2012-01-01

A Robust Real Time Eye Tracking And Gaze Estimation System Using Particle Filters

Tariq Iqbal
University of Texas at El Paso, tiqbal@miners.utep.edu

A ROBUST REAL TIME EYE TRACKING AND GAZE ESTIMATION SYSTEM USING PARTICLE FILTERS

TARIQ IQBAL

Department of Computer Science

APPROVED:

Olac Fuentes, Ph.D., Chair

Christopher Kiekintveld, Ph.D.

Sergio Cabrera, Ph.D.

Benjamin C. Flores, Ph.D.
Interim Dean of the Graduate School

to my

MOTHER and FATHER

and my WIFE

with love

A ROBUST REAL TIME EYE TRACKING AND GAZE ESTIMATION SYSTEM USING PARTICLE FILTERS

by

TARIQ IQBAL

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

August 2012

Acknowledgements

I would like to express my deep-felt gratitude to my advisor, Dr. Olac Fuentes of the Computer Science Department at The University of Texas at El Paso, for his advice, encouragement, enduring patience and constant support. He was never ceasing in his belief in me and always providing clear guidance when I was lost, constantly driving me with energy.

I would like to thank Dr. Christopher Kiekintveld of the Computer Science Department, at The University of Texas at El Paso. His suggestions, comments and additional guidance were invaluable to the completion of this work.

I also wish to thank Dr. Sergio Cabrera of the Electrical and Computer Engineering Department, at The University of Texas at El Paso, for being on my committee and for his constructive feedback.

Additionally, I want to thank The University of Texas at El Paso Computer Science Department professors and staffs for all their hard work and dedication, providing me the means to complete my degree.

Also I would like to thank Alexander Knaust, Geovany Ramirez, and the rest of the members of Vision and Learning Laboratory at The University of Texas at El Paso for their comments and suggestions.

Last but not the least, I would like to thank my wife, Liza, for her immense support during this time of my life.

Abstract

Eye tracking and gaze estimation techniques have been extensively investigated by researchers in the computer vision and the psychology community for the last few decades. Still it remains a challenging task due to the individuality of the eyes, variability in shape, scale, location, and lighting conditions. Eye tracking has many applications in neuroscience, psychology, and human-computer interaction. Gaze estimation plays a vital role in the field of human attention analysis, human factors in industrial engineering, marketing and advertising, human cognitive state analysis, gaze-based interactive user interfaces, and monitoring driver vigilance systems.

Eye-tracking technology was originally a pioneer method in reading research, but has led to the development of many different methods to track eye movements and estimate gaze. Historical eye-tracking technologies include Electrical Oculography (EOG) and Coil-Systems, which require additional electrical hardware mounted on the skin or specialized contact lenses to measure the eye movements. These methods proved quite invasive, instead current commercial eye trackers and gaze-estimation systems use video images of the eyes along with additional hardware, such as infrared sensors.

In this thesis, we design and implement a low-cost eye-tracking system using only an off-the-shelf webcam. This eye-tracking system is robust to small head rotations, changes in lighting, and all but dramatic head movements. We also design and implement a gaze-estimation method which uses a simple yet powerful machine learning-based calibration technique to estimate gaze positions.

The empirical evidence shows that our implemented eye tracker can track both eyes successfully and detect the pupil accurately in about 83% of frames regardless of the head movements and different lighting conditions. Additionally, we observe that the overall accuracy of the gaze estimation system is about 88% of the times throughout our experiments - albeit in extremely low resolution.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Eye tracking is the science of measuring the movement of the eyes. Humans move their eyes to bring a portion of their visual field into high resolution, so that they can see the fine detail of that portion. Eyes also move in response to visual, auditory or cognitive stimulus.

Eye tracking and gaze estimation have many applications in neuroscience, psychology, and human-computer interaction. A group of researchers are interested in the physiological movements of the eyes; they are not concerned about where the person is looking in space. On the other hand, another group of researchers are interested in gaze position. The gaze position is the point in the persons field of view where the person is actually looking [28]. Current commercial eye trackers use cameras along with additional hardware, such as infrared sensors. In this thesis, we present design and implementation of a low-cost eye-tracking system using only an off-the-shelf camera which is robust to small head rotations, changes in lighting, and all but dramatic head movements. We also present design and implementation of a gaze-estimation system which uses a simple yet powerful machine learning-based calibration technique to estimate gaze positions.

This chapter provides a brief introduction to the human visual system, a high-level discussion on different types of eye movements and various types of eye-tracking methods used by the research community, a statement of the contribution of this thesis, and a brief overview of the following chapters. Section 1.1 briefly describes the human visual system. Sections 1.2 and 1.3 present different types of eye movements and various eye tracking methods. The research contribution of this thesis and a high level overview of the remaining chapters are in sections 1.4 and 1.5 respectively.

1.1 Human Visual System

Humans see the world with their eyes, which are located side by side in close proximity. Both eyes see the same object in the world separately, each eye producing an independent signal representing its visual field. The human brain then produces a unified image from these signals. Different parts of a typical eye region are shown in figure 1.1.

Light rays are reflected from an object and enter the eyes through the cornea. The cornea is the transparent covering and a focusing system of the eye. In the visible portion of the eye, the iris is the colored portion which regulates the amount of light that passes into the eye through the pupil. The pupil is the small hole within the iris that allows the light to pass through. It is an adjustable opening in the center of the iris, which actually regulates the amount of light that passes through. Usually the pupil is darker than the iris regardless of the color of the iris. The light rays then pass through the lens. These lens can change its shape, so it can further bend the rays and focus them on the retina at the back of the eye. A thin layer of tissue called the retina is at the back of the lens inside the eye. Many light-sensing nerve cells called rods and cones are situated in this area. Cones, about 6% of the total number of cells, are concentrated in an area called the macula. Cones provide clear vision and help capture the colors and fine details of an object. Rods, which make up about 94% of the total light-sensing nerve cells, are located outside the perimeter of the macula and all over the outer edge of the retina. Rods are very sensitive to light; they help humans see in dim light and at night. They also provide peripheral vision which helps a person to detect motion. Light is converted into electrical signals by rods and cones. The optic nerves send these electronic signals to the brain which produces the image from these signals [1]. A mapping of eye features is shown in figure 1.2.

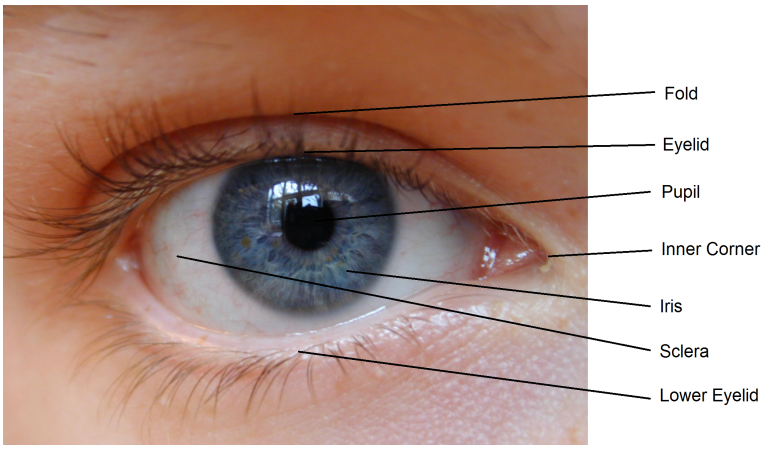

Figure 1.1: Human Eye

1.2 Types of Eye Movements

Our eyes are constantly moving, even when we are asleep. Eye movements can be divided into many different classes, and some common ones are described below:

Fixation:

Fixation is a relatively low velocity eye movement. The eye stays relatively still and is fixated on a certain point to acquire information from a scene; most of the information is acquired during fixation. Fixation may involve very small random drifting eye movements. The total duration of fixation on a point varies from 120 ms to 1000 ms, typically from 200 ms to 600 ms and the typical fixation frequency is less than 3 Hz [28].

Saccade:

Saccade movements are the rapid eye movements that the eyes make while jumping from one point to another. Saccades are also studied as the eye movements between fixations. Saccades usually have high acceleration and velocity. The usual duration of saccade movements is only 40 ms to 120 ms [28].

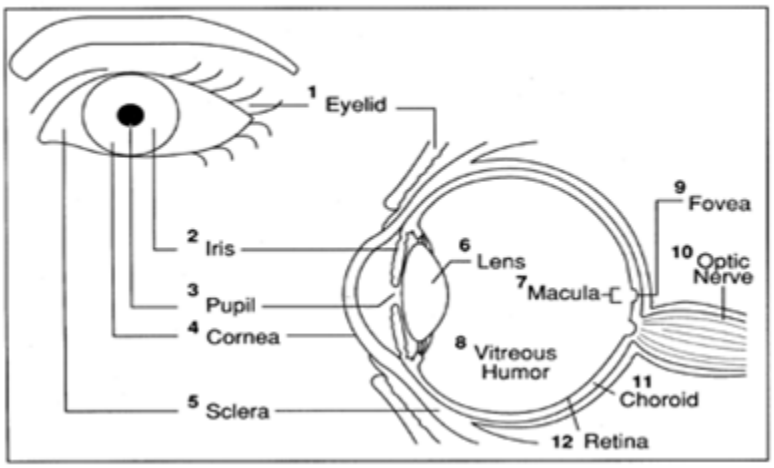

Figure 1.2: Eye Features[1]

Pursuit:

Pursuit movements occur when the eyes follow a moving object in order to keep it in focus. Pursuit movements are involuntary. Normally eyes smoothly track a moving object, but sometimes the eyes perform some rapid saccades to catch up and reacquire focus of the object [28].

1.3 Types of Eye-tracking Methods

There are several methods used by the research community for tracking the eyes. The following section is a rough taxonomy of eye tracking methods.

Electrical Oculography (EOG):

In this method some skin electrodes are placed around the eyes. These electrodes measure the changes in the electrical field as the eyes move, and from these data the eye movements

[1]Courtesy: Antti Aaltonen, Computer Human Interaction Group, Tampere University

are estimated. This method performs better when measuring relative eye movements rather than absolute movements. This system is limited in accuracy and very susceptible to noise.

Coil Systems (Mechanical Method):

A coil eye tracking system estimates the eye movements from the data obtained by a magnetic coil inserted as a part of contact lens. The user needs to wear the contact lens before the study. This method is also susceptible to noise and the coil can be fragile.

Video Image Analysis:

Video image analysis method is now dominating in the field of eye tracking and gaze estimation. A video camera is used in these methods to observe the user's eyes. Image processing techniques are then used to analyze the image and track the eye features. Based on calibration, the system determines where the user is currently looking. A detailed description of different video image analysis method is presented in Chapter 2.

1.4 Contribution of this Thesis

The contributions of this thesis are:

1. Developing a real time eye-tracking system using particle filters which is robust to small head rotations, changes in lighting, and all but dramatic head movements. The real time video image is captured with an off-the-shelf web-camera. A machine learning technique is used to evaluate the particle weights.

2. Developing a gaze-estimation system, which uses a simple yet powerful machine learning-based calibration technique to estimate the gaze region from the output of the eye-tracker.

3. Conducting quantitative study to evaluate the performance of the implemented eye-tracking and the gaze-estimation system.

1.5 Overview of the Next Chapters

In Chapter 2 relevant studies of different eye-tracking and gaze estimation methods are presented. That chapter also covers description of the different ideas and technologies used by researchers. We present a detailed discussion on the proposed design and implementation of the eye-tracking and gaze estimation method in Chapter 3. Then we present experimental results as well as analyses in Chapter 4. Finally, Chapter 5 consists of the concluding remarks and future work.

Chapter 2

Related Work

Eye detection and gaze tracking have been extensively investigated by researchers in the computer vision and the psychology community for the last few decades. Eyes are one of the most salient features of the human face. To understand a person's desires, needs, emotional state, cognitive processes and interpersonal relations, eyes and their movements are very important [7]. Again, in applications like human attention analysis, human cognitive state analysis, gaze-based interactive user interfaces, monitoring driver vigilance system and driver fatigue detection system, gaze estimation and tracking play a vital role [7][15][12].

Video-based eye tracking is divided into three aspects: detecting the eyes in the image, interpreting eye positions from the image and then tracking them in every frame. A general overview of the eye and gaze tracker is shown in figure 2.1 [7].

In the detecting and tracking of the eyes, it is essential to identify the parameters of the model of the eyes from the image data. The eye model can be identified from the intensity distribution of the eye region, iris shapes, eye shapes, head pose etc. Hansen, et al. present a very good and useful review of current progress and state of art in video-based eye detection and tracking methods in [7] and [9]. According to them the eye model detection techniques can be classified as shape-based, feature-based, appearance-based and hybrid methods.

2.1 Shape-based Approach

In shape-based approach the eye is modeled by the iris and the pupil contours and the shape of the eyelids. Many eye tracking applications model the exterior shape of the eye

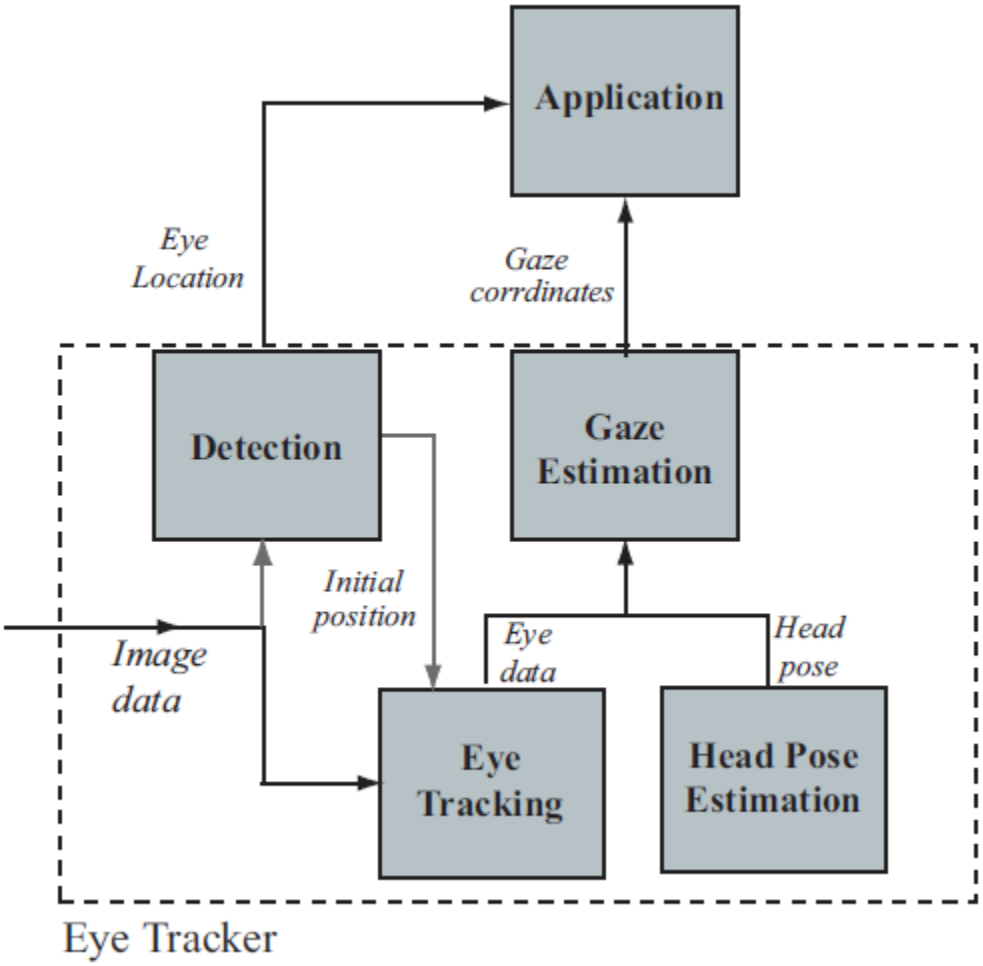

Figure 2.1: Components of Video-based Eye Detection and Eye Tracking ([7])

and the shape of the iris and the pupil as a simple elliptical model. Ellipse parameters are calculated by voting-based methods or model fitting methods [7]. Kim and Ramakrishna use a voting-based approach to model the iris as an ellipse [18]. To find the center of the iris, horizontal scanning is done in the eye region. The pupil is considered as the center of the longest horizontal line. Kothari and Mitchell also model the iris as an elliptical shape [19]. They use spatial and temporal information to detect the eyes. The iris is darker than the white portion of the eye (the sclera). So if the horizontal and vertical gradients are taken, the direction of the gradient will be outward from the center of the iris. Using

this observation, the authors propose a voting-based method to locate the center of the iris. An extrapolated line is drawn at each edge point in the direction opposite to the gradient direction. The approximate center of the iris is obtained by the area through which the maximum number of lines pass. Perez, et al. use thresholding and a contour searching method to determine the iris radius [21]. Hansen and Pece also model the iris as an ellipse, but the model is obtained through an Expectation Maximization and RANSAC optimization [9]. Another ellipse fitting algorithm is proposed by Zhu and Yang [38]. The authors assume that the iris contour points rest on an ellipse and they use these points to estimate the parameters of the ellipse by solving a least-square fitting problem. They also propose a sub-pixel edge detection method to get the accuracy in sub-pixel level.

Instead of modeling the iris as an ellipse, often the iris is modeled as a circle mostly for efficiency reasons. Tian, Kanade, and Cohn use an annulus shape half circle mask to get the radius and center of the iris [26]. The authors use the Canny edge operator [3] to get the edge map, and then use the mask to get the parameter of the circular model of the iris. Sirohey, Rosenfeld, and Duric use a similar approach to get the radius and center of the iris [23]. The author use the gradient magnitude and direction of the edge points in that annulus region to model the iris. To detect the upper-eyelid, they use the edge segments to fit a third-degree polynomial. Wang and Yang model the iris as circle [34]. They use homomorphic filtering, clustering and thresholding to detect the eye region candidate, then a classifier is used to get the eye pair regions from the candidates. Edge information of the eye pair regions is then used to detect the iris by Circular Hough transform. On the other hand, complex shape-based methods try to model the eye shape with more detailed information. Yuille, et al. model the eye region as a deformable template [37]. The authors model the two eyelids as parabolas and the iris as a circle. The deformable model changes its parameters with the change of the eye state by an energy minimization function. Tian, Kanade, and Cohn use the dual-state parametric eye model for detecting the state of the eyes [26]. The authors use their own eye model for the closed eye state, but use a modified version of the model proposed in [37] for the open eye model. A complex eye model is also

used in [33], [32], [36] and [35].

2.2 Feature-based Approach

In feature-based methods the goal is to identify local features i.e. the limbus, pupil, corneal reflection etc. for eye localization and to model the eye parameters [7]. Instead of detecting the eye features, Kawato and Tetsutani try to detect the region between the two eyes [16]. They name it "between-the-eyes" region. Usually the area between the eyes is brighter than the two sides of this region, as the eyes are on both sides of the region and this area contains the nose and the forehead region. Thus this region is easier to track, and the eyes can be localized by searching both sides of this area. They search for the 'between the eyes' template in every frame and localize the eyes depending on this area. In their work, Sirohey and Rosenfeld detect the face region using the skin color [24]. Then in the skin region, the eyes are detected using linear and non-linear filtering. They use linear filtering (four Gabor wavelets) to detect an eye in the face region and non-linear filtering to detect the eye corners.

2.3 Appearance-based Approach

The appearance-based methods are also known as image template-based methods. Single template-based methods are limited by eye movements and changes in the head pose. If the person changes the pose of the head, the single static template-based method fails. Image template-based methods also have problems when there is a change in scale and rotations. Usually a large number of different training examples of eye regions are collected with different eye states, eye orientations and head poses and with different illumination conditions. Then a classifier is modeled using the pixel information of these training examples [7]. Several modified version of Haar feature presented by Viola and Jones in [30] and [31], are also used to detect the eye and extract eye features.

2.4 Hybrid Approach

Hybrid approaches are the combination of the shape-based, feature-based and appearance-based methods. An active appearance model is used by Hansen, et al. in [10] and [8], which is referred to as the combination of shape and appearance models by [7]. Hansen, et al. also propose a combination of particle filtering and expectation maximization algorithms [9]. Particle filtering is used for iris tracking as iris movement is fast and there is no predictable pattern of iris movement. Again, expectation maximization is used for accurately estimating the pose. Color distribution of the face region and eye region is also used to detect and track the eyes. Horng, et al. use color information of the image to detect the face region [12]. Similar approach is also used by Tsekeridou, et al. [27] and Stiefelhagen, et al [25]. After detecting the face region, in [12], the face region is divided into smaller parts to search for the eyes. In the smaller windows the eyes are searched for using edge information and thresholding. Eye tracking is done by template matching. After detecting the skin region, the face region is detected by modeling the face as an elliptical shape and face mirror-symmetry in [27]. In their implementation, the authors search the eyes in the upper half of the face region. Filtering and some relative geometric constrains are used to detect the eyes as well as other facial features. The authors use a two-dimensional Gaussian distribution of a normalized skin color model in [25]. Then, pixel-wise search for skin color is reformed, and the largest connected regions of skin color pixels are considered as the face region. In the skin color region, the eyes are detected as the two dark regions satisfying several anthropometric constraints. Hansen, et al. use color information of the eye model to track the eyes using a mean-shift tracker [10].

Beside these video-based methods, eye detecting under active infrared illumination is the dominating technique in eye tracking and gaze determination. Sometimes more than one camera which can capture IR images are used. If a light source is located very close to the optical axis of the camera, the pupil reflects most of the light. As a result, in the eye image the pupil is seen as a bright area. On the other hand, when the light source is

further from the optical axis, then the pupil is seen as a dark area. Dark and bright areas are used to locate the pupil position in the active light approaches. Active light sources are used in [41], [17], [11], [15], [39]. Active light approaches are now dominating commercial eye and gaze trackers. These approaches work very well indoors, but not as well outdoors. Kawato and Tetsutani use more than one camera for accurately detecting the eye features and tracking head movements [16]. Wang, et al. use specialized camera for zoomed in high resolution eye image for accurately detecting the eye features [33] [32].

2.5 Gaze-estimation Technique

Eye gaze determination is the main goal of the gaze tracker. Eye gaze can be modeled as gaze direction or gaze point from the image data. However, the problem of estimating the gaze only from the eye model data is that usually people move their head as well as eye when gazing at something. So, somehow, the head movement should be modeled with the eye data to get the gaze directing or point of regard [7]. A very common approach to minimize the head movement is to use some sort of headrest or making the eye tracker head mounted. By these methods, the head movement can be tracked by another camera. To estimate the point of regard (for example a point in a monitor), the point in image coordinates needs to be mapped to a monitor screen coordinate system by a mapping function. The mapping function parameters are determined through a calibration procedure.

A very simple linear gaze mapping function is described by Zhu and Yang [38]. The user needs to look at several known screen points and the corresponding eye corner and iris center is recorded. They use a 2-D linear interpolation from the vector between the eye corner and the iris center to the gaze angle. To simplify, they employ the approximation $\theta \approx \sin(\theta)$ when θ is very small. Kim and Ramakrishna use internal eye geometry as well as the current eye position to determine the point of regard [18]. They use a simple marker in between the eyes as the reference model. Calculation of the eye movement is based on this reference marker. In the mapping function, the position of the marker, radius of the

iris and the vector from the marker to the iris center are used as parameters. The gaze point is calculated using these parameters by a linear approximation. Wang and Sung propose another gaze tracking approach [32]. The point of regard is determined here from the intersection between the eye gaze vector (which is considered a vector from the eyeball center to the center of the iris) and the gaze plane (the screen). The internal geometry of the eyeball and the distance from the screen to the user is used in the mapping function. Wang, et al. use one eye image to approximate the gaze [33]. The eye corners are used to determine the pupil movement. The user is supposed to look at several predefined points on the gaze plane, that is the screen, and the gaze parameters are recorded for calibration. Based on these parameters, linear approximation is done. Hansen and Pece show that only four points are necessary for gaze estimation where the geometry is fixed, but unknown to the system, under some assumptions [9]. Hansen, et al. propose a mapping function estimated based on a Gaussian process interpolation method in [10] and [8]. Mean and variance functions are used and the mean values of the interpolated points is the corresponding estimated point of gaze. The confidence of the interpolated point is estimated by the variance. Calibration is performed by twelve points on the screen. Stiefelhagen, et al. use a neural network for the gaze estimation [25]. Both eye regions of 20×10 pixels size are fed to the neural network. The output of the neural network is 2×50 output neurons for Gaussian output representation of x and y coordinates of the point of regard.

When light falls on the curved cornea, several reflections occur from the boundary between the cornea and the lens. The first corneal reflection is known as glint [7]. In the case of active illumination approaches, this corneal reflection is used to determine the pupil movement. Usually, the vector from glint to the pupil is used as the measure to determine pupil movement. Regression-based gaze mapping is very common in active illumination approaches. The regression is made based on the vector from glint to the pupil. Kiat and Ranganath propose a one-time calibration procedure [17]. The calibration data is stored for future reuse. Neural networks are used here to map the complex and non-linear relationship of pupil and glint parameters to screen coordinates. The radial

basis function neural network (RBFNN) is used as it is fast to train. Two RBFNNs are used, one to determine the row and another one to determine the column coordinates. Zhu and Ji use generalized regression neural networks (GRNN) for gaze determination [41]. Six parameters are used as the input of the input layers of the GRNN, the pupil glints displacement, the ratio of the major to the minor axis in the elliptical model of the iris, pupil ellipse orientation and the glint image coordinates. The output of GRNN is one of the eight regions on the screen. A hierarchical gaze classifier is introduced to improve the gaze estimation result. Hennessey, et al. use a high resolution camera to extract the eye features with more detail [11]. Then, eye glint points and the eye model are used in a 3-D method based calibration to estimate the point of gaze on the screen. In the implementation of Ji and Yang, the user is asked to look at nine known regions on the screen [15]. Then for every region the eye data is recorded which consists of the glint point and the pupil-glint displacement. Then, the calibration parameters are determined from these data using a least squares method. Support vector regression is another approach used to map the eye parameters to the screen coordinates. For non-linear regression, a support vector machine converts the input space into a higher dimensional space so that the problem can be solved as a linear regression problem in the higher dimensional space. Huang, Dong, and Hao use SVR for gaze estimation on the screen from the center of iris coordinate [14]. They find that a radial basis kernel function for the SVR works best with their data. Two video cameras are used by Zhu, Ji, and Bennett for detecting the pupil and glint on active illumination [40] . The point of regard is then estimated by SVR using the pupil-glint vector and the 3-D coordinate of the pupil center. A very good survey on eye gaze tracking is presented by Hansen and Ji [7]. Another good survey on mapping functions in gaze tracking is presented by Sheela and Vijaya [22].

Chapter 3

Methodologies

3.1 Overview of the Work

A high level overview of the proposed eye-tracking and gaze estimation system is presented in figure 3.1. The detailed description of the feature extraction in the first frame is presented in Section 3.2. Eye tracking, head movement compensation and gaze estimation steps are described in Section 3.3, Section 3.6 and Section 3.7 respectively.

3.2 Feature Extraction in the First Frame

For eye tracking and gaze estimation, detecting the eye is the first task to perform. In this application, in the first frame, the user needs to initialize the approximate position of the eyes manually. It is a very common approach in the eye tracking literature to initialize the locations with the user input. The initial points should be anywhere inside the eye region, but preferably in or near the iris. Now the next task is to find the pupil, which is considered as the center of the iris here. The detailed procedure to find the center of the iris is described in Section 3.2.2. Before that in Section 3.2.1, the noise removal and morphological filtering is described, which is prerequisite of Section 3.2.2, helping to detect the center of the iris more accurately.

3.2.1 Morphological Filtering

For each eye, a small rectangular window is considered, keeping the user-initialized points as the centers. As the eyes are in these two window regions, we need now only consider

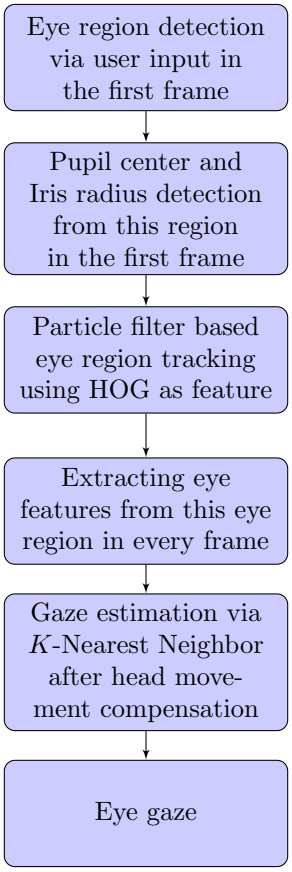

Figure 3.1: Overview of the Eye Tracking and Gaze Estimation System

these two window images for feature extraction. A right eye image is shown in figure 3.2.

The iris center and radius detection is performed only on red channel of the RGB image. This is due to the fact that, according to [29], the iris usually exhibits a very low value of red for both dark and light irises. On the other hand, the surrounding sclera and skin contain significantly higher red values. The figure 3.3 is the red channel image of figure 3.2.

The images of the eye regions are very noisy. Working with this noisy image may produce poor results, so before starting to work with the eye region images, the images need to be filtered to remove noise. We use the median filter to remove noise. Median filtering is a non-linear image filtering technique and the filtering is done only on red channel value of the eye image. The median filter takes a small neighborhood region of the image, and

Figure 3.2: Eye Region

Figure 3.3: Red Channel Eye Image of Original Figure 3.2

replaces the central pixel with the median value of this neighborhood. If there is any noisy pixel value, the median filter replaces that pixel value with the median value, which is a more probable value than the noisy one. Here, a 5×5 neighborhood structure is considered. By median filtering, not only are the noisy values of the image replaced, but also all the edges are preserved. A filtered image of figure 3.3 is shown in figure 3.4.

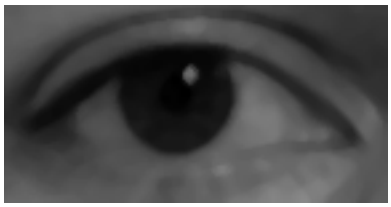

Figure 3.4: Filtered Eye Image

In the filtered image, the iris is found as the region with a circular shape, but it is still noisy. If the edge map of this image is taken, there will be many noisy edges inside the iris region which will generate false results. The cornea always reflects some light when the light source is in front of the user, this phenomenon is named as glint. A glint is shown in figure 3.5.

To eliminate the glint from this image, we use morphological erode and dilate operations.

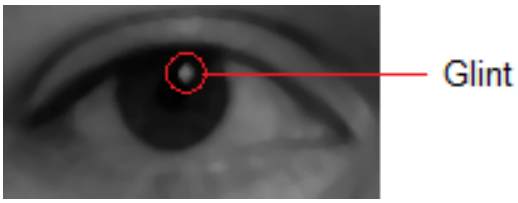

Figure 3.5: Glint in Eye Image

Erosion operation is equivalent to computing a local minimum over a neighborhood region, called the kernel. The kernel can be any shape and size. A point of the kernel is defined as an anchor point and the kernel is scanned over the image region replacing the anchor point's value with the minimum value of the region overlapped by that kernel generating a new image. On the other hand, dilation operation is the opposite operation of erosion. In dilation, the kernel is also scanned over the image region, and replaces the anchor point's value with the maximum value of the region overlapped by that kernel. The erosion and dilation operations on a gray level image can be defined by the following equations:

$$erode(x, y) = \min_{(x',y') \in kernel} image(x + x', y + y')$$

$$dilate(x, y) = \max_{(x',y') \in kernel} image(x + x', y + y')$$

Now suppose A is an image region, and B is the kernel. The erode or dilate operation can be viewed as the convolution of A and B. The erode operation is denoted as $A \ominus B$ and the dilate operation as $A \oplus B$. The erode operation is shown in figure 3.6 and dilate operation is shown in figure 3.7 (images are taken from [2]).

Now the morphological erode operation is applied on the filtered eye image. As the glint is brighter than the rest of the iris, after the erosion, glint is eliminated. However, the area of the iris expands after the erosion, as the surrounding area of the iris is lighter than the iris. To minimize this effect, the morphological dilate operation is applied on the resultant image. As the sclera is brighter than the iris, the area of the iris is shrunk by the dilate operation. Since the same kernel size is used for both operations, the iris is now in the original shape, but the glint is gone. The resultant image after applying erode and dilate is shown in figure 3.8.

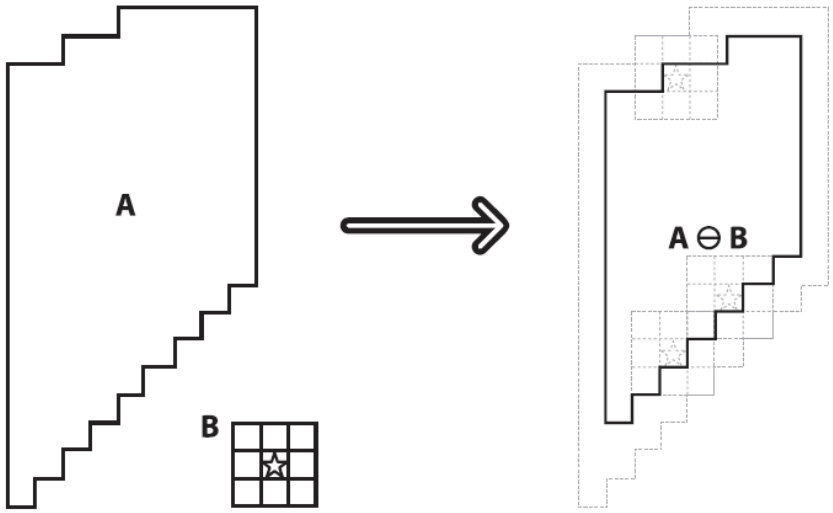

Figure 3.6: Morphological Erosion ([2])

3.2.2 Pupil Detection Using Linear Filtering

The pupil is the hole inside the iris region through which the light rays pass to the eye ball. Usually the pupil is darker than the iris, regardless of the color of the iris. The iris and pupil are both circular and concentric. Here the pupil center is determined by the iris center. In the face, the full iris is not always visible. Most of the time, a portion of it is covered by the eyelids. However as the light passes through the pupil, the pupil needs to be uncovered to see something.

Recall that the iris is always darker than the sclera, regardless of the color of the iris. Thus, the the iris and sclera edge is relatively easy to detect. The Canny edge operator ([3]) is used to get the edge map of the eye image. The result is shown in figure 3.9. Here the iris and sclera edge is relatively clear. The shape of the lower part of the iris is circular, and the upper part is occulted by the upper eyelid. So, a half circular mask is used to filter out the iris edge. A similar approach is also used in [26] and [23]. The half circular mask is considered 3 pixel thick. So, the mask is like figure 3.10. Here, $(x0, y0)$ is considered as the center of the mask and r_0 is the radius.

To detect the pupil, this half circular mask is placed in every position of the eye edge

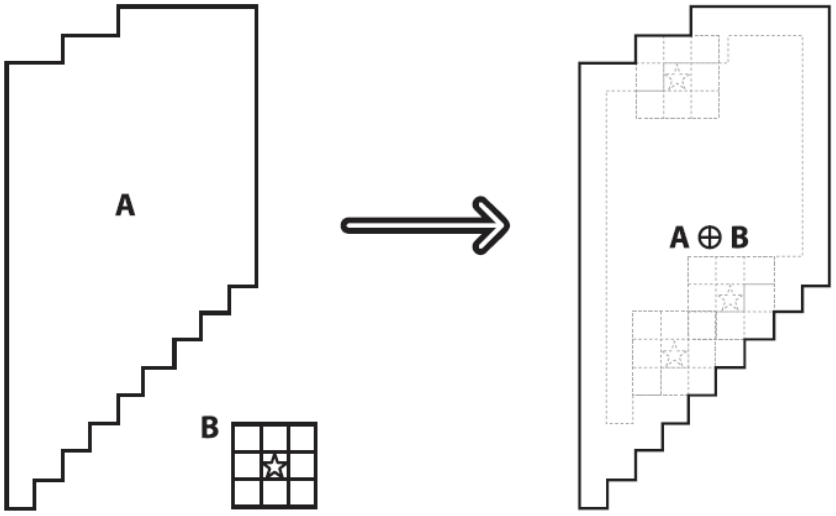

Figure 3.7: Morphological Dilation ([2])

Figure 3.8: Eye Image After Using Morphological Operations

image to compute the number of edge pixels located in the annulus region. The value of r_0 is changed in each position. The parameters of the iris are determined by the center (x_0, y_0) and the radius r_0 for which the number of edge pixels in the half circle annulus region is maximum. Figure 3.11 shows the result of the selected region in which the number of edge pixels is maximum. The result of the iris and pupil detection is shown in figure 3.12. Here the center of the half circle mask is considered as the center of the iris and the pupil.

3.3 Eye-tracking

Real time eye-tracking is implemented using particle filters. A particle filter is sophisticated Bayesian model estimation technique. The probability distribution of object states

Figure 3.9: Edge Map of Eye After Using Canny Edge Operator

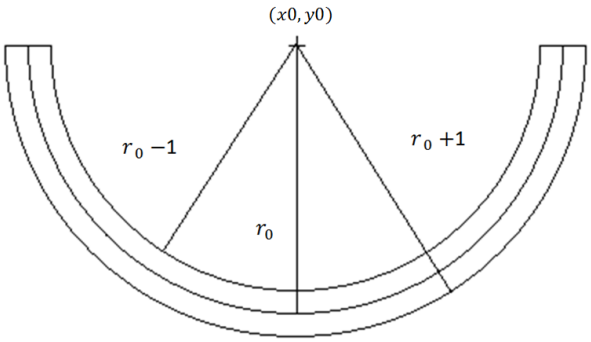

Figure 3.10: Half Circular Mask

is approximated using a set of discrete hypotheses (or particles). Each particle represents the center of a region in the image. In this section, the detailed tracking procedure is described. Prior to describing the detailed eye tracking method using a particle filter formal definition of particle filtering is presented in Section 3.3.1. Support Vector Machine classifier, described in Section 3.3.5, is used to evaluate probability of having an eye at location with a particle in the center. Training sets used to train the SVM classifier are described in Section 3.3.2. Histograms of oriented gradients is used as the feature in the SVM classifier, which is described in Section 3.3.3. The integral image technique, a very rapid feature calculation method, is described in Section 3.3.4.

3.3.1 Particle Filtering

Generalized particle filtering works as follows. First, a population of N samples is created by sampling from the prior distribution at time 0, $P(X_0)$. Then the update cycle is repeated for each time step:

Figure 3.11: Maximum Edge Pixels are Detected in the Red Region

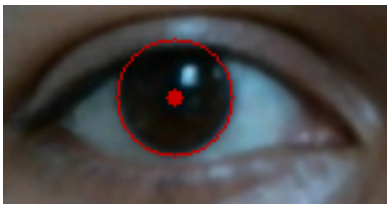

Figure 3.12: Result of the Iris and Pupil Detection

- Each sample is propagated forward by sampling the next state value x_{t+1} given the current value x_t for the sample, using the transition model $P(X_{t+1}|x_t)$.
- Each sample is weighted by the likelihood it assigns to the new evidence, $P(e_{t+1}|x_{t+1})$.
- The population is *resampled* to generate a new population of N samples. Each new sample is selected from the current population; the probability that a particular sample is selected is proportional to its weight. The new samples are unweighted.

In our implementation, each particle represents the center of a region in the image. The particles are weighted by the output of the support vector machine classifier in each step.

3.3.2 Training Set

To train the classifier, eye images taken from the BioID face database ([6]) are used. The BioID database consists of 1521 face images (384 $\times$ 288 pixel, grayscale images) of 23 different persons. The face images have been recorded during several sessions at different places. The database is available for public usage ([6]). Some examples of BioID face database are shown in figure 3.13. Manually set eye positions are also available on this

site. The positive eye examples are taken from these images. Two eye images, taken from each face image, serve as eye examples, so a total of 3042 eye images are taken as positive examples for the eye classifier. For each example, a square window around the pupil is used, with side length equal to 1/3 the distance between the pupils, thus making the training examples scale-invariant. This distance is chosen because it always encompasses the iris, but it is narrow enough to exclude other, confounding features, such as eyebrows or hair. Examples of eye images are shown in figure 3.14. The non-eye images were similarly generated, except that arbitrary locations in the image were chosen, excluding the eyes. The total number of non-eye images is also 3042. Some examples non-eye images are shown in figure 3.15.

Figure 3.13: Some Images from BioID Dace Database

Figure 3.14: Example of Eye Images (top row: right eye images, bottom row: left eye images)

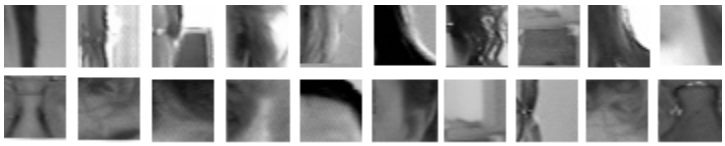

Figure 3.15: Example of Non-eye Images

3.3.3 Histograms of Oriented Gradients

Histograms of Oriented Gradients (HOG) are feature descriptors widely used in computer vision and image processing for the purpose of object detection. Local object appearance and shape can often be characterized well by the distribution of local intensity gradients. The implementation of HOG descriptor is achieved by compiling a histogram of intensity gradient directions using a fixed number of predetermined bins, weighted by the gradient magnitude. Dalal, et al. use histograms of oriented gradients (HOG) for human detection in [4] and [5]. The HOG is used here for eye detection. As the eye region, especially the iris and sclera has high contrast, HOG is very effective in eye detection. A HOG feature vector for each square window in the training set is computed. The image window is divided into four sub-windows, and a histogram of intensity gradient directions, weighted by gradient magnitude is computed for each of the sub-windows separately. For each sub-window, an 8-bar histogram is calculated. So, the total gradient directions are divided into 8 parts. The gradient magnitudes with those directions are summed up to calculate the histogram value. Then, this 8-bar histogram is normalized. The MIN_MAX normalization technique is used here. After the normalization, the four 8-bar histograms of the four sub-windows are concatenated into a 32-element feature vector of that window for classification.

Calculation of gradient magnitude and gradient direction of a sample eye image is shown in figure 3.16. In the first step the horizontal and vertical gradient of the image are calculated. From these images, the gradient magnitude and gradient direction images are obtained. After calculation of the gradient magnitude and direction images, the process of calculating the HOG is shown in figure 3.17.

HOG feature vectors are calculated for both eye and non-eye images as the training

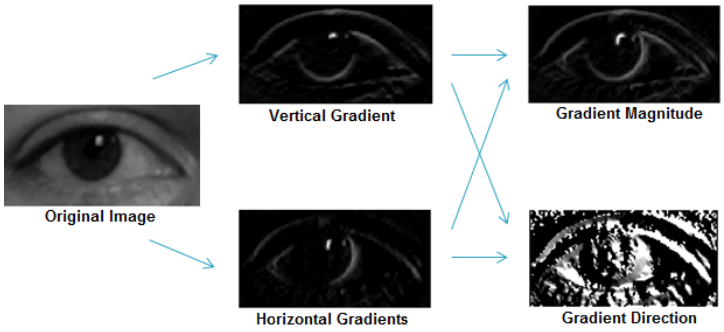

Figure 3.16: Gradient Magnitude and Director Calculation

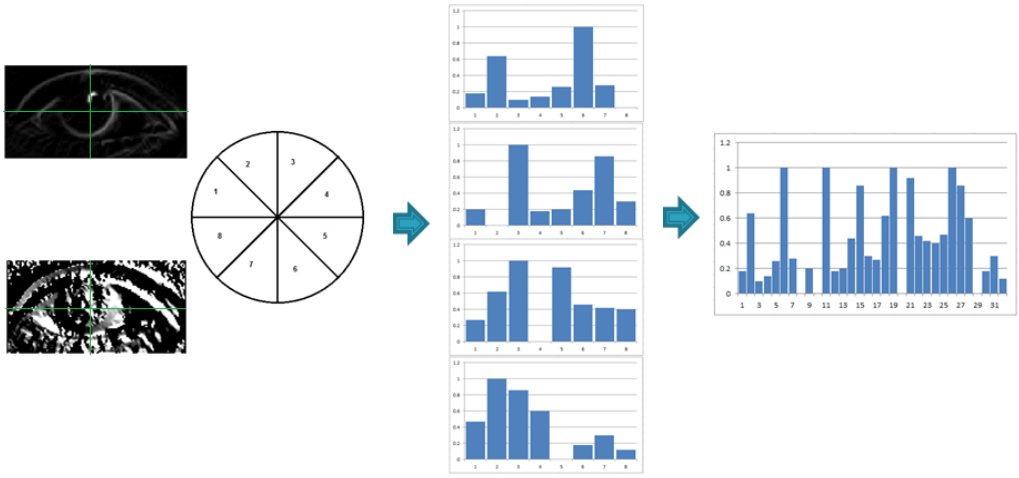

Figure 3.17: Histograms of Oriented Gradients Calculation for Four Sub-windows

example for the classifier.

3.3.4 Integral Image

Integral image techniques are used by Viola and Jones in [30] and [31], by which the features in an rectangular region can be computed very rapidly. The integral image at any location (x, y) is computed by the sum of all pixel values above and left of the pixel, including the pixel. Suppose Img is the original image, and $IntgImg$ is the corresponding integral image.

$$IntgImg(x, y) = \sum_{x' \leq x, y' \leq y} Img(x', y')$$

Now, with equation 3.1 and equation 3.2 the integral image can be computed with only one pass over the original image. Another array $CRSum$ is used for the cumulative row sum. Here, $CRSum(x, -1) = 0$ and $IntgImg(-1, y) = 0$ is defined. Integral image calculation is shown in figure 3.18.

$$CRSum(x, y) = CRSum(x, y - 1) + Img(x, y) \tag{3.1}$$

$$IntgImg(x, y) = IntgImg(x - 1, y) + CRSum(x, y) \tag{3.2}$$

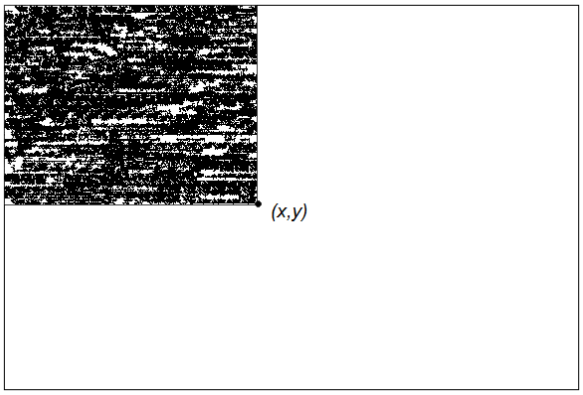

Figure 3.18: Integral Image, the Value at Point (x,y) is the Sum of All the Pixel Values Above and Left

After calculating the integral image, the sum of pixel values of a rectangular region in the original image can be calculated in constant time. Suppose the calculated integral image is shown in figure 3.19. Now, the sum of the pixel values of region C can be calculated from four points, P_1, P_2, P_3 and P_4. The value at P_1 is the sum of all pixel values of region A. Similarly, the value at P_2 is the sum of all the pixel values of region A and B, and the value P_4 is the sum of all the pixel values of the region A and D. The sum of the pixels values of region C can be calculated in constant time using equation 3.3.

$$RegionC = P_3 - (P_2 + P_4) + P_1 \tag{3.3}$$

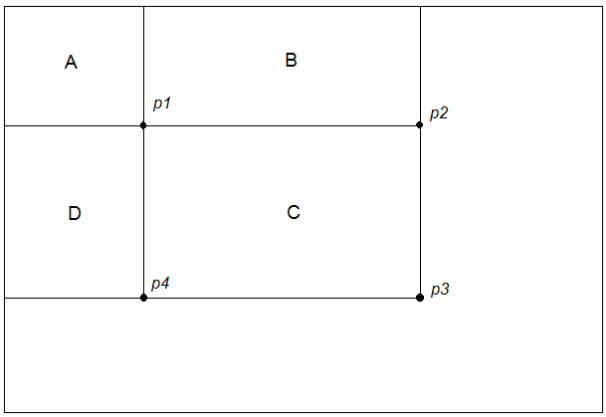

Figure 3.19: The Sum of Pixels in Region C is Calculated from the Integral Image by $P_3 + P_1 - P_2 - P_4$

The integral image technique is used for calculating the histogram of oriented gradients of a particular region. Histogram of oriented gradients is calculated for each bin using the integral image technique. As 8 bins are considered while computing HOG, the whole gradient direction range is divided into 8 parts. Then 8 gradient magnitude images, one for each part, are calculated. In each image, only those graduate magnitude values are considered whose directions are fallen in that range. From these images, 8 different integral images are calculated, one for each part. After computing these integral images, the HOG is calculated using these integral images in constant time for a particular region,. After this implementation, it is possible to calculate HOG values for several regions of interest in a larger image in constant time, after an initial cost to compute the integral images. In figure 3.20 and figure 3.21, computation of the HOG is described. After this computation, HOG value of any rectangular region of this image can be calculated in constant time. Thus it is also possible to generate a 32-element feature vector for any region in constant time.

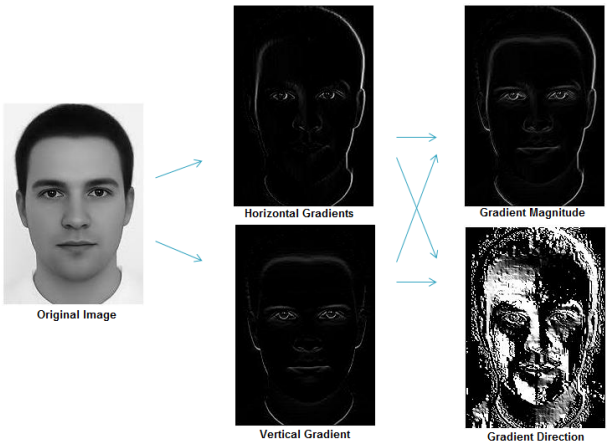

Figure 3.20: Gradient Magnitude and Director Calculation

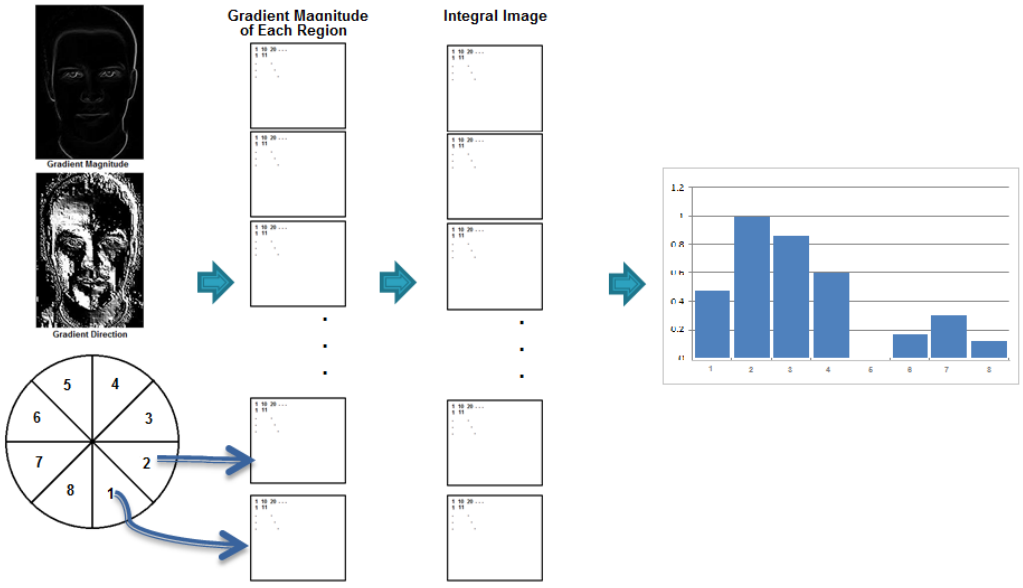

Figure 3.21: Integral HOG

3.3.5 Support Vector Machine

A support vector machine (SVM) is a supervised machine learning technique that can be used to classify different objects. The SVM classifier is trained first with different classes of object features and their labels. With this training data, a model is generated by the SVM by building a hyperplane or a set of hyperplanes in a multidimensional space which best separates the features of the training classes. It also tries to separate the different classes

so that the margin between different classes are as wide as possible. In classification, new test examples are mapped into that model space and the class is predicted based on which training class the point falls in. A very good guide of SVM is presented by Hsu, Chang, and Lin [13].

In figure 3.22, two classes of objects are present. An SVM plots the examples on a feature plane and tries to find the optimum plane that can separate the two classes. In this case, a line will suffice in the two dimensional plane. The solid line divides the classes, the dashed lines drawn parallel to the separating line mark the distance between the dividing line and the closest points to that line. The distance between the dashed lines is called the margin. The points that constrain the width of the margin are the support vectors. While generating the model, the SVM tries to maximizes the margin. The figure 3.22 is taken from [20].

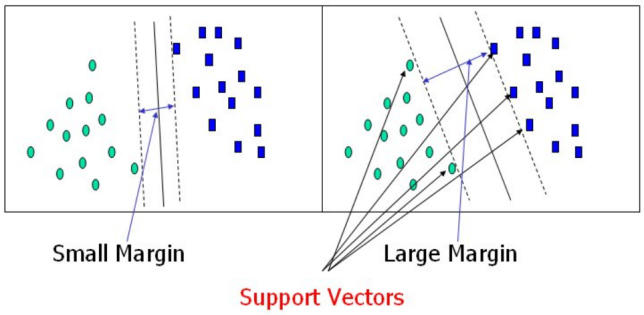

Figure 3.22: Example of a Support Vector Machine Classification

After computing the HOG feature vectors for each example, a Support Vector Machine classifier is trained with two classes of objects, eyes and non-eyes. Instead of a binary classification value, a continuous output is considered whose value increases with the likelihood of that class. It is obtained by returning the Euclidean distance of the test vector from the classification boundary.

3.3.6 Tracking the Eyes with Particle Filter

Two particle filters track both eyes separately. Particle filters are advantageous for tracking because they maintain multiple hypotheses about the location of the tracked object. A window around the particle position is considered as a candidate. On each frame, some hypotheses about the eye location (particles) may be on the eye while others are not exactly on the eyes. In subsequent frames, as more information becomes available, the algorithm can discard the particles of the poor hypotheses, and propagate only the most likely hypotheses to the next iteration. In each iteration (frame), the particle filter maintains a fixed number of particles. Generally, more particles result in better performance, but the number of particles is limited by the constraint of running in real time. Each particle is evaluated by running its HOG value through the SVM prediction described in section 3.3.5. The SVM output value, which is the distance to the classification boundary, is converted into a particle weight by first normalizing by the minimum value so that all values are positive, then adding 1, then taking an exponent to exaggerate the difference between higher and lower values. The particles for time t+1 are produced by sampling particles at time t with probability proportional to their weight. Each sampled particle is then propagated to the next generation, with the addition of a noise term. Noise considered here is Gaussian noise.

A technique is used to introduce adaptive behavior into the particle filter. The noise term in particle propagation is set to the sum of a base level, proportional to the distance between the pupils, and a variable level, proportional to the number of particles with high absolute scores. Typical particle filters normalize the particle weights, making them insensitive to absolute confidence levels, however, this implementation increases the noise when confidence is low that the particle cloud is tracking an eye. As a result, the particles are produced in a wider area when they lose the eye. Again, as in the training examples, the edge length of the sub-windows is 1/3 the distance between the pupils. This takes advantage of the constrained situation of the face to relieve the burden of determining the proper scale.

3.4 Feature Extraction in Each Frame

The particle filter tracks the eye, but does not always pinpoint the pupil location. The support vector machine does not reliably distinguish pupil points from other eye points near the pupil. For this reason, the eye features need to be extracted by other methods in each frame.

3.4.1 Searching for the Dark Region

The mean or median of the particle cloud also does not locate the pupil because some particles lag behind the eye after eye or head movements. Instead, it is assumed that the median particle position is at least within a certain radius r of the pupil. Within this radius, the intensity value of the darkest pixel is searched. It is assumed that the other dark parts of the faces, e.g. eye brows, hair, are not present in this region. Any pixels with intensity values that differ from the darkest by less than a threshold are labeled as candidates for the pupil pixels, and the median location of candidate pupil pixels is the final candidate for pupil location.

3.4.2 Finding the Iris Center

The final candidate for pupil location is obtained by the method described in section 3.4.1. A small region surrounding this location is searched using the same method described in section 3.2 for the iris center. If the distance between the camera and the eyes does not change, then this radius of the iris also does not change. Thus the radius calculated in the first frame can be used. So, the iris is searched for within a two-pixel margin of the initial radius. If the iris is not found, then the iris is searched for using a larger range of radii.

3.5 Pseudo-code of Proposed Eye-tracking Method

Following is the pseudo-code of the proposed eye-tracking method.

Eye tracking

1. **for** *each frame* **do**
2. Calculate the integral HOG for the region that contains the particles
3. **for** *each particle* **do**
4. Calculate the HOG vector for a region with the size proportional to the estimated distance between the eyes
5. Evaluate this value using the SVM classifier
6. Assign this evaluated value as the weight of the particle
7. **end for**
8. Sample all particles with probability weighted as a function of the SVM scored weight
9. Select a small radius around the median position of the particles
10. Use filtering to remove noise from this region
11. Find the darkest location within this small region
12. Count number of edge pixels by placing the half circular mask on the eye edge image in every position of a small radius around the darkest location found in previous step
13. The pupil is the center of the half circular mask with maximum number of edge pixels
14. **end for**

3.6 Head Movement Compensation

The method described in section 3.3 can reliably track both eyes and detect the eye features in each frame. Eye tracking is also robust to head movements. However, by detecting the amount of head movements accurately in each frame, it is possible to estimate the user's gaze more accurately, as it is very hard for the users to keep their head still. Although significant rotations of head, changes in distance from the camera, or large in-plane translations will still disrupt the calibration, small in-plane translations can be compensated for with this method.

The head-movement compensation works by an updating template matching method.

The midpoint between the irises is chosen in each frame. Then, a small window is chosen as the 'between-the-eyes' region, keeping the midpoint of the irises in the center of this region. This point is chosen as this is a very stable area of the face. Usually the between the eyes area is brighter than its two sides, as the eyes are on both sides of the region and this area is the nose and the forehead region. Thus, this region is easier to track [16]. Then, in each frame, the 'between-the-eyes' region is stored. In the next frame, the closest match to this template, within a fixed distance from the previous location is searched for. The difference between the two locations is the estimated head translation for that time step. Then the midpoint between the irises is calculated again for that frame and used to update the 'between-the-eyes' template. A similar approach is used in [16]. To find the head movements with respect to the first frame, the sum of head translations for time $1 \ldots t$ is calculated.

3.7 Gaze Estimation

Successful eye tracking is a prerequisite of the gaze estimation. Gaze calibration is necessary for gaze point estimation. The user is asked to look at some predefined points on the screen while keeping their head fixed. The pupil positions and corresponding screen positions are collected. After removal of the noisy data, a simple calibration is done. For this simple calibration, the K-nearest neighbor classification algorithm is used. The detailed calibration procedure is described in Section 3.7.1.

3.7.1 Calibration Procedure

The calibration procedure is very brief and simple. The calibration panel, that is, the monitor of the laptop, is divided into some vertical regions. The monitor screen is only divided into a few vertical regions. If these regions are divided by horizontal regions, then as the distance of two adjacent regions in the screen becomes very small, making gaze estimation from the iris data very difficult. For the gaze calibration, the user is asked to

look at some predefined points on the screen. These points are at the centers of these regions. The user is asked to look at each point for a short period of time. During this period the pupil positions from both eyes are taken for several frames. An example of three calibration points panel is shown in figure 3.23. Figure 3.24 presents an example of the data acquisition process. Then using the method described in Section 3.7.2, the noisy outliers are removed, and the best candidate points are determined. Using these best points, the gaze estimation is done using the K-nearest neighbor algorithm described in Section 3.7.3 and Section 3.7.4.

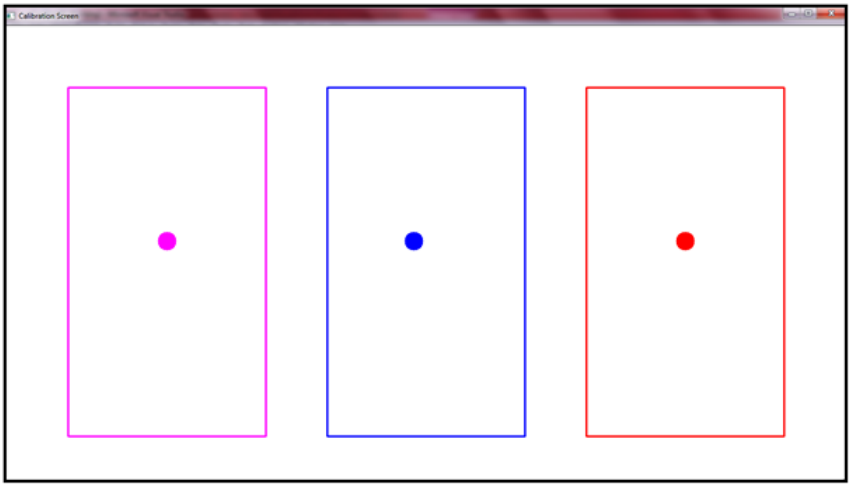

Figure 3.23: Calibration Panel

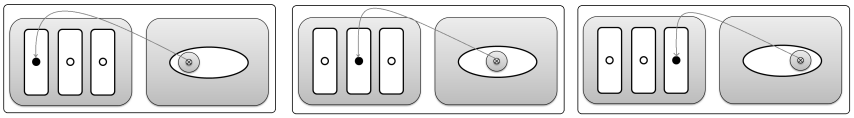

Figure 3.24: Calibration Process

3.7.2 Selecting Best Candidate Points

A two step process is used to select the best candidate pupil points for every calibration region of each eye. The first goal is to remove a certain percentage of the total set of

points as outliers. From that result, a certain percentage of points should be considered as the best candidate. So, in the first step, the center position is calculated from all the points. The outliers are classified from this set based on their distance from the center. The outliers are then removed from this point set. After the outliers removal, the center of the points is calculated again to remove the effect of the outliers on the center. From the rest of these points, the points are ranked based on their distance to the new center. From this ranked list, certain percentage of points are considered as the best candidate points.

3.7.3 K-Nearest Neighbor Algorithm

K-Nearest neighbor algorithm (KNN) is a lazy machine learning algorithm. It is based on voting of the closest samples in the feature space. Any feature is classified by the majority voting of its neighbors. K is the number of neighbors considered in the classification. The metric to measure closeness is also important. When training the classifier, only the sample features are stored, no computation is made. During the classification, all the computation is done. The training time is very small, but the classification time is relatively large depending on the number of training samples for KNN.

Suppose, as shown in figure 3.25, two classes of objects (Class A and Class B) are considered. The test example is shown in + and also the distance from this test example to its closest three training examples are shown. If 1-nearest neighbor is considered, then the text example will be classified as Class A. On the other hand, if 3-nearest neighbors are considered, then the test example point will be classified as Class B.

3.7.4 Gaze Estimation Using the KNN Algorithm

The best iris center points are obtained for each calibration region by the method described in section 3.7.2 separately for both eyes. Two KNN classifiers are trained using these iris center points, one for each eye. Training labels for these examples are the corresponding central points of the region on the screen and the region number. After training the classifier

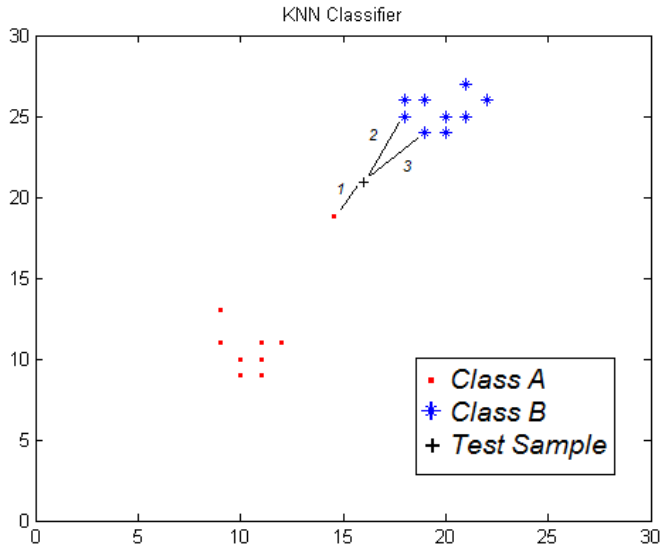

Figure 3.25: Example of K-Nearest Neighbor Classifier

for both eyes for all the regions, the calibration process is complete.

For gaze estimation, the user's iris center positions are used as the test examples. The test point is now classified using both KNN classifiers. In the ideal case, both classifiers should classify the test example as the same class and give the same output point on the screen. However, sometimes deviation from the ideal case is observed. So, another voting scheme is introduced. For each classifier, the weight of that classifier is assigned by the nearest number of points that vote positive for the output class. The estimated gaze region is determined by the output of the classifier with maximum vote. Suppose, the classifier of the left eye classified one example as output region 2, and the classifier of the right eye classified it as output region 3. Here 11 nearest neighbor is considered. For the left eye, the region 2 is classified with 6 positive votes for region 2. So, the weight of this classifier will be 6. On the other hand, suppose the right eye classifier classified that position as region 3 with 9 positive votes for this region. So the weight of this classifier will be 9. This point will be classified as region 3, from the output of the right eye classifier, and region 3 is shown on the screen.

3.8 Pseudo-code of Proposed Gaze-estimation Method

Pseudo-code of the proposed gaze-estimation method is described below.

Gaze estimation

1. **Input:** *(calibrationStatus, numCalibrationPoints, numTrainingExm, outlierPr, bestExm)*
2. **if** ($calibrationStatus == false$) **then**
3. **for** $i = 1 \rightarrow numCalibrationPoints$ **do**
4. Draw a small circle in corresponding position of i on the screen
5. **for** $j = 1 \rightarrow numTrainingExm$ **do**
6. $DL(j) =$ coordinate of left pupil as training data
7. $LBL(j) = i$, as training label of left pupil coordinate
8. $DR(j) =$ coordinate of right pupil as training data
9. $LBR(j) = i$, as training label of right pupil coordinate
10. **end for**
11. **end for**
12. Remove $outlierPr$ percent points for each calibration region as outliers from (DL,LBL), (DR,LBR) based on maximum distance from the centroid of DL and DR respectively
13. Select $bestExm$ number of points for each calibration region as best training examples based on minimum distance from the new centroid point of DL and DR
14. Train classifier $(KNN)_L$ using best training examples, data DL and label LBL
15. Train classifier $(KNN)_R$ using best training examples, data DR and label LBR
16. $calibrationStatus = true$
17. **elseif** ($calibrationStatus == true$) **then**
18. Get the coordinate of left and right pupil in LP and RP respectively
19. $OutL =$ classification result from classifier $(KNN)_L$ for test example LP
20. $WL =$ number of neighboring points that vote for test example LP in previous step
21. $OutR =$ classification result from classifier $(KNN)_R$ for test example RP
22. $WR =$ number of neighboring points that vote for test example RP in previous step
23. **if** ($OutL == OutR$) **then**
24. Highlight $OutL$ region on screen as gaze-estimation

25. **elseif** ($WL > WR$) **then**
26. Highlight $OutL$ region on screen as gaze-estimation
27. **else**
28. Highlight $OutR$ region on screen as gaze-estimation
29. **endif**
30. **endif**

Chapter 4

Experimental Results

In order to measure the validity of the design qualitatively and quantitatively, it has to be tested with several experiments. This chapter explains the environments and setup of the testing design. It starts with a description of a system to run and to test the program. Then, the evaluations of the conducted experiments of the eye-tracker and gaze determination are described separately.

4.1 Environment and Setup

The prototype system is developed using the OpenCV library and Visual C++ under Windows7. OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision, which is free for both academic and commercial use. The images are captured with a 'Logitech C615 Webcam' with a resolution of 640×480 pixels. The system is implemented on an Intel(R) Core(TM) i3 processor at 2.4GHz with 4GB of RAM.

4.2 Test Result of the Eye-tracker

The user needs to sit in front of the computer, keeping 1 to 2 feet of distance from the camera. The camera is fitted on the top of the monitor of the computer. After the system starts capturing the video sequence, the user needs to click on the video window to start the eye-tracking. The user needs to click on both eyes in the image to locate the eyes for the tracker in that frame. The points do not need to be centered exactly on the pupils,

but points near the iris regions are preferred. Then the system starts to track both eyes simultaneously in the following frames. An example of experimental setup is shown in figure 4.1.

Figure 4.1: Experimental Setup

To quantitatively analyze the performance, the pupil positions detected by the eye-tracker are stored. The video of the user is also stored for hand-labeling purposes. Then the pupil positions are hand labeled from the video. The disparity is defined here as the distance between the pupil positions detected by the tracker and hand labeled pupil positions. The eye regions are around 75×50 pixels when the user sits around 1.5 feet from the camera. Although the radius of the iris varies from person to person, it is found that the radius is within 10 to 14 pixels range if the user sits in this distance.

4.2.1 Experiment 1

In the first experiment, the user sits around 1 foot from the camera, facing the monitor. After the tracker starts tracking the eyes, the user is free to move his/her head. The data obtained from the eye tracker is recorded in this experiment. The video is also recorded and hand-labeled later. The length of the first video sequence is 500 frames, in a real time, frame rate of 10 frames per second.

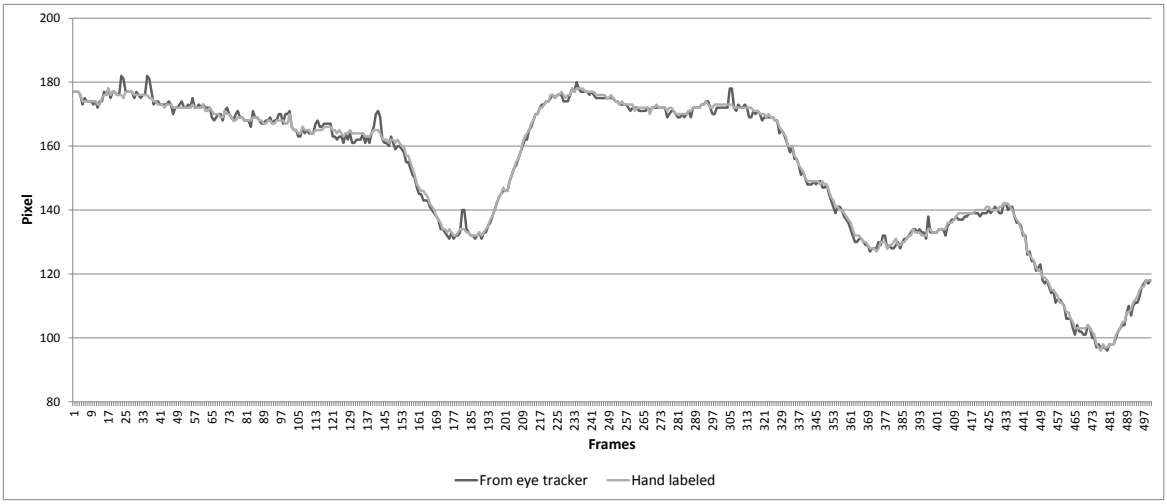

Figure 4.2: Tracked and Hand-labeled Right Eye (Row) Positions (Experiment 1)

The row positions of the right pupil in each frame for both tracked and hand-labeled pupil positions are shown in figure 4.2. Figure 4.4 presents the row positions of the left pupil in each frame for both tracked and hand-labeled pupil positions. The column positions of both pupils in each frame for both tracked and hand-labeled pupil positions are presented in figure 4.3 and figure 4.5. Figure 4.6 and figure 4.7 show the row and column difference in pixels between the hand labeled pupil positions and the data obtained from the tracker for the right pupil respectively. Figure 4.8 and figure 4.9 show the row and column disparity in pixels for the left pupil respectively. Figure 4.10 shows the Euclidean distance (error) between the tracked and hand-labeled pupil positions in each frame for the right eye. The euclidean distance (error) between the tracked and hand-labeled pupil positions in each

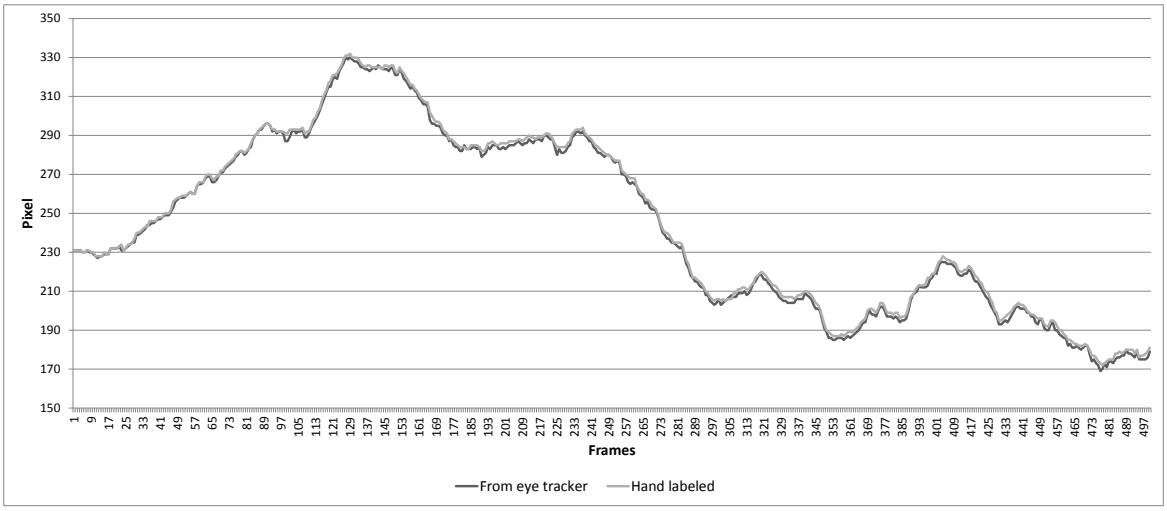

Figure 4.3: Tracked and Hand-labeled Right Eye (Column) Positions (Experiment 1)

Figure 4.4: Tracked and Hand-labeled Left Eye (Row) Positions (Experiment 1)

frame for the left eye is presented in figure 4.11.

Figure 4.5: Tracked and Hand-labeled Left Eye (Column) Positions (Experiment 1)

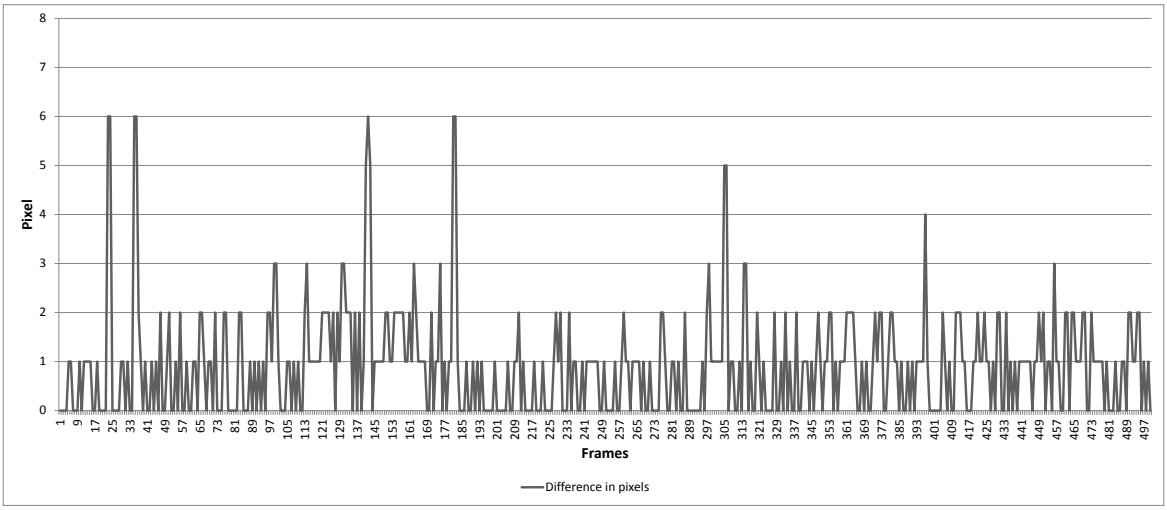

Figure 4.6: Difference in Pixels Between Tracked and Hand-labeled Right Eye (Row) Positions (Experiment 1)

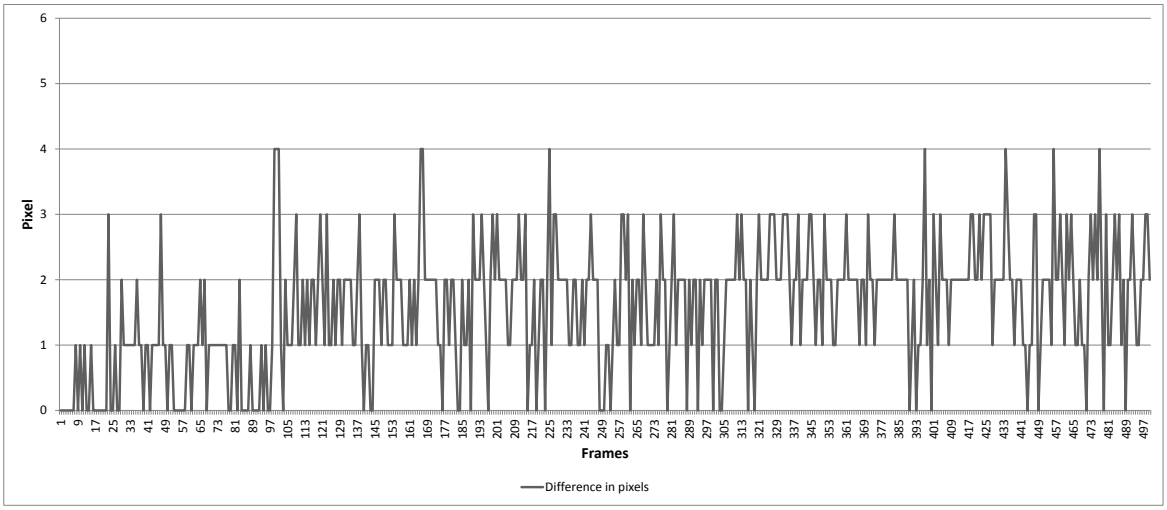

Figure 4.7: Difference in Pixels Between Tracked and Hand-labeled Right Eye (Column) Positions (Experiment 1)

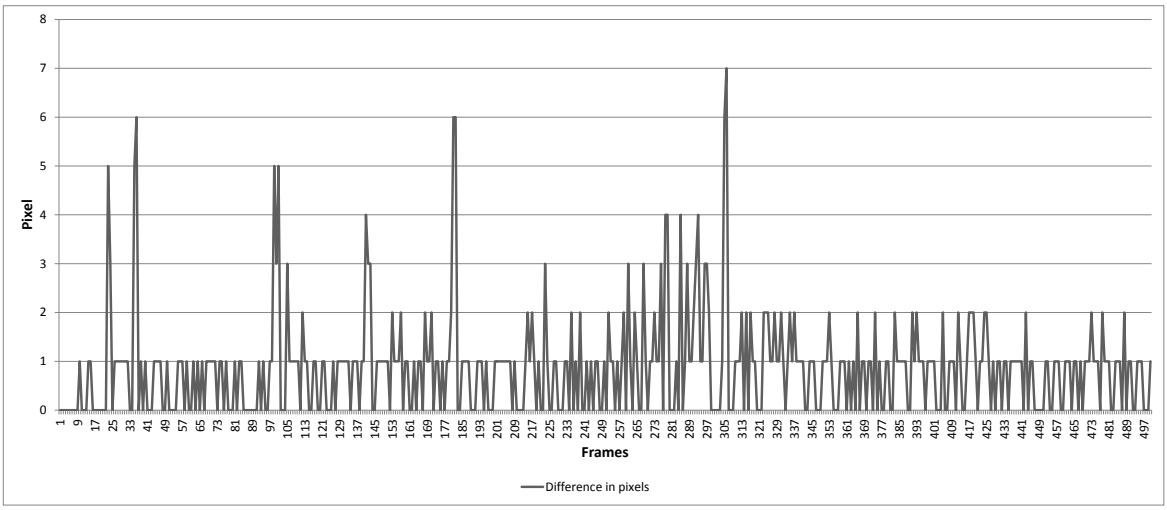

Figure 4.8: Difference in Pixels Between Tracked and Hand-labeled Left Eye (Row) Positions (Experiment 1)

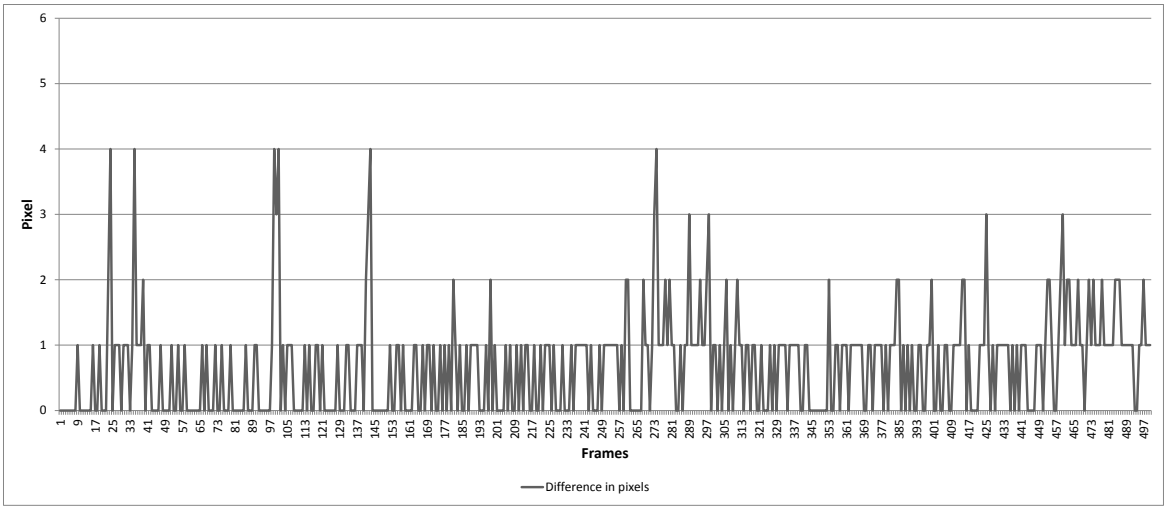

Figure 4.9: Difference in Pixels Between Tracked and Hand-labeled Left Eye (Column) Positions (Experiment 1)

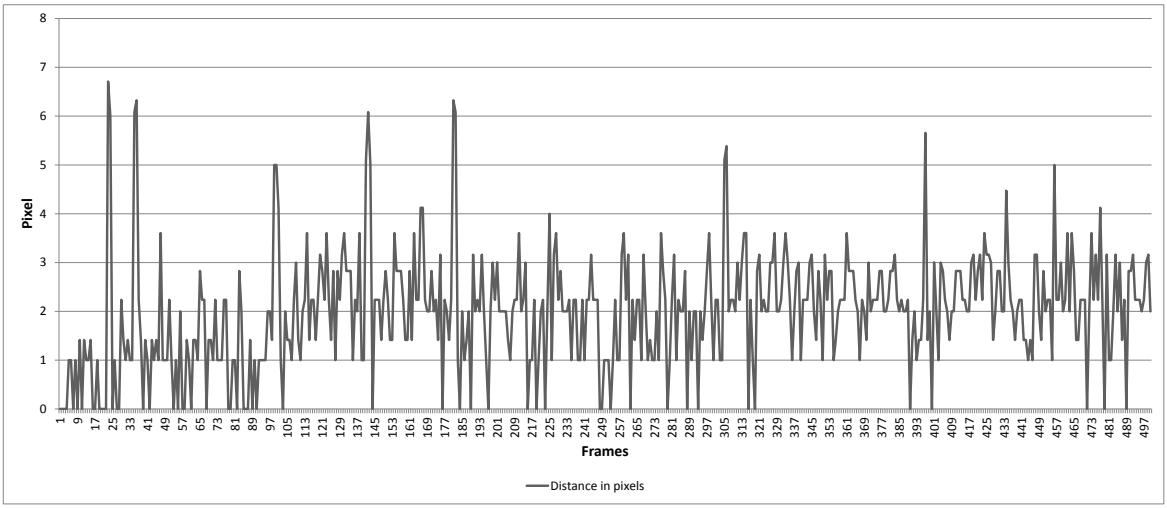

Figure 4.10: Distance in Pixels Between Tracked and Hand-labeled (Error) Right Eye Positions (Experiment 1)

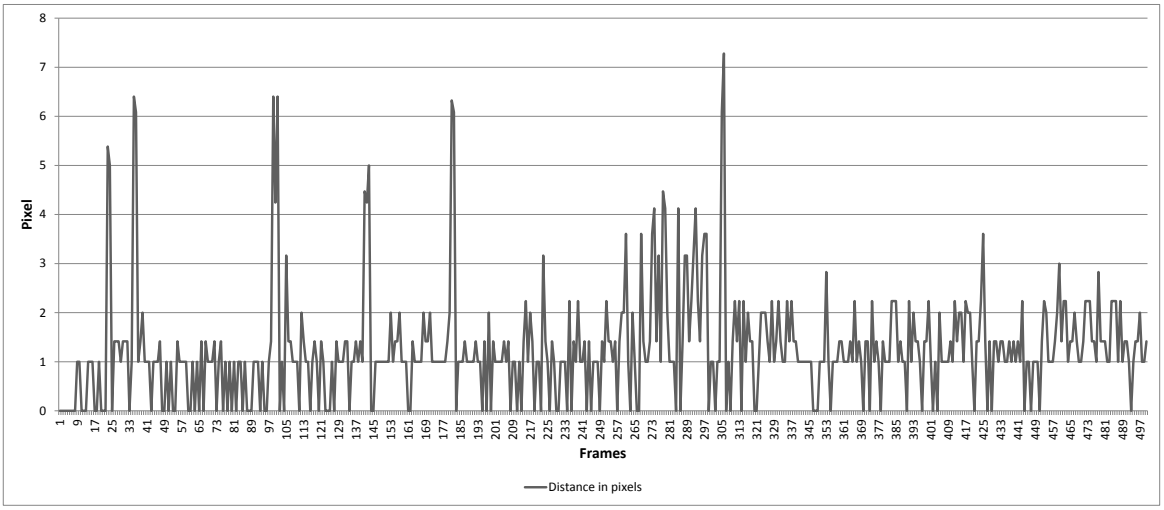

Figure 4.11: Distance in Pixels Between Tracked and Hand-labeled (Error) Left Eye Positions (Experiment 1)

4.2.2 Experiment 2

In this experiment, the user sits around 2 feet from the camera, facing the monitor. After the tracker starts, the user is also free to move his/her head here as in Experiment 1. The data from the eye tracker is recorded in this experiment and hand-labeled later. The length of the first video sequence is 500 frames, in a real time frame rate of 10 frames per second.

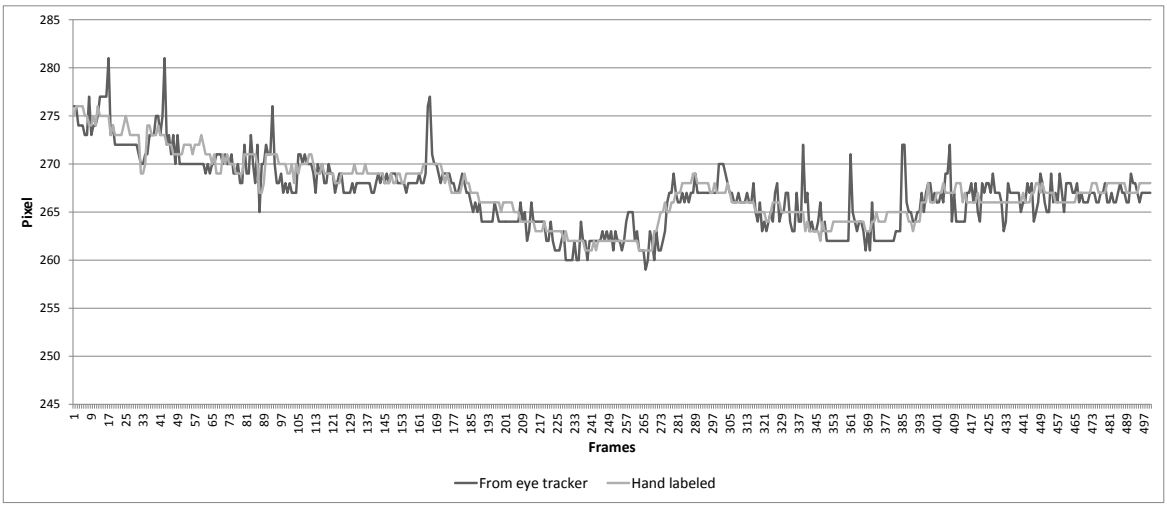

Figure 4.12: Tracked and Hand-labeled Right Eye (Row) Positions (Experiment 2)

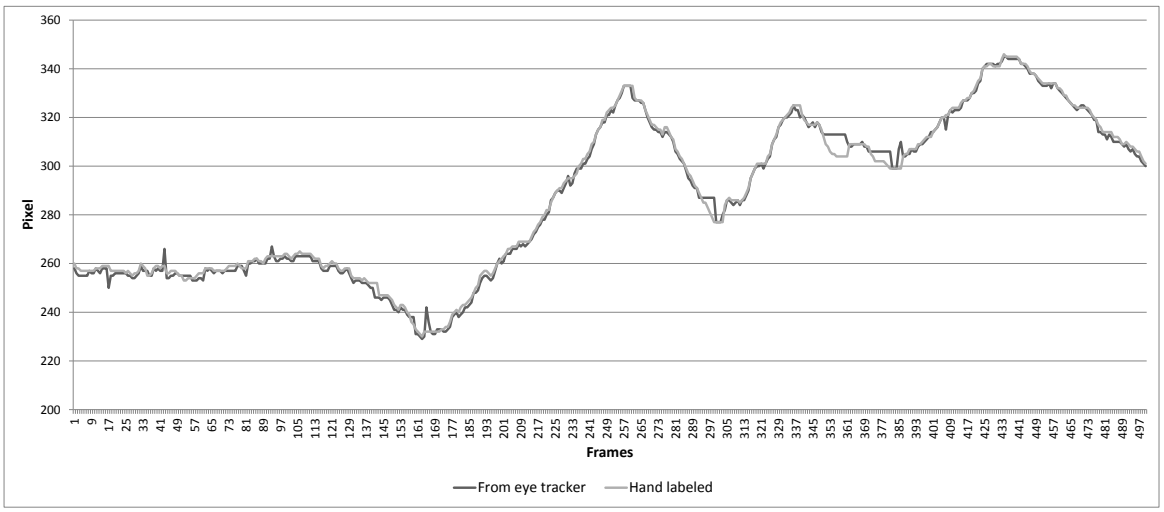

Figure 4.13: Tracked and Hand-labeled Right Eye (Column) Positions (Experiment 2)

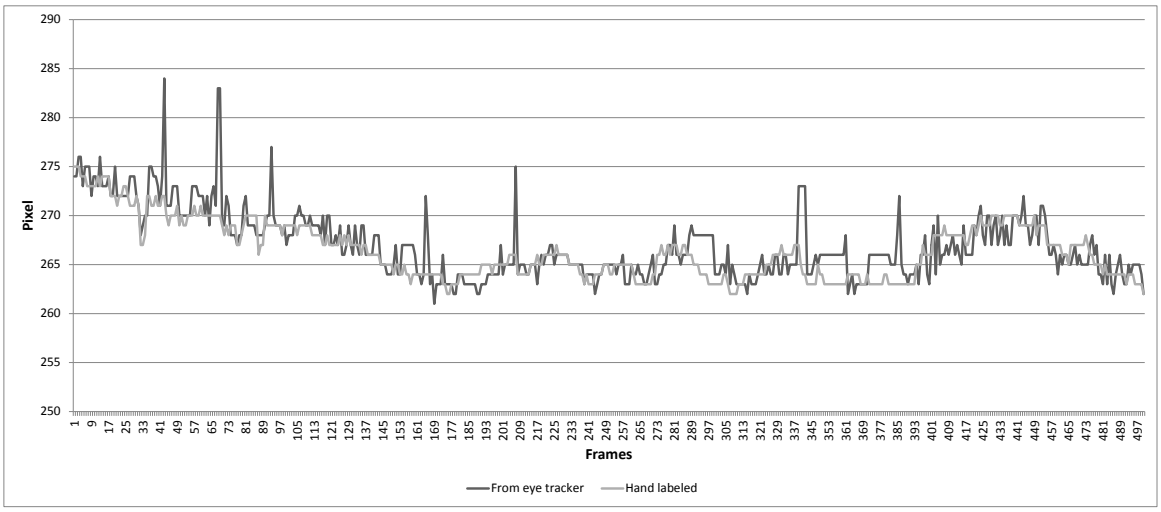

Figure 4.14: Tracked and Hand-labeled Left Eye (Row) Positions (Experiment 2)

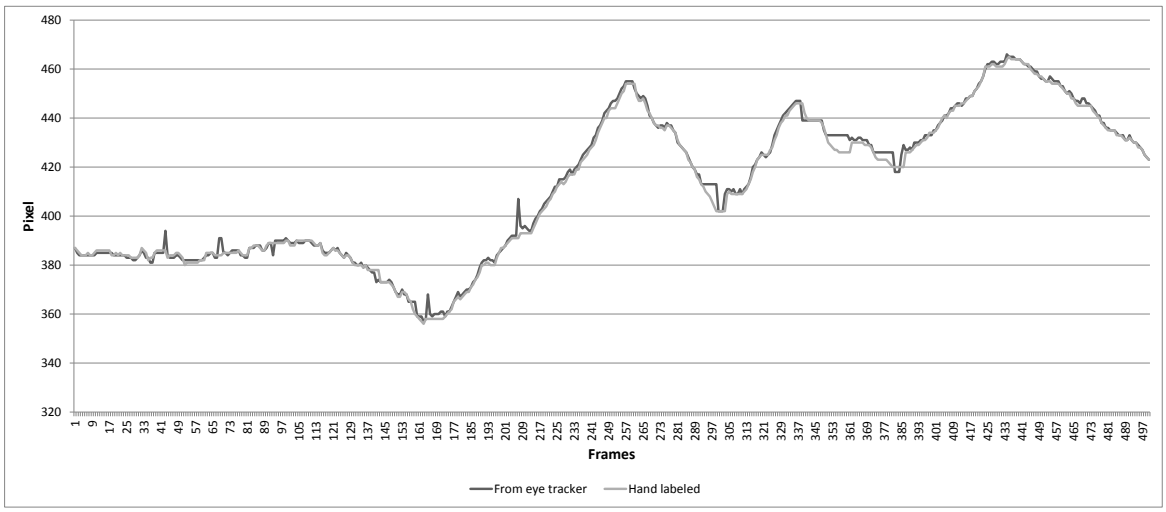

Figure 4.15: Tracked and Hand-labeled Left Eye (Column) Positions (Experiment 2)

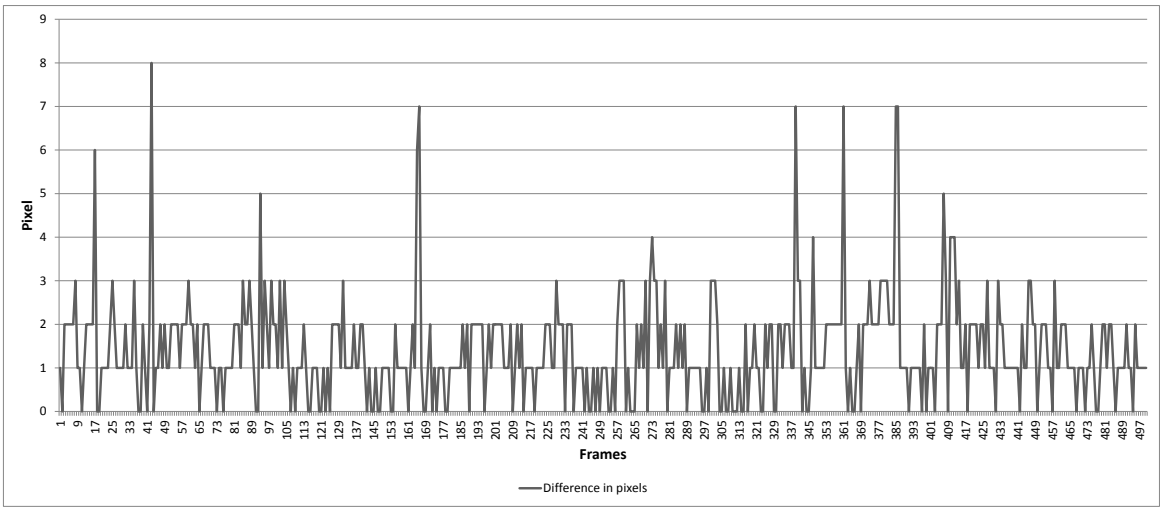

Figure 4.16: Difference in Pixels Between Tracked and Hand-labeled Right Eye (Row) Positions (Experiment 2)

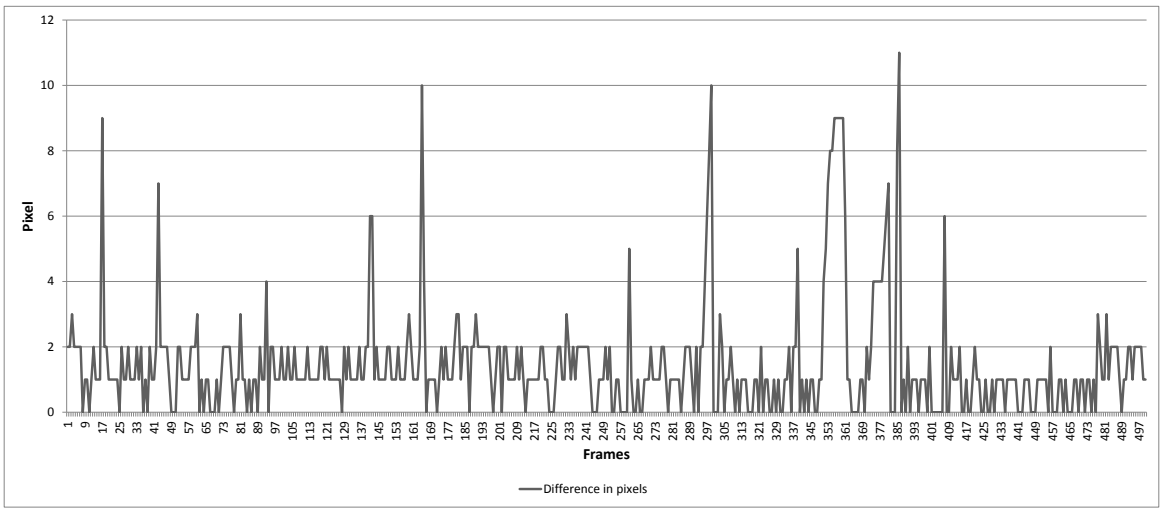

Figure 4.17: Difference in Pixels Between Tracked and Hand-labeled Right Eye (Column) Positions (Experiment 2)

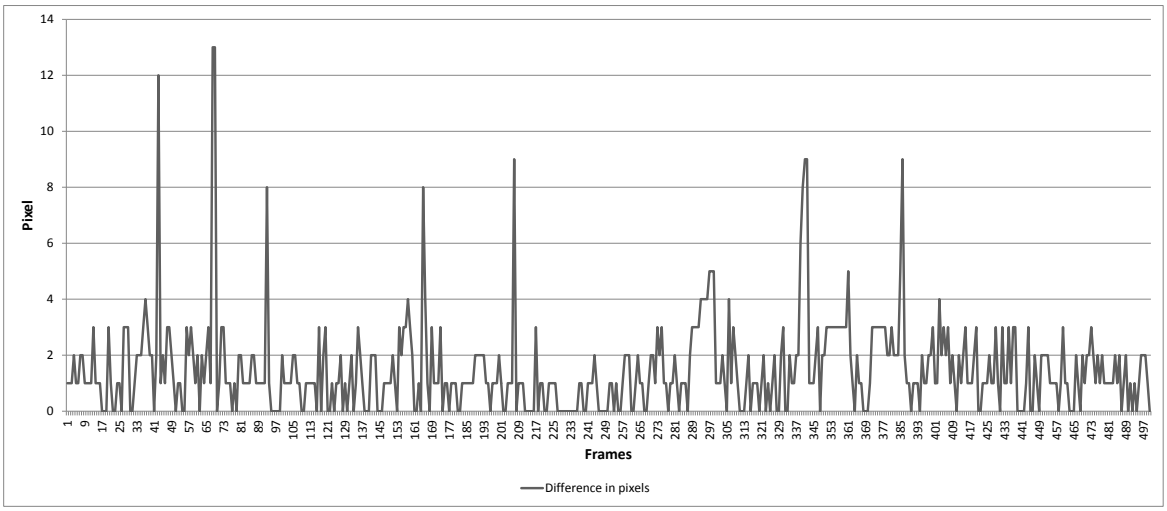

Figure 4.18: Difference in Pixels Between Tracked and Hand-labeled Left Eye (Row) Positions (Experiment 2)

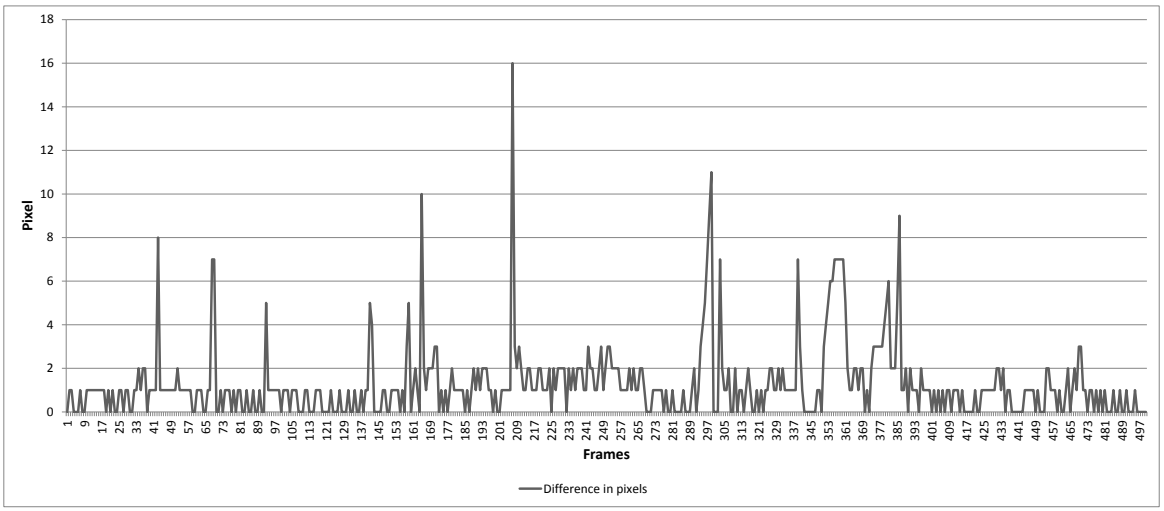

Figure 4.19: Difference in Pixels Between Tracked and Hand-labeled Left Eye (Column) Positions (Experiment 2)

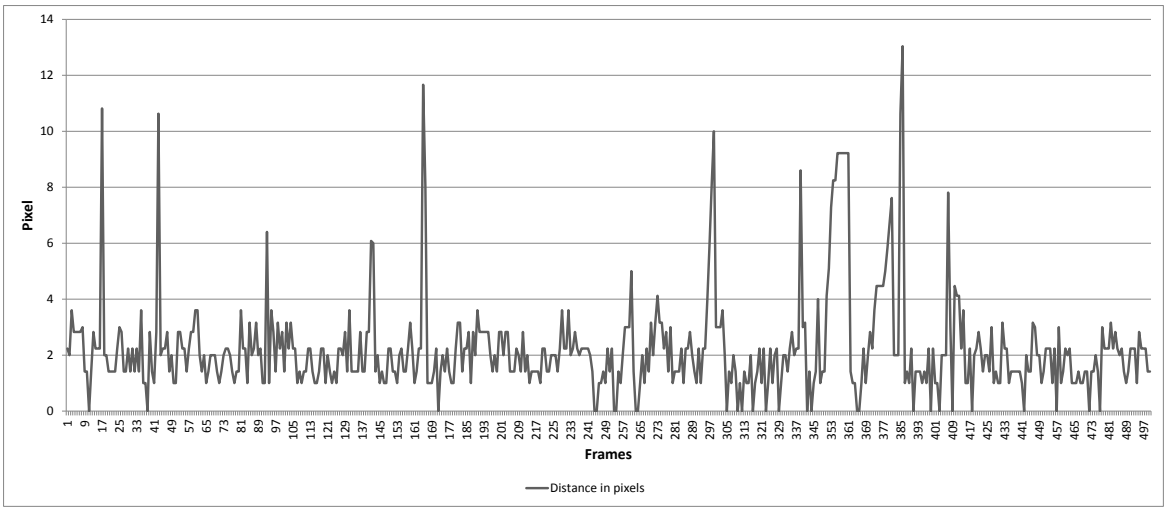

Figure 4.20: Distance in Pixels Between Tracked and Hand-labeled (Error) Right Eye Positions (Experiment 2)

The row positions of the right pupil in each frame for both tracked and hand-labeled pupil positions are shown in figure 4.12. Figure 4.14 presents the row positions of the left pupil in each frame for both tracked and hand-labeled pupil positions. The column positions of both pupils in each frame for both tracked and hand-labeled pupil positions are presented in figure 4.13 and figure 4.15. Figure 4.16 and figure 4.17 show the row and column difference in pixels between the hand labeled pupil positions and the data obtained from the tracker for the right pupil respectively. Figure 4.18 and figure 4.19 show the row and column disparity in pixels for the left pupil respectively. Figure 4.20 shows the Euclidean distance (error) between the tracked and hand-labeled pupil positions in each frame for right eye. The Euclidean distance (error) between the tracked and hand-labeled pupil positions in each frame for left eye is presented in figure 4.21.

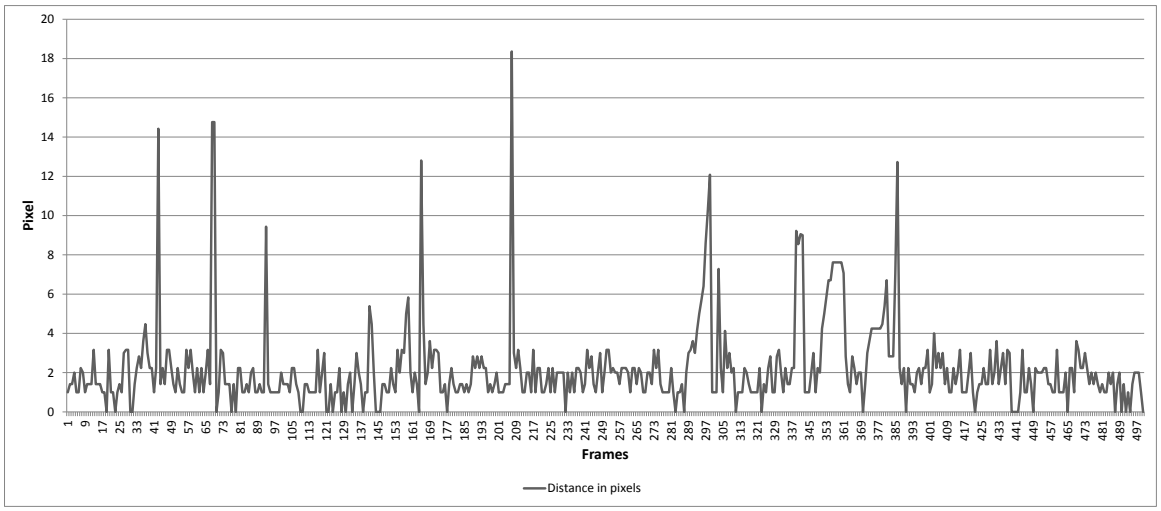

Figure 4.21: Distance in Pixels Between Tracked and Hand-labeled (Error) Left Eye Positions (Experiment 2)

4.2.3 Result Discussion

For pupil detection, from the eye region tracked by the particle filter, the iris is detected first. Then the center of the iris is considered as the pupil center in this eye-tracking design. To determine the parameter of the iris, a half circular mask is used which is described in section 3.2.2. To approximate the shape of the irises as circles when the irises are at the corner of the eyes, a three pixels thick half circular annulus mask is used to detect the iris boundary. The pupil is determined by the center of the mask, for which the number of edge pixels located in the annulus region is maximum. As a result, it is possible that the tracked pupil center position can be three pixels apart from the original pupil center. Again, while hand-labeling the pupil center, a deviation of few pixels may occur in both row and column directions. So, if the Euclidean distance (error) between the tracked and the hand-labeled pupil position is in between zero pixel to three pixels, then the pupil is considered as tracked correctly in that frame.

Table 4.1: Deviation (Error) in Pixels (Experiment 1)

	Mean Absolute Error	Standard Deviation of Errors
Right Eye	1.98	1.18
Left Eye	1.26	1.11

Recall that during experiment 1, the user sits around 1 foot from the camera, facing the monitor. For experiment 1 the Euclidean distance (error) between tracked and hand-labeled pupil position for the right and left eyes are presented in figure 4.10 and figure 4.11 respectively. Table 4.1 shows the mean absolute error and the standard deviation of errors in pixels for both eyes. From this table, it is found that the mean absolute error for the right eye is 1.98 pixels and the mean absolute error for the left eye is 1.26 pixels, which means the average error is less than two pixels for both eyes. This table also presents the standard deviation of errors for both eyes. Standard deviation of errors are 1.18 pixels for right pupil and 1.11 pixels for left pupil.

For experiment 1, the table 4.2 shows the frequency of error for both pupils. From this table it is observed that in 86.40% of frames the error are in between zero to three pixels for the right eye. The table 4.2 also shows that in 93.60% of frames the error are in between zero to three pixels for the left eye. The histogram of the deviations in pixels and the cumulative percent for experiment 1 are presented in figure 4.22 and figure 4.23 for both eyes respectively. So, from these tables and figures, it is observed that the right pupil is tracked correctly in 86.40% frames, and left pupil is tracked correctly in 93.60% frames.

Table 4.2: Frequency of Deviation (Error) for Both Pupils (Experiment 1)

Pixels Deviation	Right Pupil		Left Pupil	
	Occurrences	Cumulative %	Occurrences	Cumulative %
0	55	11.00%	103	20.60%
1	80	27.00%	194	59.40%
2	129	52.80%	134	86.20%
3	168	86.40%	37	93.60%
4	48	96.00%	13	96.20%
5	9	97.80%	10	98.20%
6	5	98.80%	1	98.40%
7	6	100.00%	7	99.80%
8	0	100.00%	1	100.00%
9	0	100.00%	0	100.00%
10	0	100.00%	0	100.00%
More	0	100.00%	0	100.00%

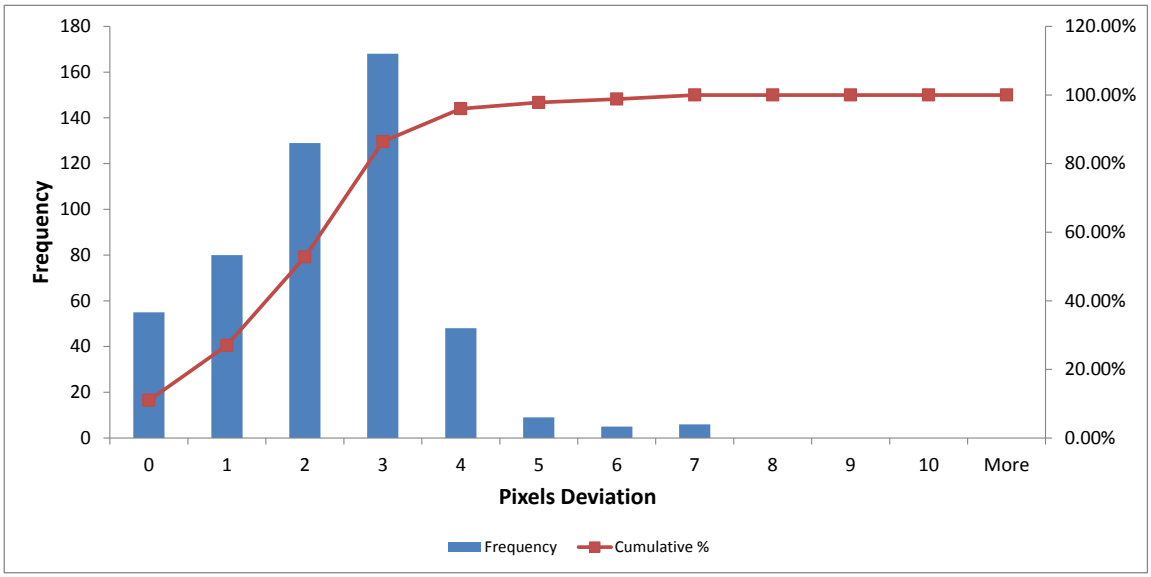

Figure 4.22: Histogram of Deviations (Error) of Right Pupil (Experiment 1)

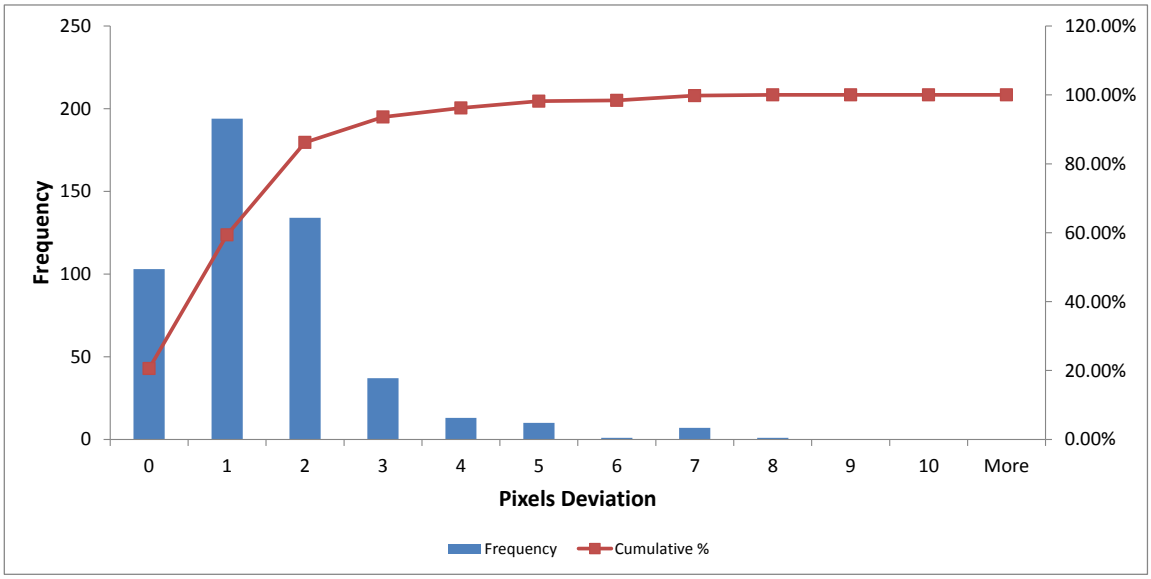

Figure 4.23: Histogram of Deviations (Error) of Left Pupil (Experiment 1)

Table 4.3: Deviation (Error) in Pixels (Experiment 2)

	Mean Absolute Error	Standard Deviation of Errors
Right Eye	2.23	1.79
Left Eye	2.19	2.15

Recall that during experiment 2, the user sits around 2 feet from the camera, facing the monitor. For experiment 2, the Euclidean distance (the error) between tracked and hand-labeled pupil position for the right and left eyes is presented in figure 4.20 and figure 4.21 respectively. Table 4.3 shows the mean absolute error and standard deviation of errors in pixels for both pupils. From this table, it is found that the mean absolute error for the right eye is 2.23 pixels and for the left eye is 2.19 pixels. This table also presents the standard deviation for both eyes for both eyes. Standard deviations of errors are 1.79 pixels for right pupil and 2.15 pixels for left pupil.

Again, for experiment 2 the table 4.4 shows that in 86.20% of frames the distance (error) between the tracked and hand-labeled pupil positions are in between zero to three pixels for the right pupil. From the same table, it is also observed that the distance (error) are in between zero to three pixels in 83.20% frames for the left pupil. The histogram of the deviations (error) in pixels and the cumulative percent are presented in figure 4.24 and figure 4.25 for both eyes respectively. So, from the table and figures, it is observed that for experiment 2, the right eye is tracked correctly in 86.20% frames, and left eye is tracked correctly in 83.20% frames.

From these tables and figures, it is observed that there are some large errors for both eyes in both experiments. After examining the video sequences, it is found that these errors are mostly due to subject's blinking. For detecting the iris, the tracker first tries to find the dark region within a radius of output of the particle filter, which is described in section 3.4. This dark region is considered to be a part of the iris. But, when the user blinks and the eyes are closed completely, in that frame the darkest regions are no longer the iris. It is

Table 4.4: Frequency of Deviations (Error) for Both Pupils (Experiment 2)

Pixels Deviation	Right Eye		Left Eye	
	Occurrences	Cumulative %	Occurrences	Cumulative %
0	28	5.60%	40	8.00%
1	80	21.60%	123	32.60%
2	170	55.60%	146	61.80%
3	153	86.20%	107	83.20%
4	31	92.40%	38	90.80%
5	11	94.60%	15	93.80%
6	4	95.40%	5	94.80%
7	3	96.00%	4	95.60%
8	3	96.60%	8	97.20%
9	5	97.60%	3	97.80%
10	7	99.00%	3	98.40%
More	5	100.00%	8	100.00%

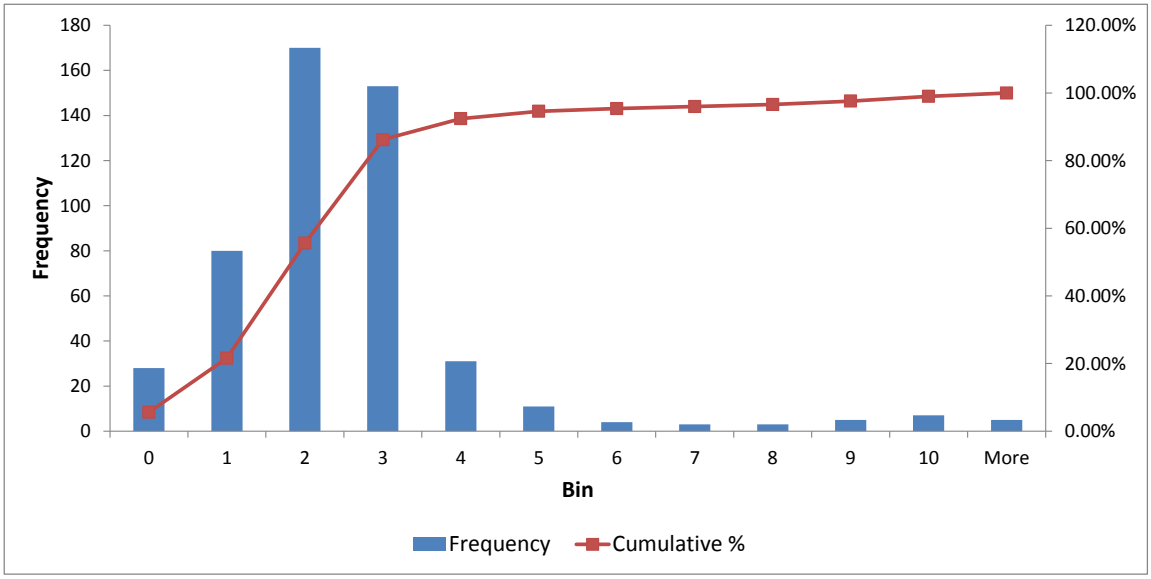

Figure 4.24: Histogram of Deviations (Error) of Right Pupil (Experiment 2)

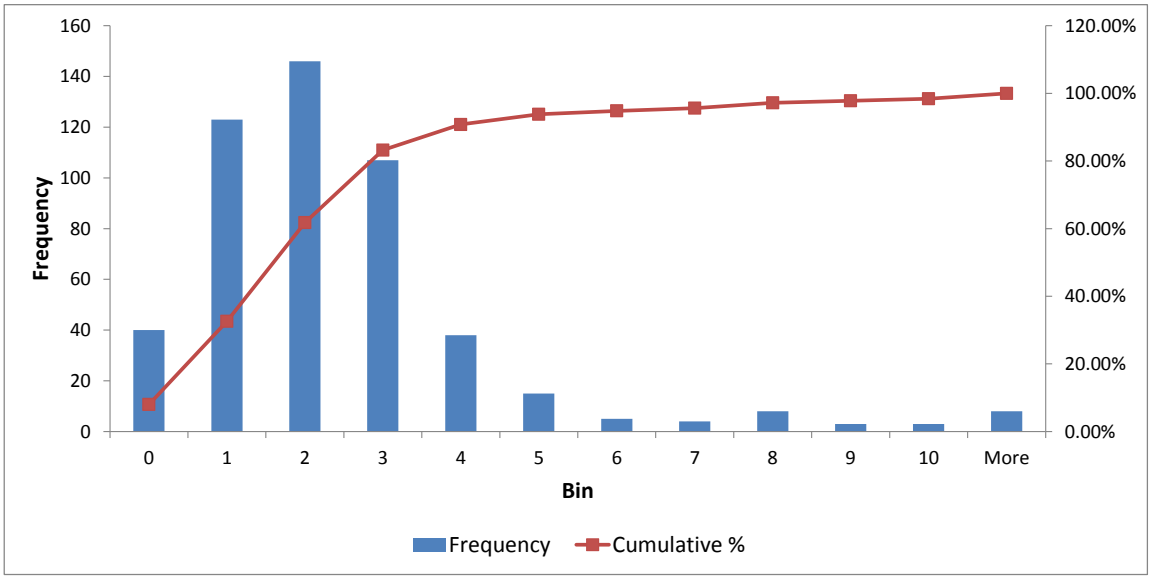

Figure 4.25: Histogram of Deviations (Error) of Left Pupil (Experiment 2)

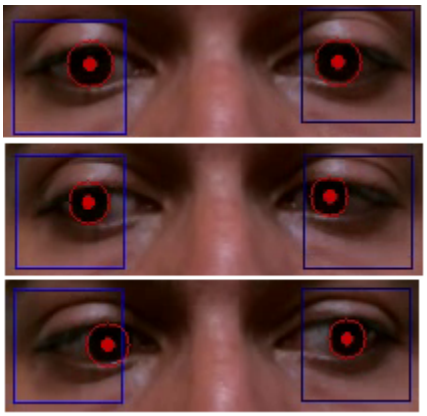

Figure 4.26: Some Good Tracking Examples

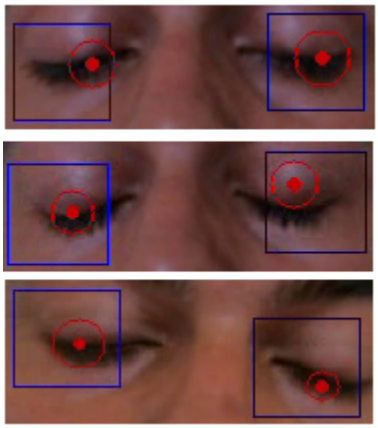

Figure 4.27: Some Bad Tracking Examples While Blinking

observed that the tracker finds a region near the eye corners or random regions near the eyelashes as the darkest in those frames. The tracker then tries to detect the iris near this darkest regions, although the iris is not present there. This phenomenon is observed in experiment 1 during frame numbers 23, 99 to 101, 141 to 142, 181 to 182 and 306 to 307 as shown in figure 4.10 and figure 4.11. From the video, it is observed that during these frames the subject is blinking and the eyes are fully or partially closed. Again, if the head is moving while the eyes are fully closed, then the tracker faces difficulty to detect the iris perfectly for some consecutive frames. In experiment 2, this phenomenon is also observed from frame number 350 to 360 for both eyes as shown in figure 4.20 and figure 4.21. During

these frames, the subject is blinking while moves his head. Some good tracking examples are shown in figure 4.26. In figure 4.27, some images are shown where the tracker faces difficulties to locate the pupil correctly.

4.3 Test Result of the Gaze Estimator

For gaze estimation, the first step is to make the eye-tracker to track both eyes. The gaze calibration is done next. For gaze calibration a white screen is shown to the user with some circles on it. The centers of these circles are predefined screen calibration points. The user needs to look at one circle for 30 frames, as the overall frame rate is around 10 fps, he/she needs to look at one circle for around 3 seconds. Then, the user should look at the next circle and so on. During this calibration process, the user is restricted from moving his/her head. However, the observation is that, although the user tries not to move his/her head, a movement of a few pixels may occur. Therefore it is necessary to find the amount of head movement, which is calculated using the method described in section 3.6. The amount of head movement is subtracted from the tracked eye positions to get the actual eye movements during the calibration process.

After the head movement compensation, for each eye the 30 eye positions for each calibration point are passed to a function to get the best candidate points described in section 3.7.2. The output of this function is then used to train the KNN classifiers. Two classifiers are used, one for each eye. The label of these training data is a number associated with the predefined position on the screen.

After the calibration process, a gaze estimation screen is shown to the user. In each frame, after head movement compensation, both pupil positions are passed to the classifiers. The output of the classifiers is a number associated with the corresponding screen calibration region. If the outputs of two classifiers are different, then a simple voting method is used as described in section 3.7.4. A highlighted rectangle is then presented on the screen to show the gaze region of the user. An example of three calibration point gaze estimation

result is shown in figure 4.28. A point (presented by the black circle) is shown to the user. Gaze estimation is presented by highlighting the background rectangle.

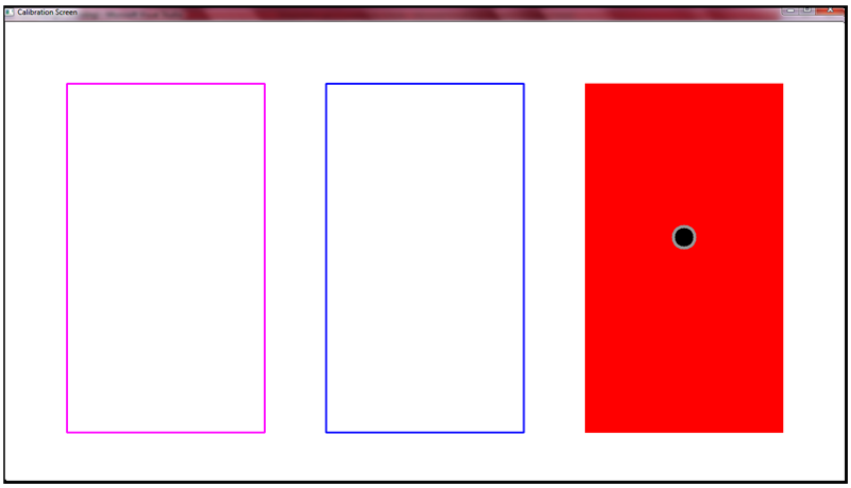

Figure 4.28: Example of an Gaze Estimation Output

4.3.1 Experiment on Two Calibration Points

In this experiment, two calibration points are used on the screen. The user needs to look at two points for around three seconds each. After the calibration process, the gaze estimation is tested. The mouse pointer is kept on one region on the monitor screen for around three seconds, and the user is asked to look at the mouse pointer. If the rectangle of that region is highlighted then it is considered that the gaze estimation is performed correctly. Otherwise, after three seconds, if the region is not highlighted, the estimation is considered as wrong, and the mouse pointer is moved to another region for testing. For the gaze estimation, 7-nearest neighbor points are considered by the classifiers.

During one experiment, the mouse pointer is kept on each region to test the gaze estimation. Each region is considered randomly for 30 times during one experiment. This experiment is repeated 10 times, meaning a total of 300 tests are conducted for each region. Table 4.5 shows the result of the experiments on two calibration points.

Table 4.5: Gaze Estimation Result of Two Calibration Points

Ground truth	Estimated Result		Accuracy
	1	2	
1	287	13	95.67%
2	9	291	97%
Overall Accuracy			96.33%

4.3.2 Experiment on Three Calibration Points

In this experiment, three calibration points are used on the screen for gaze calibration. The mouse pointer is kept on one region for around three seconds and the user is asked to look at the mouse pointer. For gaze estimation, the classifiers consider 7-nearest neighbor points are also considered by the classifiers. If a rectangle of the region over which the mouse is hovering is highlighted then it is considered that the gaze estimation is done correctly. Otherwise, after three seconds, if the region is not highlighted, the estimation is considered as wrong, and the mouse pointer is moved to another region for testing. During one experiment, here also the mouse pointer is kept for 30 times by turns on each region to test the gaze estimation. Result of 10 conducted experiments on three calibration points are presented in table 4.6.

4.3.3 Experiment on Four Calibration Points

In this experiment, four calibration points are used on the screen for gaze calibration. The same procedure is used for gaze estimation described in section 4.3.1 and section 4.3.2. Here also 7-nearest neighbor is used for estimation by the classifiers. During one experiment, here also the mouse pointer is kept on each region to test the gaze estimation. Result of 10 conducted experiments on four calibration points are presented in table 4.7.

Table 4.6: Gaze Estimation Result of Three Calibration Points

Ground truth	Estimated Result			Accuracy
	1	2	3	
1	286	11	3	95.33%
2	9	278	13	92.67%
3	6	10	284	94.67%
Overall Accuracy				94.22%

Table 4.7: Gaze Estimation Result of Four Calibration Points

Ground truth	Estimated Result				Accuracy
	1	2	3	4	
1	273	18	3	6	91%
2	27	254	14	5	84.67%
3	3	20	260	17	86.67%
4	2	11	9	278	92.67%
Overall Accuracy					88.75%

4.3.4 Result Discussion

The summary of the gaze estimation results are presented in table 4.8. From this table it is observed that, the highest accuracy of 97% is achieved for two calibration points, and lowest accuracy of 84.67% is achieved for four calibration points.

In case of the gaze estimation, the screen is divided only into vertical regions. This is due to the fact that, when the iris is near the corners of the eye, the vertical movement of the pupil is very small and very hard to detect accurately. So, only the horizontal movements are considered here for gaze estimation.

In these experiments, the errors are found mostly when the calibration points of different

Table 4.8: Summary of Gaze Estimation Results

Calibration Points	Lowest Accuracy	Highest Accuracy	Overall Accuracy
2	95.67%	97%	96.33%
3	92.67%	95.33%	94.22%
4	84.67%	92.67%	88.75%

regions are overlapped. In that case, the KNN classifier tries to calculate the nearest K points and gets erroneous neighbors. This observation is proved by the results presented in table 4.6 and table 4.7; most errors are found in between consecutive regions. It is also found that the accuracy of gaze estimation is higher in the terminal regions. This is because the chances of overlapping gaze calibration points are less than those in the middle regions. Another cause is that if the pupil positions are beyond the terminal gaze calibration points, the closest regions are only the terminal regions.

Another cause of error is that sometimes the eye tracker provides erroneous people data. As the gaze calibration points are very close, when the iris movements are too small, for one or two pixels, then the tracker sometimes provides the pupil positions one or two pixels apart from the original pupil positions, which results in erroneous gaze estimation.

In these experiments, it is also observed that, if the gaze calibration points are significantly apart, then the gaze estimations are nearly 95% correct. However, some errors still occur due to the wrong eye tracking. But, the gaze estimator faces difficulties when there are large overlaps between the gaze calibration points. It is worth noting that the extent of head movement is a crucial factor for gaze estimation - even a movement of 5 pixels might adversely affect the performance.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis an prototype eye tracking system is demonstrated which reliably track the pupil in real-time from an off-the-shelf webcam video and is robust to small rotations, translations, changes in lighting, and all but dramatic head movements. A simple machine learning-based gaze estimation system is also described.

The empirical evidence shows that the implemented eye tracker can track both eyes successfully and detect the pupil correctly in more than 86% of the frames regardless of the head movement when the subject is around one foot away from the camera. Another experiment shows that the eye tracker can track both eyes successfully and detect the pupil correctly in more than 83% of the frames regardless of the head movement when the subject is around two feet away from the camera.

Additionally the empirical evidence shows that the overall accuracy of the gaze estimation system is about 96% for experiments with two calibration points. It is also found that the overall accuracy of the gaze-estimation system is about 94% for experiments with three calibration points and about 88% for experiments with four calibration points.

However, the tracker faces difficulties to detect the pupil correctly if the eyes are fully closed. It is also faces some difficulties for couple of frames when the eyes are closed during the head movements. In more difficult situations where the eyes are covered by the hair, the tracker cannot detect the eye features because it finds the hair as the darkest location rather than the iris. Gaze estimation also faces difficulties when the monitor screen is divided in horizontal rows. Overlapping gaze calibration points and head movements during gaze

estimation also hamper gaze estimation.

5.2 Future Work

There is a need to develop an accurate model of facial position, information which could help restrict the possible eye locations, and allow the tracker to recover if the eye is momentarily lost, i.e. after a sudden head movement. A 3-D facial model using Microsoft Kinect can be used in this purpose. Future work will be needed to integrate super resolution technique for more accurate eye feature extraction. Future work will also be needed to obtain a reliable model of eye states in order to detect blinking, another phenomenon that hampers eye tracking. A better eye state model could detect the eye parameters more precisely, resulting in better tracking and also better gaze estimation. There is a need to implement a method for detecting the eye corners, which will improve accuracy of the gaze estimation. There is also a need to provide higher degree of tolerance toward eye glasses and contact lenses.

References

[1] American Optometric Association. http://www.aoa.org/x6024.xml.

[2] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

[3] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 8(6):679–698, 1986.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. Ieee, 2005.

[5] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. *Computer Vision–ECCV 2006*, pages 428–441, 2006.

[6] BioID Face Database. http://www.bioid.com.

[7] Dan Witzner Hansen and Qiang Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3), 2010.

[8] Dan Witzner Hansen, David J. C. MacKay, John Paulin Hansen, and Mads Nielsen. Eye tracking off the shelf. In *Proceedings of the 2004 symposium on Eye tracking research & applications*, ETRA '04, pages 58–58, New York, NY, USA, 2004. ACM.

[9] Dan Witzner Hansen and Arthur E.C. Pece. Eye tracking in the wild. *Computer Vision and Image Understanding*, 98(1):155 – 181, 2005. Special Issue on Eye Detection and Tracking.

[10] D.W. Hansen, J.P. Hansen, M. Nielsen, A.S. Johansen, and M.B. Stegmann. Eye typing using markov and active appearance models. In *Applications of Computer Vision, 2002. (WACV 2002). Proceedings. Sixth IEEE Workshop on*, pages 132 – 136, 2002.

[11] Craig Hennessey, Borna Noureddin, and Peter Lawrence. A single camera eye-gaze tracking system with free head motion. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, ETRA '06, pages 87–94, New York, NY, USA, 2006. ACM.

[12] W.B. Horng, C.Y. Chen, Y. Chang, and C.H. Fan. Driver fatigue detection based on eye tracking and dynamk, template matching. In *Networking, Sensing and Control, 2004 IEEE International Conference on*, volume 1, pages 7–12. IEEE, 2004.

[13] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, Taipei 106, Taiwan, 2003.

[14] Yaqin Huang, Xiucheng Dong, and Minggang Hao. Eye gaze calibration based on support vector regression machine. In *Intelligent Control and Automation (WCICA), 2011 9th World Congress on*, pages 454 –456, june 2011.

[15] Qiang Ji and Xiaojie Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5):357 – 377, 2002.

[16] Shinjiro Kawato and Nobuji Tetsutani. Detection and tracking of eyes for gaze-camera control. *Image and Vision Computing*, 22(12):1031 – 1038, 2004.

[17] L.C. Kiat and S. Ranganath. One-time calibration eye gaze detection system. In *Image Processing, 2004. ICIP '04. 2004 International Conference on*, volume 2, pages 873–876, oct. 2004.

[18] Kyung-Nam Kim and R.S. Ramakrishna. Vision-based eye-gaze tracking for human computer interface. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, volume 2, pages 324 –329 vol.2, 1999.

[19] R. Kothari and J.L. Mitchell. Detection of eye locations in unconstrained visual images. In *Image Processing, 1996. Proceedings., International Conference on*, volume 3, pages 519–522, 1996.

[20] Example of Support Vector Machine Classification. http://www.dtreg.com/svm.htm.

[21] A. Perez, M.L. Cordoba, A. Garcia, R. Mendez, M.L. Munoz, J.L. Pedraza, and F. Sanchez. A precise eye-gaze detection and tracking system. In *11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2003.

[22] S.V. Sheela and P.A. Vijaya. Article: Mapping functions in gaze tracking. *International Journal of Computer Applications*, 26(3):36–42, July 2011. Published by Foundation of Computer Science, New York, USA.

[23] S. Sirohey, A. Rosenfeld, and Z. Duric. A method of detecting and tracking irises and eyelids in video. *Pattern Recognition*, 35(6):1389 – 1401, 2002.

[24] S. A. Sirohey and A. Rosenfeld. Eye detection in a face image using linear and nonlinear filters. *Pattern Recognition*, 34(7):1367–1391, 2001.

[25] R. Stiefelhagen, J. Yang, and A. Waibel. Tracking eyes and monitoring eye gaze. In *Proceedings of the Workshop on Perceptual User Interfaces (PUI97)*, pages 98–100, 1997.

[26] Ying-li Tian, T. Kanade, and J.F. Cohn. Dual-state parametric eye tracking. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 110 –115, 2000.

[27] S. Tsekeridou and I. Pitas. Facial feature extraction in frontal views using biometric analogies. In *Proceedings of the IX European Signal Processing Conference*, volume 1, pages 315–318, 1998.

[28] iView X System Manual Version 2.4. December 2009.

[29] Vladimir Vezhnevets and Anna Degtiareva. Robust and accurate eye contour extraction. *International Conference Graphicon*, 2003.

[30] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511 – I–518 vol.1, 2001.

[31] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[32] Jian-Gang Wang and Eric Sung. Gaze determination via images of irises. *Image and Vision Computing*, 19(12):891 – 911, 2001.

[33] Jian-Gang Wang, Eric Sung, and Ronda Venkateswarlu. Eye gaze estimation from a single image of one eye. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 136 –143 vol.1, oct. 2003.

[34] Q. Wang and J. Yang. Eye location and eye state detection in facial images with unconstrained background. *Journal of Information and Computing Science*, 1(5):284–289, 2006.

[35] X. Xie, R. Sudhakar, and H. Zhuang. A cascaded scheme for eye tracking and head movement compensation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(4):487 –490, jul 1998.

[36] Xangdong Xie, R. Sudhakar, and Hanqi Zhuang. Real-time eye feature tracking from a video image sequence using kalman filter. *Systems, Man and Cybernetics, IEEE Transactions on*, 25(12):1568 –1577, dec 1995.

[37] Alan L. Yuille, Peter W. Hallinan, and David S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8:99–111, 1992.

[38] Jie Zhu and Jie Yang. Subpixel eye gaze tracking. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 124 –129, may 2002.

[39] Z. Zhu and Q. Ji. Robust real-time eye detection and tracking under variable lighting conditions and various face orientations. *Computer Vision and Image Understanding*, 98(1):124–154, 2005.

[40] Z. Zhu, Q. Ji, and K.P. Bennett. Nonlinear eye gaze mapping function estimation via support vector regression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 1132–1135. IEEE, 2006.

[41] Zhiwei Zhu and Qiang Ji. Eye and gaze tracking for interactive graphic display. *Machine Vision and Applications*, 15:139–148, 2004. 10.1007/s00138-004-0139-4.

Curriculum Vitae

Tariq Iqbal was born in Kishoreganj, Bangladesh on December 15, 1983. He graduated from Bangladesh University of Engineering and Technology (BUET) with a Bachelor's in Computer Science and Engineering degree in 2007. After completing his undergrad study, he joined the largest telecommunication company in Bangladesh, Grameen Phone Ltd. He worked there for three and half years, first working as a System Engineer and then as a Senior System Engineer.

In the spring of 2011, Tariq resumed his studies at the Graduate School of The University of Texas at El Paso (UTEP). While pursuing his Master's degree in Computer Science, he worked as a Teaching Assistant in Computer Science Department at UTEP. At the same time, Tariq became a member of the Vision and Learning Laboratory (VLL) at UTEP and also started graduate research under the supervision of Dr. Olac Fuentes.

Permanent address: 1118 Los Angeles Drive, Apt 3
El Paso, Texas 79902