shown in Figure 4.14 b.

To solve this problem, a knowledge based method has been implemented; “The area formed by eye pupil and iris may be accepted as a perfect circle”. So we have developed a circle detection algorithm based on pixel voting which is explained in section 4.3.1.1. This algorithm uses the edge information in the image. So before applying the circle detection, edge image of the region segmented image must be obtained and Canny edge detection is used to detect intensity changes correspoding to edges in binary image.

Figure 4.15 a) Segmented image, b) Edge image

4) **Circle Detection** **:** We are interested in finding the edge points the edge image that best fits to form a circle. So the whole edge image is searched using the algorithm given in section 4.3.1.1 and the most voted circle is accepted as the boundary of region#1.

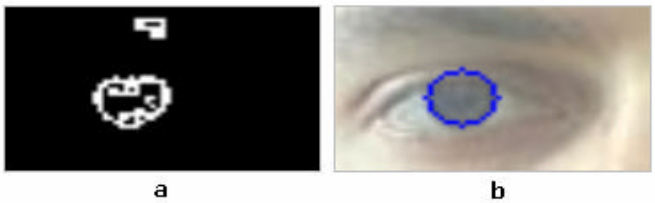

Figure 4.16 a) Edge image, b) Final output

As a result of circle detection algorithm, radius and coordinates center of the perfect circle is also provided. Note that location of the center compared to the width of the eye area may be accepted as an indication of the direction of gaze. In fact width and coordinates of the perfect ellipse fitted onto the visible eyeball area i.e. region#2 with the information of circle fitted onto the region#1 gives more accurate results but we have used the width information of the eye area which is supplied by adaptive eigeneye method and coordinates of the circle fitted onto the region#1 instead. Reasons and conclusions about measuring eye movements is given section 6.1.

4.3.1.1 Circle Detection Algorithm

Implemented circle detection algorithm is based on a voting method in the edge image. The most voted pixel is accepted as the center of the circle.

Block diagram of the algorithm and necessary explanations can be outlined as follows;

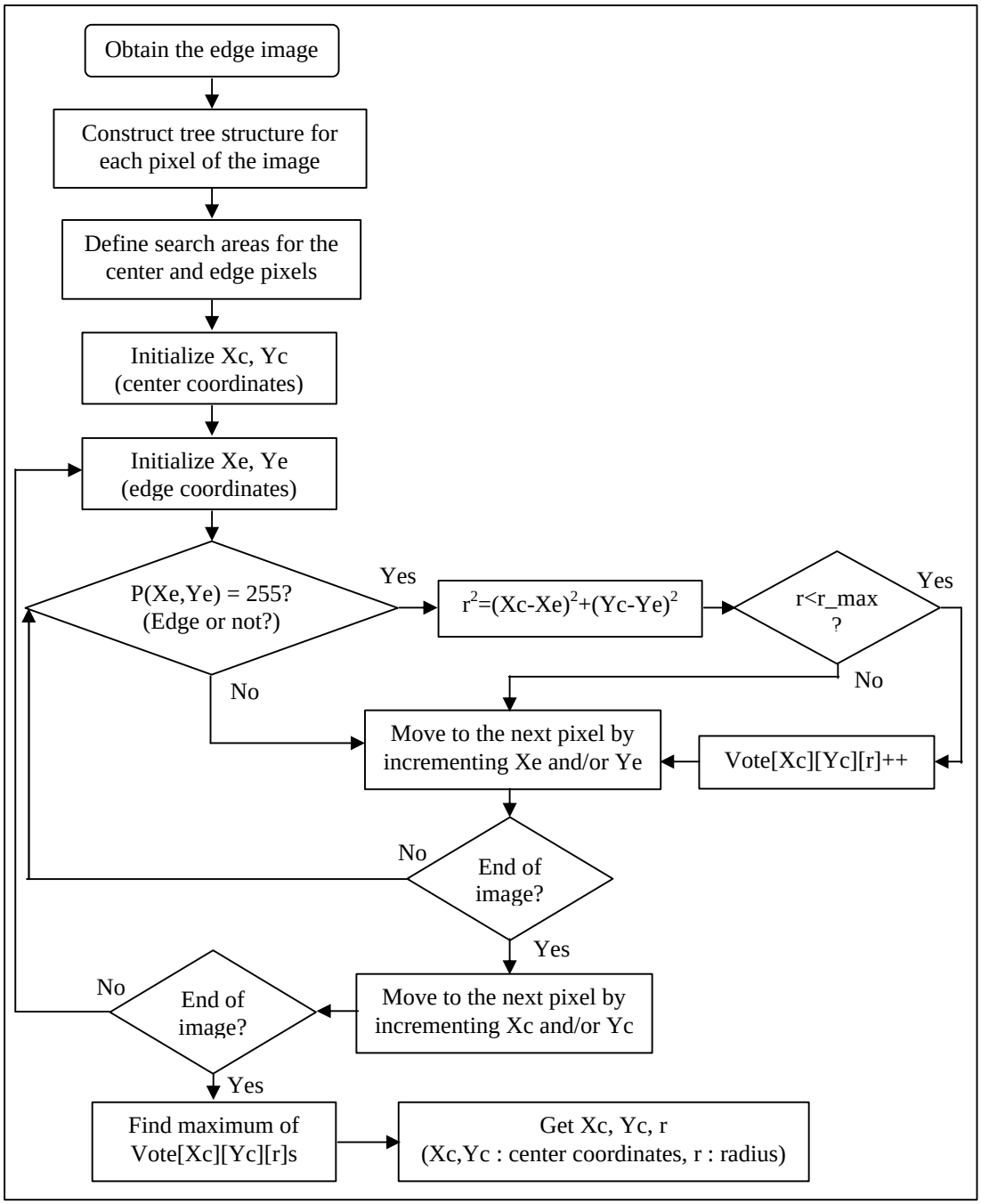

Figure 4.17 Circle detection algorithm

1) Circle detection algorithm is based on a voting mechanism where edge pixels vote for the center pixel. Main idea of circle detection algorithm is; Edges, white pixels in the edge image, are the boundaries of the circle and the pixel which is surrounded by most of the edge pixels with equal distances is the center of the circle.
2) Using the whole image as the search area for center coordinates and edges is a time consuming process, so limiting the search areas by some rules which are found experimentally reduces the time to calculate the center significantly;
    a) Center is chosen from the eye area given below;

$$Minimum\ X\ \ for\ Center = \frac{Width\ of\ \ eye\ area}{4}$$
$$Maximum\ X\ \ for\ Center = \frac{Width\ of\ \ eye\ area \times 3}{4}$$
$$Minimum\ Y\ \ for\ Center = \frac{Height\ of\ \ eye\ area}{4}$$
$$Maximum\ Y\ \ for\ Center = \frac{Height\ of\ \ eye\ area \times 3}{4} \qquad (4.5)$$

    b) Edges voting for the center are chosen from the eye area given below;

$$Minimum\ X\ \ for\ Edges = \frac{Width\ of\ \ eye\ area}{6}$$
$$Maximum\ X\ \ for\ Edges = \frac{Width\ of\ \ eye\ area \times 5}{6}$$
$$Minimum\ Y\ \ for\ Edges = \frac{Height\ of\ \ eye\ area}{6}$$
$$Maximum\ Y\ \ for\ Edges = \frac{Height\ of\ \ eye\ area \times 5}{6} \qquad (4.6)$$

3) During the implementation phase we sometimes have observed that edge points which do not belong to the boundary of region#1 may form a perfect circle. So in order to avoid this, checking the radius of the best fitted circle with the rule

$$Radius < \frac{\min(image\_width, image\_height)}{3} \qquad (4.7)$$

is necessary.

4.3.2 Region#2 Detection

Detection of region#2 has been a challenging problem during our study as it contains region#1. But using the fact “whole eye area may be accepted as visible eyeball area surrounded by skin so somehow eliminating the skin region will output the eyeball area seen by the imaging system”.

The part of the program related with the detection of region#2 is shown in figure 4.18;

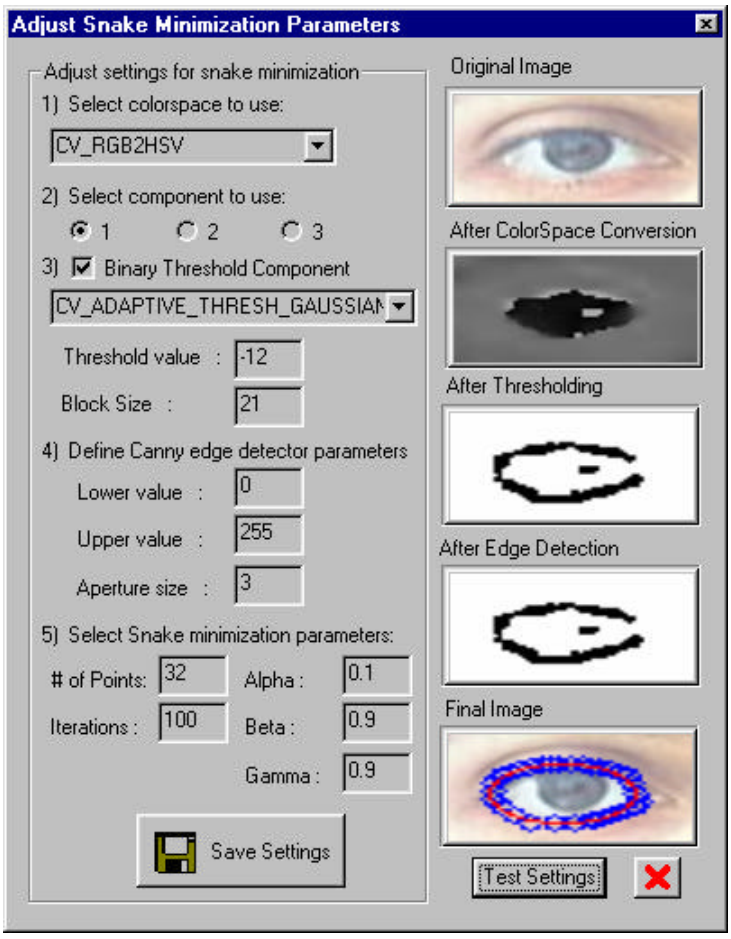

Figure 4.18 Extraction of eye features – 2 : Region#2

Steps for detecting boundary of region#2 are quite similar to the ones given in the preceding section;

1) **Color Space Conversion** **:** Color space is mandatory when dealing with images for thresholding, external energy minimization of a snake, region

partitioning etc. but also the type of color space conversion and component used in further steps play a very important role. As mentioned above; eye area during detection od region#2 must be seen as eyeball area surrounded by skin. So using the color information of human skin, skin region may be eliminated as a first preprocessing step. To obtain the color information of the whole area we have used the HUE value as in the CAMSHIFT algorithm and obtained quite satifying results before thresholding even if the images were noisy and taken under bad lightening conditions.

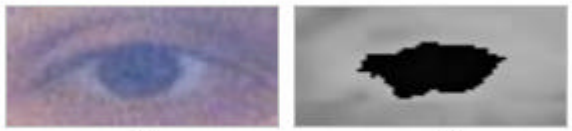

Figure 4.19 a) Original eye area, b) HUE component of HSV color space

2) **Thresholding :** Thresholding is used for converting gray-scale image obtained as a result of the HUE component of the HSV color space into a binary image. By thresholding, objects of interest, here reigon#2, are separated from the background.

Although binary thresholding with a fixed value may lead to good results in most cases, adaptive thresholding is a better type of thresholding as strong illumination gradient may occur in the eye area during the movement of the head.

Figure 4.20 a) Gray-scale image, b) Adaptive thresholded image with a 15x15 window, C=-12

3) **Active Contour Energy Minimization / Ellipse Fitting :** Final step in detection of region#2 is indicating the boundary. Visible eyeball area is geometrically almost an ellipse. Instead of complex ellipse fitting algorithms we have implemented a snake minimizing its energy on the binary image for boundary detection.

The developed snake needs to be initialized by an external procedure. In order to provide the initial locations of the control points, we have developed an initialization procedure based on the rectangular box bounding the whole eye area. Control points are uniformly placed around this bounding box as shown in Figure 4.21–b.

Two types of energy functions were used as the total energy of the snake during energy minimization; Internal energy and external energy.

$$E_{snake} = E_{\text{int}\,ernal} + E_{external} \qquad 4.8$$

Minimization of internal energy provides the snake to have a smooth circular shape whereas minimization of external energy is used to exploit the edge potentials of the underlying image. As a result of the minimization procedure the final shape of the snake is expected to be the smooth buondary of the reigon#2.

Thresholded image has a very important role during snake minimization; uneven illumination causes uneven distribution of gray values after computing the HUE value of the eye area and simple binary thresholding with a fixed value may result in small regions like eyeball area around the real eyeball area which causes the snake to build an invalid shape. So during our study we have observed that adaptive thresholding in the preprocessing step#2 is a solution for this kind of problems.

Values of energy coeeficients *alpha, betha*, *gamma*, which were explained in section 2.5, also have an important role during each epoch. These values are determined by the user before the initialization of the tracker. The effect of selecting different coefficient values will be discussed later.

After the final locations of the control points are obtained simple ellipse fitting algorithms are used to indicate the boundary of region#2 with a perfect ellipse for the sake of better indication.

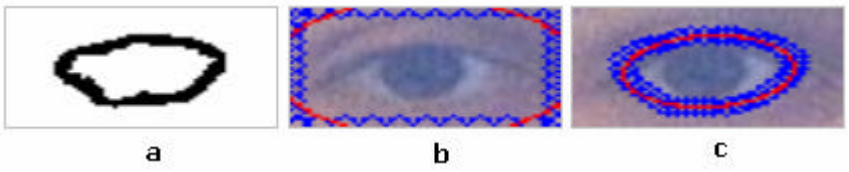

Figure 4.21 a) Thresholded image, b) Initial locations of control points, c) Final locations of control points

Some outputs of the program, extracting the features of the eye are given in figure 4.22 and 4.23.

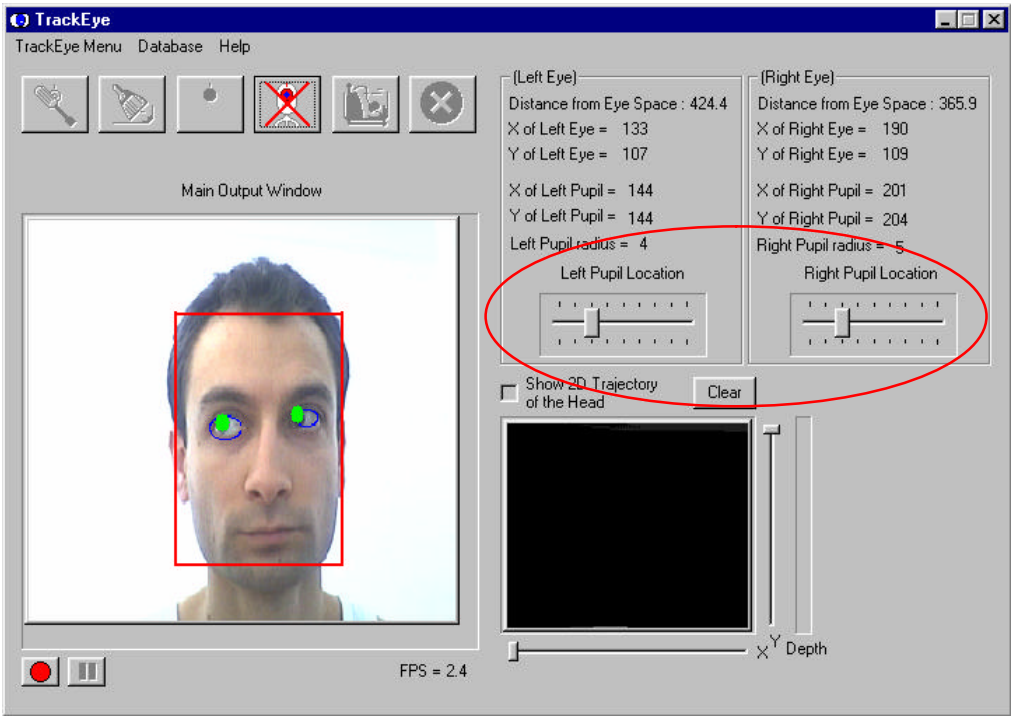

Figure 4.22 Extraction of eye features – Sample output-1

Coordinates of the eye areas, center of the pupils, radius informations for both left and right eyes are given as output in the final window. Also using the idea given in

section 4.3.1 directions of the gazes are obtained and shown on sliders for both left and right eyes.

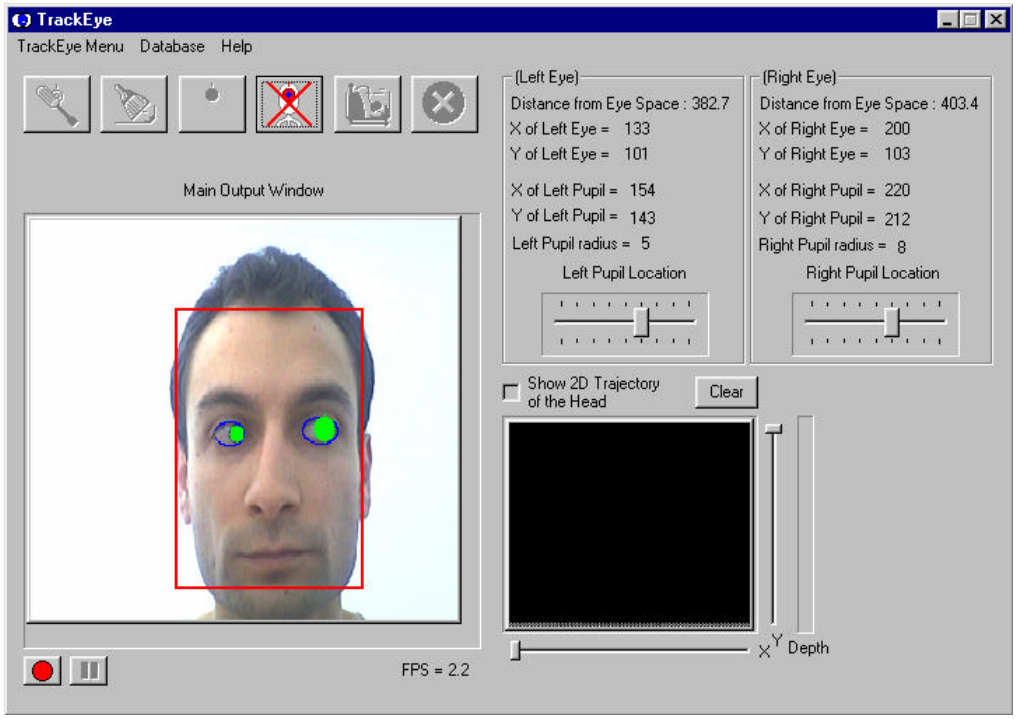

Figure 4.23 Extraction of eye features – Sample output-2

More results under different conditions and with different parameters will be given in the following chapter.

CHAPTER 5

RESULTS AND COMMENTS

In this chapter performance of the system is presented. First of all, properties of the system that the tests were conducted at are given. Then, performance of each step is explained and lastly the performance of the system as a whole is discussed.

Performance of the whole system is dependent on each step and performance of each step is effected by the results obtained in the previous steps. For example, to track eye features; the system must first localize the face correctly, after localization of the face, eye areas are searched in the face area. Finally after the localization of the eye areas, features of the eyes are determined.

5.1 Properties of the Test Setup

Performance of the system was tested with a Philips PCVC750K ToUcam video camera, providing frames of resolution 320x240, attached to a standard PC with PentiumII® 400MHz processor and 192MB of RAM. Performance of each step is tested under different conditions such as the ones with variable lightening, distractors around the image areas of interest, different size of face image etc.

5.2 Performance of the Face Tracker

CAMSHIFT is a computationally efficient, colored object and face tracker. Although its simplicity causes limitations, it's face tracking performance was enough to use it as a face tracker in our work. The results of the tests are given below;

5.2.1 Tracking in the Presence of Distractions

CAMSHIFT's search window converges to span the nearest dominant connected probability distribution i.e., only one connected area having the color properties

defined in the calibration stage will be tracked. Regions overlapping with the face and having skin color will not be ignored by CAMSHIFT.

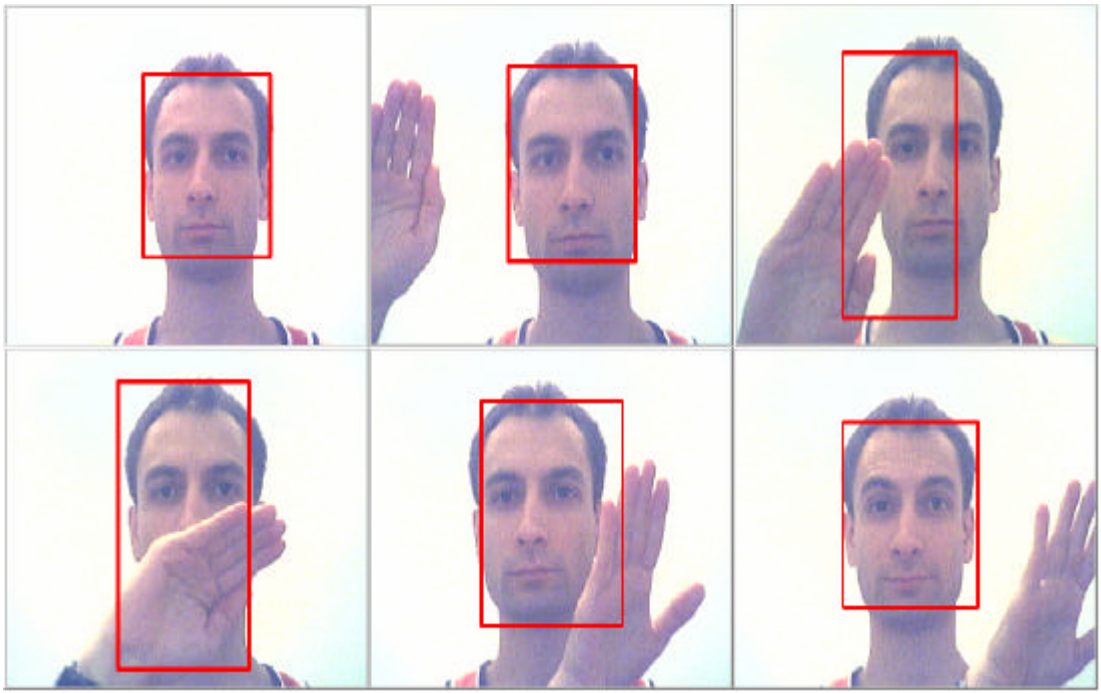

Figure 5.1 Tracking a face in the presence of a passing hand

Implemented face tracker ignores outliers allowing it to be robust against distractions. Once it is locked to the face it will tend to ignore nearby but non-connected color distributions. As a result; presence of other faces, hand movements in the scene will not cause the algorithm to loose the original face unless the other faces or hands occlude the original face.

During our experiments we have observed that tracking in complex backgrounds is the main disadvantage of the implemented algorithm. As the face moves, background objects producing high color probability distributions will cause the face area grow erroneously. Eye feature detection of more than one person or detecting features of eyes in overlapping face areas and using complex backgrounds during tracking are not feasible for a human-computer interface application. So the above property makes our face tracking algorithm suitable for a real-time eye feature detection system as only one face is of interest and human face is moving on a simple, such as whole white background, during the application. On the other hand, performance of

eye localization using adaptive eigeneyes is dependent on the width information of the rectangle surrounding the color distribution so if distractors occur this will cause incorrect database size which will probably make the system fail to detect eye areas.

5.2.2 Tracking in Variable Ambient Lighting

Implemented face tracking algorithm uses only hue component of the HSV color space. High or low brightness causes errors in color so this will cause degraded performance.

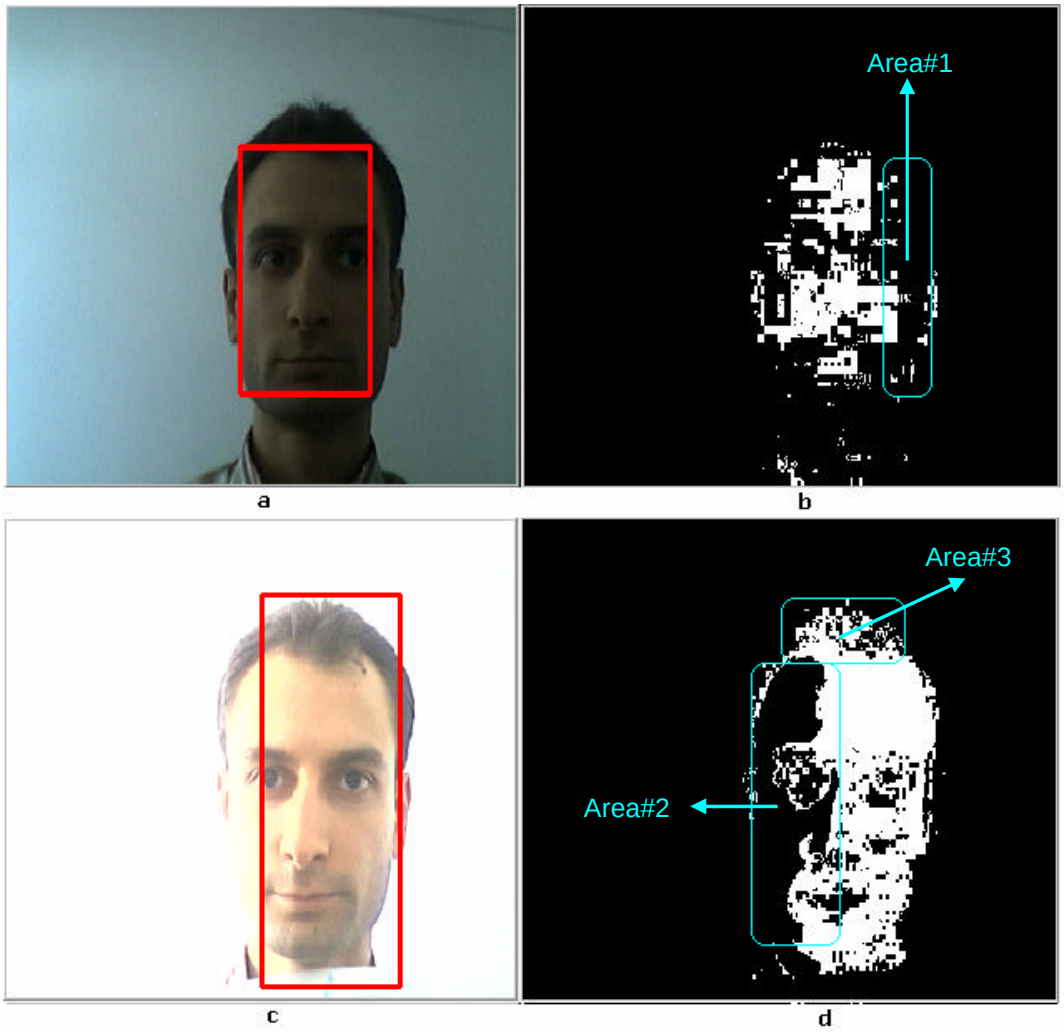

Figure 5.2 Tracking a face under different lighting conditions

The implemeted algorithm has been applied under different illumination conditions. Figure 5.2 summaries the results we have obtained in our experiments. Figure 5.2 – b is the color probability distribution of Figure 5.2 – a and similarly Figure 5.2 – d is

the color probability distribution of Figure 5.2 – c. As seen in Figure 5.2 – b dark areas of the face (Area#1) is ignored by the algorithm causing the face rectangle to be smaller in width. Similarly as seen in Figure 5.2 – d left of the face area (Area#2) which is applied to high brightness is ignored by the algorithm. Another effect of high illumination is error in hue as shown with Area#3. This causes the face rectangle to be longer in height.

During our experiments we have observed lighting variation as another disadvantage of the implemented algorithm. Pixels with high or low brigthness are ignored by the CAMSHIFT. Ignoring parts of the face under high or low illumination causes the face to be partially detected and this will make the eye detector fail as the width information of face rectangle is used for resizing of the eye database before the calculation of eigen basis.

Calibration of the camera before tracking process is an important step in color object tracking; so in our system. Sudden color shifts must be avoided by necessary adjustments and the image brightness must be neither too dim nor saturating.

5.3 Performance of the Eye Detector

Theoretical information on Eigenface method is given in section 3.6 and details of the adapted method, ‘Adaptive Eigeneye Detection’ is explained in section 4.2.2. When the face area is detected, width information of the face area is used to calculate the new size of the training images. After some preprocessing steps face area is search and eye areas are found. During the search, using the distance-from-eye-space measure, the area is either rejected as not an eye or determined to be an eye. Because eye areas do not change radically when projected into the eye space whereas noneye areas appear quite different.

There are three important parameters in Adaptive Eigeneye method;

1) The number of training images in the database to be used which is *M*
2) The number of eigenvectors i.e., eigenfaces which is *M’*
3) The color space component to be used before images are processed

Variance check is also another important parameter during the search of eyes; choosing low variance value means more candidates for decomposition and results in more computation time. On the other hand, choosing high variance value means most of the areas are rejected before decompsiton so eyes may be rejected as noneye.

Effect of each parameter is disccused in detail in the following sections;

5.3.1 Effect of Number of Training Images

Number of training images used during the detection is a very important parameter for Adaptive Eigeneye method. After succesfully detection of face area, to provide scale invariance depending on the width property of the face, eyedatabase is resized. So increasing the number of training images increases the computational time dramatically.

Tests were conducted using the eye database given in Appendix B.1. The number of eigeneyes used were taken as equal to the number of training images and maximum allowable distance from eye space, which was experimentaly obtained before the test, was chosen as 1250. Training images and the face area where the search will be conducted were converted into the XYZ color space and Z component of the images were used in eigeneye method. Detection accuracy and required computation time were measured over 60 consecutive frames. For a frame to be accepted as succesful both eyes shall be located correctly.

Table 5.1 Eye detection performance for different number of training images

Number of Training Images	Detection Rate (%)	Computation time (sec)
3	65	13.84
6	67	14.43
9	81	15.16
12	76	18.34
15	80	19.46

Figure 5.3 Eye tracking with 3 training images

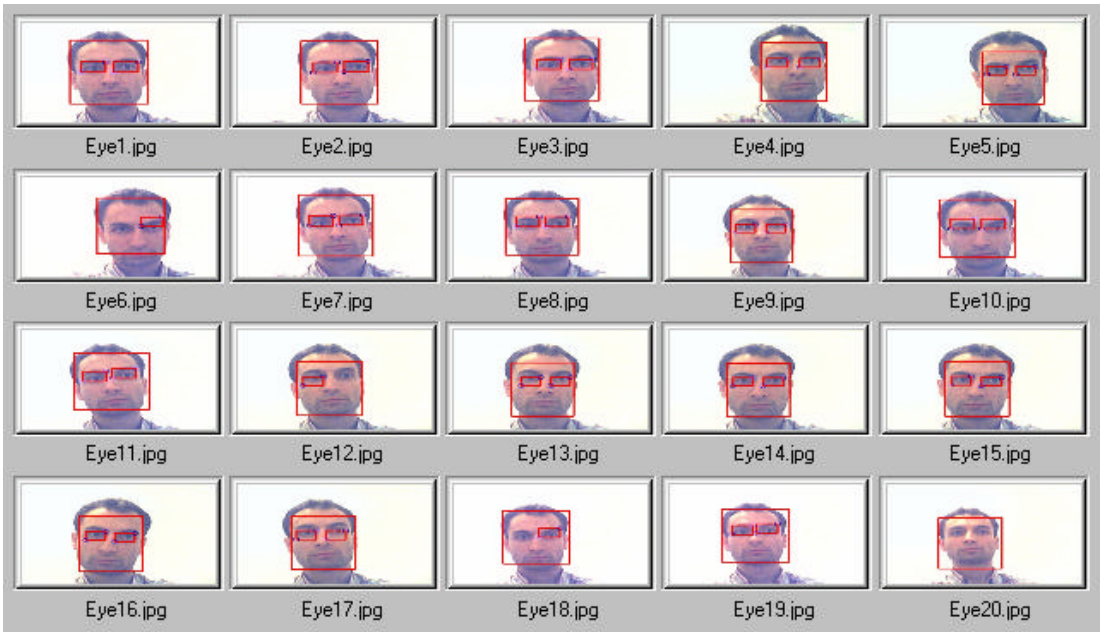

Figure 5.4 Eye tracking with 9 training images

Some test outputs are presented in Figures 5.3 and 5.4. Eye detection accuracy of the adaptive eigeneye method is slightly dependent on the number of the training images because distance of the eye areas to the eye space is usually smaller than other face regions i.e., eye areas do not change radically when projected into the eye space whereas noneye areas appear different. The areas that may be closer to the eye space in some situations are eye brow area and the area where hair begins. However computation time increases with increasing number of training images so using large number of training images should be avoided.

Figure 5.3 is presented to show how the rotation of head affects the results; which is one of the main disadvantages of the eigeneye method.

5.3.2 Effect of Number of Eigenvectors

Maximum number of eigenvectors that can be used is the number of training images M. However, in practice a smaller set of eigenvectors is sufficient for detection processes because eigenvectors with small eigenvalues have very little effect on the detection accuracy.

After observing the results of the previous section, the number of training images used was taken as 9 and the number of eigenvectors as *M'=9* during the test. Maximum allowable distance from eye space was chosen as 1250. Training images and the face area where the search will be conducted were converted into the XYZ color space and Z component of the images were used in eigeneye method. Detection accuracy and required computation time were measured over 60 consecutive frames. For a frame to be accepted as succesful both eyes shall be located correctly.

Table 5.2 Eye detection performance for different number of eigenvectors

Number of Eigenvectors	Detection Rate (%)	Computation time (sec)
1	81	14.12
3	84	12.51
5	85	15.92
7	84	16.48
9	75	16.94

Inspecting Table 5.2 we can see that increasing the number of eigenvectors does not increase the performance of the detection. Moreover using a small set of eigenvectors such as 2 or 3 give quite satisfying results for the number of detections and computation time.

Test results obtained in sections 5.3.1 and 5.3.2 made us choose the number of training images as 9 and the number of eigenvectors as 5 in the system.

5.3.3 Effect of Color Space

Image enhancements before continuing with more complex algorithms provide better results in further steps. In our work one of the enhancements we tried was using different components of the image after it was converted from RGB color space to another color space. Results of several color spaces during adaptive eigeneye detection is presented below.

Figure 5.5 Eye tracking using gray-scale images

Figure 5.6 Eye tracking using Z component of XYZ color space

Figure 5.7 Eye tracking using Hue component HSV color space

Figure 5.8 Eye tracking using L component of Lab color space

Most of the eigenface methods use gray scale images during decomposition process as the image has to be 2 dimensional. However we obtained better results than gray scale images with other color space components. Z component of the XYZ color space is the most effective component. It is robust to noise and reduces the effect of low or high ambient lightining.

5.4 Performance of the Eye Feature Extractor

5.4.1 Performance of Region#1 Detection

Binary thresholding the Z component of the XYZ color space and using the edge image for circle detection is a powerful way of detecting the area formed by the eye pupil and iris. The color model eliminates much of the noise and irregular lightining in the eye area and by thresholding whole image is partitioned into two regions allowing to distinguish region#1 from the rest of the image. To prove the robustness of the XYZ color model and thresholding, eye area was partitioned under different lightining conditions.

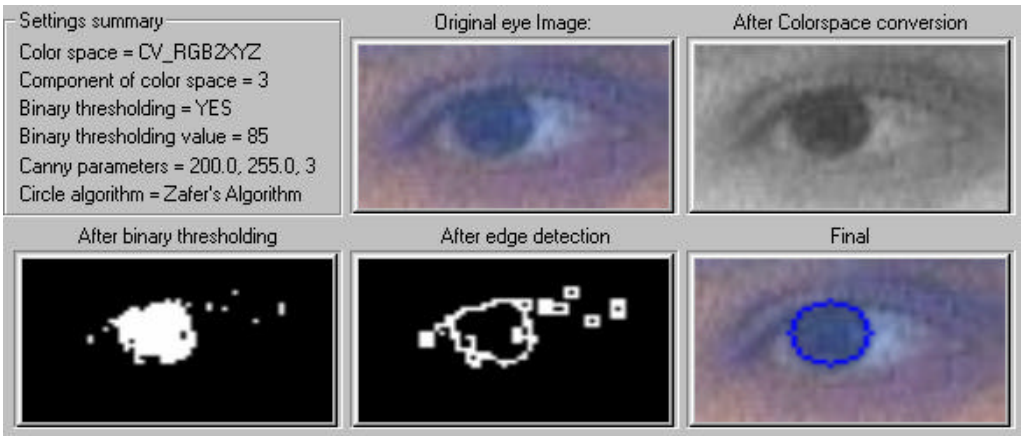

Figure 5.9 Detecting Region#1 under normal illumination

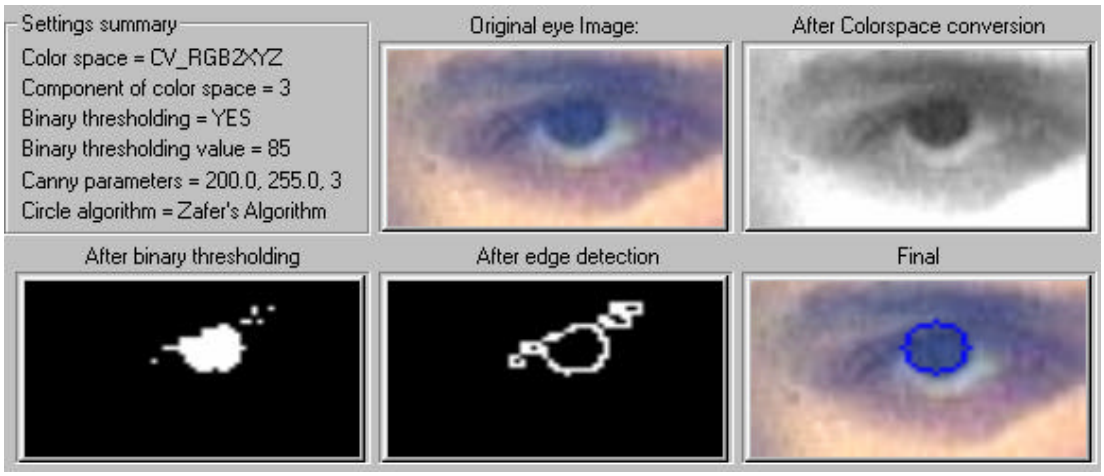

Figure 5.10 Detecting Region#1 under high illumination

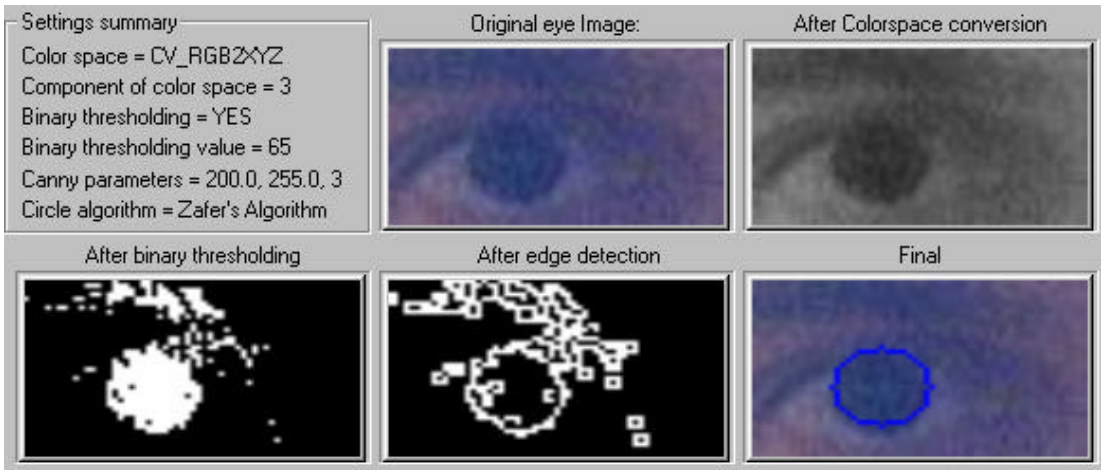

Figure 5.11 Detecting Region#1 under low illumination

Looking at Figures 5.9 – 5.11 we see that thresholding value has to be small under poor illumination. Also small areas around region#1 are also partitioned as region#1 but limiting the search area for center candidates and edges voting for center coordinates eliminates their effect succesfully.

5.4.2 Performance of Region#2 Detection

In this section first of all, performance of the color space component used to detect region#2 under different lightining conditions is given and then the effect of using different coefficients for snake internal and external energies are explained.

5.4.3 Performance of the Color Space

Adaptive thresholding of the hue component of HSV color space is used for detecting the region formed by whites and blacks of the eye. Eliminating the skin color by thresholding the hue component is an effective way of detecting region#2 but color is effected by low and high illumination which makes the detection give erroneous results.

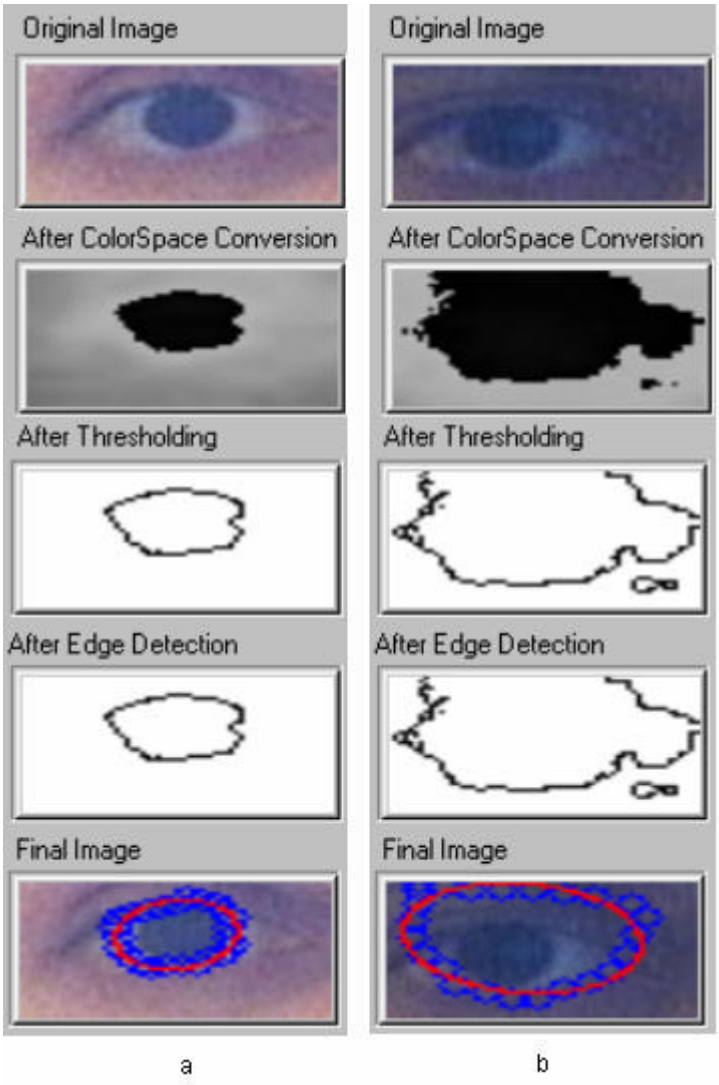

Figure 5.12 a) Detecting Region#2 under high illumination, b) Detecting Region#2 under low illumination

As we can see from the low illuminated image, dark skin areas around the eye are also partitioned as eyeball area which causes problems in outline detection using snakes.

5.4.4 Effect of Coefficients During Snake Energy Minimization

An active contour (Snake) is used for detecting the boundary of the region#2. Our template have three types of energies; elastic and bending energies which are internal energies and external energy. Effect of each energy term can be increased/decreased by adjusting the coefficient related with the energy term. Some results are presented in Figure 5.13.

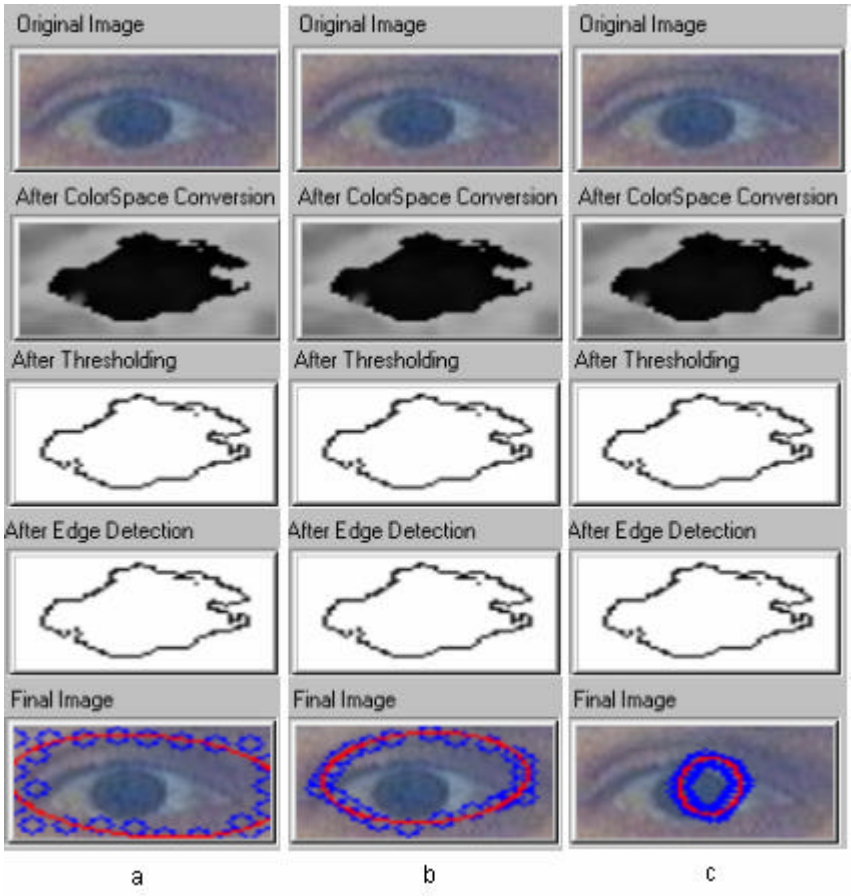

Figure 5.13 a) alpha=0, beta=0.15, gamma=0.2, b) alpha=0.2, beta=0, gamma=0.2, c) alpha=0.2, beta=0.15, gamma=0

Gamma is the coefficient related with image gradient energy. So it must be chosen as a positive value to detect edges. Ignoring the external energy means ignoring the edges in the image.

5.4.5 Performance of the Whole System

In this study, a real-time eye tracking and feature extracting system is developed. The performance of each step play an important role for the final output. In the final step features of eyes region#1 and region#2 are extracted and other indications such as direction of gaze that can be useful for human-computer interaction applications are calculated. In this section performance of the whole system and some outputs will be given.

Final system has been tested with the optimum parameters obtained above and we achieved a frame rate of about 5 frames/second. The developed system was tested with the setup given in section 5.1 and we believe that the frame rate will be higher with increasing processor speeds.

One advantage of the system is that; it is independent of the user. The system gives almost the same result for different people because the whole system is dependent on skin color detection and eye area detection in the this skin region. As eyes are almost the same for most of the people, outputs and performance will not vary too much from person to person. A result for a different person is given in Figure 5.14.

Figure 5.14 Eye area detection results for an unknown person

Variaous frames obtained during the test demonstrate the accurate tracking performance the system when ideal conditions are available (Right to left and then down). As it can be seen from picture#6, features are found incorrectly when the face is localized incorrectly. Also the region#2 detection performance is highly effected with the varying lightining. Parameters used for this test were;

Table 5.3 Test parameters for ideal condition testing

Parameter	Value
Number of training images	9
Number of eigen eyes	5
Minimum standard deviation of eye area	10
Maximum allowable distance from eye space	1250
Color space and component used during the test	Z component of XYZ for eigeneye detection and region#1 H component of HSV for region#2
Binary threshold value for region#1 detection	105
Adaptive threshold parameters for region#2 detection	Threshold value : -12 Block size : 3

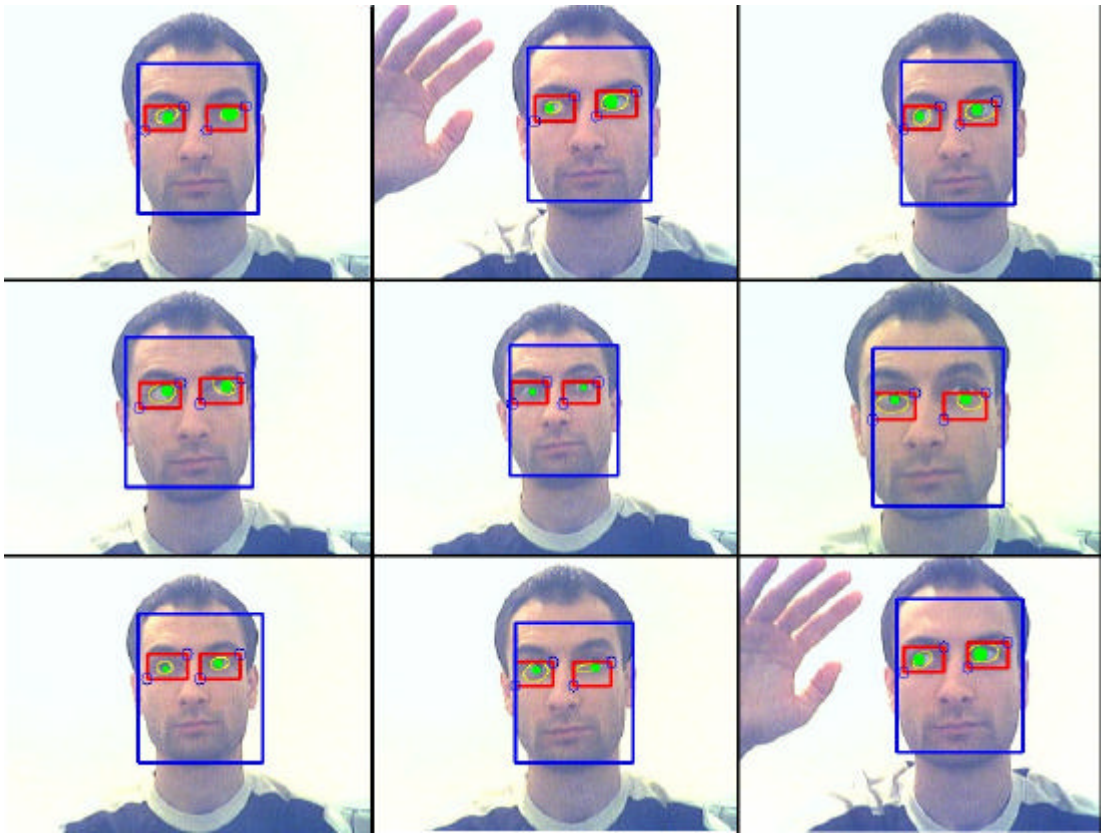

Figure 5.15 Performance of the system – Ideal conditions

To depict the scale and rotation invariance capability of the system, we have tested it with the face located far away from the camera and having a roll angle = 45^{o}. Eyes in face images having roll angles greater than 45^{o} probably will not be located beacuse of the incorrect width information supplied to the adaptive eigeneye detection. On the other with small angles, eyes can be located and region#1 can be detected whereas region#2 is not suitable for small eye areas. Scale invariance property of the system is provided by both the rotated training eye images and the eigen method itself. Some frames of the test are shown in figure 5.16.

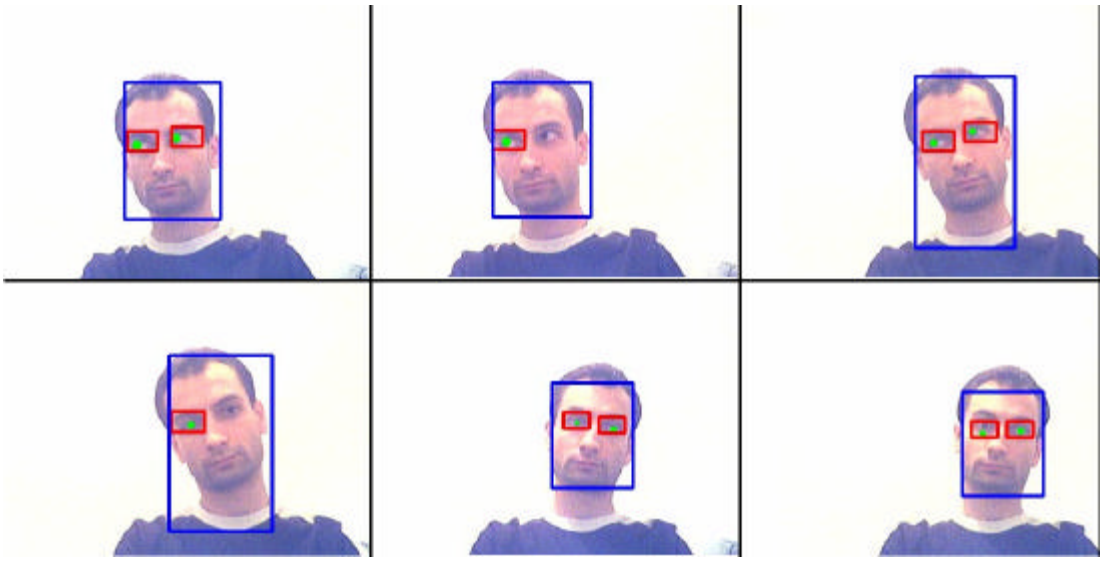

Figure 5.16 Performance of the system – When face is far away from the camera

The system has also been tested with people having occulusions on their faces such as glasses. In most of the cases occlusions make the algorithm fail to detect faces, even when the eye areas localized correctly, features of the eyes can not be correctly extracted. A sequence of frames is presented in Figure 5.17.

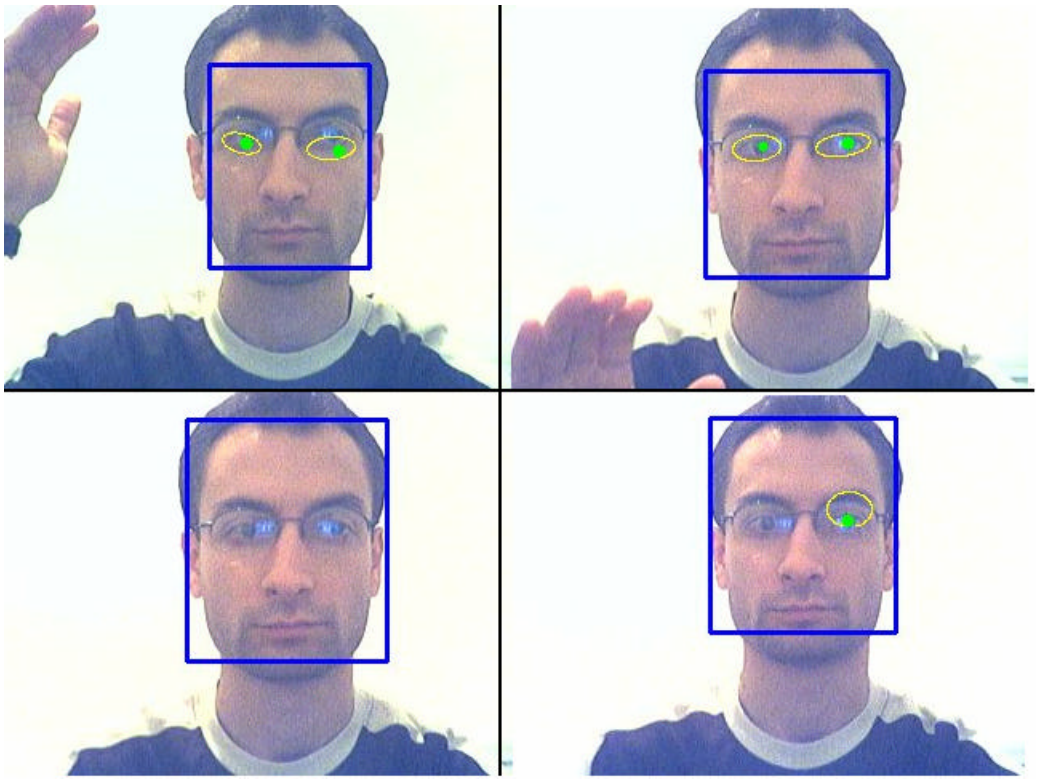

Figure 5.17 Performance of the system when the user has glasses

CHAPTER 6

CONCLUSIONS

In this study, a real-time system capable of face tracking, eye area tracking and extracting eye features was developed. The system's face tracking algorithm is based on CAMSHIFT algorithm developed by [4] which is a feature-based face tracking algorithm searching for the color of the human skin in images. Detection of eye areas after face localization is based on Principal Components Analysis technique which was previously used for face recognition. In order to detect eyes with different sizes without so much computational work and without using databases of different sizes for each image, we have developed an eigenface based method called adaptive eigeneye method. After detection of the eye areas, features of the eyes, region#1 which is the area containing eye pupil and iris and region#2 which is the visible eyeball area, are determined. The extraction of detailed location and shape information of eye features is performed by applying different image preprocessings and region segmentation based on boundary estimation using edge features. Boundary of the final regions are found by circle detection algorithms or by an active contour (Snake).

The system works well with cheap cameras and does not require camera lens calibration. Although the system performance is highly dependent on the system settings done by the user before the tracking process, it is very easy to make these adjustment by the help of friendly user interface.

6.1 Limitations and Future Work

The system we have developed is not fully automatic; there are several adjustment the user has to perform before the tracking process;

- Settings of the video camera shall be done to avoid sudden colors shifts which effects face tracking negatively.
- The user must define the hue of the face before face tracking algorithm.
- The eye database must be built before eye detection
- The user must define the thresholding levels for region eye feature detection.

These steps makes our system user dependent. Future work may be a less user dependent system by performing automatic adjustments.

Considering each step of the system;

- The performance of the whole system is highly dependent on the face area detected by the CAMSHIFT algorithm. Errors occuring in this step result in failing to detect or detecting very small areas as eye areas. Main disadvantages of the face tracking step are; the system is highly effected by the lighting variations and distractors around the person's face. Especially tracking in complex backgrounds with cheap and low quality video cameras is the main disadvantage of the implemented algorithm. So better approaches eliminating the lightening and distractors may be used in future implementations.
- Adaptive eigeneye method is a fast way of eye detection however its performance is dependent on the width information supplied by the face detection step.
- Features of eyes are detected by region partitioning using edge properties of the eye areas. Deformable template matching may be a better way to extract the eye features.

With this study it is possible to calculate the face, eye locations on the whole image and determine the eye movements indicating the direction of gaze. By processing these outputs, future work may be the design of user interfaces based on eye tracking.

6.2 Application Areas

Eye tracking and eye movement analysis may be particularly interesting for the applications depending on user-computer dialogue since they represent measures which can provide valuable information about the visual and attentional aspects of human.

Jacob [1] presented many application areas of eye tracking in advanced interface designs. Applications using our work as a starting point may be used in a varity of areas including: Computer Science, Physhology, Industrial Engineering, Human Factors, Marketing and Advertising. It can be used as a high speed positioning tool as it tells where the user's interest is focussed. It is an effective method when the hands of the user are somehow disabled.

Some specific applications may be listed as;

- Aviation : Analysis of eye movements illustrates the importance of the Primary Flight Display in a modern glass cockpit as they are the primary source of information during a flight. So analysis of the pilots eye movements may be used in pilot performance testing and training of the novice pilots. Eye movements may also be used to evaluate the usability of new instruments on the aircraft such as target selection, arming systems and Electronic Moving Maps (EMM) in which the pilot navigates the whole map using eye movements.
- Driver monitoring systems : Lack of visual attention plays an important role in road traffic accidents. So monitoring the eye states and gaze direction can be used as an accident countermeasure system.
- Marketing and Advertising : Eye tracking can provide insight into how much visual attention the consumer pays in different forms of advertisings in print medias or television systems.
- Interactive information systems : Eye movement based informational display systems in which the user uses his/her eyes as a pointing device such as a mouse pointer to navigate visually selectable items.

- Research related applications : It can be used as a tool during the researches for the relation between the eye movements and cognitive tasks; eye movements are believed to be an important part of modeling a person's inner strategies for decision making, learning, motivation and memory. According to the results of this research it can be used to determine the mental state and pyshology of the person or how truthful a person is being.
- It can be used for helping disabled people, such as quadriplegics, who can move their eyes much more effectively then they could operate any other input device.
- Security systems based on iris recognition.
- Clinical diagnostics.
- Computer gaming.

REFERENCES

[1] R.J.K. Jacob, (1991). “The use of eye movements in human-computer interaction techniques”. ACM Transactions on Information Systems, 9(3):152–169.

[2] Howard Anton, (1987). “Elementary Linear Algebra 5e”. John Wiley & Sons Inc, ISBN 0-471-85223-6.

[3] J.C. Terrillon, M. David, S. Akamatsu, (1998). “Automatic Detection of Human Faces in Naturel 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.

[4] Gary R. Bradski, “Computer Visison Face Tracking for Use in a Preceptual User Interface”. Microcomputer Research Lab, Santa Clara, CA, Intel Corporation

[5] K. M., W. A. and T. D., (1996). “Snakes: Active Contour Models”. 1st Conference on Computer Vision, pp. 259-268.

[6] M. Sonka, V. Hlavac, and R. Boyle, (1999). Image Processing, Analysis, and Machine Vision. Brooks/Cole Publishing Company, second ed.

[7] J. Ivins and J. Porrill, (2000). “Everything you always wanted to know about snakes (but were afraid to ask)". AIVRU Technical Memo 86, Artifcial Intelligence Vision Research Unit, University of Shefield, England.

[8] D. J. Williams and M. Shah, (1992). “A fast algorithm for active contours and curvature estimation", CVGIP: Image Understanding, vol. 55, pp. 14-26.

[9] E. Hjelmas and B. K. Low, (2001). “Face detection: A survey”. Computer Vision and Image Understanding, vol. 83, pp. 236-274.

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

[11] K. Lam and H. Yan, (1994). “Fast Algorithm for Locating Head Boundaries”. J. Electronic Imaging, vol. 3, no. 4, pp. 351-359.

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

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

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

[15] J. Canny, (1986). “A Computational Approach to Edge Detection”. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 679-698.

[16] S.A. Sirohey, (1993). “Human Face Segmentation and Identification”. Technical Report CS-TR-3176, Univ. of Maryland.

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

[18] T.S. Jebara and A. Pentland, (1997). “Parameterized Structure from Motion for 3D Adaptive Feedback Tracking of Faces”. Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 144-150.

[19] Y. Miyake, H. Saitoh, H. Yaguchi, and N. Tsukada, (1990). “Facial Pattern Detection and Color Correction from Television Picture for Newspaper Printing”. J. Imaging Technology, vol. 16, no. 5, pp. 165-169.

[20] D. Saxe and R. Foulds, (1996). “Toward Robust Skin Identification in Video Images”. Proc. Second Int’l Conf. Automatic Face and Gesture Recognition, pp. 379-384.

[21] R. Kjeldsen and J. Kender, (1996). “Finding Skin in Color Images”. Proc. Second Int’l Conf. Automatic Face and Gesture Recognition, pp. 312-317.

[22] K. Sobottka and I. Pitas, (1996). “Face Localization and Feature Extraction Based on Shape and Color Information”, Proc. IEEE Int’l Conf. Image Processing, pp. 483-486.

[23] H. Wangand S.-F. Chang, (1997). “A Highly Efficient System for Automatic Face Region Detection in MPEG Video”. IEEE Trans. Circuits and Systems for Video Technology, vol. 7, no. 4, pp. 615-628.

[24] Y. Dai and Y. Nakano, (1995). “Extraction for Facial Images from Complex Background Using Color Information and SGLD Matrices”. Proc. First Int’l Workshop Automatic Face and Gesture Recognition, pp. 238-242.

[25] Q. Chen, H. Wu, and M. Yachida, (1995). “Face Detection by Fuzzy Matching”. Proc. Fifth IEEE Int’l Conf. Computer Vision, pp. 591-596.

[26] T. Sakai, M. Nagao, and S. Fujibayashi, (1969). “Line Extraction and Pattern Detection in a Photograph”. Pattern Recognition, vol. 1, pp. 233-248.

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

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

[29] M. Turk and A. Pentland, (1991). “Eigenfaces for Recognition”. J. Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86.

[30] R. O. Duda, P. E. Hart, and D. G. Stork, (2001). “Pattern Classification”. John Wiley and Sons, Inc., second ed.

[31] F.S. Samaria, (1994). “Face Recognition Using Hidden Markov Models”. PhD thesis, Univ. of Cambridge.

[32] R. Kothari and J.L. Mitchell (1996). “Detection of eye locations in unconstrained visual images”. Int Proc. International Conference on Image Processing, volume I, pages 519–522, Lausanne, Switzerland.

[33] A. Yuille, C. D.S., and H. P.W., (1989). “Feature extraction from faces using de formable templates". Int. Proc. CVPR, pp. 104-109.

[34] A. Blake and A. Yuille, (1992). “Active Vision”. Massachusetts Institute of Technology.

[35] P. Fieguth and D. Terzopoulos, (1997). “Color-based tracking of heads and other mobile objects at video frame rates”. In Proc. of IEEE CVPR, pp. 21-27.

[36] C. Wren, A. Azarbayejani, T. Darrell, A.Pentland, (1995). “Pfinder: Real-Time Tracking of the Human Body”, SPIE Vol. 2615.

[37] K. Fukunaga, (1990). “Introduction to Statistical Pattern Recognition”. Academic Press, Boston.

[38] M. Önder, (2004). “Face Detection in active Robot Vision”, Msc Thesis. METU.

[39] H. Serçe, (2003). “Facial Feature Extraction Using Deformable Templates”, Msc Thesis. METU.

[40] R. Jain, R. Kasturi, B. G. Schunck, (1995). “Machine Vision”, pp. 73-75. Co-published by the MIT Press and McGraw-Hill, Inc.

[41] Lindsay I. Smith, (2002). “A tutorial on Principal Components Analysis”. www.cs.otago.ac.nz/cos453/student_tutorials/principal_components.pdf

[42] Intel® Open Source Computer Vision Library. www.intel.com/research/mrl/research/opencv

APPENDIX A

OBJECT RECOGNITION LIBRARY OF INTEL OPENCV

Trackeye software was developed in Microsoft Visual C++ 6.0 environment and uses the object recognition library of Intel OpenCV [42]. In this section details of the functions are presented.

A.1. CalcEigenObjects

This function calculates orthonormal eigen basis averaged object for group of input objects.

```
void cvCalcEigenObjects( int nObjects, void* input, void* output, int
ioFlags, int ioBufSize, void* userData, CvTermCriteria* calcLimit, IplImage*
avg, float* eigVals );
```

`nObjects`
: Number of source objects.

`input`
: Pointer either to the array of *IplImage* input objects or to the read callback function according to the value of the parameter *ioFlags*.

`output`
: Pointer either to the array of eigen objects or to the write callback function according to the value of the parameter ioFlags .

`ioFlags`
: Input/output flags.

`ioBufSize`
: Input/output buffer size in bytes. The size is zero, if unknown.

`userData`
: Pointer to the structure that contains all necessary data for the callback functions.

`calcLimit`
: Criteria that determine when to stop calculation of eigen objects.

`avg`
: Averaged object.

`eigVals`
: Pointer to the eigenvalues array in the descending order; may be *NULL* .

The function cvCalcEigenObjects calculates orthonormal eigen basis and the averaged object for a group of the input objects. Depending on *ioFlags* parameter it may be used either in direct access or callback mode. Depending on the parameter *calcLimit*, calculations are finished either after first *calcLimit.maxIters* dominating eigen objects are retrieved or if the ratio of the current eigenvalue to the largest eigenvalue comes down to *calcLimit.epsilon* threshold. The value *calcLimit -> type* must be *CV_TERMCRIT_NUMB*, *CV_TERMCRIT_EPS*, or *CV_TERMCRIT_NUMB | CV_TERMCRIT_EPS* . The function returns the real values *calcLimit -> maxIter* and *calcLimit -> epsilon* .

The function also calculates the averaged object, which must be created previously. Calculated eigen objects are arranged according to the corresponding eigenvalues in the descending order. The parameter *eigVals* may be equal to *NULL*, if eigenvalues are not needed. Example usage of the function is given below;

```
const int nImages = 10;

IplImage** images = (IplImage**)malloc(sizeof(IplImage*)*numOfImages);
IplImage** eigens = (IplImage**)malloc(sizeof(IplImage*)*numOfImages);

for (int i=0; i<nImages; i++)
{
        fileName.Format("C:\\Eye%d.jpg", i);
        images[i] = cvLoadImage(fileName, -1);
}

for (i=0; i<settings->params->nImages; i++)
        eigens[i] = cvCreateImage(cvGetSize(images[0]), IPL_DEPTH_32F, 1);

CvTermCriteria criteria;
criteria.type = CV_TERMCRIT_ITER|CV_TERMCRIT_EPS;
criteria.maxIter = 13;
criteria.epsilon = 0.1;

IplImage* averageImage;
averageImage = cvCreateImage(cvGetSize(images[0]), IPL_DEPTH_32F, 1);

float* vals = (float*)malloc(sizeof(float)*nImages);

cvCalcEigenObjects(nImages, images, eigens, 0, 0, 0, &criteria,
averageImage, vals );
```

A.2. EigenDecomposite

This function calculates all decomposition coefficients for input object

```
void cvEigenDecomposite( IplImage* obj, int nEigObjs, void* eigInput, int
ioFlags, void* userData, IplImage* avg, float* coeffs );
```

`obj`
: Input object.

`nEigObjs`
: Number of eigen objects.

`eigInput`
: Pointer either to the array of *IplImage* input objects or to the read callback function according to the value of the parameter *ioFlags*.

`ioFlags`
: Input/output flags.

`userData`
: Pointer to the structure that contains all necessary data for the callback functions.

`avg`
: Averaged object.

`coeffs`
: Calculated coefficients; an output parameter.

The function cvEigenDecomposite calculates all decomposition coefficients for the input object using the previously calculated eigen objects basis and the averaged object. Depending on *ioFlags* parameter it may be used either in direct access or callback mode. Example usage of the function is given below;

```
IplImage* Image2Comp; // The image for which the coeffs will be calculated

const int nEigens = 9;

// eigens were calculated using “cvCalcEigenObjects” before
// averageImage was calculated using “cvCalcEigenObjects” before

float* weights = (float*)malloc(sizeof(float)*nImages);

cvEigenDecomposite( Image2Comp, nEigens, eigens, 0, 0, averageImage, weights
);
```

A.3. EigenProjection

This function calculates object projection to the eigen sub-space

```
void cvEigenProjection( int nEigObjs, void* eigInput, int ioFlags, void*
userData, float* coeffs, IplImage* avg, IplImage* proj );
```

`nEigObjs`
: Number of eigen objects.

`eigInput`
: Pointer either to the array of *IplImage* input objects or to the read callback function according to the value of the parameter *ioFlags*.

`ioFlags`
: Input/output flags.

`userData`
: Pointer to the structure that contains all necessary data for the callback functions.

`coeffs`
: Previously calculated decomposition coefficients.

`avg`
: Averaged object.

`proj`
: Decomposed object projection to the eigen sub-space.

The function cvEigenProjection calculates an object projection to the eigen sub-space or, in other words, restores an object using previously calculated eigen objects basis, averaged object, and decomposition coefficients of the restored object.

Depending on *ioFlags* parameter it may be used either in direct access or callback mode. The functions of the eigen objects group have been developed to be used for any number of objects, even if their total size exceeds free RAM size. So the functions may be used in two main modes. Direct access mode is the best choice if the size of free RAM is sufficient for all input and eigen objects allocation. This mode is set if the parameter *ioFlags* is equal to *CV_EIGOBJ_NO_CALLBACK* . In this case *input* and *output* parameters are pointers to arrays of input/output objects of *IplImage** type. Exmaple usage of the function is given below;

```
// eigens, nEigens, weights, averageImage were declared and calculated using
// the functions above

IplImage* projection = cvCreateImage(cvGetSize(images[0]), IPL_DEPTH_8U, 1);

cvEigenProjection( eigens, nEigens, CV_EIGOBJ_NO_CALLBACK, 0, weights,
averageImage, projection );

// As a result of the steps above euclidian distance between the images
// image2comp and projection is calculated and result is compared to the
// threshold to find whether image2comp is e.g. an eye or not.
```

APPENDIX B

TRAINING EYE IMAGES USED DURING THE TESTS

As explained in section 4.2.1 The training database contains different poses and orientations of both eyes. A typical eye database contains eye images for left and right eyes when the person is staring at different directions and when the head has a negative/positive roll angle. In our system training database can be easily built by the user before the tracking process begins. Eye databases do not have to be complex eye images like the ones used in recognition processes and a training database for a person can also be used in tracking the eyes of another person as the appearance of eyes does not differ much from person to person.

B.1. Eye Database – 1

Total number of images : 20

Size of each image : 40x25

Number of images for each eye : 10

Figure B.1 Eye database – 1

B.2. Eye Database – 2

Total number of images : 17

Size of each image : 40x25

Number of images for each eye : 8

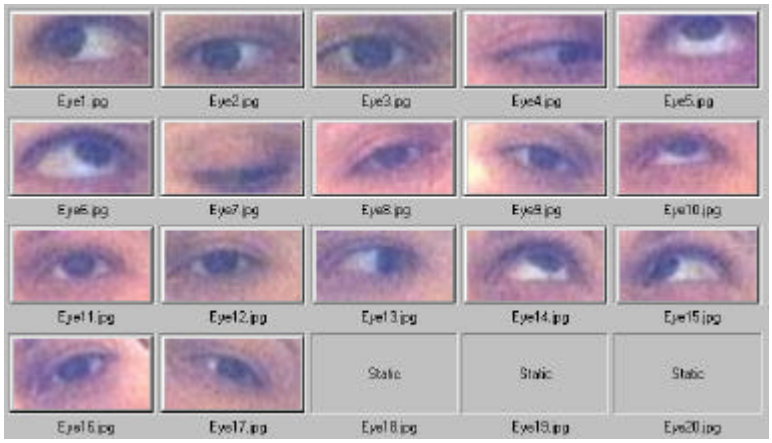

Figure B.2 Eye database – 2