UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

DEVELOPMENT OF A ROBUST ACTIVE INFRARED-BASED EYE TRACKING SYSTEM

by

Reinier Casper Coetzer

Submitted in partial fulfilment of the requirements for the degree

Master of Engineering (Computer Engineering)

in the

Faculty of Engineering, Built Environment and Information Technology
Department of Electrical, Electronic and Computer Engineering
UNIVERSITY OF PRETORIA

October 2011

"Engineering: where the noble, semi-skilled laborers execute the vision of those who think and dream. Hello, Oompa Loompas of science!"

Sheldon Cooper PhD, The Big Bang Theory

SUMMARY

DEVELOPMENT OF A ROBUST ACTIVE INFRARED-BASED EYE TRACKING SYSTEM

by

Reinier Casper Coetzer

Promoters:	Prof. G.P. Hancke
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Computer Engineering)
Keywords:	Eye detection, eye tracking, bright/dark pupil effect

Eye tracking has a number of useful applications ranging from monitoring a vehicle driver for possible signs of fatigue, providing an interface to enable severely disabled people to communicate with others, to a number of medical applications. Most eye tracking applications require a non-intrusive way of tracking the eyes, making a camera-based approach a natural choice. However, although significant progress has been made in recent years, modern eye tracking systems still have not overcome a number of challenges including eye occlusions, variable ambient lighting conditions and inter-subject variability. This thesis describes the complete design and implementation of a real-time camera-based eye tracker, which was developed mainly for indoor applications. The developed eye tracker relies on the so-called bright/dark pupil effect for both the eye detection and eye tracking phases. The bright/dark pupil effect was realised by the development of specialised hardware and near-infrared illumination, which were interfaced with a machine vision camera. For the eye detection phase the performance of three different types of classifiers, namely neurals networks, SVMs and AdaBoost were directly compared with each other on a dataset consisting of 17 individual subjects from different ethnic backgrounds. For the actual tracking of the eyes, a Kalman filter was combined with the mean-shift tracking algorithm. A PC application with a graphical user interface (GUI) was also developed to integrate the various aspects of the eye tracking system, which allows the user to easily configure and use the system. Experimental results have shown the eye detection phase to be very robust, whereas the eye tracking phase was also able to accurately track the eyes from frame-to-frame in real-time, given a few constraints.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

OPSOMMING

ONTWIKKELING VAN 'N ROBUUSTE AKTIEWE INFRAROOI GEBASEERDE OOGVOLGINGSSTELSEL

deur

Reinier Casper Coetzer

Promotors:	Prof. G.P. Hancke
Departement:	Elektriese, Elektroniese en Rekenaaringenieurswese
Universiteit:	Universiteit van Pretoria
Graad:	Magister in Ingenieurswese (Rekenaaringenieurswese)
Sleutelwoorde:	Oog deteksie, oog volging, helder/donker pupil effek

Oogvolging het 'n beduidende aantal toepassings wat wissel van die deteksie van bestuurderuitputting, die voorsiening van 'n rekenaarintervlak vir ernstige fisies gestremde mense, tot 'n groot aantal mediese toepassings. Die meeste toepassings van oogvolging vereis 'n nie-indringende manier om die oë te volg, wat 'n kamera-gebaseerde benadering 'n natuurlike keuse maak. Alhoewel daar alreeds aansienlike vordering gemaak is in die afgelope jare, het moderne oogvolgingstelsels egter nogsteeds verskeie uitdagings nie oorkom nie, insluitende oog okklusies, veranderlike beligtingsomstandighede en variansies tussen gebruikers. Die verhandeling beskryf die volledige ontwerp en implementering van 'n kamera-gebaseerde oogvolgingsstelsel wat in reële tyd werk. Die ontwikkeling van die oogvolgingsstelsel maak staat op die sogenaamde helder/donker pupil effek vir beide die oogdeteksie en oogvolging fases. Die helder/donker pupil effek was moontlik gemaak deur die ontwikkeling van gespesialiseerde hardeware en naby-infrarooi illuminasie. Vir die oogdeteksie fase was die akkuraatheid van drie verskillende tipes klassifiseerders getoets en direk vergelyk, insluitende neurale netwerke, SVMs en AdaBoost. Die datastel waarmee die klassifiseerders getoets was, het bestaan uit 17 individuele toetskandidate van verskillende etniese groepe. Vir die oogvolgings fase was 'n Kalman filter gekombineer met die gemiddelde-verskuiwings algoritme. 'n Rekenaar program met 'n grafiese gebruikersintervlak was ontwikkel vir 'n persoonlike rekenaar, sodat al die verskillende aspekte van die oogvolgingsstelsel met gemak opgestel kon word. Eksperimentele resultate het getoon dat die oogdeteksie fase uiters akkuraat en robuust was, terwyl die oogvolgings fase ook hoogs akuraat die oë gevolg het, binne sekere beperkinge.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

ACKNOWLEDGEMENTS

The author would like to thank the following persons and institutions for their support:

- Professor G.P. Hancke for his continual advice and financial support for equipment and international conference attendance during the course of this research.
- Centre for Teletraffic Engineering in an Information Society (CeTEIS) for their bursary during 2010.
- Asheer Buchoo and Jason de Villiers at DPSS at the CSIR for their suggestions and insights into some of the image processing problems I encountered.
- Jason Page at EBV Elektronik for all his help during the development of the hardware.
- Noelanie, my fianceé, who patiently supported me during the writing of this thesis.
- My family who also supported me during the writing of this thesis.
- The Lord Almighty who provided me with the opportunity and ability to perform this research.

LIST OF ABBREVIATIONS

AAM	Active Appearance Model
AdaBoost	Adaptive Boosting
ANN	Artificial Neural Network
API	Application Programming Interface
CIE	International Commission on Illumination
COTS	Commercial Off-The-Shelf
CSIR	Council for Scientific and Industrial Research
DPSS	Department of Defense, Peace, Safety and Security
EKF	Extended Kalman Filter
FNR	False Negative Rate
FPR	False Positive Rate
FPS	Frames Per Second
GigE	Gigabit Ethernet
GUI	Graphical User Interface
HMM	Hidden Markov Models
ISR	Interrupt Service Routine
LED	Light Emitting Diode
MLP	Multi-layer perceptron
NIR	Near-Infrared
PCA	Principle Component Analysis
PCB	Printed Circuit Board
PPCA	Probabilistic Principle Component Analysis
PWM	Pulse Width Modulation
RBF	Radial Basis Function
ROI	Region Of Interest
SDK	Software Development Kit
SVM	Support Vector Machine
TNR	True Negative Rate
TPR	True Positive Rate
UKF	Unscented Kalman Filter

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 MOTIVATION

This research started out as problem posed by Eskom, the major electricity public utility company of South Africa. Eskom noticed that they were losing millions of rands and human lives every year due to a number of fatigue related accidents involving their coal carrying trucks. As a result they were interested in a system that would be able to non-intrusively monitor their truck drivers to pro-actively detect fatigue. Consequently an extensive literature survey [1] was performed to identify the most suitable fatigue detection approaches and the conclusion was made that a hybrid fatigue detection approach, consisting of a camera-based driver monitoring component (directly) and a component that monitors how the driver handles the vehicle (indirectly), would be the most suitable.

However, following the power shortages in South Africa during 2007 and 2008, Eskom lost interest in such a system and had to channel their resources elsewhere to contain the nation-wide power shortages. In addition, consultation with the local trucking industry revealed that installing a camera in a truck to monitor the driver would be unacceptable for the trade unions, since it implies that the employer could at all times "keep an eye" on the employee, which could be perceived as a violation of human rights. Therefore Eskom could no longer afford the development of a driver fatigue detection system and a camera-based approach for monitoring driver fatigue was in any case ruled out as a possibility, although not even on technical grounds!

Nevertheless, the literature also revealed that there is a number of other applications for eye tracking systems and despite extensive research in the field, eye tracking remains a challenging

problem that has not yet been completely solved [2]. The main challenges that eye tracking systems still face are eye occlusions, variable ambient lighting conditions and inter-subject variability. Currently, a number of commercial eye tracking systems are available, but they are regrettably still expensive and also still suffer from the aforementioned limitations.

As a result eye tracking was indentified as an interesting and challenging research problem with a number of applications besides driver fatigue monitoring, which were ultimately the main motivations for the research presented in this thesis.

1.2 OBJECTIVES

The main objective of this research was therefore to develop a robust eye tracking system, based on active near-infrared illumination that would be capable of tracking the eyes of a user with little to no initialization and under as many different situations as possible. Due to a lack of resources in terms of manpower, time and funding it was never expected that this research would be able solve all of the above mentioned challenges that current eye tracking systems still face. Eye tracking is a real-world problem, which implies that an eye tracking system is typically only considered useful if it is capable of functioning with as little constraints as possible. However, based on the magnitude of the problem, it was decided to impose the following constraints on the operating conditions of the system:

- The system shall only be used for indoor applications.
- The user shall be at a relatively fixed distance from the camera, depending on the actual lens being used (1400mm to 1600mm in this case).
- The system shall be capable of tracking the eyes under reasonable free head movements. Reasonable head movements are defined as smooth head movements that are not excessively fast and do not rapidly change direction.
- Apart from normal eye tracking without glasses, the system shall only be capable of tracking the eyes when the user is wearing selected types of glasses.

Given these constraints, the system shall be capable of tracking the eyes in real-time by means of two phases:

1. **Eye detection**: The bright/dark pupil effect shall be used to produce eye candidates

at a very low computational cost. The resulting eye candidates shall then have to be classified as either eyes or non-eyes using a strong classifier such as Support Vector Machines or AdaBoost. The final step shall then be the verification of the detected eyes, by using geometric constraints.

2. **Eye tracking**: Once the eyes have been successfully detected, it shall be tracked from frame to frame using a combination of Kalman filtering and mean-shift tracking to compensate for their relative weaknesses.

It was decided to follow this particular approach based on an extensive literature survey performed on existing eye tracking techniques.

1.3 CONTRIBUTIONS

During the course of the work presented in this thesis, the following research outputs have been produced:

1. "Driver fatigue detection : A survey" in the proceedings of IEEE AFRICON 2009, Nairobi, Kenya [3].
2. "Driver fatigue detection based on eye tracking" a work-in-progress paper in the proceedings of the South African Telecommunication, Networks and Applications Conference (SATNAC 2010), Stellenbosch, South Africa [4].
3. "Eye detection for a real-time vehicle driver fatigue monitoring system" in the proceedings of the IEEE Intelligent Vehicles Symposium (IV 2011), Baden-Baden, Germany [5].

Besides the above mentioned research papers, the work presented in this thesis made the following contributions:

- A complete proof-of-concept eye tracking system (hardware and software) was designed and developed from first principles in which much practical research had to be performed to determine the best possible combination of different hardware components, to ultimately obtain a strong bright pupil effect. The complete and detailed design of this system is also presented in this thesis, which would enable a competent engineer to almost exactly reproduce the system.

- In terms of the bright pupil effect, a number of practical limitations (presented throughout this thesis) were identified during the course of this research. It was found that these limitations were very seldomly mentioned in the literature (perhaps for the sake of good results) and the author feels that by presenting these results, albeit negative results in some instances, a significant contribution has been made.

- In terms of eye detection, the classification accuracy and the ability to generalize on unseen examples were tested for three strong machine learning techniques including Support Vector Machines (SVMs), Adaptive Boosting (AdaBoost) and Artificial Neural Networks (ANN). These machine learning techniques were directly compared to each other, specifically for the purpose of eye classification. To the author's knowledge this comparison has not been presented the literature before [5].

- In terms of eye tracking, the author believes that a novel approach has been followed in tracking the eyes, by combining Kalman filtering with mean-shift tracking.

- The developed eye tracking system can therefore serve as a platform for various other research activities (not related to eye tracking research itself), which only requires an eye tracker to enable the particular research.

1.4 STRUCTURE OF THIS THESIS

This thesis is structured as follows: *Chapter 2* provides the reader with a background of the most important concepts and techniques that were used in this research, while *Chapter 3* provides a literature survey of the current state of eye tracking, as well as related work. *Chapter 4* provides the complete detailed hardware and software design of the developed eye tracking system, while *Chapter 5* provides the results thereof. Finally, *Chapter 6* draws some conclusions and suggests some future work.

Chapter 2

BACKGROUND

2.1 THE BRIGHT/DARK PUPIL EFFECT

The concept of using near-infrared (NIR) illumination to track the human eye was first suggested in the late 1980s by Hutchinson *et al* [6] with the development of their eye-gaze tracker Erica, which served as a human interface for severely handicapped people. Hutchinson *et al* found that if the NIR illumination (a LED in their case) was placed on-center with the camera lens, a fraction of the light was reflected off the corneal surface of the human eye (called the first Purkinje image) to produce an intense area of NIR light in the captured images, called the glint. In addition, some of the NIR light also entered the pupil and was reflected of the retina to produce the so-called bright pupil effect. This bright pupil was found to be much larger than the glint and although less intense than the glint, both the bright pupil and the glint were significantly more intense than the surrounding iris of the eye.

As a result of this contrast it was relatively easy to detect the position of both the glint and the bright pupil in an image and this was exactly what Hutchinson *et al* used to determine the direction of a person's eye-gaze. If the center of a person's glint coincided with the center of the bright pupil, the person was looking directly at the camera and this could be used as the reference eye-gaze position. The relative positions of the bright eye and the glint could then be used to determine the direction in which the person was looking. Thomas Hutchinson also filed a patent [7] for this technique.

The bright pupil effect was already at that point in time not an unknown phenomenon, since it is very similar to the red-eye effect that frequently occurs in general photography. The red-eye effect is usually a result of the combination of relatively low ambient light and the

flash being too close to the camera's lens. Yoshinobu Ebisawa and Shin-ichi Satoh [8] realized that they could use this red-eye effect (or bright pupil effect for grayscale images) to robustly detect the pupils in an image, by using two NIR light sources that were synchronized with a CCD camera.

For their NIR light sources, Ebisawa and Satoh simply used two NIR LEDs that were placed in a specific manner with regards to the camera's lens. One LED was placed on-center with the camera's lens, while the second LED was placed 23mm away from the center of the lens. These LEDs were then synchronized with the even and odd field signals from the camera, with the on-center LED being switched on with odd fields (to produce the bright pupil effect) and the off-center LED with even fields (to produce the dark pupil effect). The NIR light from the off-center LED is reflected differently from the retina and typically away from the camera's lens. An example of the bright pupil effect is shown in Figure 2.1, whereas an example of the dark pupil effect is shown in Figure 2.2.

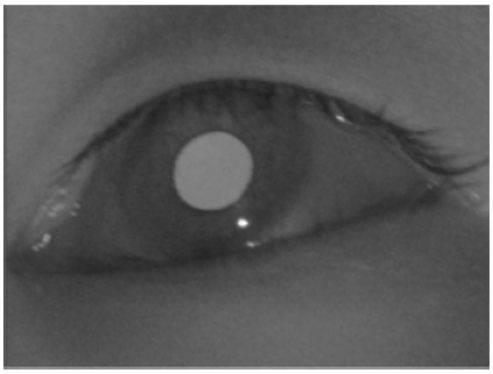

Figure 2.1: The bright pupil effect as produced by on-center NIR illumination.

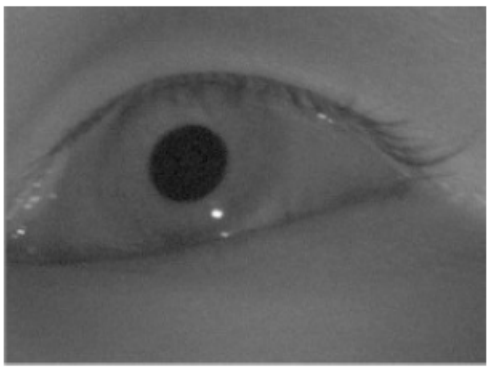

Figure 2.2: The dark pupil effect as produced by off-center NIR illumination.

Normally the even and odd fields of a camera are combined to form a single image, but in this case they were used as two separate images. These images were essentially the same, but with an important difference: the pupils. By subtracting the dark pupil image from the bright pupil image (i.e. the pixel intensity values), the pupils were almost effectively segmented from the background in the resulting difference image.

Ebisawa further improved the pupil segmentation by thresholding the difference image and also attempted to remove light reflected from normal transparent eye glasses, which was successful under some conditions according to his article [9]. Morimoto *et al* [10] developed a system that was very similar to the one developed by Ebisawa, but used rings of NIR LEDs as opposed to single LEDs to obtain the bright/dark pupil effect. By using this LED ring configuration, Moritmoto *et al* found that the glint detection was more robust. In addition Morimoto *et al* developed an eye tracker by means of the bright/dark pupil difference images. However, this was not exactly a tracking algorithm in the true sense of the word, but more of a repeated eye detection process.

The implication of using the bright/dark pupil effect is that the eyes can be robustly detected with minimal computational effort at the expense of some additional hardware. This is an important advantage for real-time eye tracking systems, since the eye candidates are readily available as opposed to first scanning the entire image in order to detect the eyes. Although the bright pupil effect is more easily obtained under low ambient lighting conditions, it has been proven to be relatively insensitive to ambient lighting conditions. In addition, NIR light is barely visible to the human eye and will therefore not immediately interfere with the given task at hand, while being used for either detecting or tracking the eyes.

Although the bright/dark pupil effect is indeed a very robust method for eye segmentatation, it is not without limitations. The first problem that has not yet been completely overcome is in situations where a person is wearing glasses. Glasses also reflect the incident NIR light to form intense spots on both the bright and dark pupil images. Since the position of the NIR light sources differs, the position of the reflected spots from the glasses are also located at different positions in a bright/dark pupil image pair. As a result, the image differencing step does not necessarily diminish the effect from the light reflected of the glasses. If the position of the reflected light does not overlap with the actual position of the eye, this problem can be overcome. However, the problem becomes particularly severe if the position of the reflected light coincides with the actual position of the eye. Eye detection even becomes impossible

when certain sunglasses are worn. In particular, high quality polarized sunglasses almost completely prevents NIR light from passing through the glass, nullifying the bright/dark pupil effect. This is of course a deliberate feature of the polarized sunglass design in order to prevent glare. These problems are further discussed in *Chapter 5*.

A second potential problem with the bright pupil effect is that it is not consistent among certain groups of people. These inconsistencies were first observed by Nguyen *et al* [11] while developing their gaze tracker and as a result Nguyen *et al* did a further study to determine which factors influence the bright pupil effect (i.e. average pupil intensity for on-center illumination). The first factor that they considered was how far the NIR light source can be horizontally translated, away from the center of the lens to still obtain the bright pupil effect. They found that for a horizontal translation beyond approximately 20 mm from the center of the lens (for their setup at least), the bright pupil effect proved difficult to obtain. However, this particular problem can easily be mitigated since its controllable by the designer.

From all the other factors that Nguyen *et al* considered, the two factors that resulted in the most variation among subjects were the pupil size and ethnicity. It is well known that pupil sizes are not uniform among people and as a result larger pupil sizes are expected to produce a more intense bright pupil effect. However, in their experiments Nguyen *et al* forced the pupil sizes of different subjects to be of the same size, but still found large variations in the average pupil intensities. Nguyen *et al* noted that a possible cause for these variations could perhaps be attributed to ethnicity, where they found large variations in the bright pupil effect among Hispanic, Caucasian and Asian people. Hispanic people had the highest average bright pupil intensity, followed by Caucasians and finally Asian people who had the lowest average bright pupil intensity. The authors did note that their sample size was too small to conclusively prove this observation, but in this research a similar trend was observed among Caucasian, African and Indian people.

The final potential limitation of the bright/dark pupil effect was not so much a technical problem but more of a practical one. The human eye may not be able to perceive light at wavelengths above 700 nm, but this of course does not mean that the light is not potentially harmful. In fact the potential risk can then even be higher, since the eye cannot adapt to the light or trigger an aversion reaction. For example, a person is typically not able to directly look into the sun since it will result in a painful and unpleasant experience and it is exactly this aversion reaction that helps prevent damage to the human eye.

It is generally accepted that short bursts of NIR light used in the current eye tracking systems are not harmful to the human eye. However, there is still quite a lot of uncertaintity surrounding the continuous exposure to NIR light for extended periods of time. This is typically the case for eye trackers used by severely disabled people, which use their eyes as the only means of interaction. Another example is driver fatigue detection systems based on eye tracking, where the eyes have to be tracked for long periods while driving.

To determine how safe the continuous exposure to NIR light is, Mulvey *et al* [12] did a survey on existing eye tracking systems. From the five separate potential eye hazards specified by the optical radiation safety guidelines, Mulvey *et al* identified thermal injury to the retina (400 nm to 1400 nm) and NIR thermal hazards to the eye lens (800 nm to 3000 nm) as the only hazards relevant to NIR based eye tracking. From their measurements, the authors found that for brief periods (minutes to hours) the current available eye trackers did not pose any hazards to the eyes. However, for extended periods (days to years) their findings were inconclusive. As a result of these inconclusive findings Division 6 of the International Commission on Illumination (CIE) formed a technical committee (TC6-64) to perform research on this topic. At the time of writing this thesis their findings have not yet been published.

2.2 SUPPORT VECTOR MACHINES

2.2.1 History

The concept of Support Vector Machines (SVMs) was initiated in the late 1970's by Vladimir Vapnik, but was only fully developed during the early 1990's while he worked at AT&T Bell Labs and was subsequently published in [13] and [14]. SVMs are a set of supervised learning methods for pattern recognition in machine learning, which started out in its simplest form as binary classification but was subsequently extended to multi-class classification and regression analysis. SVMs have been successfully used in a number of real-world applications including handwritten character recognition [14], [15], spam classification [16], speaker verification [17], facial recognition [18] and eye detection [19] to name only a few.

2.2.2 SVM classification

SVMs start out by trying to solve the general binary classification problem of estimating a function $f : \mathbb{R}^N \rightarrow \{-1, +1\}$ from a set of training examples with unknown probability

distribution $P(\mathbf{x}, y)$:

$$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n) \in \mathbb{R}^N \times Y, \qquad Y = \{-1, 1\} \tag{2.2.1}$$

in such a way so that f will be able to correctly classify an unseen example, $(\mathbf{x}, y)$, assumed to be drawn from the same probability distribution $P(\mathbf{x}, y)$ as the training examples. If $f(\mathbf{x}) \geq 0$, the example will be assigned to the $+1$ class and otherwise to the -1 class. The optimal function f can be obtained by minimizing the expected error or risk, but this cannot be directly performed since the underlying probability distribution $P(\mathbf{x}, y)$ is unknown. As a result, a function should be estimated that is as close as possible to the optimal solution, based upon the given training examples and the properties of the function class F from which f is chosen.

Perhaps the most simple induction rule to approximate the minimum risk, is by minimizing the empirical risk. However, with a small set of examples, large deviations are typical, which can result in overfitting. A possible way to mitigate the problem of overfitting is by restricting the complexity of the function class F from which the function f is chosen. This implies that a simple function, such as a linear function, is preferable to a complex function.

It is at this point where Vapnik-Chervonenkis (VC) theory and the structural risk minimization (SRM) principle come into play, which is a particular method of controlling the complexity of the function class F. The complexity of the function class F is described by the VC dimension, which in essence measures the amount of training examples that can be shattered (i.e. separated) for all possible functions in each function class. Next the SRM proceeds by selecting the function class, say F_i, and the particular function of that class, say f_i, that minimizes the upper bound of the generalization error. However, in practice this bound is typically difficult to compute or not very useful, for example the VC dimension of the class might be unknown or infinite in which case the bound cannot be minimized. Fortunately in cases where the training examples can be separated by the function class of hyperplanes, it was shown that the VC dimension itself can be bounded by the margin. The margin is defined as the minimal distance of a sample to the decision surface (i.e. the hyperplane).

Although the VC dimension itself can now be bounded by using hyperplanes as classifiers, the next obvious problem is that in practice it is rarely the case that the training examples can be properly separated by such a simple linear classifier (resulting in underfitting). Fortunately,

there is again an elegant way that this problem can be overcome by making use of kernels.

Let's first consider why one would use kernels in the first place. As mentioned before, it is frequently the case that training examples in the input space cannot be separated by a linear classifier. However, if this training examples can somehow be mapped (non-linearly) to a potentially much higher dimensional feature space, $\mathcal{F}$, the training examples might then be effectively linearly separable:

$$\Phi : \mathbb{R}^N \rightarrow \mathcal{F} \tag{2.2.2}$$

$$\mathbf{x} \mapsto \Phi(\mathbf{x}) \tag{2.2.3}$$

where Φ is the mapping function from the input space to the feature space. The implication is that the same linear classifier (or any other classifier for that matter) can now be used to separate the non-linear training examples in the feature space, $\mathcal{F}$, instead of in the input space, $\mathbb{R}^N$. At first glance this concept might seem like a bad idea when considering the curse of dimensionality from statistics, which in short states that the complexity of an estimation problem increases severely when mapping to a higher dimension. Fortunately, statistical learning theory also states that the contrary can be true, i.e. if the complexity of the classifier is low (read linear) then learning can actually be easier in the feature space.

Although learning in a higher dimensional feature space can be easier (given a simple linear classifier), there are practical problems associated with mapping to a higher dimension. In the real-world it is often the case that the patterns that have to be classified are large and highly non-linear, which implies that the training examples might only be linearly separable in an extremely high dimensional feature space. As a result it becomes computationally infeasible to first map and then execute the algorithm in the feature space. It is at this point where the concept of kernels comes to rescue. In the case of a linear classifier, the separating hyperplane is of the general form

$$f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b \tag{2.2.4}$$

From equation 2.2.4 it is evident that the computation of the scalar product of two input vectors will be a frequent occurrence in the feature space. Without going into any of the mathematical formulation, a kernel will directly compute the scalar product of the vectors

in $\mathcal{F}$, without explicitly using or even knowing the mapping function, Φ. As a result, kernels make it possible to compute scalar products in higher dimensional spaces, where it would have hardly been possible otherwise. This is one of the most important concepts of SVMs.

SVMs is based upon all the concepts discussed thusfar. When considering hyperplanes (as defined in equation 2.2.4) as a classifier, the decision function to classify an unseen example is:

$$y = \text{sign}((\mathbf{w} \cdot \mathbf{x}) + b) \tag{2.2.5}$$

which implies that

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \quad \text{if} \quad y_i = 1, \tag{2.2.6}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \quad \text{if} \quad y_i = -1 \tag{2.2.7}$$

Consequently the conditions for classification without any training error are

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \quad i = 1, \ldots, n \tag{2.2.8}$$

However, as previously mentioned, the linear classifier will typically not be able to separate the training examples in the input space, so it should first be mapped to the feature space. The resulting conditions for perfect classification in the feature space are therefore

$$y_i((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) \geq 1, \quad i = 1, \ldots, n \tag{2.2.9}$$

The goal of SVM learning is to obtain $\mathbf{w} \in \mathcal{F}$ and b, so that the expected risk is minimized. As previously discussed, the expected risk cannot be directly obtained so the bound will instead be minimized. By means of some rigorous math (which is far beyond the scope of this research) and the introduction of Lagrange multipliers and kernel functions, the minimization of the bound results in a dual quadratic optimization problem:

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) \tag{2.2.10}$$

$$\text{subject to} \quad \alpha_i \geq 0, \quad i = 1, \ldots, n, \tag{2.2.11}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.2.12}$$

where the α_i's are the Lagrange multipliers and $(\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$ was replaced by some kernel function, $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$. By solving the dual optimization problem, the coefficients $\alpha_i, i = 1, \ldots, n$ are obtained which are in turn used to minimize the bound of the expected risk. Therefore the resulting decision function that will be used to classify an unseen sample is:

$$f(\mathbf{x}) = \text{sgn}(\sum_{i=1}^{n} \alpha_i y_i \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) + b) \tag{2.2.13}$$

Up to this point, only the perfectly separable case of training examples was considered, corresponding to an empirical risk of zero. In reality however, the examples will be noisy and typically not perfectly separable and if the aim remains to obtain an empirical risk of zero, overfitting is likely to occur. In order for SVMs to generalize better on noisy (real-world) data, the use of slack-variables was first suggested by [20] to relax the hard-margin constraints. These slack-variables are introduced in equation 2.2.9:

$$y_i((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n \tag{2.2.14}$$

The slack-variables from equation 2.2.14 will now allow for some classification errors. The goal of the SVM solution is still to minimize the upper bound of the VC dimension, but now the upper bound, $\sum_{i=1}^{n} \xi_i$, on the empirical bound is additionally minimized. In the process of minimization a regularization constant, $C > 0$, is also introduced which controls the tradeoff between the empirical error and the complexity term of the VC bounding function. As a consequence, the dual quadratic optimization problem from equations 2.2.10 - 2.2.12 are slightly modified:

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) \tag{2.2.15}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n, \tag{2.2.16}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.2.17}$$

By making use of the slack-variables, box constraints are formed that limit the size of the Lagrange multipliers ($\alpha_i \leq C$) and by tuning C, the generalization performance of the SVM can be enhanced.

The solution to most of these optimization problems are based on the Karush-Kuhn-Tucker (KKT) second order optimality conditions, which state necessary and in some cases sufficient conditions for a set of variables to be optimal for a given optimization problem. In turn there are a number of proposed approaches (based on the KKT conditions) to solve the particular SVM dual optimization problem including chunking, decomposition methods and sequential minimal optimization (SMO), with the latter approach being used in the SVM implementation of OpenCV. SVM optimization techniques are also far beyond the scope of this research, and the reader is referred to the SVM literature for more details.

2.3 ADAPTIVE BOOSTING

2.3.1 History

Compared to all the techniques presented in this chapter, boosting has the most recent origins dating back to the late 1980's when Robert Schapire [21] published the first boosting algorithm. Boosting is the notion of combining a number of *"weak"* classifiers to form a single *"strong"* classifier. In effect the *"weak"* classifiers are *"boosted"* to obtain an arbitrarily accurate learning algorithm. In this context a classifier is considered *"weak"* if it performs only slightly better than random guessing (e.g. the flipping of a coin). The accuracy of the resulting learning algorithm is described as arbitrarily since the accuracy can be controlled by the amount of *"weak"* classifiers used.

Boosting deliberately does not specify the *"weak"* classifiers, since any type of classifier can be *"boosted"* to improve its accuracy. In addition, the chosen base classifier does not even have to be *"weak"*, it can be a *"strong"* classifier in its own right. In fact, Li *et al* [22] have illustrated how the SVM classifier can be used as the base classifier for the AdaBoost

algorithm. However, such an approach probably defies the goal of simplicy of boosting, but just illustrates its flexibility in essentially improving *any* classifier. In practice *"stumps"* (single-split trees with only two terminal nodes) have been found to be very effective *"weak"* classifiers for boosting.

The first boosting algorithm that was presented by Schapire [21] was further improved by Freund [23], but both these algorithms still required previous knowledge of the *weak* classifiers to obtain a single accurate prediction rule. However, these requirements were no longer necessary with the adaptive boosting (AdaBoost) algorithm developed by Freund and Schapire [24]. With their version of the boosting algorithm, the algorithm adapts to the accuracies of the *weak* classifiers to obtain a weighted majority prediction rule with the weight of each *weak* classifier being a function of its accuracy. Their AdaBoost algorithm for the binary classification problem is explained in Table 2.1.

Table 2.1: The AdaBoost algorithm as first presented by Freund and Schapire [24].

The Adaptive Boosting (AdaBoost) algorithm

1. Given: $(x_1, y_1), ..., (x_m, y_m)$ where $x_i \in \mathbf{X}, y_i \in \mathbf{Y} = \{-1, +1\}$
2. Initialize $w_i = \frac{1}{m}$
3. For $t = 1, ..., T$:
   (a.) Train the *weak* classifier using the distribution w_i
   (b.) Get the *weak hypothesis* $h_t : \mathbf{X} \to \{-1, +1\}$:
   $$\epsilon_t = Pr_{i \sim w_i} = [h_t(x_i) \neq y_i]$$
   (c.) Choose $\alpha_t = \frac{1}{2} ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
   (d.) Update:
   $$w_{t+1}(i) = \frac{w_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t}, & \text{if } h_t(x_i) \neq y_i \end{cases}$$
   Where Z_t is a normalization factor, for w_{t+1} to be a distribution
4. Output the final prediction rule:
   $$H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

2.3.2 AdaBoost Classification

AdaBoost starts out by taking as input the training set $(x_1, y_1), ..., (x_m, y_m)$ where each x_i is an example from some feature space $\mathbf{X}$ and y_i is the corresponding label of the example and belongs to the label set $\mathbf{Y}$. For the binary classification problem $\mathbf{Y} = \{-1, +1\}$. Given some *weak* classifier, AdaBoost then calls this *weak* classifier repeatedly in a number of rounds $t = 1, ..., T$. A main goal of AdaBoost is to obtain a distribution over the training set and therefore the weight of a distribution of a training example i for round t is denoted as $w_t(i)$. All of these weights are initialized to $1/m$, but for each round the weights are increased for the example distributions that were incorrectly classified by the *weak* classifier. In effect the *weak* classifier is forced to concentrate on the difficult examples in the training set.

For each round t, the *weak* classifier is trained to find a *weak hypothesis* $h_t : \mathbf{X} \to \{-1, +1\}$ based on the weights w_i for that round. The goodness of the *weak hypothesis* is measured by its classification error:

$$\epsilon_t = Pr_{i \sim w_t} = [h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i \neq y_i)} w_t(i) \tag{2.3.1}$$

Given the classification error of the *weak hypothesis* h_t, AdaBoost calculates the parameter:

$$\alpha_t = \frac{1}{2} ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{2.3.2}$$

Intuitively, the parameter α_t serves as a weight for h_t to indicate its importance in the final classifier. From 2.3.2, it can be seen that $\alpha_t \geq 0$ if $\epsilon_t \leq \frac{1}{2}$ and as ϵ_t becomes smaller, the weight α_t becomes larger. This simply means that a more accurate *weak* classifier will have more “say" in the final majority vote.

The next step is to update the weights w_t according to the following rule:

$$w_{t+1}(i) = \frac{w_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t}, & \text{if } h_t(x_i) \neq y_i \end{cases} \tag{2.3.3}$$

The effect of this step is necessary to increase the weights of the training examples that were misclassified by h_t and to decrease the weights of the examples that were actually correctly classified. Consequently the weights w_t force AdaBoost to focus on the *difficult* examples.

The final prediction rule $H(x)$ is thus the weighted majority vote of the T *weak hypotheses* together with their associated weights:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{2.3.4}$$

2.3.3 AdaBoost Classification Error

As with most classification techniques, AdaBoost is concerned with the reduction of the training and generalization error. What is particularly important about the relatively simple AdaBoost algorithm is that it is seemingly resistant to overfitting. In recent years it has been shown to eventually overfit with enough boosting rounds, but this can mainly be attributed to noise in the data. Due to this very interesting characteristic of AdaBoost, this section attempts to provide some insight into this occurrence.

Given the binary classification problem, a hypothesis that guesses the class of a sample at random, has an error rate of $\frac{1}{2}$. Freund and Schapire [24] defined the training error of AdaBoost as $\epsilon_t = \frac{1}{2} - \gamma_t$, which essential measures how much better than random guessing the *weak hypotheses* h_t are. They then proved that the upper bound of the training error of the final classifier $H(x)$ is:

$$\prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_t \gamma_t^2\right) \tag{2.3.5}$$

From Equation 2.3.5 it can therefore be seen that if the individual *weak hypotheses* are only slightly better than random guessing, the training error reduces exponentially as the number of boosting rounds are increased. An important advantage of AdaBoost is that it does not require the lower error bound (which is in practice difficult to obtain) of the individual *weak hypotheses a priori*, but adapts (hence the name) to these individual error rates.

The generalization error of a classifier is defined as the classification errors made on previously unseen examples, in other words how well the classifier can formulate a general prediction rule, based only on training on a sub-set of the actual data distribution. To initially characterize the performance of AdaBoost, Freund and Schapire [24] found an upper bound on the generalization error in terms of the training error (Equation 2.3.5), the training sample size m, the VC dimension d of h_t and finally the number of boosting rounds. Recall from

the previous section on SVMs that the VC dimension is a measure of the complexity of a function class, which is the space of the *weak hypotheses* in the case of AdaBoost. This upper bound of the generalization error was found to be:

$$\hat{P}r[H(x) \neq y] + \hat{O}\left(\sqrt{\frac{Td}{m}}\right) \tag{2.3.6}$$

where $\hat{P}r[\cdot]$ is the empirical training error probability. From Equation 2.3.6 it is evident that AdaBoost should be prone to overfitting as the number of rounds T are increased. Although overfitting does occur in a few instances, AdaBoost is remarkably resistant to overfitting in most cases. A number of authors, including [25], [26] and [27], have empirically observed AdaBoost not to overfit even when executing it on thousands of rounds.

Based on these findings, Schapire *et al* [28] provided an alternative analysis of the upper generalization bound in terms of the margins of the training examples. They defined the margin of a training example (x, y) as:

$$\text{margin}_{(x,y)} = \frac{y \sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \tag{2.3.7}$$

This training margin is a number in the range $[-1, +1]$ and is positive if and only if H classifies the example correctly. In addition the magnitude of the margin can be seen as an indication of the confidence of the prediction. For larger margins Schapire *et al* proved that the training set provided a much improved upper bound on the generalization error:

$$\hat{P}r[\text{margin}_{(x,y)} \leq \theta] + \hat{O}\left(\sqrt{\frac{d}{m\theta^2}}\right) \tag{2.3.8}$$

The upper bound from Equation 2.3.8 is defined for any $\theta > 0$ with high probability. The reader should notice that this bound is completely independent of the number of boosting rounds T, which gives an indication as to why the upper bound from Equation 2.3.6 could not explain why the generalization error bound decreased (instead of increasing) as the number of boosting rounds increased.

However, the upper bound from Equation 2.3.8 is still rather conservative and in practice AdaBoost typically performs significantly better than this bound would suggest. In a further attempt to explain why AdaBoost is relatively immune against overfitting, Friedman *et al* [29]

described AdaBoost from a statistical point of view as an additive logistic regression model, that optimizes a criterion similar to binomial log-likelihood by means of an adaptive Newton method. Their work provides new insights into the workings of AdaBoost, but are quite involved and far beyond the scope of this research. However, the interested reader are referred to [29] for more details.

2.4 KALMAN FILTERING

2.4.1 History

The Kalman filter was developed by Rudolf E. Kalman and Richard S. Bucy in the 1960's and was originally used in the Apollo navigation computer for trajectory estimation. After its success in the Apollo program, the Kalman filter has been widely used in space and military applications but has since been found useful in a number of additional applications, in particular computer vision. The aim of this section is not to provide an extensive description of Kalman filtering, but rather a brief introduction (based upon [30]) to the concepts that are relevant to this research. For a more in-depth discussion of Kalman filtering, especially with regards to the mathematical deduction of the equations, the reader is referred to [31], [32] and [33].

2.4.2 Kalman Filtering for Object Tracking

Given some discrete-time controlled process with noisy measurements of the process, the aim of the Kalman filter is to estimate the actual state of the process. The Kalman filter estimates the process by making use of feedback control: the current state of the process is first predicted and then corrected by an actual measurement made (which is noisy). In cases where the process to be estimated is linear, it is governed by the linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \qquad x \in \mathbb{R}^n \tag{2.4.1}$$

with measurements

$$z_k = Hx_k + v_k, \qquad z \in \mathbb{R}^m \tag{2.4.2}$$

The process and measurement noise are represented by the random variables w_k and v_k, respectively. These variables are assumed to be white noise, independent of each other and with normal probability distributions:

$$p(w) \sim N(0, Q), \tag{2.4.3}$$

$$p(v) \sim N(0, R) \tag{2.4.4}$$

Matrix Q is the process noise covariance and matrix R is the measurement noise covariance both of which will in practice change with each time step. However, for this research they are assumed to be constant. The $n \times n$ matrix A in equation 2.4.1 relates the state of the previous time step $(k-1)$ to the state of the current time step (k), with no presence of a driving function or process noise. In practice, A might also change with each time step, but is also assumed to be constant in this case. The $n \times l$ matrix B relates the optional control input, $u \in \mathbb{R}^l$, to the state x. When considering the measurement equation from 2.4.2, the $m \times n$ matrix H relates the state (x_k) to the measurement (z_k). Once again, in practice H might change with each time step or measurement but this is also assumed to be constant in this case.

Due to the feedback control that is employed, the discrete Kalman filter equations are divided into two groups: time update equations and measurement update equations. The time update (or predictor) equations are responsible for forward projecting the current state and error covariance estimates, which is the *a priori* estimates that will be used in the next time step. The measurement update (or corrector) equations are responsible for the feedback, which will then be combined with the *a priori* estimate to obtain a more accurate *a posteriori* estimate of the process. The time update equations are:

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_{k-1} \tag{2.4.5}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2.4.6}$$

Firstly note that equation 2.4.5 is the *a priori* estimate (as indicated by the superscript minus) of the actual process state (equation 2.4.1). Next the *a priori* error covariance estimate is given by equation 2.4.6. Once the time update step has finished, the results from this step is

combined with an actual measurement in order to correct the predicted process state. The measurement update equations are:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{2.4.7}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.4.8}$$

$$P_k = (I - K_k H) P_k^- \tag{2.4.9}$$

The reader should firstly note the definition of a new matrix, K_k, in equation 2.4.7. K_k is a $n \times m$ matrix and is known as the Kalman gain or blending factor, which is chosen to minimize the *a posteriori* error covariance. As mentioned previously, the derivation of the Kalman gain is beyond the scope of this work. Secondly note that equation 2.4.8 is the *a posteriori* estimate (as indicated by the lack of any superscript) of the actual process state.

In essence equation 2.4.8 is the predicted state of the process (equation 2.4.5) corrected with the noisy measurement of the process (z_k). The purpose of the Kalman gain in equation 2.4.8 is to provide a weighting factor for the process that is "trusted" more. If the measurement process error covariance R approaches zero (i.e. the measurement is considered accurate), the Kalman gain will increase the weight of the residual in equation 2.4.8 and the measurement will be "trusted" more. If the estimation process error covariance P_k^- approaches zero (i.e. the forward predicted state is considered accurate), the Kalman gain will decrease the residual in equation 2.4.8 and the forward prediction will be "trusted" more.

Finally equation 2.4.9 is the updated or *a posteriori* estimate of the error covariance. The Kalman filter is typically first initialized with the estimates for $\hat{x}_{k-1}$ and P_{k-1} (reasonable initial estimates depend on the application). The process state and error covariance are then projected ahead in the time update step and are then fed to the measurement update step.

In the measurement update step the first task is to calculate the Kalman gain. The Kalman gain is then combined with an measurement of the state of the observed process, together with the projected state from the previous time update step in order to obtain an updated estimation of the actual process state. Next the error covariance matrix is then updated with the calculated Kalman gain and the previously projected error covariance. Finally the feedback loop is closed with the updated state estimation and the error covariance being fed back to the next time update step.

The performance of the Kalman filter can often be increased by tuning the measurement noise covariance (R) and the process noise covariance (Q) parameters, with tuning typically being performed off-line by means of another distinct Kalman filter. In general it is possible to measure R, since it should be possible to measure the process in any case. It is however often the case that Q cannot be easily determined, since it is typically not possible to directly observe the process that is being estimated (if this was possible, there would be no need for the Kalman filter in the first place). In cases where Q and R are in reality constant (as usually assumed), the estimation error covariance, P_k and the Kalman gain, K_k will quickly stabilize and then remain constant.

Up to this point the aim of the Kalman filter was to estimate the actual state of a process governed by the linear stochastic equation from 2.4.1. However, it is frequently the case that the process itself that must be estimated is non-linear or the measurement relationship to the process is non-linear. In such cases the normal discrete Kalman filter will fail to accurately estimate the process. Solutions to this problem is to either linearize about the current estimate, known as the extended Kalman filter (EKF), or to use the unscented transformation to propagate mean and covariance information through non-linear transformations, known as the unscented Kalman filter (UKF). The extended and unscented Kalman filters are beyond the scope of this research, but the inquisitive reader is referred to [30] and [34] for more information.

2.5 MEAN-SHIFT

2.5.1 History

The mean-shift algorithm is a non-parametric estimator of the density gradient of some density distribution of a data set, and is consequently very robust in finding the local extrema of such a density distribution. The algorithm was originally developed by Fukunaga and Hostetler [35] in 1975, but it was only in the late 1990's when its application to video object tracking was realized. As with the other sections in this chapter, the aim of this section is not to produce a rigorous mathematical deduction of the mean-shift procedure, but rather to give a higher level explanation of how the mean-shift procedure can be applied to object tracking. For a more indepth discussion on the mean-shift procedure, the reader is referred to [35] and [36].

In the previous paragraph the mean-shift procedure was described as a robust method for find local extrema. If the data distribution is continuous, this process is quite simple since it is essentially 'hill climbing' applied to the the histogram of the data. However, in the case of discrete data sets the problem becomes less trivial. The word robust is used in the formal statistical sense of the word, in that mean-shift ignores outliers in the data. This is accomplished by only processing points within a local window of data and then shifting the window.

The mean-shift procedure is related to the field of kernel density estimation, where some kernel function (a function that is mostly concerned with local processing) is used to express the distribution of data in terms those kernels. Mean-shift is different from kernel density estimation, in the sense that it is only concerned with the estimation of the gradient of the data distribution. Consequently, when the mean-shift vector is calculated it will always point in the direction of maximum increase in density. This behavior of mean-shift in estimating the direction of change, allows it to seek local maxima. By repeatedly executing the mean-shift procedure the gradient estimation will eventually converge to a value of 0, which indicates a peak (global or local) in the distribution. The prove of a sufficient condition for convergence can be found in die appendix of [36].

2.5.2 Mean-shift Tracking

The mean-shift procedure starts out by estimating the probability distribution by means of a kernel density estimator:

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i) \tag{2.5.1}$$

given n data points $\mathbf{x}_i$, some kernel $\mathbf{K}(\mathbf{x})$ and calculated in the point $\mathbf{x}$. For the purpose of mean-shift, only the special class of radially symmetric kernels have to be considered:

$$K(\mathbf{x}) = c_{k,d}k(\|\mathbf{x}\|^2) \tag{2.5.2}$$

where the function $k(x)$ is called the profile of the kernel and is only defined for $x \geq 0$. The normalization constant, $c_{k,d}$, results in $K(\mathbf{x})$ integrating to one. By parameterizing $K(\mathbf{x})$ with a bandwidth matrix proportional to the identity matrix, $\mathbf{H} = h^2\mathbf{I}$, only one bandwidth

parameter $h > 0$ is necessary. In the context of mean-shift tracking, the parameter h is used to control the window size. By incorporating the kernel from equation 2.5.2 with the bandwidth parameter h into equation 2.5.1, the kernel density estimator can be rewritten as:

$$\hat{f}(\mathbf{x}) = \frac{c_k}{nh} \sum_{i=1}^{n} k(\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \|^2) \tag{2.5.3}$$

Now given the underlying density $f(\mathbf{x})$, the mean-shift procedure attempts to find the modes (i.e. maxima) of the distribution in an elegant way, without actually estimating the density. The modes of the distribution are located at the zeros of the gradient (i.e. $\nabla f(\mathbf{x}) = 0$) and therefore a natural way to obtain the distribution modes is to compute the gradient of the density estimator from equation 2.5.3 due to its linearity:

$$\hat{\nabla} f(\mathbf{x}) \equiv \nabla \hat{f}(\mathbf{x}) = \frac{2c_k}{nh} \sum_{i=1}^{n} (\mathbf{x} - \mathbf{x}_i) k'(\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \|^2) \tag{2.5.4}$$

By assuming that the derivative of $k(x)$ exists for almost all $x \in [0, \infty)$, a new function can be defined:

$$g(x) = -k'(x), \tag{2.5.5}$$

which is then substituted into equation 2.5.4, which results in:

$$\begin{aligned} \hat{\nabla} f(\mathbf{x}) &= \frac{2c_k}{nh} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}) g(\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \|^2) & (2.5.6) \\ &= \frac{2c_k}{nh} \sum_{i=1}^{n} \left[g(\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \|^2) \right] \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g(\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \|^2)}{\sum_{i=1}^{n} g(\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \|^2)} - \mathbf{x} \right] & (2.5.7) \end{aligned}$$

The two terms that are bracketed in equation 2.5.7 both have an important meaning, with the first bracketed term being proportional to the density estimate at $\mathbf{x}$ computed with the kernel G and the second bracketed term being the actual mean-shift vector:

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i g(\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \|^2)}{\sum_{i=1}^{n} g(\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \|^2)} - \mathbf{x} \tag{2.5.8}$$

Equation 2.5.8 effectively calculates the difference between the weighted mean of the kernel (window) and $\mathbf{x}$, the center of the kernel. Therefore by repeatedly calculating the mean-shift

vector and then adjusting the window with this difference vector, implies that the window will move and eventually settle at a local maxima of the distribution. Adjusting h, controls how much of the distribution the window *'sees'*.

Two of the first successful applications of the mean-shift procedure were discontinuity preserving smoothing (similar to bilateral filtering) and image segmentation, both of which are presented in [36]). However, for this research the mean-shift procedure was found to be very effective for object tracking (i.e. the eyes). The mean-shift procedure adapted for tracking, as implemented by the OpenCV library [37], was used for this research and is given in Table 2.2.

Table 2.2: The mean-shift procedure for object tracking as implemented in OpenCV [37].

The mean-shift tracking algorithm

1. Initialize the search window:
   (a.) its starting position
   (b.) its type (i.e. uniform, polynomial, exponential or Gaussian)
   (c.) its shape (i.e. symmetric or skewed, possibly rotated, rounded or rectangular)
   (d.) its size (i.e. the extent to which it rolls off or is cut off)
2. Calculate the window's center of mass
3. Center the window at the calculated center of mass
4. Return to Step 2 until the window converges (i.e. a mean-shift vector of 0)

For the mean-shift tracking algorithm from Table 2.2 to function properly, the first step is to detect the object to be tracked and secondly to pre-process the image with the probability distribution of this object, in order to effectively separate the object from the image background. This pre-processing is typically the backprojection of the histogram of the object, applied to the original image. In effect the histogram of the object to be tracked serves as a lookup table for pixel brightness values by assigning the associated histogram bin value to the pixels of the output image, based on the associated pixel values in the original image. In terms of statistics, it implies that the pixel values of the output image characterize the probability of the corresponding pixels in the original image to belong to the object being tracked.

Once the image have been pre-processed, the initial search window is placed at the position of the detected object to be tracked in the pre-processed image. The mean-shift algorithm is

then applied to the pre-processed image, resulting in the search window to converge to the position of the object to be tracked from frame to frame. Since back projection is a very effective method for color segmentation in an image, the mean-shift algorithm is frequently used to track objects of a specific color from frame to frame. However, for this research only grayscale images were available but fortunately the objects to be tracked (i.e. the eyes) could be easily separated from the background by modeling the eyes as circles. More details on this approach can be found in *Chapter 4*.

2.6 CONCLUDING REMARKS

The aim of this chapter was to present a thorough introduction to the most important techniques used for this research in the development of a robust eye tracker. The first section introduced the bright/dark pupil effect, which is a physical characteristic of how the human eye responds to near-infrared light. This particular property of the human eye have been exploited to easily and robustly obtain the potential eye candidates for eye detection.

Given a number of potential eye candidates, these candidates have to be classified as either eyes or non-eyes and for this reason support vector machines (SVMs) and AdaBoost were presented in sections 2.2 and 2.3, respectively. Both of these classifiers have been proven to be highly accurate in similar image processing applications and were therefore natural choices. The classification results of SVMs and AdaBoost are shown in *Chapter 5* of this thesis.

Once the eyes have been detected, it should then be tracked from frame to frame and to this extend Kalman filtering and the mean-shift procedure have been identified as suitable techniques for eye-tracking and were therefore presented in sections 2.4 and 2.5, respectively. The chapters to follow will describe how these techniques can be combined to produce a robust eye-tracker that was able to track the eyes under various environmental conditions. The next chapter will present an overview of the existing eye detection and eye tracking techniques, as well as some of the current commercially available eye tracking systems.

Chapter 3

EYE TRACKING TECHNIQUES

This chapter will present an overview of current video-based eye detection and eye tracking techniques, with the purpose of putting this research into context. Video-based eye detection and tracking have been an active research field since the early 1980's and although major progress has been made since then, there still remains a number of challenging problems that have not yet been effectively solved. The current main challenges of eye tracking systems are the inter-subject variability of the human eye, occlusion, variability in scale as well as ambient lighting conditions. This chapter is divided into three main sections with the first section covering current eye detection techniques, while the second section covers current eye tracking techniques. The third and final section provides some concluding remarks.

3.1 EYE DETECTION

As mentioned before, the first phase of an eye tracking system is to first detect the eyes accurately. An effective eye detection approach should identify some model for the eyes that will able to handle large variability in appearance, while also being computationally effective in order to achieve real-time detection. Eye detection faces some very challenging issues, including occlusion of the eyes due to the eyelids, the degree of eye openness and, depending on the approach being followed, variability in either size, reflectivity or head pose. Although occlusions and shape variations are also common problems in other computer vision tasks, such as people tracking and face detection, it is seldomly so severe as in eye detection.

Hansen and Ji [2] did a comprehensive survey on different approaches followed for eye detection and gaze estimation, and much of the work presented in this chapter is based on their survey. According to their survey eye detection techniques can be divided into four main

categories, including shape-based approaches, feature-based approaches, appearance-based approaches and hybrid approaches. In turn, the techniques used in each of these categories will be discussed in the sections to follow.

3.1.1 Shape-Based Approaches

As the name suggests, shape-based approaches are concerned with modeling the shape of the eye, which is characterized by the iris and pupil contours as well as the exterior shape of the eye (i.e. the eyelids). The most simple shape-based approach is to model the iris or pupil as an ellipse and then attempt to detect ellipses in the image, which should in turn be identified as either an eye or a non-eye using some similarity measure.

Kim and Ramakrishna [38] detected the center of the iris of the eye by modeling the iris as an ellipse and then used the notion that the center of an ellipse lies at the center of the longest line inside the boundary of the ellipse. They achieved this by first thresholding the input image into a binary image according to typical pixel intensities of the iris. Canny edge detection was then performed on the binary image and longest line scanning was then performed on the resulting image to detect the center of the iris.

Young *et al* [39] followed a similar approach by modeling the eyes as an ellipse, due to the orthogonal projection of the imaging system. Their eye model had four fixed parameters, namely (X_c, Y_c) which is the center coordinates of the eye in the image, R_i which is the radius of the outer boundary of the iris and R_e which is the distance from the center of the eye to the plane of the iris boundary. Their eye model also had two varying parameters which specified the position of the iris center in the image. As with [38], Canny edge detection was then also performed, but the Hough transform was instead used to detect ellipses (i.e. the eyes) based on their eye model parameters.

Yuille *et al* [40] used a more complex eye model for eye detection, based on a deformable template, essentially consisting of two parabolas that represent the eyelids and a circle that represents the iris. Their eye model had a total of 11 parameters that were allowed to vary during template matching. This model was fitted to the image by means of an update rule and their experimental results have shown that the initial position of the template is critical to its success.

Although deformable templates are in general an accurate way of modeling the eyes, Hansen

and Ji [2] identified the following important limitations:

- They are computationally intensive.
- They typically require high contrast in images.
- Initialization typically has to be close to the eye to obtain successful localization, which implies that they usually struggle with large head movements.
- In situations where infrared light is used, the boundary between the scelera (the white of the eye) and the face may appear weak, which can cause problems for deformable contour models.
- In general they also struggle with significant face pose changes and eye occlusions.

Shape-based approaches are in general an accurate method for detecting the eyes in applications where the subject is relatively stationary, but they tend to struggle with large variability in the eye shape as a result of large head movements or eye occlusions.

3.1.2 Feature-Based Approaches

Feature-based approaches are concerned with the characteristics of the human eye and its surroundings, as opposed to modeling the eye. With these approaches the limbus (the border of the cornea and the sclera), pupil (based on bright/dark pupil images) and cornea reflections are frequently used for localizing the eyes.

Feng and Yuen [41] proposed variance projection functions to describe the change of variance in the vertical and horizontal directions, which they then used for image segmentation. They defined six eye landmarks, which were detected from the variance projection functions of a given eye image. In turn these landmarks were used to detect the positions of the iris, the upper eyelid and the lower eyelid. Although variance projection functions have been shown to be orientation and scale invariant, experimental results have indicated that this approach was ineffective when the eyes were closed or partially occluded by hair or facial orientation.

Kawato and Ohya [42], [43] followed an interesting alternative approach for eye detection, by first detecting the area between the eyes. The area between the eyes is characterized by dark parts to the left and right (as a result of the eyes and eyebrows) and comparably bright upper and lower parts (as a result of the forehead and the nose bridge, respectively). Kawato and

Ohya argued that this area between the eyes is viewable from a wider range of angles and is also more stable and easier to detect than the actual eyes. However, experimental results have shown that this approach was prone to failure when hair covered the forehead or when the subject wore black rimmed glasses.

The pupil is a popular feature used for eye detection (especially when viewed sufficiently close), since the pupil and iris are together much darker than the surrounding scelera. This high contrast between the pupil and its surroundings is typically exposed for eye detection by means of thresholding. Pupil detection is usually made more effective by using active infrared light to obtain the so-called bright/dark pupil effect as a result of the illumination being placed either on-center (bright pupil) or off-center (dark pupil) with regards to the camera's sensor. The eyes can then typically be detected from the difference image resulting from the subtraction of the dark pupil image from the bright pupil image. A number of authors have employed this approach, including [11], [8], [9], [10] and [44].

The bright/dark pupil effect has already been discussed in the previous chapter and shall not be repeated here. However, it is worth noting that this approach is mostly suitable for indoor usage and is particularly robust in situations where the ambient light is relatively low. In situations of high ambient light (e.g. outdoors during day time), the pupils will typically contract in order to limit the amount of light entering the eye, which implies that the bright pupil effect is not very apparent in such cases. Nevertheless, despite this limitation the bright/pupil effect remains a very robust method of detecting the eyes and was therefore used in this research.

3.1.3 Appearance-Based Approaches

The third main category of eye detection approaches is based upon eye appearance, which is also known as image template matching or holistic methods. Appearance-based approaches are concerned with the color distribution or filter responses of the actual eye and therefore detect the eyes directly, as opposed to either modeling the eye as a certain shape or relying on certain features of the eye (or its surroundings).

Appearance-based approaches are a general way of detecting almost any type of object in an image and is therefore not limited to detecting eyes. These methods can either operate in the spatial domain (i.e. template-based) or in some transformed domain (i.e. holistic). Template-based methods preserve the individual pixel intensity information, whereas holistic-based

methods are concerned with the pixel intensity distribution of the entire object appearance.

Detecting eyes (or any object for that matter) by means of template-based correlation maximization is simple and effective, but is also inherently more sensitive to scale and rotational changes. A major advantage of detecting eyes through a holistic approach in some transformed domain, is that the significant impact of variable illumination can be suppressed to a certain extend. However, this type of approach is typically only computationally viable in a low-dimensional transformed domain.

Grauman *et al* [45] used the motion of eye blinks to detect the eyes. Given a video stream of frames, Grauman *et al* subtracted the previous frame from the current frame and thresholded the resulting difference image. This produced a binary image containing white blobs, which indicate significant differences between the frames, in other words the motion. This binary image is passed through a number of filters to detect the eyes, subject to anthropomorphic constraints. Although this is a novel approach for detecting the eyes in the spatial domain, it is highly limited for general purpose eye detection, since the subject's head is assumed to be stationary with eye blinks being the only major form of motion.

Samaria and Young [46] were confronted with the problem of face identification, which they attempted to solve by using stochastic modeling in the form of Hidden Markov Models (HMMs) to holistically describe frontal facial information. By using a HMM, the authors have shown how the face image could be segmented from which various features (including the eyes) could then be extracted for identification. This approach only made coarse-scale eye location possible, and therefore further processing was necessary if the exact positions of the eyes were required.

Huang and Wechsler [47] used optimal wavelet packets for eye representation, which were used to first find candidate eye regions. These candidate eye regions were then classified as either eyes or non-eyes by using the Radial Basis Function (RBF). Optimal eye wavelet packets were derived from the filter response of the Daubechies family of order two, which Huang and Wechsler [47] found to improve the RBF classification performance when compared to using raw eye intensity images.

3.1.4 Hybrid Approaches

All of the eye detection approaches discussed thusfar are more suitable for specific types of applications, mainly due to their respective limitations. A natural approach therefore would be to combine different techniques to compensate for their individual weaknesses, to form a hybrid eye detection approach. A good example of a hybrid eye detection approach, combining appearance and shape, was presented by Ishikawa *et al* [48]. In their approach they first modeled the face using Active Appearance Models (AAMs), which allowed them to detect and track the entire face as a single object. From the detected face the locations of the eye corners were then estimated, which were in turn used to detect the iris using template matching. As with deformable models, AAMs also have to be initialized close the eyes to obtain an accurate fit, which also implies that AAMs struggle with large eye appearance variability.

Hansen *et al* [49] also used AAMs, but instead of modeling the face, they modeled the eyes directly. The shape and texture formed the main properties of their eye model and by using prior knowledge of the optimization space, their model is able to rapidly fit to an unseen example, if provided with reasonable initialization.

Arguably, one of the most effective eye tracking approaches that is currently available, is achieved by using the bright/dark pupil effect (i.e. feature-based) to extract eye candidates from the original image for classification (i.e. appearance-based). Zhu and Ji [19] illustrated such a system, in which they used active near-infrared illumination to obtain the bright/dark pupil effect. The dark pupil image was then subtracted from the bright pupil image to obtain a difference image, which was then thresholded to obtain a binary image. In the resulting binary image a number of white blobs were formed, which indicated potential eye candidates. The locations of these white blobs were mapped back to the original dark pupil image, from which the corresponding sub-images were extracted. The extracted sub-images were then classified as either eyes or non-eyes, using a Support Vector Machine (SVM) classifier. Zhu and Ji's work was based on previous work by done by [50] and [51].

Due to the robust nature of the eye tracking system presented by Zhu and Ji [19], much of the work presented in this thesis is based on their ideas. In terms of eye detection, their work was extended by considering different classifiers as well as using anthropometric constraints on detected eyes in order to verify correct eye classification, in an attempt to further reduce

the number of false positives.

3.2 EYE TRACKING

There is a lot of overlap between the eye detection techniques discussed in the previous section and the actual tracking of the eyes. This is due to some of the eye detection approaches being repeatedly executed for each frame, which implies that eye tracking is simply the re-detection of the eyes from frame to frame. This may seem like a logical approach, and indeed a number of authors have successfully demonstrated such systems, but in general the eye detection process is computationally intensive, which in turn limits its use for real-time eye tracking applications. This is especially true for applications that have to perform a higher level or processing over and above tracking to draw a conclusion from the eyes, for example determining in which direction the user is looking at (i.e. eye gaze) or the percentage of eye closure as a metric for monitoring fatigue.

For real-time eye tracking systems, the approach is therefore usually to first detect the eyes with some approach discussed in the previous section, and then initialize a simpler eye tracking approach with the detected eyes, which then tracks the eyes from frame to frame. By using this type of approach, the task of tracking is moved into the realm of general purpose tracking techniques, which can typically be used to track almost any type of object. As a consequence, the eye tracking approaches presented in this section cannot be as easily categorized as the eye detection approaches from the previous section and will be presented as a whole. However, the focus will be mainly of real-time eye tracking systems.

Talmi and Jiu [52] used the Principle Component Analysis (PCA) to detect the eyes directly (as opposed to first detecting the face), by transforming the luminance description of the eyes into a new coordinate system. The principle components of this new coordinate system were the calculated eigenvectors with the highest associated eigenvalues, which were referred to as the eigeneyes. The typical characteristics of eigenvectors that represent the eyes were learned beforehand and a similarity measure was used to detect the eyes during operation. Their system tracked the eyes from frame to frame by repeating this process, but severely limited the search region for the eyes once they have initially been detected. The goal of their system was to track the eye gaze of a user on an interactive screen to determine the fixation position, in order to produce a limited depth of focus (similar to human binocular focus) by only displaying the fixation areas on the screen in full resolution and gradually blurring

the surrounding areas. Mainly due to the computational cost of the eye gaze tracker, their system could only achieve near real-time performance.

Morris *et al* [53] presented a real-time eye tracking system that could detect blinks to enable people with motor difficulties to interact with a computer. In fact, their system used involuntarily eye blinks to detect the eyes. This was achieved by first detecting the face, in order to obtain a bounding box for detecting and tracking the eyes. Once the bounding box have been established, a variance map from this section of the first frame was initialized to zero and a mean image was created from the pixels in the bounding box from this first frame. Then for each following frame, both the variance map and the mean image were updated and a thresholded version of the variance map was calculated. A decision on blink detections were made by looking at the ratio of the number of thresholded pixels to the number of pixels in the bounding box. Once a blink have been detected the eye corners were located, which were in turned passed to the eye tracking phase as the feature points to track from frame to frame.

For eye tracking Morris *et al* [53] used the Lucas-Kanade feature tracker, which tracks feature points to sub-pixel accuracy by means of optical flow based on a pyramid representation of the image. The Lucas-Kanade method was first proposed in [54], and rests on three assumptions: brightness consistency, temporal persistence (i.e. small movements) and spatial coherence (i.e. neighboring points belong to the same surface and have similar motion). The Lucas-Kanade method is widely used in different tracking applications in computer vision, but is usually quite limited in handling large movements of the object being tracked.

Haro *et al* [55] exploited the bright/dark pupil effect for both eye detection and tracking. Adaptive thresholding was applied to the difference image formed by the subtraction of the bright and dark pupil images. From this binary image, pixels that were at least three connected were considered as eye candidate regions. All of the resulting candidate regions (eyes and non-eyes) were then tracked from frame to frame using Kalman filtering. A Probabilistic Principle Component Analysis (PPCA) (based on texture information) was performed for all of the candidate regions being tracked, in order to provide the probability of the particular candidate regions being eyes or non-eyes. Finally, the tracked candidate regions were actually classified as eyes or non-eyes, by means of a weighted probability function based on the probabilities obtained from the PPCA and the Kalman tracker's covariance matrix. Their system could track the eyes in real-time for indoor environments, under relatively smooth

head movements. Experimental results have shown that for fast and jerky head movements, their system struggled to accurately track the eyes. For a more in-depth discussion of the Kalman filter as applied to object tracking, the reader is referred to the *Chapter 2*.

Hansen and Pece [56] did not automatically detect the eyes, but relied on the users to initialize the eye tracking process by requesting the to position their eyes within a rectangle that was displayed on the screen. This would then be used to initialize an iris model, from which the state variables were estimated recursively in each following frame, in order to enable the tracking of the iris. Their model consisted of three components:

- A dynamic model defining the probability density function (pdf) over the iris in the current frame, based on the state from the previous frame.ity function (pdf) over the iris in the current frame, based on the state from the previous frame.
- A geometric model defining the pdf over the contours of the current frame, based on the iris state in the current frame.
- An observation model defining the pdf over the gray-level difference, based on the contours in the current frame.

Hansen and Pece [56] made some assumptions for the observation model, which meant that no explicit features had to be first detected and then matched to the model for tracking purposes. The actual tracking of the iris was then performed by the combination of particle filtering with the Expectation Maximization (EM) algorithm. Particle filtering was chosen based on its ability to maintain multiple hypotheses, enabling the system to robustly track the irises in the presence of noise and also to recover from occlusions (e.g. eye blinks).

Particle filtering was first introduced into the field of computer vision by Isard and Blake [57] with their proposed conditional density propagation (CONDENSATION) algorithm, which is a general purpose algorithm for tracking the contour of an object in a cluttered environment. It is interesting to note that although the CONDENSATION algorithm (or particle filtering in general) appears to be well suited for eye tracking applications, at the time of writing this paper and to the authors knowledge, it has been very seldomly used to this extend.

As mentioned in Section 3.1.4, Zhu and Ji [19] used a hybrid approach for eye detection by exploiting the bright/dark pupil effect to obtain eye candidates, which were then classified as

either eyes or non-eyes through a SVM classifier. Once their system has successfully detected the eyes, the locations of the detected eyes were used to initialize a Kalman tracker. From this point onwards, the Kalman tracker was used to track the bright pupils from frame to frame. Their experimental results have shown that the Kalman tracker works reasonably well under smooth head movements, as long as the bright pupils were present in the image. If the bright pupils disappeared or became weak due to eye occlusion (as a result of eye closure or large head rotations) the Kalman filter would fail.

Zhu and Ji compensated for this limitation of the Kalman tracker, by combining it with mean-shift tracking. An eye appearance model was created for their mean-shift tracker based upon the intensity distribution of eyes and non-eyes. As a result, the mean-shift tracker was therefore not dependent on the bright pupils. However, they noticed that the mean-shift tracker could in certain situations be easily distracted by objects close to the eyes that had a similar intensity distribution, which implied that the mean-shift tracker would eventually drift off and loose track since there was no feedback mechanism. For this reason the Kalman tracker was combined with the mean-shift tracker to compensate for their relative weaknesses. The Kalman filter was therefore used to track the eyes for as long as the bright pupils were present and when the bright pupils dissapeared, the mean-shift tracker would take over the tracking responsibilities until either the bright pupils again reappeared, or the mean-shift tracker drifted off. If the bright pupils reappeared, the Kalman tracker would again be responsible for eye tracking, otherwise the eye detection process would be re-initiated to restart the entire tracking phase.

The hybrid eye tracking approach followed by Zhu and Ji were able to robustly track the eyes under various illumination conditions and head poses, as well as in situations where the subject wore glasses. Their robust eye tracking results also inspired the tracking approach followed in this research, by combining Kalman filtering with mean-shift tracking. However, the way in which Kalman filtering and mean-shift tracking are combined in this research is slightly different as proposed by Zhu and Ji. The details of how these two approaches were combined are presented in the next chapter.

3.3 CONCLUDING REMARKS

The aim of this chapter was to present an overview of the existing work done on eye tracking and as a result placed the work done for this research into context. This chapter was divided

into two main parts, with the first part discussing the most influential approaches that have been followed for eye detection and the second part discussing some of the most successful approaches followed for tracking the eyes in real-time.

As can be seen from the literature presented in this chapter, the challenges of eye detection and eye tracking are rather complex and are yet to be completely solved. A vast number of approaches have already been suggested, with some working better than others, depending on the particular application. The conclusion that can be drawn from this chapter is that no single approach will be robust enough to detect and track the eyes under all of the multitude of scenarios, and therefore currently a hybrid approach seems like the most likely technique of compensating for the weaknesses of the individual approaches. Indeed, hybrid approaches were followed for both the eye detection and tracking phases in this research and will be presented in the next chapter.

Chapter 4

DESIGN AND IMPLEMENTATION

In this chapter the detailed design and implementation of the eye tracking system will be presented. The system required both hardware and software components that had to be developed and therefore the first section provides a functional overview of the complete system, followed by a section that covers all the aspects of the hardware design and implementation, with the third section covering the software development. The fourth and final section provides some concluding remarks on the complete design and implementation.

4.1 FUNCTIONAL OVERVIEW

In this section the system will be discussed at a high level to provide the reader with an overview of all the functional units of the implemented eye tracking system, as well as how these functional units interface with each other. The system consists of four main functional units, which are shown in the functional block diagram of Figure 4.1.

4.1.1 Functional Unit 1: Embedded System

Description

The first functional unit is the embedded system and its main purpose was to synchronize the NIR illumination with the camera. To achieve this functionality, the embedded system that was developed consisted of three separate sub-units:

- **Microcontroller (FU 1.1)**: The microcontroller which receives commands from the personal computer (FU 3), switches the NIR LED drivers and triggers the camera for image acquisition. The firmware developed for the microcontroller will be discussed

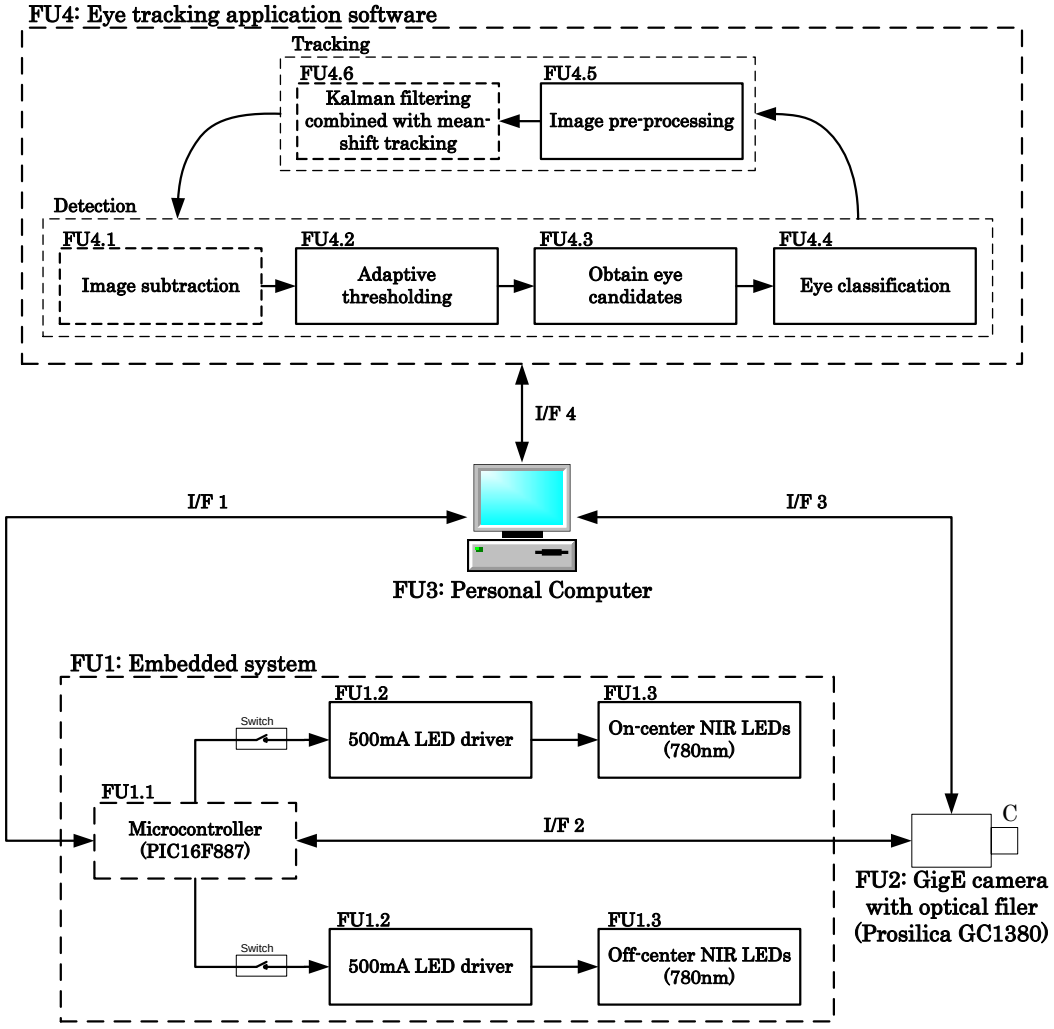

Figure 4.1: The complete functional block diagram of the designed eye tracking system.

later in this chapter. The PIC16F887 microcontroller from Microchip was used to this extend.

- **500mA LED drivers (FU 1.2)**: This sub-unit is in fact two identical 500mA constant current sources to drive the on-center and off-center NIR LEDs (FU 1.3) separately. The 500mA LED drivers were specifically designed and manufactured for the particular LEDs that were used.

- **NIR LEDs (FU 1.3)**: The NIR LEDs which provide the actual illumination to obtain the bright/dark pupil effect. The specific NIR LEDs emit light at a wavelength of 780nm and were obtained from Roithner Lasertechnik (H11A1-780-30).

4.1.2 Functional Unit 2: Camera

The second functional unit was a commercial off-the-shelf (COTS) camera that was used to capture images of a subject. The actual triggering of the camera was controlled by the embedded system (FU 1), while the transferring of images and other camera configurations (e.g. sensor exposure control, capturing mode, trigger settings, etc.) were controlled by the personal computer (FU 3). As previously mentioned, the camera that was used was the Prosilica GC1380 (monochrome).

4.1.3 Functional Unit 3: Personal Computer

The third functional unit was a normal personal computer (PC) that was used to control and ci the camera and to execute the image processing algorithms. The PC that was used had the following specifications:

- Intel Core 2 Quad CPU (Q8300 with a clock speed of 2.5 GHz)
- Gigabit Ethernet adapter
- 2GB of a RAM (later upgraded to 6GB)
- Operating system: Ubuntu 10.04 (Lucid Lynx)
- A USB-to-serial converter was used for RS-232 communication

4.1.4 Functional Unit 4: Eye Tracking Application Software

The final functional unit was the application software that had to be developed to perform the image processing necessary for tracking the eyes and were executed on the PC (FU 3). The application software was divided into two phases, namely detection and tracking, both with its own sub-units. As input the first phase received a bright and dark pupil image pair and as output produced the locations of the individual eyes in the dark pupil image. The eye detection phase consisted of four sub-units:

- **Image subtraction (FU 4.1)**: The corresponding pixel intensity values of the bright and dark pupil images were subtracted from each other to form a difference image.

- **Adaptive thresholding (FU 4.2)**: Given the difference image, an adaptive threshold was calculated and was then used to binarize the difference image, i.e. only black or white pixels. A fixed threshold will not be optimal, since the ambient lighting conditions can vary greatly in different scenarios.
- **Obtain eye candidates (FU 4.3)**: The resulting binary image will contain a number of binary blobs, which may or may not be eyes. A connected component analysis was performed on the binary image and the locations of the resulting blobs were mapped back to the dark pupil image and the corresponding sub-images at those locations were extracted, which served as the potential eye candidates.
- **Eye classification (FU 4.4)**: Given the extracted sub-images, a trained classifier was used to classify the sub-images as either eye or non-eye images.

The second eye tracking phase received as input the locations of the detected eyes and were used to initialize the tracker, which from thereon attempted to track the eyes from frame to frame, based upon the bright pupil effect. The eye tracking phase consisted of two sub-units:

- **Image pre-processing (FU 4.5)**: As discussed in the *Chapter 2*, the mean-shift tracker is non-parametric and therefore the bright pupil frames first had to be pre-processed to effectively segment the feature to be tracked (i.e. the bright pupils) from the background. In applications that use mean-shift tracking to track color objects, color histogram backprojection is typically used. However, in this case the images were grayscale and therefore this approach was not very effective. An alternative pre-processing approach was thus followed by first smoothing the image and then applying Canny edge detection, which effectively presented the bright pupils as circles. These circles, in effect the eyes, were then detected using the Hough transform. The details of this pre-processing step is presented later in this chapter.
- **Mean-shift and Kalman tracking (FU 4.6)**: The mean-shift procedure (discussed in Section 2.5) was then applied to the pre-processed image and will typically move to, and settle at the locations of the pupils. These locations then served as the measurements for Kalman filtering (discussed in Section 2.4). The combination of mean-shift tracking with Kalman tracking proved to be very effective. Finally metrics (to be discussed later) were applied in this sub-unit to detect when eye tracking was lost, in which case the eyes will have to be detected again.

4.1.5 Interfaces

The eye tracker system has four important interfaces to facilitate communication between the functional units:

- **RS-232 (I/F 1)**: A standard RS-232 serial interface between the PIC16F887 and the PC (FU 3), through which the embedded system received commands. The communication protocol can be found in Appendix A.
- **Triggering I/O's (I/F 2)**: This was the control signals necessary to trigger the camera (FU 2) for image acquisition. The timing diagrams, as specified by the camera's manufacturer, as well as the timing realization with the GPIO pins of the PIC16F887 can be found in Appendix B.
- **Gigabit Ethernet (I/F 3)**: Gigabit Ethernet (GigE) is an industry standard specified in IEEE 802.3-2008 and was used to transmit Ethernet frames between the camera (FU 2) and the PC (FU 3). Due to GigE's high data rate, it was ideally suited for streaming high quality images from the camera. In addition it also provided a high level of flexibility on the physical placement of the camera, since the typical Cat5 twisted pair cable length can be up 100 meters long.
- **Application programming interface (I/F 4)**: The application programming interface (API) was an abstract interface to the PC (FU 3) that provided the necessary functionality for the eye tracking application software (FU 4) to access the camera (FU 2), in order to communicate with the embedded system (FU 1) and to perform image processing on the resulting images. In reality there were actually a number of distinct APIs used that will be discussed later in this chapter.

4.2 HARDWARE DEVELOPMENT

As discussed in previous chapters, the eye tracker was based upon the bright/dark pupil effect. To achieve the bright/dark pupil effect, near-IR LEDs had to be synchronized with a camera. In terms of hardware, there were consequently two needs: firstly hardware to drive the LEDs and secondly hardware to interface with the camera so that the LEDs could be synchronized with the camera. This section will discuss how these two needs were addressed and the subsequent design and implementation of the associated hardware.

4.2.1 Hardware Selection

Camera Selection

In any real-world scenario there are a number of constraints that have to be considered when developing hardware, with financial considerations typically being the most profound constraint. For this research it was no different, with the camera being the most expensive piece of hardware. Although the ultimate goal of any commercial camera-based eye tracking system would be to use an as affordable (and perhaps customized) camera as possible, the aim of this research was rather to illustrate the proof-of-concept.

As a result there was a need for a general purpose, high resolution machine vision camera. In this case the need for a general purpose camera, implied a camera that provided a relative simple interface so that it could easily be synchronized with some external source. The specific need for a high resolution camera implied that the performance of the developed software will be increased since the images would contain more information. The camera also had to be monochrome, since color images would provide no significant additional benefits for this particular application, in fact color images would actually increase the complexity of the image processing. A camera with all of these characteristics would also result in a more accurate comparison between the different classification and tracking techniques that were considered in this research.

Given the lack of knowledge on which specific camera would be the most suitable for this application, experts in the field of computer vision had to be consulted to obtain recommendations. A few individual experts in the field were consulted and the consensus recommendation was the range of Prosilica Gigabit Ethernet (GigE) machine vision cameras from Allied Vision. Machine vision cameras are in general expensive equipment and an important consideration was to first have access to some of these cameras for testing purposes before actually purchasing one. Fortunately the Department of Defense, Peace, Safety and Security (DPSS) at the CSIR provided unrestricted access to their Prosilica cameras for this research.

Admittedly, the range of Prosilica cameras that were considered in this research were by no means the only cameras that would have been suitable for the application of eye tracking. In fact, cameras with sensors that were more sensitive in the NIR range would probably result in the use of more standardized and easily obtainable NIR LEDs (discussed later). However, the

major challenges were firstly just to get access to any machine vision camera and secondly to obtain a camera with a proper hardware interface for synchronizing the NIR LEDs in order to capture images. The latter proved to be an important consideration in choosing a suitable camera.

Since the illumination that had to be synchronized with the camera emitted light in the NIR range, the chosen camera's sensor also had to be sensitive to light in the NIR range. At the time of this research, DPSS had the following Prosilica GigE cameras available:

- **GC1380** - Monochrome, high sensitivity 1.4 megapixel (1360 × 1024) CCD sensor
- **GC1600** - Monochrome, ultra compact 2 megapixel (1620 × 1220) CCD sensor
- **GC1900** - Monochrome, high resolution 2 megapixel (1920 × 1080) CCD sensor

The next step was to determine what the sensor responses of these cameras were, and this information was found in the datasheets of each camera (see [58], [59] and [60] for the relevant figures of the sensor responses). Both the GC1380 and GC1600 had Sony CCD sensors, while the GE1900 had a Kodak CCD sensor. The International Commission on Illumination (CIE) defined the NIR band as light with a wavelength between 700nm and 1400nm and it was therefore decided that the camera which was the most sensitive in the 750 nm to 900 nm range would be chosen. This particular requirement was mainly due to the results obtained for similar eye tracking systems found in the literature.

When considering the figures of the three camera sensor responses from their respective datasheets, it is evident that the Sony ICX285AL CCD sensor from the GC1380 [58] was the most sensitive of the three cameras in the NIR range, with a quantum efficiency of close to 30% at 750nm and close to 10% at 900nm. Eventhough the GC1380 had the lowest resolution of the three cameras, a 1.4 megapixel resolution was still deemed more than sufficient for this application.

NIR Illumination Selection

Once the camera was selected, associated illumination in the wavelength range of 750nm to 900nm had to be obtained. The natural choice was LED illumination due to the wide variety of LEDs that were readily available, but as it turned out LEDs in the NIR range were not nearly as widely used when compared to LEDs in the visible light waveband. As a result the

NIR LEDs were not as easily obtainable as was expected. In addition high powered LEDs (which had to be within safety regulations) were also required, since the person's face had to be properly illuminated even in complete darkness. This implied that high-powered NIR LEDs were expensive and not available in a wide variety of wavelengths between 750nm to 900nm. Fortunately it was possible to obtain 850nm high-powered NIR LEDs from EBV Elektronik.

The specific type of LED that was sampled was the SFH4232 High Power Infrared Emitter from Osram. The SFH4232 had an emitting area of $1 \times 1\ mm^2$, a center of spectral emission at 850nm and could typically be driven at a forward current of 1A as specified in its datasheet [61]. The SFH4232 was specifically designed for infrared illumination for cameras in surveillance, driver assistance and machine vision systems and was therefore expected to be well suited for this application. Since these LEDs were high powered, only a few were required for the on-center and off-center illumination sets and therefore it was decided to use four of these LEDs per set. Also due to the high powered nature of these LEDs, a specific constant current driver circuit had to be developed. The design of a 1A constant current LED driver are presented later in this chapter.

As it turned out, preliminary test results have shown that the Prosilica GC1380 sensor was not sensitive enough to light at 850nm, as necessary to produce the bright pupil effect. Consequently NIR LEDs at lower wavelengths had to be considered, where the camera's sensor was more sensitive, in particularly wavelengths at 780nm and 810nm were identified as suitable candidates. At that point in time, LEDs at these wavelengths could not be found in South Africa and were sourced from an Austrian-based company called Roithner Lasertechnik. Both the 780nm (H11A1-780-30) and 810nm (H2A1-H810) LEDs were designed to operate at a constant current of 500mA and therefore the initial 1A LED driver designed for the 850nm LEDs was not be suitable for these LEDs. Consequently a second 500mA constant current LED driver had to be developed and its design is also presented later in this chapter.

Optical Filter Selection

The ambient lighting conditions under which the eye tracker operates will have a major influence on the image processing and since one of the initial goals of this research was to develop an eye tracker that could potentially be used in a vehicle for fatigue detection, the

system would typically have no control over the ambient light. To this extent it was decided to use an optical bandpass filter to suppress all light with wavelengths outside of the near-infrared band (750nm to 1400nm), in an attempt to at least have some control over which wavelengths of light would enter the camera's sensor.

Mainly due to financial constraints, but also due to the initial uncertainty of the ideal specifications, it was not possible to develop a customized optical bandpass filter for the eye tracking system. Fortunately, DPSS at the CSIR had an assortment of optical filters available that they graciously made available for this research. Since the selected type of NIR LEDs for the final design of the eye tracker have a center of spectral emission at a wavelength of 780nm, the goal was to find an optical bandpass filter that had good transmission characteristics for wavelengths in this region. From the available optical filters that met this requirement, the final two candidates are shown in Figure 4.2 and Figure 4.3, respectively.

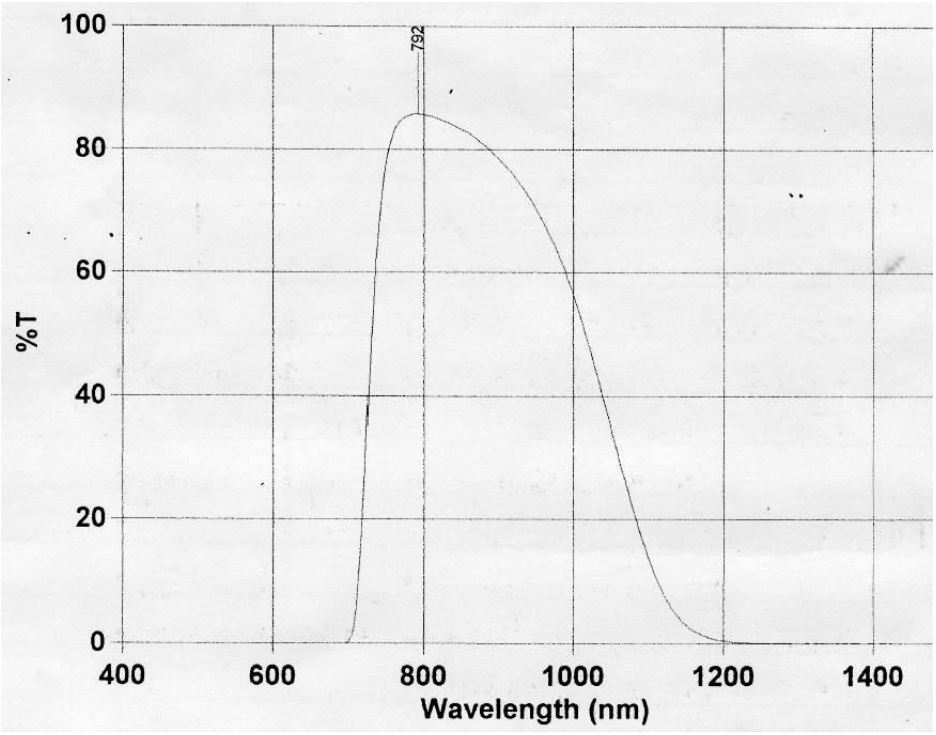

Figure 4.2: The filter response graph of the first optical bandpass filter candidate that was considered, but was ultimately not chosen in the final design. This optical filter was supplied by DPSS, a department at the CSIR.

Note that the low quality of these images was a result of the original electronic versions of the graphs not being available, and therefore the original printouts had to be scanned. The reader will notice that the filter response graphs of the two optical filters are very similar in

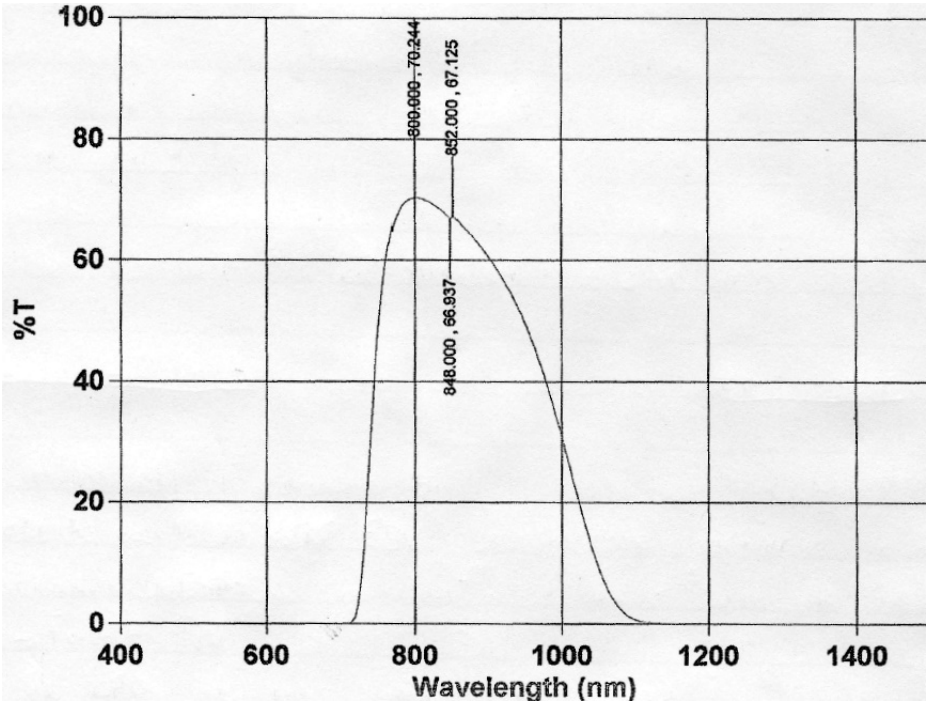

Figure 4.3: The filter response graph of the optical bandpass filter that was used in the final design of the eye tracker, as a result of some field trials. This optical filter was supplied by DPSS, a department at the CSIR.

shape, with the filter from Figure 4.2 having a larger passband and also less attenuation for incident light that falls within the passband, when compared to the filter from Figure 4.3.

The bright pupil effect was critical for both eye detection and the actual tracking of the eyes, and therefore the chosen optical filter had to maximize the bright pupil effect for various ambient lighting conditions. This could of course only be determined once the LED drivers and the embedded system were fully functioning, as necessary for synchronizing the NIR LEDs with the camera to ultimately obtain the bright pupil effect. After this phase was completed, a few trials under different daylight conditions were performed and the results suggested that the optical bandpass filter from Figure 4.3 was more effective in suppressing daylight and therefore enhancing the bright pupil effect.

These results were mainly due to the fact that the passband of the filter from Figure 4.3 started at a wavelength that was also closer to the 780nm wavelength of the NIR LEDs. In addition, the passband for the filter from Figure 4.3 was also narrower and the attenuation more. This was important since normal everyday light sources, in particular the sun, also emit light in the NIR spectrum of light and should therefore also be suppressed to a certain

extend.

The reader should note that the chosen optical bandpass filter from Figure 4.3 was not necessarily the ideal optical filter for this application, but was the best suited from the available optical filter. It is expected that an optical bandpass filter, with a narrower passband centered around 780nm would be more appropriate and might also result in requiring less NIR illumination. However, this would most likely also require a camera with a different sensor that would be more sensitive to NIR light in the region of these wavelengths.

Microcontroller Selection

Given the camera and the NIR LEDs (with their driver circuits), there was a need for a simple embedded system that would be able to synchronize the LEDs with the camera, based on commands it received from a personal computer (PC). It was decided that the simplest method to achieve this controlling functionality was to interface a microcontroller with a PC through a serial RS-232 port, which could at a later stage perhaps be upgraded to USB communication. Based upon the user manual of the Prosilica GC1380 [62], the capturing of images could be triggered through the general purpose I/O port. The LED driver circuit should therefore also be designed so that the NIR LEDs could be switched through the normal I/O pins of the microcontroller.

Consequently, the only important requirements for the microcontroller were a few I/O pins and an UART. Since the processing speed of almost any modern microcontroller would be sufficient to realize this controlling functionality, the main consideration was a microcontroller with a programmer that was readily available, which in this case was the PIC16F887 together with the PICkit 2 programmer from Microchip. Admittedly, the PIC16F887 was superfluous for this particular scenario, but could be tolerated since this was not a critical aspect of the complete system. More details on the camera synchronization are given later in this chapter.

4.2.2 Constant Current LED Driver Design

The current-to-voltage characteristics of LEDs are very similar to that of other diodes, and therefore the current through the LED is exponentially dependent on its voltage. The implication is that small changes in the voltage will result in large changes in current. Large variations in current can damage an LED, in particular the high powered LEDs used for this research, but can be solved by using a constant current driver. To this extend, the on-

line WEBENCH Designer from National Semiconductor [63] was used to design the constant current drivers required for this research.

1A Constant Current Driver Design

The SFH4232 by OSRAM is an infrared light source that emits light with a center of spectral emission at 850nm, which was specifically designed for the purpose of infrared illumination for cameras [61]. The SFH4232 LED was designed for a safe maximum DC current of 1A, and therefore it was decided to design 1A constant current driver for up to four SFH4232 LEDs in series. The design parameters, as were required by the WEBENCH Designer, can be found in Table 4.1.

Table 4.1: The design parameters (and design consideration) for the 1A constant current LED driver, as required by the WEBENCH Designer.

Parameter	**Value**	**Design Consideration**
Input voltage	9V - 16V	The typical output voltage of a car battery.
Ambient temperature	30°C	An overestimate of the average environmental operating ambient temperature.
LED operating current	1A	As specified in the datasheet.
Part number	Custom	WEBENCH does not have the specific LED.
$V_{Forward}$	1.5V @ 1A	As specified in the datasheet.
$R_{Dynamic}$	0.7Ω	$\frac{\Delta V_F}{\Delta I_F} = \frac{1.5-0.8}{1-10^{-2}} \approx 0.7\Omega$ (datasheet graphs)
LEDs in series	4	The maximum LED set size (can be less).
LEDs in parallel	1	Only LEDs in series.

Given the specified parameters, the WEBENCH Designer provided a number of designs with a trade off between the highest efficiency and the smallest footprint. However, the main design considerations were the availability of components and a footprint that did not require specialized equipment for component placement, and as a result the chosen design was based upon the LM3404 constant current buck regulator. The LM3404 regulator also provided LED dimming through pulse width modulation (PWM), but for this purpose it was only required to switch the LEDs on or off and to this extend the chosen design was slightly modified, with a 2N7000 MOSFET transistor that was placed between the RON pin and ground to enable on and off switching. The resulting schematic of the design is shown in Figure 4.4.

The 1A constant current LED driver from Figure 4.4 was duplicated (since there were two

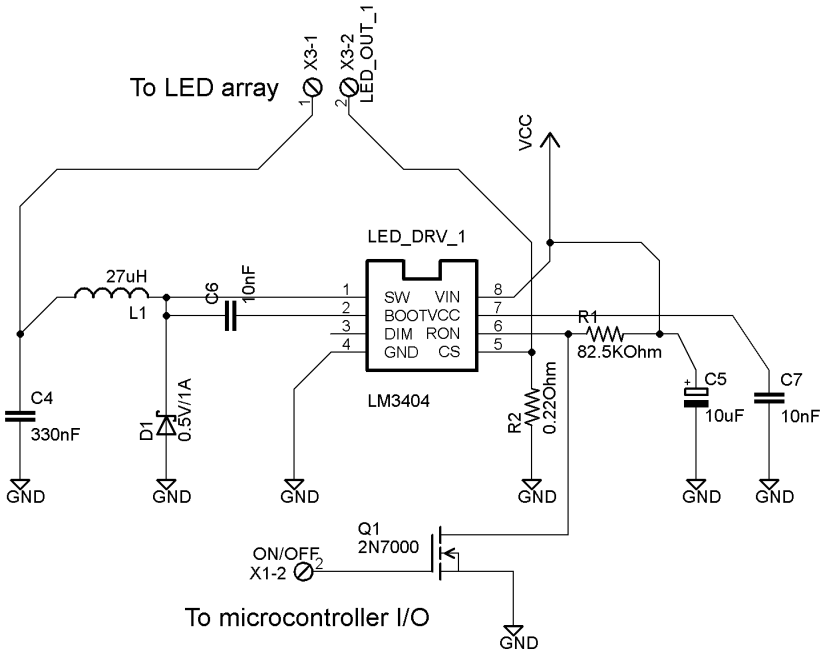

Figure 4.4: The 1A constant current driver that was designed to drive up to four 850nm high-powered LEDs (SFH4232) in series.

separate sets of NIR LEDs that had to be driven) and then integrated with a PIC16F887 microcontroller based embedded system that was able to control the LED drivers, trigger the camera for image acquisition and communicate with a PC through its UART interface. The design of the PIC based embedded system is not shown in this section, since the LED driver and the embedded system was separated in the final design, with the latter design being presented later in this chapter. The PCB design and layout was realized with EAGLE Light.

500mA Constant Current LED Driver

The 1A constant current LED driver from the previous section worked well in driving and switching the SFH4232 NIR LEDs, but when tested with the Prosilica GC1380 camera it was found that it was not really effective in producing the bright pupil effect. This was not a result of the SFH4232 LEDs not being bright enough in the NIR range (in fact they were probably brighter than required), but due to a mismatch between the emitting wavelength and the camera's sensor sensitivity, i.e. the Sony ICX285AL CCD sensor of the Prosilica GC1380 was simply not sensitive enough to NIR light at 850nm.

The simple solution was to use NIR illumination at a lower wavelength, but as mentioned

earlier, it turned out not be so easily obtainable in South Africa and had to be sourced from Austria. The chosen LEDs had to emit light just outside of the visible spectrum (i.e. above 750nm) so that it could not be perceived by the human eye, but that it could still be effectively detected by the Prosilica GC1380's sensor. At that point in time, there was uncertainty as to exactly which wavelength would be the most effective and as a result two types of LEDs at 780nm (H11A1-780-30) and 810nm (H2A1-H810) were chosen, both manufactured by Roithner Lasertechnik.

These LEDs were also chosen so that they had the same driving current and forward voltage drop requirements, so that only a single LED driver had to be developed that could be used for both the LED configurations. As with the development of the 1A LED driver from the previous section, the WEBENCH Designer was used to generate designs for the 500mA constant current LED driver and the design parameters are shown in Table 4.2.

Table 4.2: The design parameters (and design consideration) for the 500mA constant current LED driver, as required by the WEBENCH Designer.

Parameter	**Value**	**Design Consideration**
Input voltage	9V - 16V	The typical output voltage of a car battery.
Ambient temperature	40°C	An overestimate of the average environmental operating ambient temperature.
LED operating current	500mA	As specified in the datasheet.
Part number	Custom	WEBENCH does not have the specific LED.
$V_{Forward}$	1.6V @ 500mA	As specified in the datasheet.
$R_{Dynamic}$	0.8Ω	An estimate (datasheets did not specify).
LEDs in series	5	The maximum LED set size (can be less).
LEDs in parallel	1	Only LEDs in series.

As with the 1A LED driver from the previous section, the main requirements for the generated WEBENCH designs were also component availability and a footprint that did not require specialized equipment for component placement. The chosen design was based upon the LM3401 hysteretic PFET switching controller. There was just one modification made to this design: the Fairchild p-channel MOSFET (FDC602P) was difficult to obtain, and was just exchanged with a different p-channel MOSFET (FDC642P) with similar specifications. The LM3401 switching controller also provided a CMOS logic level PWM input pin for dimming,

which was just used to switch the LEDs on or off. The resulting schematic of the design is shown in Figure 4.5.

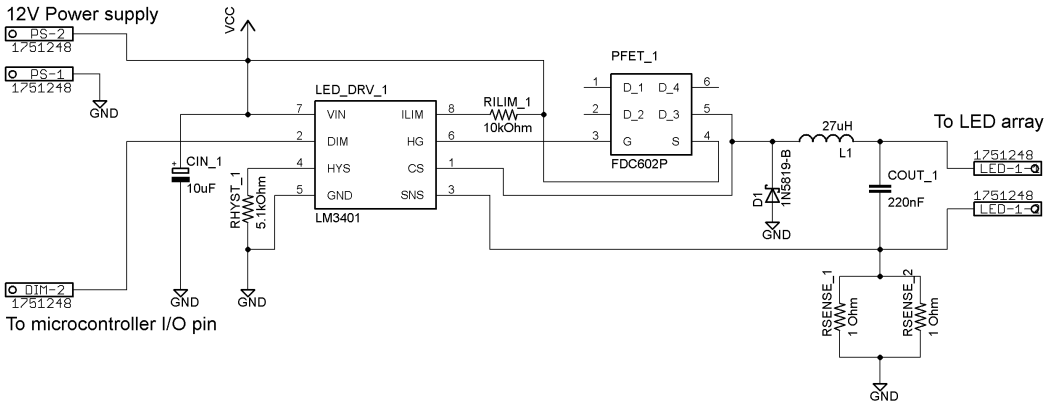

Figure 4.5: The 500mA constant current driver that was designed to drive up to five 780nm (H11A1-780-30) or 810nm (H2A1-H810) LEDs in series.

The 500mA LED driver as shown in Figure 4.5 was duplicated, as necessary to drive the inner and outer sets of NIR LEDs separately, and both drivers were placed together on a single PCB. The 500mA LED drivers were decoupled from the PIC based microcontroller embedded system, in order for both systems to be more general purpose, i.e. so that the same PIC based embedded system could control different types of LED drivers or that the same 500mA LED drivers could be controlled by any embedded system, provided that CMOS logic level signals were provided. The PCB layout of the 500mA LED driver was realized with EAGLE Light.

4.2.3 Embedded Controller Circuit Design

Given the LED drivers from the two previous sections, it was now possible to individually switch the two sets of NIR LEDs, and therefore it was now also possible to synchronize these sets with the camera to obtain the bright/dark pupil effect. Naturally, the next step was to develop an embedded system that could receive commands from the application software running on the PC, and switch the appropriate NIR LED set and finally to provide the control signals for external triggering of the camera.

As mentioned earlier, the most important design consideration for the embedded system was the availability of a microcontroller and its associated programmer, and to this extend the PIC16F887 microcontroller (40-pin PDIP package) and the PICKit 2 programmer were

selected. The only important requirements for the microcontroller were a UART module for serial communication, a number of general purpose I/O pins for the control signals and a timer module for the timing of the camera's triggering signals. Admittedly, the PIC16F887 has considerably more features than required, but since the aim of the current work was more research orientated than developing an optimized commercial product, it could be tolerated. In fact, the additional features could prove useful for future work, in particular the PWM module could perhaps be used to adapt the brightness (through dimming) of the NIR LEDs to the current ambient light conditions.

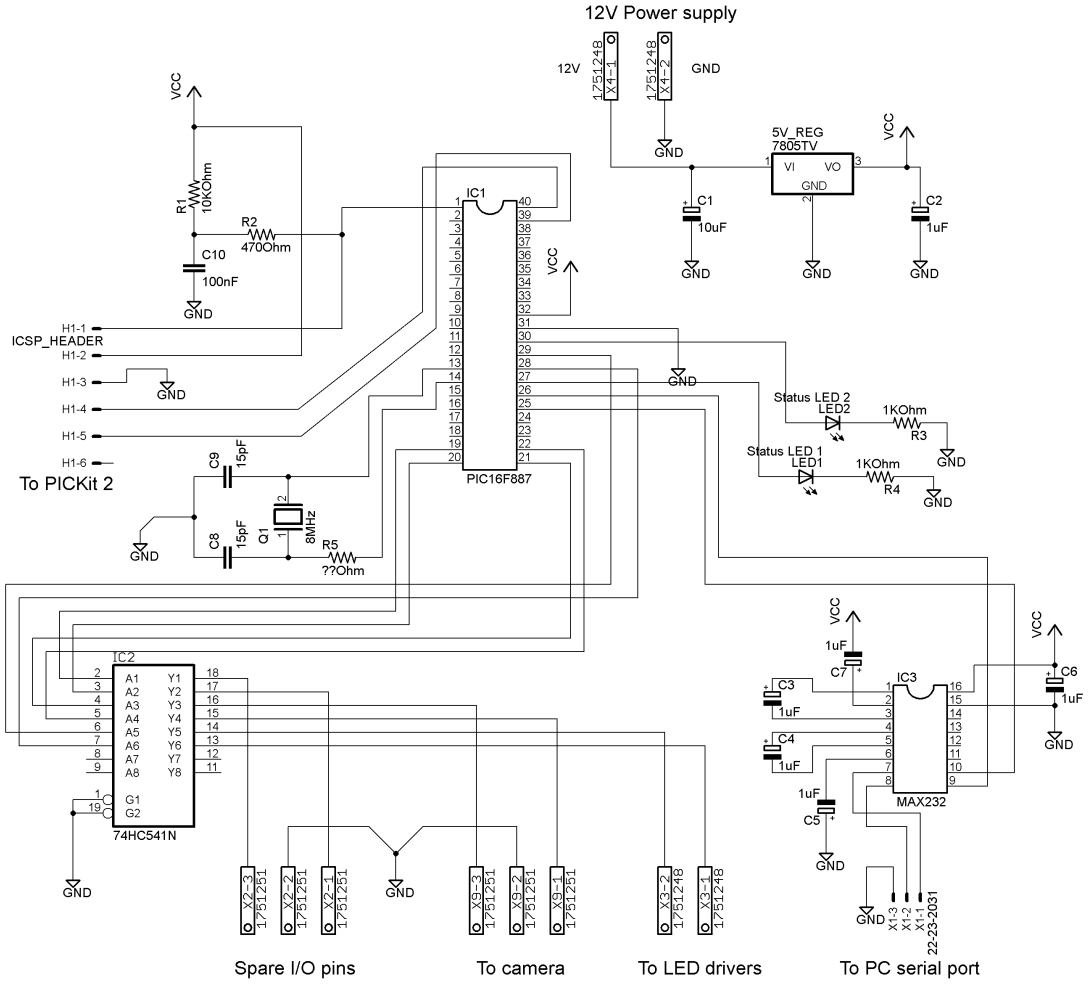

Figure 4.6: The embedded controller circuit (based on the PIC16F887) that received commands from the PC and triggered the camera accordingly.

The complete embedded controller circuit schematic is shown in Figure 4.6 and is used from the point onwards to explain the individual parts of the design. The first step of the design was to ensure that system received a regulated power input, since large variations in the

input voltage could potentially damage the PIC microcontroller. For this reason the 7805, 3-terminal 5V voltage regulator (in a TO-220 package) was used to allow an input DC voltage of a typical car battery (9V - 16V). The regulator circuit is shown in the top right corner of Figure 4.6.

The next important step was to ensure that the PIC16F887 could be programmed in-circuit, so that any modifications to the firmware could be easily made without first removing the PIC from the circuit and programming it separately. To enable serial in-circuit programming, a simple isolation circuit had to be inserted, as specified in the user guide of the PICKit 2 [64] and can seen in the top left corner of Figure 4.6.

To ensure that the timing of the triggering control signals to the camera were accurate enough, an external 8MHz quartz crystal oscillator was used and was connected to the PIC16F887 as specified in its datasheet [65] (see middle left of Figure 4.6). The resistor R5 was deliberately not given a value, but only placed there since a series resistor may be required for some crystals with a low drive level (see the PIC16F887 datasheet). The use of an external crystal with PIC microcontrollers is in general good practice and will improve the timing accuracy of other modules within the PIC, in particular the baud rate generator as required by the UART module for serial communication.

Once it was possible to program the PIC16F887 in-circuit, it was necessary to enable serial communication between a standard RS-232 serial port (from a PC in this case) and the PIC. Fortunately, the PIC16F887 was equipped with an UART module that enabled RS-232 control signals. However, the voltage levels of these UART signals were not according to the RS-232 standard ($\pm$10V to $\pm$12V) and it was necessary for a level shifting circuit, which was realized with the MAX232 IC that was specifically designed for this purpose. The MAX232 level shifter circuit is shown in the bottom right corner of Figure 4.6.

Finally, it was necessary to enable the actual transmission of the control signals necessary to switch the LEDs and trigger the camera. Both the LED driver circuit and the triggering input circuit of the camera would work with the TTL logic level signals as directly provided by the I/O ports of the PIC16F887. However, it was decided to buffer the I/O ports to protect the PIC microcontroller from any power surges on these lines. The chosen IC was the high speed, non-inverting 74HC541N octal buffer, and its connections to the PIC16F887 can be seen in the bottom left corner of Figure 4.6. Two status LEDs were also connected to the

an I/O port of the PIC16F887, which were just used to indicate activity on the embedded system (see middle right of Figure 4.6).

All of the ICs from the schematic shown Figure 4.6 (i.e. the PIC16F887, MAX232 and 74HC541N) were also connected through the appropriate DIP sockets, to enable effortless replacement in case of an IC being damaged. The PCB layout of the designed embedded controller circuit was realized with EAGLE Light.

4.3 SOFTWARE DEVELOPMENT

The software developed for the eye tracking system can be divided into two parts: the embedded firmware and the application software. The firmware was developed for the PIC16F887 microcontroller and was mainly concerned with triggering the camera and synchronizing the sets of NIR LEDs with the triggering signals of the camera. The application software was mainly concerned with obtaining the images resulting from the camera triggering and performing the image processing necessary to detect and track the eyes in these images. The two sections to follow will discuss the details of the developed firmware and application software, respectively.

4.3.1 Embedded Firmware

Table 4.3: The UART configuration of the PIC16F887 for serial communication.

UART parameter	Value
Baud rate	19,200 bps
Data bits	8 bits
Parity	None
Stop bits	1 bit
Flow control	None

The primary tasks of the embedded firmware, running on the PIC16F887, were to control the NIR illumination and the triggering of the camera based upon commands received on the serial interface from the PC. The first step was to enable the serial communication by configuring the UART module of the PIC16F887 with its setup shown in Table 4.3. A baud rate of 19,200 bits per second was deemed sufficient since the serial interface was only used for transmitting short commands from the PC to the PIC16887, and not for any data transfer.

It should also be noted that in order to achieve an accurate higher baud rate, a crystal with a higher oscillating frequency should be used. The reader is referred to the PIC16F887 datasheet [65] for more information.

All of the embedded firmware was developed using the C programming language, in particular using the HI-TECH C Compiler for PIC10-12-16 microcontrollers (version 9.80) together with Microchip's MPLAB IDE (version 8.63). The flowchart describing the embedded firmware is shown in Figure 4.7.

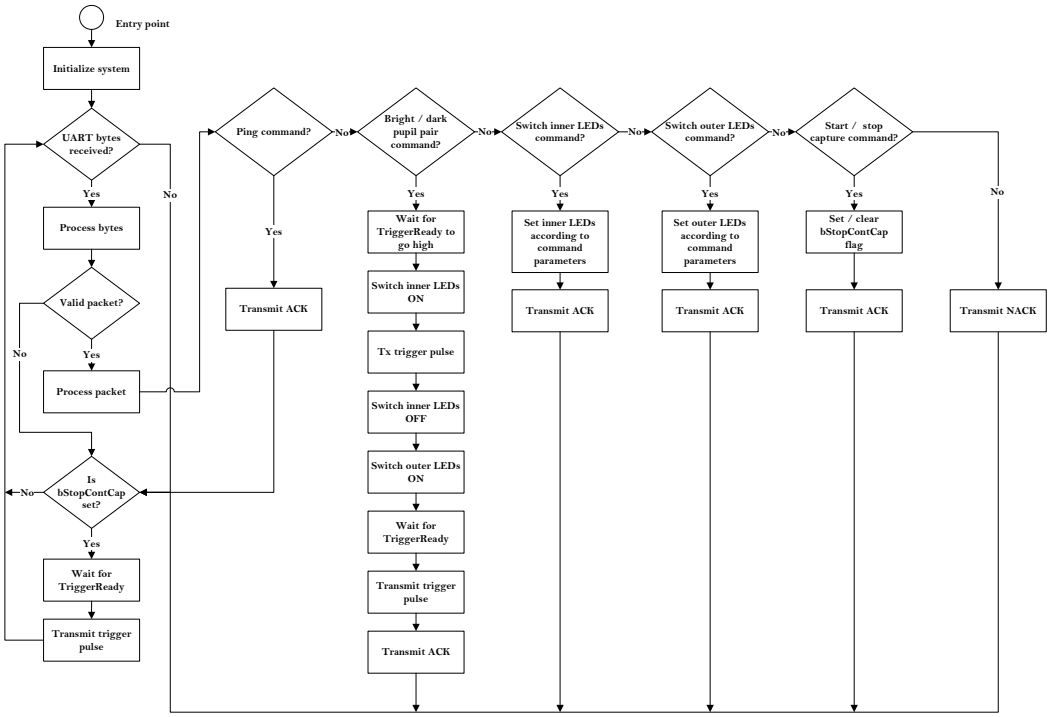

Figure 4.7: The flowchart of the firmware running on the PIC16F887.

The UART module was configured to generate an interrupt with every byte received, upon which the byte was packed into an implemented circular buffer during the interrupt service routine (ISR). Meanwhile in the foreground, the firmware would poll the circular buffer to see if any new bytes were written into it, and process the bytes as they were received. If valid packet bytes were received, the packet would then be parsed (without processing the packet contents) and would be written into a generic packet structure. Only once an entire packet has been received, it would be processed and the resulting command would then be executed. With the current design, the embedded firmware supports five commands as described in Table 4.4. For more details on the serial protocol and the triggering of the camera, the

reader is referred to Appendix A and Appendix B, respectively.

Table 4.4: The serial commands currently supported by the embedded firmware.

Command	Action	Description
Ping	Transmit ACK	A ping packet is transmitted to the embedded system and is simply acknowledged.
Capture bright/dark pupil image pair	- Set RD5, clear RD6 - Wait for RD0 to go high - Pulse RD1 for 15 μs - Clear RD5, Set RD6 - Wait for RD0 to go high - Pulse RD1 for 15 μs - Clear RD5 and RD6 - Transmit ACK	The inner set of NIR LEDs is first switched on and the camera is then triggered for the bright pupil image, followed by switching the outer set of NIR LEDs on, and again triggering the camera for the dark pupil image.
Switch inner set of NIR LEDs	- Set or clear RD5 - Transmit ACK	Switch the inner set of NIR LEDs on or off, based on the parameter.
Switch outer set of NIR LEDs	- Set / clear RD6 - Transmit ACK	Switch the outer set of NIR LEDs on or off, based on the parameter.
Start or stop video capture	- Set / clear bStopContCap - Transmit ACK	Start or stop the continuous capturing of images (i.e. video)

4.3.2 Classifier Selection

Before continuing onto the design of the application software, it is worth discussing the reasoning behind the selection of the particular classifiers that were used during the eye detection phase of the application software. The first steps of the eye detection phase were concerned with obtaining a number of eye candidates by means of the bright/dark pupil effect. Given a number of possible eye candidates, it was therefore necessary to use a classifier to discriminate between actual eye and non-eye images. The first requirement of the chosen classifier was that it had to be strong enough to be able to generalize well on unseen eye candidates, i.e. eye candidates that were not used during training. The second requirement was that there should be a readily available implementation for the chosen classifier, preferably in OpenCV,

since this research was more concerned with the ultimate goal of eye tracking.

At the time of performing this research the current version of OpenCV was 2.1, which implemented 9 different machine learning techniques. Therefore, if possible, the suitable classifiers had to be selected from OpenCV's implemented algorithms. During the course of the literature study (*Chapter 2*), the author came across various eye detection techniques, in particular the work performed by Zhu and Ji [19] on which much of this research was based. In their proposed system, they used various SVM classifiers to discriminate between eyes and non-eyes and obtained a very high accuracy rate of 95.5%. Based on the high accuracy rate that they obtained, the SVM classifier was a natural choice and was also already implemented by OpenCV.

Besides comparing the SVM classification results obtained in this research with the similar results obtained by Zhu and Ji, it was also decided to compare the results with other strong classifiers since there was an intuitive expectation that other strong classifiers should have comparable discriminating abilities for eye images. Face detection is a closely related problem to eye detection and was a topic that frequently occurred during the literature study. In particular, the author inevitably came across the highly referenced work of Viola and Jones [66], which used the AdaBoost classifier for feature selection in their face detector. Given that an average detection rate of above 90% was achieved with their face detector, it was expected that similar results could potentially be obtained for eye detection based on AdaBoost. For this reason and the fact that AdaBoost was already implemented in OpenCV, it was selected as the second classifier.

Finally, it was decided that there should also be some baseline for comparing the classification results of both SVM's and AdaBoost. The artificial neural network is arguably one of the most well known and widely used type of classifier, which forms a standard topic in any undergraduate course on artificial intelligence and based purely on its status and the fact that it was already implemented in OpenCV, made it the final selected classifier.

For this research the following classifiers, as implemented by OpenCV (version 2.1), were used:

- Support Vector Machines (SVM) (see Chapter 2)
- Adaptive Boosting (AdaBoost) (see Chapter 2)

- Multi-layer perceptron (MLP) feedforward artificial neural network

As already mentioned, SVM and AdaBoost were the classifiers of interest while the MLP feedforward artificial neural network classifier was merely used as a baseline for comparison. For each category of classifier there were different configurations available for selection (as provided by their OpenCV implementations):

- **SVM kernels:**
  - Polynomial
  - Radial basis functions (RBF)
  - Sigmoid
- **AdaBoost types:**
  - Discrete
  - Real
  - Gentle
  - Logit
- **MLP feedforward artificial neural network types:**
  - Sequential random backpropagation
  - Batch resilient backpropagation

The configuration selection of each classifier is discussed in *Chapter 5*.

4.3.3 Application Software

The application software was the final major component of the eye tracker that had to be developed, and executed all of the image processing algorithms that performed the eye detection and ultimately the tracking of the eyes. The application software was mainly dependent on the proper functioning of the embedded system and therefore the embedded system (hardware and firmware) first had to be completed before the development of the application software could commence.

The application software was developed with a graphical user interface (GUI), with the following main design goals in mind to enable a relatively unskilled person to use the software:

- The entire system should be easily configurable (especially the camera),
- Classifier training samples should be obtainable with the click of a button,
- The process of training the classifier should be simple and finally,
- Perform the actual tracking of the eyes.

Table 4.5: The additional SDKs and libraries that were used in the development of the application software.

SDK / Library	Version	Description
Qt SDK	4.7	An open source, cross-platform application framework for developing the GUI.
AVT PvAPI SDK	1.24	An extensive cross-platform application programmers interface (API) developed by Allied Vision Technologies as necessary for interfacing with their range of GigE cameras (the Prosilica GC1380 in this case).
OpenCV Library	2.1	An open source, cross-platform library used for real time computer vision applications.
cvBlobsLib	6.1	A library for OpenCV to perform connected component labeling in binary images.
QextSerialPort	1.0.0	A cross-platform serial port class written by Stefan Sander.

The application software was written in the C++ programming language under the Ubuntu 10.04 (Lucid Lynx) operating system and made use of a few additional software development kits (SDKs) and libraries, that are shown in Table 4.5. This section is divided into five sub-sections, which describes the five main components of the application software as necessary to track the eyes.

Extracting Eye Training Samples

As mentioned before, the first step of eye tracking is detecting the eyes and in order to detect the eyes there was a need for an accurate classifier that would be able to determine if a given

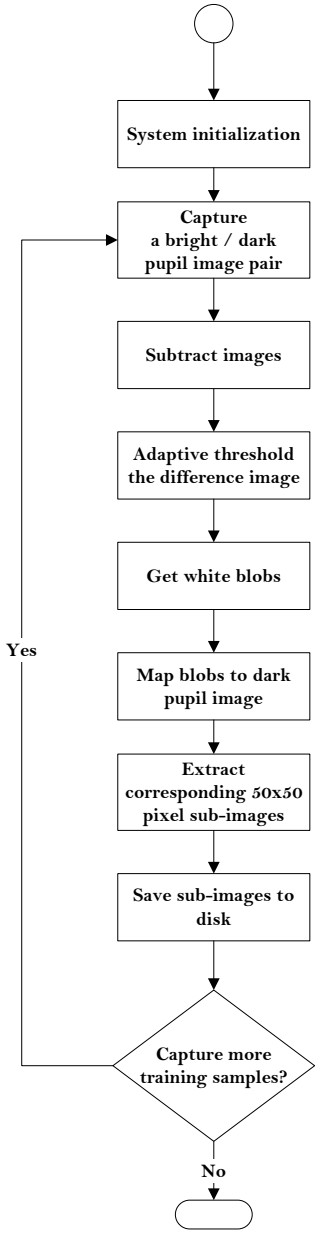

Figure 4.8: The application software flowchart for extracting training samples for eye classification.

sub-image is either an eye or a non-eye. To ensure that the classifier was accurate enough, it required a number of training images that were representative of the actual operating conditions. For this reason, the first major component of the application software was to extract the possible eye candidates as produced by the bright/dark pupil effect, which were then used to perform the supervised training of the classifiers. The flowchart for this process is shown in Figure 4.8, with the first step of the process being the proper initialization of the system, which included the following:

- **Camera initialization:**
  - Initialization of the PvAPI SDK.
  - Creation of a thread that continuously scans the Gigabit Ethernet interface of the PC for connected cameras.
  - Configuration of a connected camera to enable external triggering.
- **Serial port initialization:**
  - Configuration settings of the serial port (i.e. baud rate, data bits, stop bits, parity and flow control).
  - Detection of the available serial ports on the PC.
  - Creation of a thread that continuously monitors the serial port connected to the embedded system for received bytes, which also process valid packets as they are received.
- **Frame buffer initialization:**
  - Allocation of a circular frame buffer to store frames as they were received from the camera.

Once the system has been correctly configured, the user could initiate the capturing of a bright/dark pupil image pair with the click of button. This would result in the application software transmitting a command to the embedded system, instructing it to trigger the camera with the appropriate NIR illumination as necessary to obtain the bright/dark pupil effect (see Figure 4.7).

The resulting bright/dark pupil image pair would then be subtracted from each other to obtain a difference image. This was simply achieved by creating an empty image with the same dimensions as a bright (or dark) pupil image and populating the individual pixel intensities with the absolute difference between the corresponding pixel intensities in the bright and dark pupil images. Adaptive thresholding would then be applied to this difference image, which resulted in a binary image containing a black background with a number of white blobs.

The locations of the white blobs in the binary image were then detected using a connected component analysis and the location of each blob were mapped back to its corresponding location in the original dark pupil image. For each detected blob, a 65x65 pixel sub-image

was extracted from the dark pupil image, which may or may not be an actual eye. The number of sub-images extracted from the dark pupil image can vary greatly and is typically a function of how the objects within the captured images reflect the on-center and off-center NIR illumination. These extracted sub-images therefore served as the training examples for the classifiers and were finally saved to disk, where it had to be manually classified as either eyes or non-eyes. The user of the application software can capture as many training examples as deemed necessary, by simply repeating this entire process with the click of a button.

Eye Classifier Training

Since supervised learning classifiers were used, it was necessary to first manually classify the resulting extracted sub-images from the previous section. For each of the subjects that was used to extract training examples, the resulting sub-images were divided into two separate directories containing eyes and non-eyes. One of the research goals was to develop a classifier for eye detection that would be able to accurately classify the eyes of a wide variety of people from different ethnic backgrounds. To determine the feasibility of this goal, it was therefore necessary to train the classifiers with the data of a subset of the total amount of subjects and then test the classifiers with the data from the remaining unseen subjects.

Once the two final training directories (one for eyes and one for non-eyes) have been prepared, the application software could then be used for training the classifiers. The flowchart for training the classifiers are shown in Figure 4.9. As can be seen in Figure 4.9, the top half of the flowchart is dedicated to pre-processing the training data into a single array, which will be the input to the classifier (as required by OpenCV). From the flowchart the steps in obtaining the eye and non-eye training data are shown as two separate branches working in parallel, whereas in the actual implementation of the application software this occurred sequentially. The only reason for this representation was the lack of space, and functionally it makes no difference since these branches were essentially independent of each other and could easily have been implemented to be executed in parallel.

The first step in the flowchart is to determine how many training sub-images were available for each class in its associated directory, and this was obtained by using the Linux operating system API. Once the total number of images has been determined, each image would then be loaded from disk to obtain its feature vector. The feature vector for each training image was the pixel intensities normalized with maximum pixel intensity of each image, in order

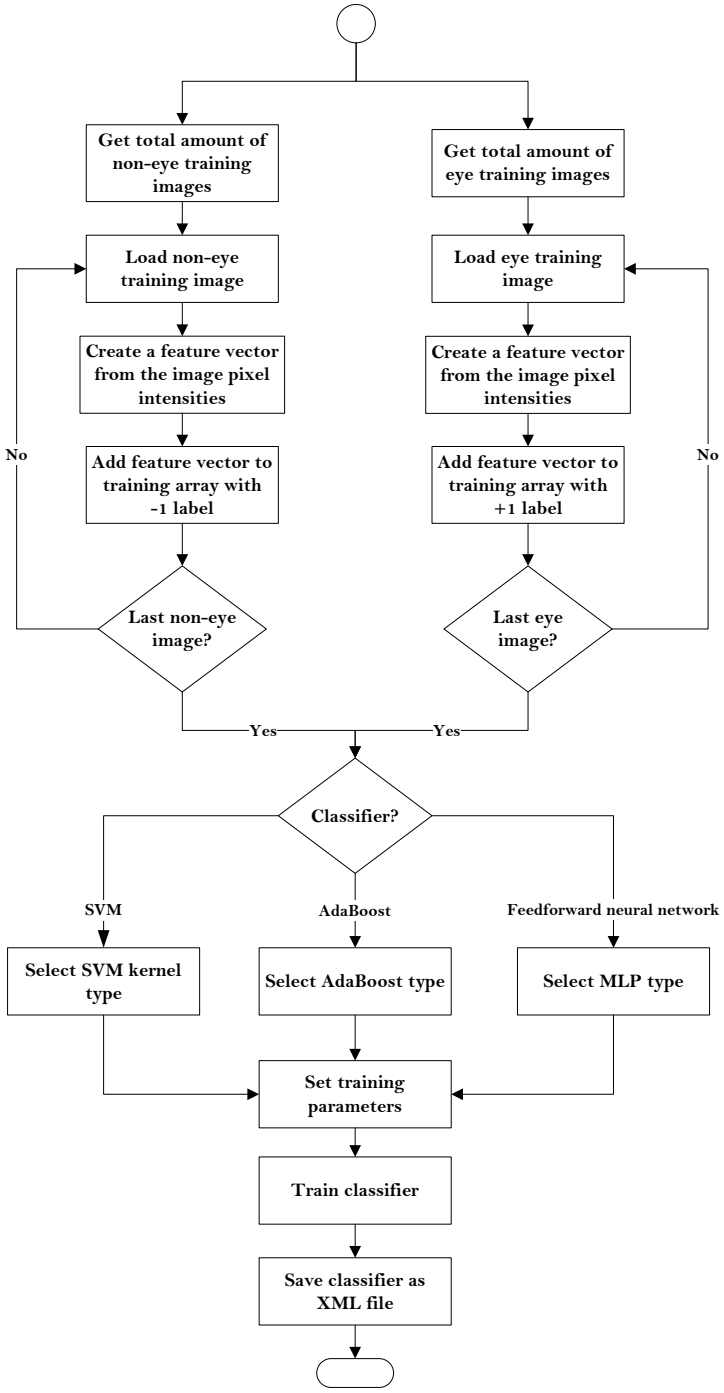

Figure 4.9: The application software flowchart for training an eye classifier.

to compensate (to some degree at least) for the effect of varying ambient light during the capturing of the images. In other words, the feature vector for each training image had 65x65 elements, with the value of each element being in the range [0,1]. Each feature vector would then be assigned a label, which told the classifier if the current training example belonged

to either the eye class (a label value of +1) or to the non-eye class (a label value of -1). As soon as the complete array has been created, the selected classifier would then be trained with this array. The values of +1 for an eye (positive) label and -1 for a non-eye (negative) label, was a requirement of the particular OpenCV implementation.

Given a selected configuration type, the classifier specific training parameters had to be setup before training could commence. The classifier specific training parameters that were used for each configuration can be found in Chapter 5. Finally, the training of the classifier was then started, which could take a considerable amount of time depending on the selected classifier and the amount of training data. Since a training time of even a couple of seconds would make the GUI non-responsive (i.e. it will not be possible to refresh the GUI while training), a separate thread had to be created that executed the actual training of the classifier, which then simply notified the main GUI thread upon completion of the training. If the training process finished successfully, the resulting classifier would then be written to disk in the XML file format, so that it could be loaded at a later stage for the class prediction of a given feature vector.

Eye Classifier Testing

For a given trained classifier, the next important step was to test its classification accuracy. As mentioned in the previous section, only a subset of the total subjects were used for training the classifiers, with the remaining subjects being used for testing the classifiers. As soon as the testing data has been placed in two separate directories, one for eyes and one for non-eyes, the application software would use this data to test a classifier as shown in the flowchart of Figure 4.10. As with the training flowchart, the classifier performance testing with the eye and non-eye images are shown in Figure 4.10 as two separate processes running in parallel, whereas in the actual implementation this occurs sequentially. Again, the only reason for this representation is the lack of space and functionally it makes not difference.

Firstly the trained classifier was loaded from a XML file stored on disk, followed by determining how many eye and non-eye testing images were located in their respective directories. Next, a testing image was loaded and its feature vector (i.e. the normalized pixel intensities) was extracted in exactly the same way as with the training process. The feature vector was then presented to the classifier, which predicted the class that it belonged to. Since the testing images were manually sorted as eyes and non-eyes, and were placed in separate

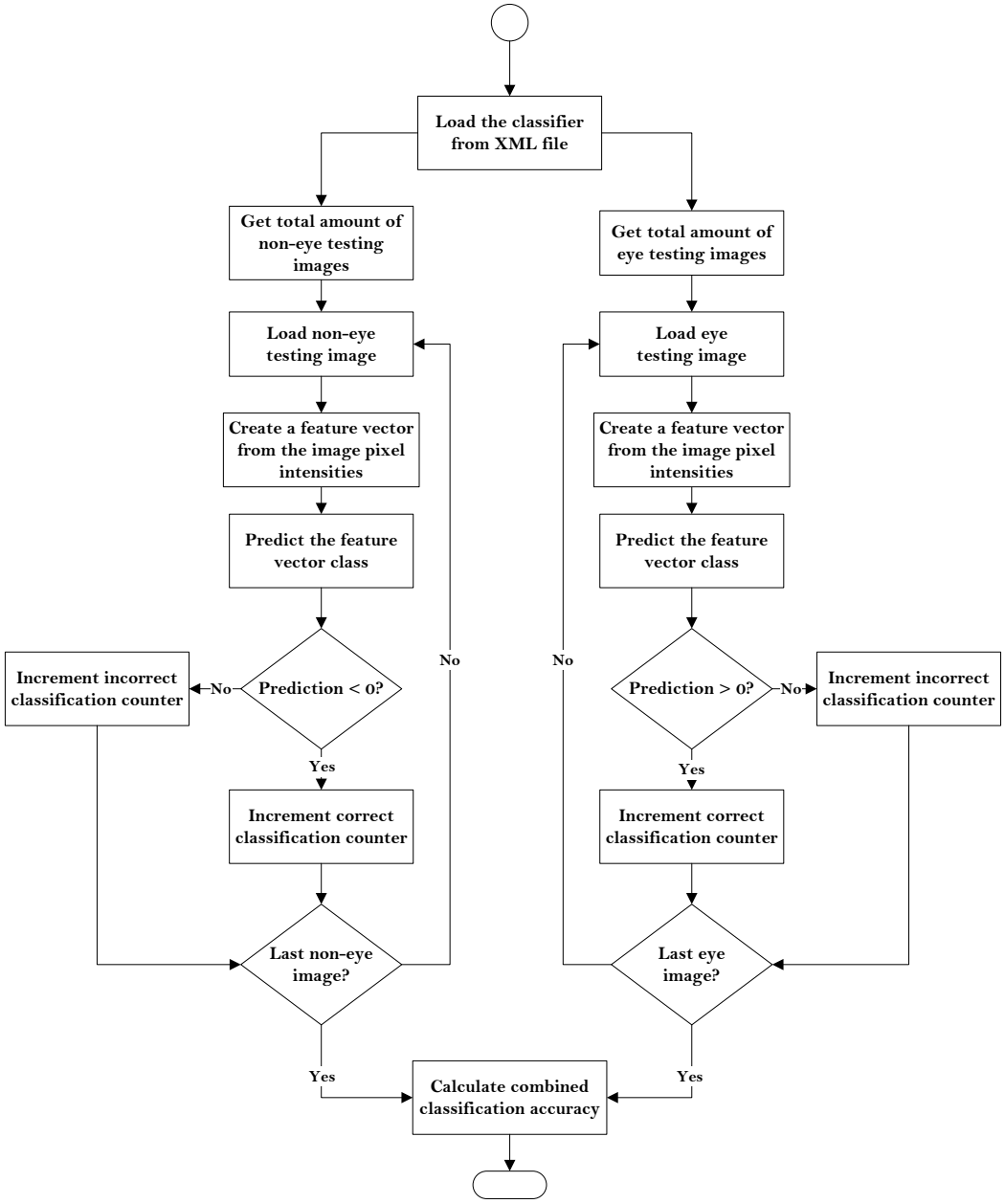

Figure 4.10: The application software flowchart for testing an eye classifier.

directories, the application software knew what the output of the classifier ought to be.

A classification attempt of a testing image was considered correct when the prediction outcome was larger than zero for an eye image, or smaller the zero for a non-eye image. In such cases, the respective correct classification counters were incremented by one, otherwise the incorrect classification counters were incremented by one. This was repeated for all of the eye and non-eye testing images and the final classifier accuracy was then calculated as

a percentage of the number of testing images that were correctly classified out of the total number of testing images.

Once tested, the accuracy of a classifier for a single subject could be improved by re-training the classifier with the testing examples that were incorrectly classified with the previous testing round. The accuracy of a classifier used for unseen subjects can perhaps also be further improved by simply using more training data from more individual subjects. However, increasing the size of the training set will only improve the accuracy of the classifier up to a point, in which case more training data will actually decrease the accuracy. The experimental results from Chapter 5 have shown that it is challenging to obtain a classifier that can generalize well for all types of people. This indicates just how complex the underlying eye model being learned is, which is mainly due to the large inter-subject variability.

Eye Detection

Once a classifier was considered accurate enough, which for this research was initially decided to be a correct classification rate of above 95%, it could be used for the real-time eye detection phase of eye tracking. A correct classification rate of above 95% was chosen as a benchmark, due to the similar results obtained by Zhu and Ji [19] for their SVM classifiers. The process for eye detection is shown in the flowchart of Figure 4.11 and the reader will notice that this flowchart is simply a combination of some of the functions already described in the three previous sections. The eye detection process, as described earlier in this section, was used to initialize the eye tracking phase and will only be shown as the *Eye detection* block in the flowchart of the complete eye tracking system (see Figure 4.12, in an attempt to provide a more easily understandable view of the complete system.

The first step of the real-time eye detection phase was to properly initialize the system, as described in Section 4.3.3, which was followed by the loading of the trained classifier from its XML file. The actual classifier that was used in the final system, was Discrete AdaBoost (the reasoning behind this choice can be found in Chapter 5.

Given that the system was correctly initialized, a bright/dark pupil image pair could then be captured as described in Section 4.3.3. The resulting bright and dark pupil images were then subtracted from each other to produce a difference image, which was then adaptively thresholded to obtain a binary (i.e. black and white) image, which contained white blobs. These white blobs, which represent the possible eye candidates, were detected by means of

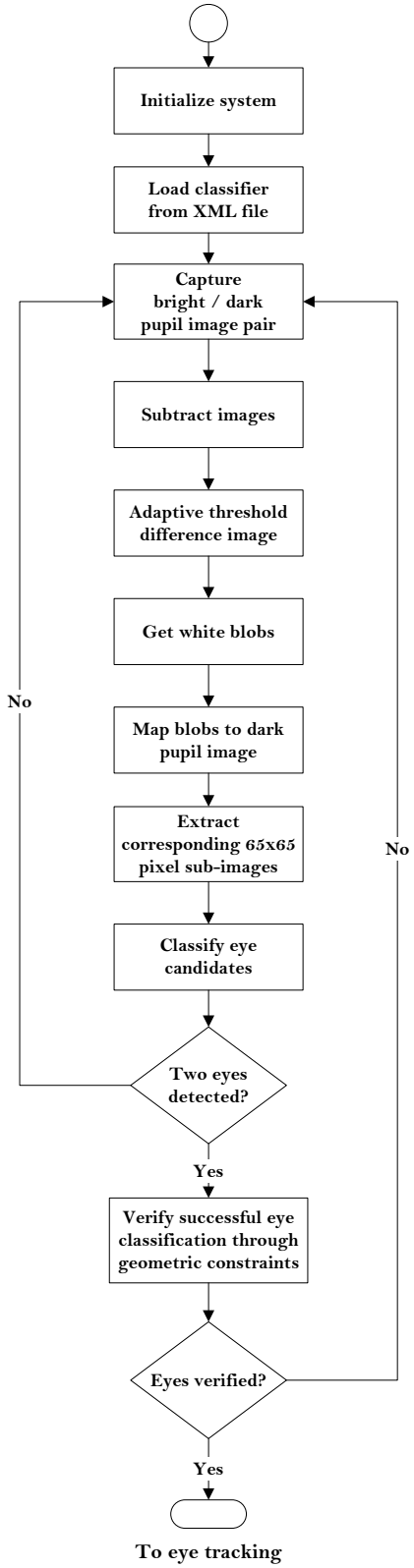

Figure 4.11: The application software flowchart for detecting eyes in real-time.

a connected component analysis and their locations were mapped back to the original dark pupil image from which the corresponding 65x65 pixel sub-images were extracted.

For each of the extracted sub-images its feature vector was obtained, which was then presented to the Discrete AdaBoost classifier to predict if the sub-image was either an eye or a non-eye. It was likely that multiple detected white blobs could be different parts of the same eye, so therefore if the classifier predicted that a given sub-image was an eye, it should first be checked if its location did not overlap with a previously detected eye. It was also observed that the Discrete AdaBoost classifier would occasionally produced false positives (i.e. non-eyes classified as eyes) and therefore the classified eyes were verified through the use of geometric constraints based upon the angle and distance relationship between the two detected eyes. The actual metric used for this verification process was determined experimentally and can be found in Chapter 5.

If the eyes were not successfully detected from all of the eye candidates for a given bright/dark pupil image pair, the detection process was repeated until both the eyes were detected. If both the eyes were successfully detected, the locations of the individual eyes were presented to the tracking phase, which were then used to track the eyes from frame to frame. As soon as the tracking phase realized that it had lost track of the eyes (which was inevitable) the eye detection phase, as described ins from frame to frame. As soon as the tracking phase realized that it had lost track of the eyes (which was inevitable) the eye detection phase, as described in Figure 4.11, was again executed to re-initialize the eye tracking phase.

Kalman Filtering Dynamics

Before continuing onto the design of the eye tracking application software, it is first necessary to describe the dynamics of the Kalman filter and the requirements for the OpenCV mean-shift tracking algorithm (described in the next sub-section), which were the key components in the eye tracking algorithm. The motion model used for the Kalman filter in this research was based upon the similar model used by Zhu and Ji [19]. Recall from Chapter 2 that the current Kalman process state can be represented with the following equation:

$$x_k = Ax_{k-1} + Bu_k + w_k \tag{4.3.1}$$

where x_k is the current state, x_{k-1} the previous state and A the transfer matrix relating the

previous state to the current state. The variable u_k is the control input, which is simply zero in this case since the system has no control over how the subject will move around. The variable w_k is known as the process noise, which was chosen to be fixed as 1e-5, which was based on some trial-and-error experimenting, since it was difficult to actually measure the process noise. To further improve the Kalman tracker, the process noise should be measured for a number of different scenarios and should then be set accordingly. The process noise might actually change over time, so it might even be necessary to update this at runtime. However, for the proof-of-concept the fixed value of 1e-5 was deemed accurate enough and the actual measurement of the process noise is left as future work.

Since any object tracking in an image is essentially two dimensional motion, the Kalman state can be represented as two position variables x and y, and two velocities v_x and v_y:

$$x_k = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \tag{4.3.2}$$

This representation of the state implies that the transfer matrix, A, should have the following form to relate previous state to the current state, according to time steps:

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3.3}$$

For this particular setup, dt was chosen as 200 after some trial-and-error experimenting. Also recall that the actual measurements made of the state is represented with the following equation:

$$z_k = Hx_k + v_k \tag{4.3.4}$$

where the matrix H relates the measurement to the state with some added measurement noise, v_k, which was chosen to be a constant value of 1e-1. Similar to the process noise, the constant value chosen for the measurement noise was determined on a trial-and-error basis.

In practice, the measurement noise is likely to change over time, but for this proof-of-concept the fixed value of 1e-1 was considered accurate enough. As future work, it might be useful to adapt the measurement noise to how well the Hough transform detects the circles associated with the eyes, since this will directly affect the measurement made by the mean-shift tracker.

By using a camera, it is typically only possible to directly measure the position of the object being tracked, and therefore z_k only contains position variables:

$$z_k = \begin{bmatrix} z_x \\ z_y \end{bmatrix}_k \tag{4.3.5}$$

The matrix H relates the measurement to the state and therefore has the form

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.3.6}$$

Given the Kalman dynamics, the *a priori* estimate can therefore be calculated as:

$$x_k^- = Ax_{k-1} + w_k \tag{4.3.7}$$

and after making the measurement z_k^-, the predication can be corrected with this measurement:

$$x_k = x_k^- + K_k(z_k^- - H_k x_k^-) \tag{4.3.8}$$

where K_k is the Kalman gain as discussed Chapter 2. The Kalman filter will therefore be used to predict the state of the eyes $[x, y, v_x, v_y]$ and the mean-shift tracking to measure only the positions, $[z_x, z_y]$, of the eyes (as will be discussed shortly).

Mean-shift Tracking Requirements

Also recall from Chapter 2 that the mean-shift procedure is a non-parametric estimator of the density gradient of some density distribution of a data set, which were pixel intensities in

this case. The mean-shift tracking algorithm as implemented by OpenCV was used for this research and therefore the implication of the previous sentence was that the input image to the mean-shift tracking algorithm had to be pre-processed to effectively segment the object to be tracked (the eyes in this case) from the background.

The OpenCV implementation only accepted grayscale input images and when initialized at some starting location, the search window would then be shifted with each iteration until it reached a maximum pixel intensity distribution. To simplify matters, it was decided that the input image should be pre-processed to produce a binary image with a black background (i.e. zero pixel intensity) containing white blobs (i.e. 255 pixel intensity) that represented the eyes to be tracked. The approach followed to pre-process the input image in this format, is described in the next section. As mentioned before, the mean-shift tracking algorithm was effectively used to take the measurements for the Kalman filtering.

Eye Tracking

Only once all of the three previous application software components had been successfully developed, could the actual eye tracking algorithm be implemented, which is shown in the flowchart of Figure 4.12. As expected, the first half of the eye tracking flowchart consists of a number of functional blocks that was described in the previous sections. The flowchart starts out by initializing the system as described in Section 4.3.3, followed by the loading of the Discrete AdaBoost classifier from a XML file. The eye detection process was then initiated by the capturing of a bright/dark pupil image pair, which was used to detect the eyes as described in Section 4.3.3. If both the eyes were successfully detected the eye tracker could be initialized with the detected eyes, otherwise the eye detection process was repeated until both the eyes were detected.

As soon as the eye tracker was initialized, the eyes could be tracked from frame to frame, and to achieve this the eye tracker required bright pupil images. It was decided to track the bright pupils since they produced the most distinct feature that could be robustly tracked. A continuous stream of bright pupil images was simply achieved by leaving the inner set of NIR LEDs on (and the outer set off) while capturing images. The developed eye tracker algorithm was a combination of a Kalman filter (used for eye location prediction and correction) with the mean-shift procedure (used to make *"measurements"* of the eye locations).

Since the eyes had to be tracked in real-time, it was important to spent as little time as

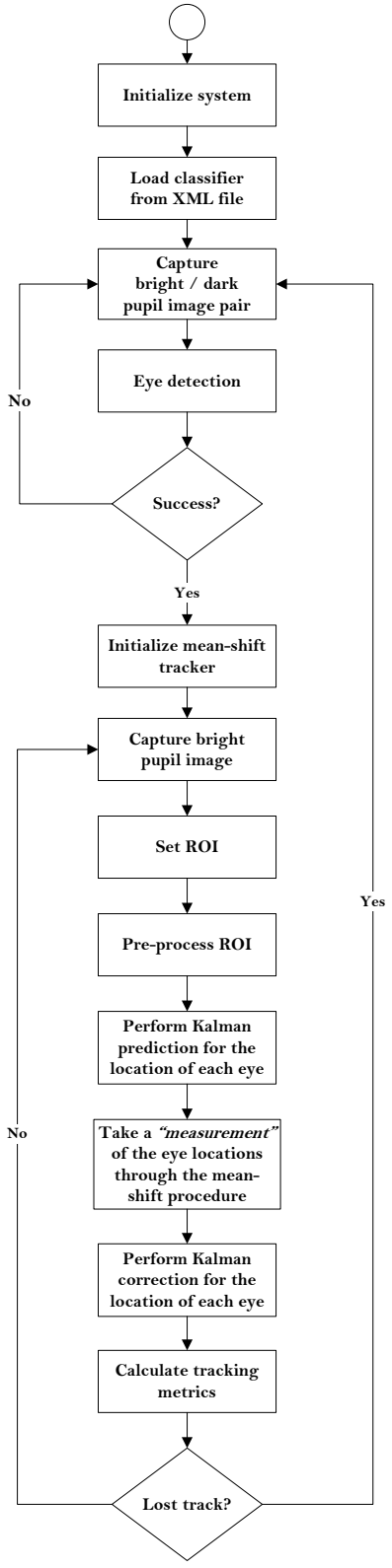

Figure 4.12: The application software flowchart for tracking the detected eyes.

possible on processing the image and to this extend it was decided to only process a certain region of interest (ROI). This was of course a reasonable approach, since the locations of the eyes from the previous frame could be used to set the ROI in the current frame, and if the ROI was large enough (without defeating the purpose) it was highly unlikely that the eyes would have moved out of the current ROI in just one frame.

Before the eye tracker could be applied, the ROI first had to be pre-processed for taking mean-shift *"measurements"*. In cases where the mean-shift procedure was used for tracking objects in a color image, the image would usually be pre-processed by backprojecting the hue histogram of the object (calculated during initialization) onto each received frame. This would effectively segment the object (based on its color) from the background and would result in the mean-shift procedure easily converging to the location of the object in each pre-processed frame. However, since only grayscale images were used in this research, experiments have confirmed that histogram backprojection was not very effective in segmenting the bright pupils from the background.

Since the bright pupil pixel intensities were significantly higher than the surrounding face pixel intensities, there were strong edges in the region of the bright pupils. These edges were therefore detected by first smoothing the ROI with a Gaussian kernel and then applying Canny edge detection as implemented in OpenCV. As the reader might already expect at this point, the Canny edge detection revealed the bright pupils as circles. Other regions of the eyes, eyebrows and the nose also exhibited strong edges and would typically interfere with the mean-shift procedure and therefore the eyes (modeled as circles) had to be segmented from the other detected edges.

The circles that were formed after Canny edge detection were not perfect circles, which had to be taken into account in order to detect the circles. For this reason it was decided to use the Hough transform for circle detection, which is a method for finding imperfect instances of objects within a certain class of shapes (circles in this case). OpenCV had an implementation of the Hough transform that was specifically developed for detecting circles, and was therefore used to this extent.

This specific Hough transform implementation would typically detect multiple circles from the edges (which were not always actual circles), but sporadic detections were filtered out to a large extent based upon the following criteria related to the previously discussed geometric

constraints:

- The minimum distance between any two detected circles had to be at least 100 pixels, since for this particular setup the distance between two actual eyes would never be below 100 pixels.
- The radius of the detected circles had to be between 5 and 25 pixels, since the circles formed as a result of the bright pupil effect would vary in size according to the ambient light, i.e. the pupils would contract as the ambient light becomes brighter. The size of the bright pupils would also vary in size as the subject moved closer or further away from the camera, but since the distance to the camera was quite restricted, this did not have a major influence. The chosen minimum and maximum pixel radiuses were specific to this particular setup.

After applying these constraints most of the sporadic circle detections would be filtered out, but there would usually still remain a number of detected circles that were not actual eyes. This was typically the case for other objects within the ROI that had a similar circular shape, e.g. certain types of earrings. Finally, the ROI was cleared and the remaining detected circles were redrawn as perfect flood-filled circles within the ROI. The typical result of this pre-processing step was a black ROI with solid white circles at the locations of the pupils and other randomly (but scarcely) placed solid white circles. The eye detection phase would produce the locations of the actual eyes and therefore the mean-shift tracker would initially *'lock'* onto the correct redrawn circles that correspond to actual eyes, and would then track the correct redrawn circles from frame to frame.

This pre-processed ROI was now ready for applying the mean-shift procedure to take *"measurements"* of the location of each eyes, but before this actually occurred two Kalman filters (one for each eye) were used to predict the locations of the eyes. When the eye tracker was initialized, the Kalman filters and the mean-shift trackers (also one for each eye) were initialized with the coordinates of the centers of the two detected eye windows, as obtained from the eye detection phase. Once the Kalman prediction has been made, the mean-shift trackers were used to *"measure"* the actual location of each eye in the ROI. Since the pre-processed ROI essentially contains a black background with white solid circles at the locations of the eyes, the mean-shift trackers easily converged to the center of each circle as long as the bright

pupils were present. The centers of the circles were then used as the *"measurements"*, which were in turn used by the Kalman filters to correct the initial location predictions.

By combining Kalman filtering with mean-shift tracking, the *"measurements"* for the eye locations were usually very accurate as long as the bright pupils were present (due to mean-shift tracking), but when the bright pupils dissapear momentarily the eye locations could still be accurately predicted (due to Kalman filtering) as long as the subject did not make erratic head movements. This proved to be a very robust approach for tracking the eyes, as will be shown in Chapter 5.

For the human observer it is quite simple to determine when eye tracking has been lost, but for the system itself it was of course less obvious. Therefore it was necessary to have some mechanism that would be capable of detecting when the system has lost track of the eyes (which was inevitable). One possible approach that was initially used, was to use the trained classifier, as used for eye detection, to verify the predicted eye location in each frame. However, this proved to be cumbersome since by design the variations allowed in eye appearance are significantly larger in the tracking phase when compared to the detection phase, which implied that the classifier struggled much more to correctly classify the eyes (i.e. a lot of false negatives occurred) resulting in unnecessary re-detection of the eyes. In addition, the inherent false negative rate of the trained classifier (even for simple frontal face images) would also too easily disrupt the tracking process.

For these reasons it was necessary to use some other metric and it was decided to apply the same geometric constraints from frame to frame, as proposed for the verification of the detected eyes as described in the previous section. However, for the tracking phase it was again observed that there could be much more variation in the distance between the eyes when compared to the eye detection phase, and therefore the distance thresholds were slightly increased. By using these geometric constraints meant that the system could quite accurately detect when eye tracking has been lost, so that the re-detection of the eyes could again be initiated.

4.4 CONCLUDING REMARKS

This chapter presented the reader with the complete design of the developed eye tracker, covering all of the hardware and software aspects of the system. The aim of this chapter was to enable a technically competent person to reproduce the entire system based only on the

information contained in this chapter and the associated appendices. The chapter started out by providing the reader with a functional overview of the entire system and how the various components of the system interfaced with each other.

This was followed by an in depth discussion of the hardware design of the system, which firstly described the rationale behind the selection of the various hardware components and secondly the designs of the hardware components (that were not COTS) were presented, in particular the designs of the constant current LED drivers and the PIC16F887-based embedded system. Where applicable, further details of some of the hardware aspects were placed in appendices.

Finally, the design of the various software components were presented, which were divided into the embedded firmware running on the PIC16F887 and the application software running on the PC. The designs of the various software components were only explained in flowcharts, since it was argued that presenting the reader with a number of code snippets or pseudo code would add unnecessary detail, which would only result in complicating the explanation of the core concepts of the software. In addition, the documentation for the various SDKs and libraries that were used in the software were extensive and sufficient in most cases.

The final configuration of the eye tracking system is shown in Figure 4.13 and Figure 4.14. Admittedly, there is room for improving and optimizing the design of the final system, especially the embedded system and the mechanical configuration, but this was of course not the goal of this research and is left as future work.

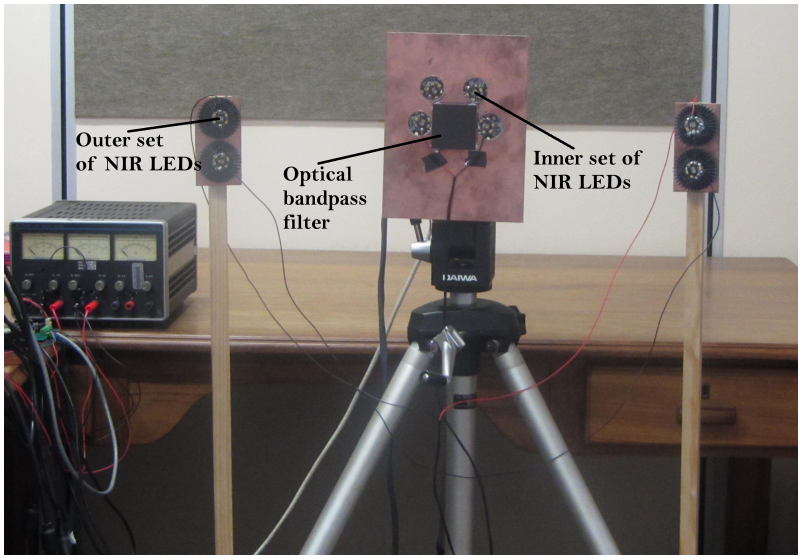

Figure 4.13: A frontal view of the final eye tracking system.

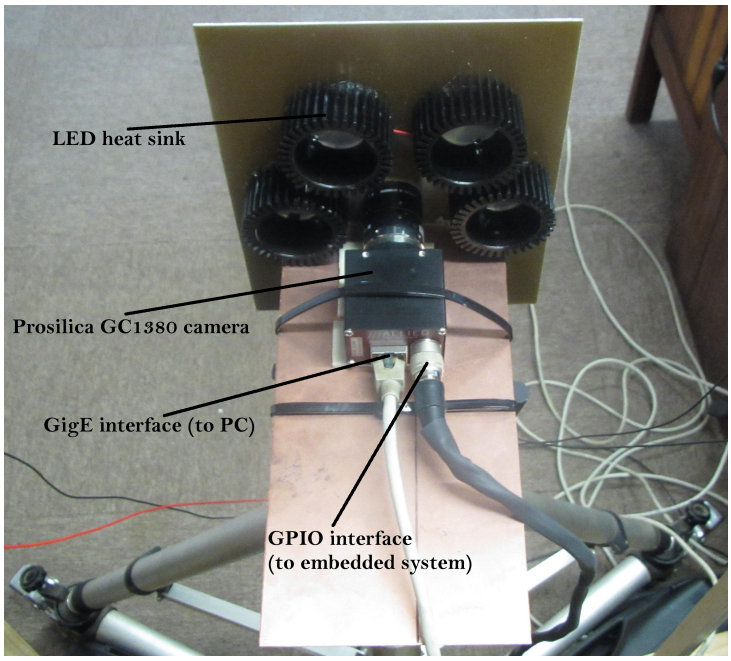

Figure 4.14: A top view of the final eye tracking system.

Chapter 5

RESULTS AND APPLICATIONS

In this chapter the results from the various aspects of the eye tracking system will be presented. The results are divided into five sections, with the first section presenting the results from using the bright/dark pupil effect. The second section presents the results from the three different classifiers used to detect the eyes, given the potential eye candidates as produced by the exploitation of the bright/dark pupil effect. The third section presents the results from the complete eye tracking system, which was heavily dependent on the results from the previous two mentioned sections. The fourth section provides some typical applications for the eye tracking system, while the final section provides the concluding remarks on the results obtained in this chapter.

5.1 THE BRIGHT PUPIL EFFECT

Although the bright pupil effect is quite well documented in the literature, the various considerations surrounding the hardware setup, in particular the camera and the NIR illumination, are seldomly presented in much detail. Even in the cases where the actual hardware used were given, it was not possible or desirable to reproduce the same configuration.

In almost all of the literature found on eye tracking based on the bright/dark pupil effect, in particular [50], [51] and [19], an analog camera was used with interlaced scanning where the NIR illumination was synchronized with the even and odd fields to produce the bright and dark pupil effect, respectively. Although these systems worked well with this approach, the implication was that an analog camera had to be used with a separate frame grabber to obtain the images from the camera. However, in recent years with the advent of high performance digital cameras with direct camera interfaces (e.g. USB, Firewire and GigE),

the use of analog cameras with frame grabbers have significantly decreased.

In addition, from the literature it appeared as though the various authors had very little control over the camera and also did not attempt to control the input to the camera, for example, by using optical filters or dynamically adapting the camera's sensor exposure to the ambient lighting conditions. In this light it was decided not to use a camera based on technology that was considered by many to be outdated and resulted in the selection of a GigE Vision camera, which was combined with an optical bandpass filter. The implication of this selection of hardware was that there was essentially no information available in the public domain that described the ideal configuration for effectively obtaining the bright pupil effect and therefore this had to be determined on a trial-and-error basis. The focus was on determining the ideal configuration for obtaining the bright pupil effect, since this was a critical requirement for detecting and tracking the eyes.

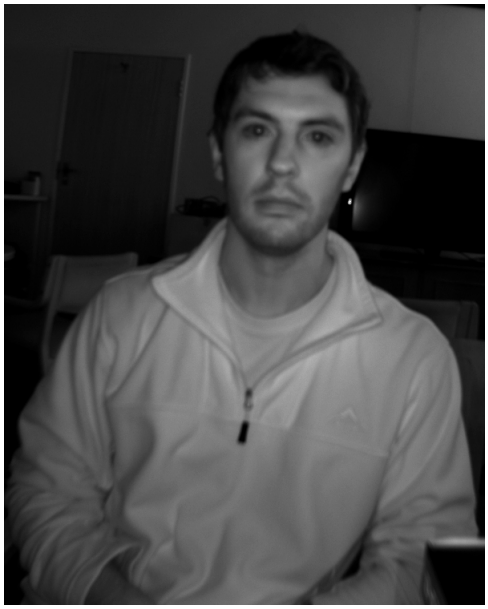

Figure 5.1: An example image of an attempt to obtain the bright pupil effect with 850nm NIR LEDs in a Caucasian subject. The bright pupil effect was present, but not strong enough for practical purposes.

Based on the reasoning presented in Section 4.2.1, the Prosilica GC1380 GigE camera was selected with the goal to determine which combination of NIR illumination and an optical filter would yield the best results for obtaining the bright pupil effect. Mainly due to availability, the NIR illumination that was first considered was the SFH4232 LEDs from Osram, which had a center of spectral emission at 850nm. The initial experiments were performed on

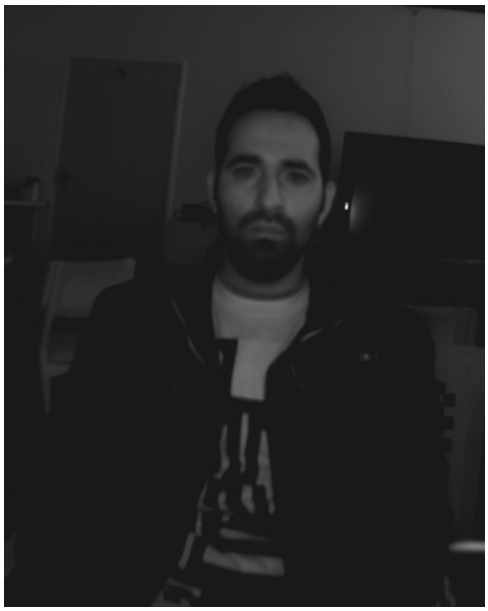

Figure 5.2: An example image of an unsuccesful attempt to obtain the bright pupil effect with 850nm NIR LEDs in a subject from Indian decent.

Figure 5.3: An example image of an unsuccesful attempt to obtain the bright pupil effect with 850nm NIR LEDs in a subject from African decent.

a Caucasian male (the author of this thesis), which was situated a distance between 550mm to 650mm away from the camera. It was found that the bright pupil effect was the strongest when used with the optical bandpass filter of Figure 4.3.

However, the bright pupil effect obtained with the combination of the Prosilica GC1380

camera, the particular optical bandpass filter and the 850nm NIR illumination was not nearly as prominent as illustrated in the literature. In fact, with this configuration it was found that the bright pupil effect was barely visible with certain subjects, most notably subjects from African or Indian decent that in general have much darker eyes when compared to Caucasian subjects. For example, consider Figure 5.1, which shows an attempt to obtain the bright pupil effect with a Caucasian subject. From this figure it can be seen that the bright pupil effect was present, but will typically not be strong enough to robustly detect the eyes. Figure 5.2 and Figure 5.3 show examples of attempts to obtain the bright pupil effect in an Indian and an African subject, respectively. With the the Indian subject the bright pupil effect was, for all practical purposes, not present while the bright pupil effect was only barely present with the African subject.

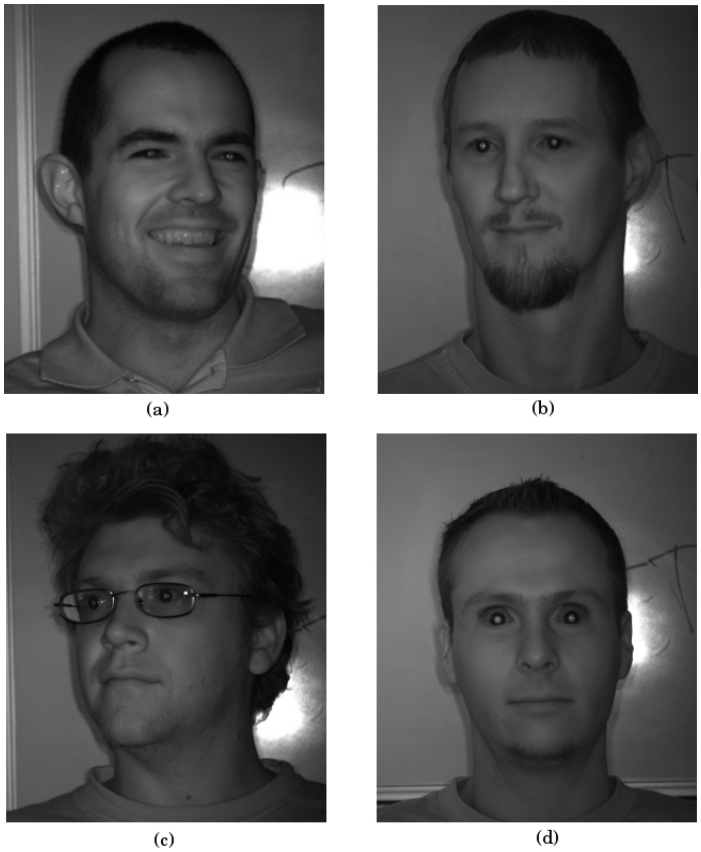

Figure 5.4: Four example images, (a)-(d), of a strong bright pupil effect with Caucasian subjects, using 780nm LEDs.

The weak bright pupil effect based on the 850nm NIR illumination can mainly be attributed to the low sensitivity of the camera's sensor to light at this wavelength and therefore the logical next step was to use NIR illumination at lower wavelengths. When considering the

Figure 5.5: Two example images, (a) and (b), of a relatively weak bright pupil effect with African subjects using 780nm LEDs.

sensitivity of the camera's sensor and the peak of transmission of the chosen optical bandpass filter, coupled with the requirement that the illumination still had to fall within the NIR band of wavelengths, resulted in the selection of of LEDs at wavelengths of 780nm and 810nm.

The experiment to obtain the bright pupil effect was again performed on the author at a distance of between 550mm and 650mm from the camera, but this time with both the 780nm and 810nm NIR LEDs. At both of these wavelengths the bright pupil effect was much stronger when compared to the 850nm NIR LEDs, but it was the 780nm NIR LEDs that ultimately produced the strongest bright pupil effect, which was very similar to the results shown in the literature. Figure 5.4 shows example images for four Caucasian subjects for which the bright pupil effect was very strong at 780nm, with (d) showing that the bright pupil effect could still easily be obtained when the subject was wearing transparent glasses. Figure 5.5 shows two example images of the bright pupil effect in African subjects at 780nm, for which the bright pupil effect was present but relatively weak. This weak bright pupil effect was observed in most of the African subjects tested, but was still enough in most cases to still produce actual eye candidates for eye detection. However, the weak bright pupil effect did place a limitation on the accuracy of the eye tracking phase.

It should be noted that all of the results presented in this section were obtained with a Prosilica GC1380 camera together with a JF4.8 high resolution machine vision lens, which had a focal length of 4.8mm. This particular camera and lens were the property of DPSS, which were initially used to determine the best possible configuration before actually purchasing this expensive piece of equipment. In the final design, as presented in Chapter 4, the same

camera model was used but the Pentax C2514-M lens was used (instead of the JF4.8), which had a focal length of 25mm. In the final design, all of the equipment was the property of the University of Pretoria.

5.1.1 Discussion

The experimental results on the bright/dark pupil effect presented in this section have shown that illumination at 780nm produced the strongest bright pupil effect, which was critical to both the eye detection and the eye tracking phases. However, the chosen NIR illumination to obtain the bright pupil effect was strongly dependent on the sensitivity range of the actual camera being used, together with the response of the optical band-pass filter that was used. It is believed that there still exists a slight mismatch between the 780nm illumination used, the camera's sensor and the optical band-pass filter since none of the components were custom made for each other. In particular, a different camera sensor, which is more sensitive in the NIR spectrum of light would more easily produce the bright pupil effect, which implies that less NIR LEDs might be necessary to obtain the same effect.

A very interesting observation that was made, is the variability of the bright pupil effect among different ethnic groups. In almost all of the Caucasian subjects that were tested the bright pupil effect was very strong, whereas the bright pupil effect was in general weak among African and Indian subjects. Asian and Hispanic subjects were not available for testing, but their results would be compelling. Nguyen *et al* [11] also observed this phenomenon, and based on their results it can be expected that the bright pupil effect might even be stronger in Hispanic subjects when compared to Caucasian subjects, whereas Asian subjects are expected to have a weaker bright pupil effect when compared to Caucasian subjects. This variability of the bright pupil effect was mentioned a couple of times in the literature, but the cause of this variability is apparently still unknown, since it was not very well documented in the literature at the time of writing this thesis. Given that the majority of the current eye tracking systems are based on the bright pupil effect, it is perhaps strange that this phenomenon did not already received much more attention.

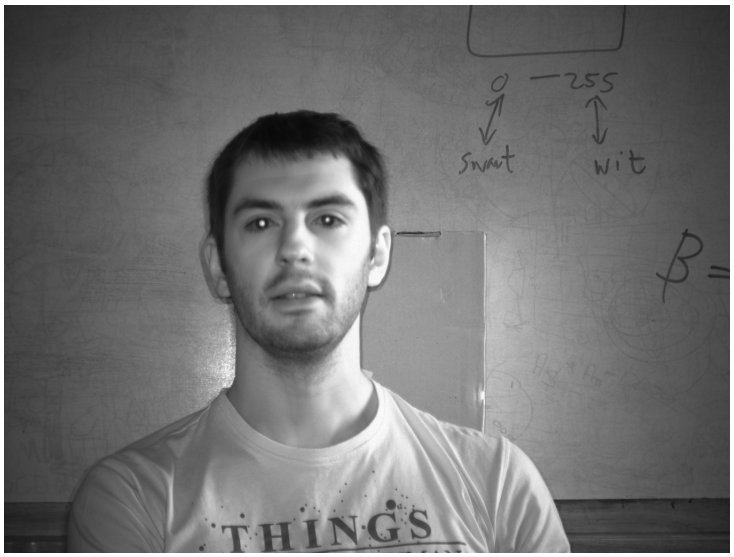

Figure 5.6: The bright pupil image as produced by on-center illumination from 780nm NIR LEDs.

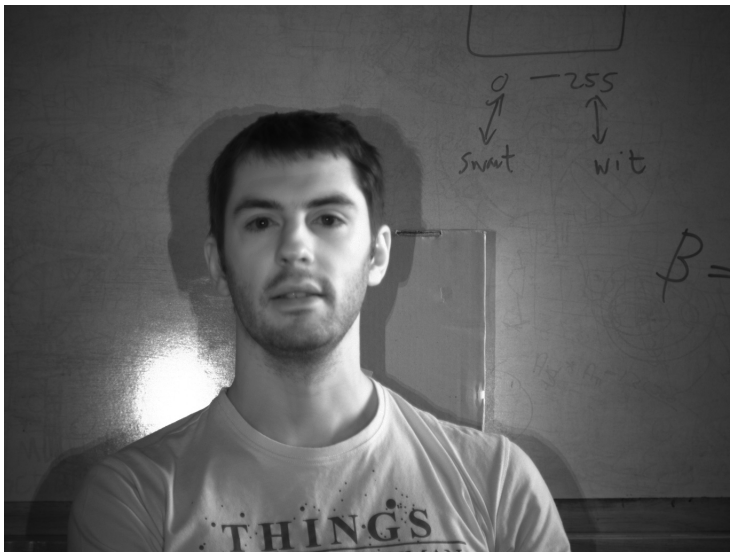

Figure 5.7: The dark pupil image as produced by off-center illumination from 780nm NIR LEDs.

5.2 EYE DETECTION

5.2.1 Eye Candidate Extraction

Given the strong bright pupil effect that was obtained with the configuration of 780nm NIR LEDs, it was relatively simple to obtain the possible eye candidates. For example, consider the bright and dark pupil images from the author as illustrated in Figure 5.6 and Figure 5.7, respectively. The pixel intensities from the dark pupil image were subtracted from

Figure 5.8: The resulting difference image from subtracting the dark pupil image (Figure 5.7) from the bright pupil image (Figure 5.6).

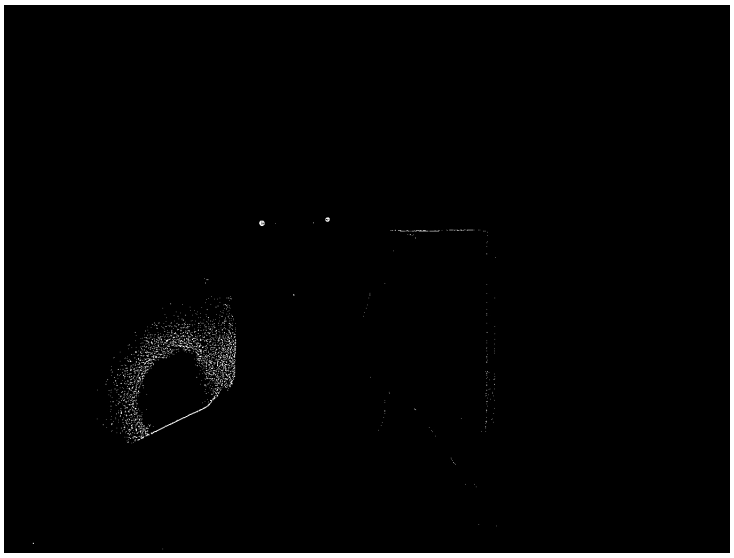

Figure 5.9: The resulting binary image from adaptively thresholding the difference image (Figure 5.8).

the corresponding pixel intensities in the bright pupil image, which yielded the difference image of Figure 5.8. From this difference image it is already apparent where the eyes are located. The next step was to adaptively threshold the difference image to obtain a binary image (i.e. only black and white pixels), which is shown in Figure 5.9.

The aim of the final binary image was to significantly reduce the number of possible eye candidates, as opposed to scanning the entire image, as necessary to detect the eyes. A

connected component analysis was performed on the binary image to obtain the white blobs, which were then mapped back to the original dark pupil image to extract the corresponding sub-images. From Figure 5.9 it can be seen that the white blobs in the regions of the eyes were the most prominent, with a number of other smaller white blobs also being present due to background objects reflecting the on-center and off-center NIR illumination in a similar fashion than the eyes. It was observed that a dark background that typically contains almost no objects that strongly reflect light, would result in a binary image that essentially only contained the white blobs that were associated with the eyes. In such cases the eyes could then almost be detected directly, making the eye detection process extremely efficient.

5.2.2 Preliminary Eye Classification Results

Given the extracted sub-images as a result of the white blobs in the binary image, a classifier was required to discriminate between the eye and non-eye sub-images. In similar work presented by Zhu *et al* [19], SVM classifiers were used, which achieved high accuracy rates and was therefore the first type of classifier that was used. Also based on the high accuracies achieved in face detection in the work presented by Viola and Jones [66], the different variants of the AdaBoost classifier were also used for discrimination. Finally, as a baseline for comparison, two types of artificial neural network classifiers were used.

Various configurations for each type of classifier were available and, depending on the application, some configurations will perform better than others. For this research the following SVM kernels were considered (as provided by the OpenCV SVM implementation): polynomial, Gaussian radial basis functions (RBF) and sigmoid. In the case of AdaBoost the different implementations that were considered (as provided by the OpenCV Boosting implementation) were: Discrete AdaBoost, Real AdaBoost, LogitBoost and Gentle AdaBoost. More details on these AdaBoost variants can be found in Friedman *et al* [29]. For the neural network classifiers the most widely used random sequential backpropagation (BACKPROP) and resilient backpropagation (RPROP) methods were considered (also provided by the OpenCV).

Due to some uncertainty of how well these various classifiers would perform at the task of eye classification, preliminary tests had to be performed to narrow down the options for the final classifier that would ultimately be used. The preliminary classification tests were performed on a dataset consisting of a single Caucasian subject, obtained with the

hardware configuration from DPSS, with the only significant difference from the final design (as presented in Chapter 4) being the lens that was used with the camera. The effect of using the JF4.8 lens was that the subject was between 550mm and 650mm away from the camera, which resulted in a suitable resolution for the sub-images (to be classified) to be 35x35 pixels.

Table 5.1: A summary of the preliminary dataset used for the classifier performance evaluation.

Preliminary dataset	
Number of subjects:	1
Subject information:	Only male and Caucasian
Training set:	739 eye (positive) / 660 non-eye (negative)
Testing set:	236 eye / 210 non-eye
Image resolution:	35x35

There was also uncertainty (mainly due to the author's inexperience with the specific classifiers) as to which configuration for each type of classifier would yield the best performance results, and therefore these preliminary tests were conducted with all of the possible configurations provided by their respective OpenCV implementations.

A total of 975 eye and 870 non-eye sub-images (at a resolution of 35x35 pixels) were extracted from the captured images, with the raw pixel intensities of the sub-image being the feature vector. The extracted sub-images were then randomly divided into a training set and a testing set. The training set consisted of 739 eye (positive) and 660 non-eye (negative) images, whereas the testing set consisted of 236 eye and 210 non-eye images. The considered classifiers were trained on the training set and their respective accuracies were measured with the classification of the "unseen" testing set. A summary of the dataset used for the preliminary classification performance evaluation is shown in Table 5.1.

The preliminary results from the considered classifiers are shown in Table 5.2 to Table 5.7. For the neural network classification, the RPROP method of training was also considered but the classical BACKPROP method consistently outperformed RPROP and since neural networks were only used a baseline for comparison, it was considered unnecessary to also include the RPROP results. For SVM classification the Sigmoid kernel was also considered, but its performance was very poor and was therefore considered unsuitable for eye classification. The parameters for the neural networks and AdaBoost classifiers were manually selected, whereas the optimal parameters for the SVM classifiers were automatically obtained by means of a

grid search. The manual selection of the parameters for the neural network and AdaBoost classifiers were made on a trial-and-error basis until the best possible results were obtained.

As mentioned previously, the personal computer that was used for training had an Intel Core 2 Quad CPU (Q8300 with a clock speed of 2.5 GHz) with 2GB of RAM. The Ubuntu 10.04 (Lucid Lynx) operating system was installed on the computer and during training no programs other than the normal operating processes were running. Besides classification accuracy, the training time was also considered to be a performance metric of the classifier. Although the training process occurs offline, the training time should still be reasonable for very large training sets from multiple subjects, as assumed necessary to produce a classifier that would be able to generalizes well. One of the goals of the chosen classifier was that it would be possible to train it with an average PC, not requiring a high-performance CPU or a vast amount of RAM (training time was related to the RAM usage).

The best performing neural network had three hidden layers, with 200 neurons in each hidden layer. For this configuration the True Positive Rate (TPR) was 93.9% (i.e. actual eyes correctly classified) and the True Negative Rate (TNR) was 94.0% (i.e. actual non-eyes correctly classified). Futher results for the neural networks are presented in Table 5.3, which shows that the overall accuracy of the best performing configuration was also 93.9%. The False Positive Rate (FPR) was 6.0% (i.e. actual non-eyes incorrectly classified as eyes) and the False Negative Rate (FNR) was 6.1% (i.e. actual non-eyes incorrectly classified as eyes).

The best performing SVM configuration, in terms of both accuracy and training time, was achieved with the RBF kernel, which achieved a TPR of 96.3% and a TNR of 95.9%. Additional results on the SVM classification results are shown in Table 5.5, which indicates that the overall accuracy of the RBF kernel was 96.1%, while the FPR was 4.1% and the FNR was 3.7%. In terms of overall accuracy, the second degree polynomial kernel performed slightly better than the RBF kernel, but took almost 4 times as long to train. Similarly, the RBF kernel only slightly outperformed the third degree polynomial kernel in terms of overall accuracy, but the third degree polynomial kernel took more than 11 times as long to train.

All of the AdaBoost algorithms performed exceptionally well, with high accuracy and a training time of well below a minute. Discrete AdaBoost was the best performing classifier and achieved a TPR of 97.5% and a TNR of 96.8%. Further results for the AdaBoost classifiers are shown in Table 5.7, which indicate that for Discrete AdaBoost the overall accuracy was

97.2%, while the FPR was 3.2% and the FNR was 2.5%. These preliminary results for the group of AdaBoost classifiers confirmed the expectation that AdaBoost would be well suited for the task of eye classification. Due to the very similar performance results, all of the AdaBoost algorithms (except LogitBoost) were considered for further testing.

When considering the performance results from Table 5.3, Table 5.5 and Table 5.7, the best performing configurations from each type of classifier were selected to be tested again on an extended dataset, which contained various subjects from a number of different ethnic groups.

Table 5.2: Initial classifier performance results of the artificial neural networks, using sequential backpropagation, on a single subject with 739 eye and 660 non-eye training images, and 236 eye and 210 non-eye testing images.

Classifier	**Parameters**	**Training time (HH:MM:SS)**	**Accuracy**	
			Eyes	**Non-eyes**
ANN				
BACKPROP	2 hidden layers (50 neurons per layer)	± 00:01:50	91.8%	93.1%
BACKPROP	2 hidden layers (100 neurons per layer)	± 00:05:24	92.2%	90.8%
BACKPROP	2 hidden layers (200 neurons per layer)	± 00:18:38	91.8%	90.8%
BACKPROP	3 hidden layers (50 neurons per layer)	± 00:01:57	92.2%	91.7%
BACKPROP	3 hidden layers (100 neurons per layer)	± 00:06:28	92.6%	91.7%
BACKPROP	3 hidden layers (200 neurons per layer)	± 00:21:23	93.9%	94.0%

Table 5.3: Initial classifier performance results of the artificial neural networks on 236 eye (positive) and 210 non-eye (negative) testing images.

Classifier	**Overall accuracy**	**False positive rate (FPR)**	**False negative rate (FNR)**
ANN			
BACKPROP (2 H/L, 50 neurons)	92.4%	6.9%	8.2%
BACKPROP (2 H/L, 100 neurons)	91.6%	9.2%	7.8%
BACKPROP (2 H/L, 200 neurons)	91.3%	9.2%	8.2%
BACKPROP (3 H/L, 50 neurons)	92.0%	8.3%	7.8%
BACKPROP (3 H/L, 100 neurons)	92.2%	8.3%	7.4%
BACKPROP (3 H/L, 200 neurons)	93.9%	6.0%	6.1%

Table 5.4: Initial classifier performance results of SVMs on a single subject with 739 eye and 660 non-eye training images, and 236 eye and 210 non-eye testing images.

Classifier	**Parameters**	**Training time (HH:MM:SS)**	**Accuracy**	
			Eyes	**Non-eyes**
SVM				
RBF kernel	$\gamma = 0.03375$, $C = 12.5$	± 00:09:16	96.3%	95.9%
Polynomial kernel (Degree = 1)	$\gamma = 0.00225$, $coef0 = 0.1$, $C = 62.5$	± 00:37:02	91.0%	89.9%
Polynomial kernel (Degree = 2)	$\gamma = 0.03375$, $coef0 = 0.1$, $C = 2.5$	± 00:36:01	96.7%	95.9%
Polynomial kernel (Degree = 3)	$\gamma = 0.03375$, $coef0 = 0.1$, $C = 1.4$	± 01:43:34	95.5%	96.3%

Table 5.5: Initial classifier performance results of the SVMs on 236 eye (positive) and 210 non-eye (negative) testing images.

Classifier	**Overall accuracy**	**False positive rate (FPR)**	**False negative rate (FNR)**
SVM			
RBF kernel	96.1%	4.1%	3.7%
Polynomial kernel (degree = 1)	90.5%	10.1%	9.0%
Polynomial kernel (degree = 2)	96.3%	4.1%	3.3%
Polynomial kernel (degree = 3)	95.9%	3.7%	4.5%

Table 5.6: Initial classifier performance results of AdaBoost on a single subject with 739 eye and 660 non-eye training images, and 236 eye and 210 non-eye testing images.

Classifier	**Parameters**	**Training time (HH:MM:SS)**	**Accuracy**	
			Eyes	**Non-eyes**
AdaBoost				
Discrete AdaBoost	Weak count = 100, max depth = 5	± 00:00:29	97.5%	96.8%
Real AdaBoost	Weak count = 100, max depth = 5	± 00:00:33	95.9%	97.7%
LogitBoost	Weak count = 100, max depth = 5	± 00:00:34	94.3%	91.3%
Gentle AdaBoost	Weak count = 100, max depth = 5	± 00:00:35	96.7%	97.2%

Table 5.7: Initial classifier performance results of AdaBoost on 236 eye (positive) and 210 non-eye (negative) testing images.

Classifier	**Overall accuracy**	**False positive rate (FPR)**	**False negative rate (FNR)**
AdaBoost			
Discrete AdaBoost	97.2%	3.2%	2.5%
Real AdaBoost	96.8%	2.3%	4.1%
LogitBoost	92.9%	8.7%	5.7%
Gentle AdaBoost	97.0%	2.8%	3.3%

5.2.3 Final Eye Classification Results

Table 5.8: A summary of the final dataset used for the classifier performance evaluation.

Final dataset	
Number of subjects:	17
Subject information:	- 3 African female subjects
	- 1 African male subject
	- 4 Caucasian female subjects
	- 9 Caucasian male subjects
Training set:	2632 eye (positive) / 3681 non-eye (negative)
Testing set:	909 eye / 2101 non-eye
Image resolution:	65x65

The next step was therefore to again perform the classifier performance testing, but on a larger dataset in order to determine how well these classifiers could generalize on various subjects that had not been used for training. To obtain this dataset, the camera setup was slightly modified from the one used at DPSS. At this point in time, the University of Pretoria purchased its own Prosilica GC1380 camera and lens for the purpose of this research. The camera was the same as the one used at DPSS, but the lens was different. The selected lens was the Pentax C2514-M high resolution lens, which was specifically designed for machine vision applications.

This lens had a focal length of 25mm, which implied that the subjects had to be located further away from the camera to obtain properly focused images. Pentax recommended that an ideal object distance for this lens was approximately 1.5 meter, and therefore the subjects used to obtain this training set was located between 1400mm and 1600mm away from the camera, as opposed to the 550mm to 650mm used in the initial setup at DPSS. This setup also resulted in an increased resolution of the extracted sub-images to be classified, which was now 65x65 pixels as opposed to the 35x35 pixels in the initial setup.

Given this new setup, the dataset of images were captured from 17 different subjects. This dataset consisted of 3 African females, 1 African male, 4 Caucasian females and 9 Caucasian males. The images were captured indoors under various ambient lighting conditions (i.e. different times during the day) with some of the subjects also wearing glasses, in an attempt to keep the dataset as general as possible. For training purposes the images from 13 of the 17

subjects were used, with the images of the remaining 4 subject being used for testing purposes. The testing set consisted of one African female, the only African male, one Caucasian female and one Caucasian male, in order to be representative of both ethnical background and sex.

As a result the final training set consisted of 2632 eye and 3681 non-eye images, whereas the final testing set consisted of 909 eye and 2101 non-eye images. Since in practice there would typically be much more actual non-eyes than actual eyes, it was decided that the testing set of non-eyes should contain considerably more examples. The feature vector of an image was again the raw pixel intensities, which implied that a feature vector consisted of 65x65 elements (the image resolution), which was much larger than the 35x35 element feature vector used in the preliminary classifier performance testing. As a result, it was expected that the classifier training times would be significantly longer. A few examples images that were used for training the different classifiers are shown in Figure 5.10 (eyes) and Figure 5.11 (non-eyes). A summary of the final dataset that was used for the classifier performance evaluation is shown in Table 5.8.

The final eye classifier performance results are shown in Table 5.9 and Table 5.10. At a glance, it can already be seen that these classifier performance results were significantly worse when compared to the preliminary classifier performance results for a single subject. The sequential backpropagation neural network had a TPR of 72.7% and a TNR of 92.7%, which was a reduction of 21.2% for the TPR (from 93.9%) and a reduction of 1.3% for the TNR (from 94.0%). Therefore the observation that can be made, was that the neural network classifier struggled much more with correctly classifying eyes than non-eyes. From Table 5.10 the overall classifier accuracy of the neural network was 86.6%, but this was actually quite misleading since there were more than twice as many non-eye testing images, which biased the overall accuracy towards the TNR. Had there been an even amount of eye and non-eye testing examples, the overall accuracy would have been considerably lower. The training time of the neural network also drastically increased by almost 16 times, from 21:23 minutes to 5:36:35 hours, but the reader should take into account that the size of the feature vector together with the number of training examples increased drastically.

A similar trend was observed with the SVM classification on the larger data set, but in this case the volatile memory requirement was a serious hampering factor. When the preliminary classification testing was performed, the PC that was used had 2GB of RAM installed. However, when the SVM training (with a RBF kernel) was performed on this larger data set,

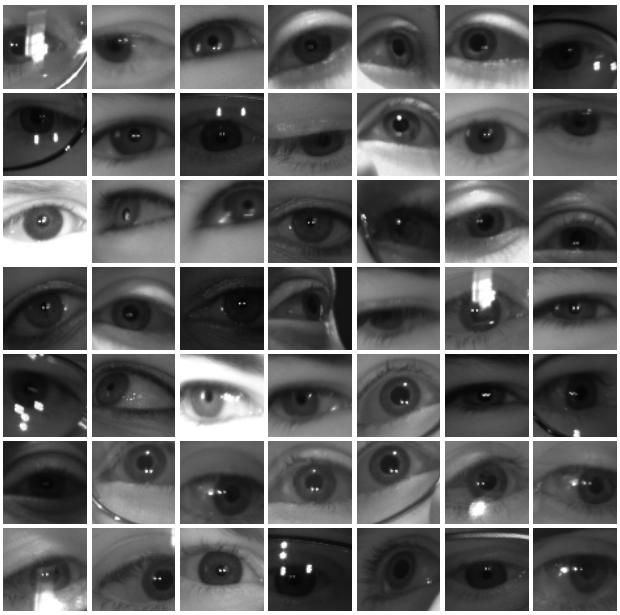

Figure 5.10: A few eye training examples (i.e. positive examples) as used in the training set for the final classification testing.

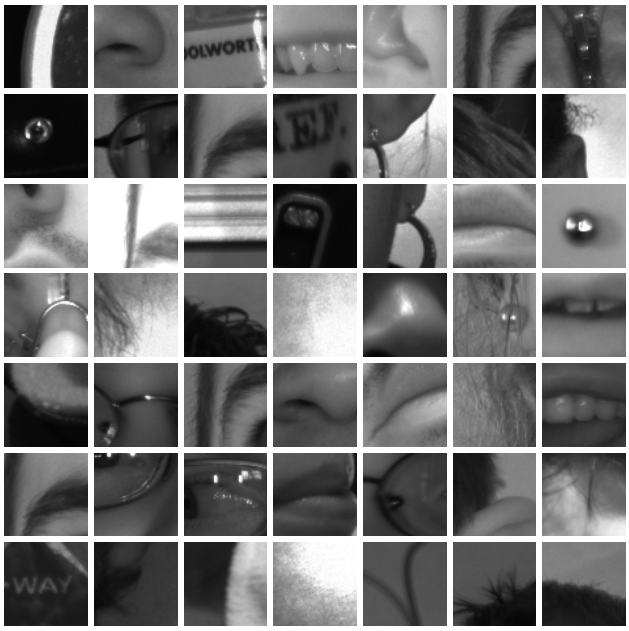

Figure 5.11: A few non-eye training examples (i.e. negative examples) as used in the training set for the final classification testing.

the training thread crashed repeatedly as a result of running out of memory. Consequently an additional 4GB of RAM had to be installed and therefore the PC had a total of 6GB of memory, which enabled the SVM training to be completed. For comparison a second degree polynomial kernel was instead used for SVM training, but even the 6GB of RAM was not

enough for the training to complete! The total time for SVM training with a RBF kernel increased by almost 7 times from 9:16 minutes to 1:03:33 hours.

Table 5.9: The final classifier performance results on 17 subjects with 2632 eye and 3681 non-eye training images from 13 subjects, and 909 eye and 2101 non-eye testing images from 4 subjects.

Classifier	**Parameters**	**Training time (HH:MM:SS)**	**Accuracy**	
			Eyes	**Non-eyes**
ANN				
BACKPROP	3 hidden layers (200 neurons per layer)	± 05:36:35	72.7%	92.7%
SVM				
RBF kernel	$\gamma = 0.03375$, $C = 62.5$	± 01:03:33	74.3%	97.3%
AdaBoost				
Discrete AdaBoost	Weak count = 200, max depth = 5	± 00:16:18	85.0%	95.5%
Real AdaBoost	Weak count = 200, max depth = 5	± 00:17:36	79.1%	95.5%
Gentle AdaBoost	Weak count = 200, max depth = 5	± 00:18:38	83.9%	95.6%

Table 5.10: The final classifier performance results on 17 subjects with 2632 eye and 3681 non-eye training images from 13 subjects, and 909 eye and 2101 non-eye testing images from 4 subjects.

Classifier	**Overall accuracy**	**False positive rate (FPR)**	**False negative rate (FNR)**
ANN			
BACKPROP (3 H/L, 200 neurons)	86.6%	7.3%	27.3%
SVM			
RBF kernel	90.3%	2.7%	25.7%
AdaBoost			
Discrete AdaBoost	92.3%	4.5%	15.0%
Real AdaBoost	90.5%	4.5%	20.9%
Gentle AdaBoost	92.1%	4.4%	16.1%

In terms of performance, the RBF SVM classifier had a TPR of 74.3% and a TNR of 97.3%.

This was a reduction of 22.0% for the TPR (from 96.3%) and an increase of 1.4% for the TNR (from 95.9%). As with the neural network, it was observed that the SVM classifier struggled more to correctly classify actual eyes, whereas it had seemingly very little trouble in correctly classifying non-eyes. From Table 5.10 the overall accuracy of the RBF SVM classifier was 90.3%, which was biased towards the TNR (as mentioned with the neural network performance results).

A similar performance trend was again observed in the AdaBoost classifier performance, but to a lesser extent. As mentioned before, Discrete AdaBoost, Real AdaBoost and Gentle AdaBoost were all considered for further testing on the final dataset due to their very similar preliminary performance results. However, it should be noted that the parameters used for these AdaBoost classifiers were slightly changed from those used in the preliminary classifier evaluation. The amount of weak classifiers for AdaBoost were doubled, after some trial runs revealed an increase in performance. Since AdaBoost trains much faster when compared to the other classifiers, it was possible to experiment with different parameters to obtain the best possible results. This parameter 'tweaking' was not necessary in SVM training, since the optimal parameters were already obtained through a grid search during the training process and due to the extremely long training time, the neural network classifier was unattractive to start with.

Discrete AdaBoost took almost 34 times longer to train from 29 seconds to 16:18 minutes, while Real AdaBoost took 32 times longer to train from 33 seconds to 17:36 minutes. Similarly, Gentle AdaBoost took almost 32 times longer to train from 35 seconds to 18:38 minutes. This drastic increase can again be attributed to a much larger training set and the fact that the size of feature vectors was more than tripled from 35x35 to 65x65.

In terms of accuracy the Discrete AdaBoost classifier had a TPR of 85.0% (a decrease of 12.5% from 97.5%) and a TNR of 95.5% (a decrease of 1.3% from 96.8%). The Real AdaBoost classifier had a TPR of 79.1% (a decrease of 16.8% from 95.9%) and TNR of 95.5% (a decrease of 2.2% from 97.7%). Likewise, the Gentle AdaBoost had a TPR of 83.9% (a decrease of 12.8% from 96.7%) and a TNR of 95.6% (a decrease of 1.6% from 97.2%). From Table 5.10 the overall accuracy was 92.3% for Discrete AdaBoost, 90.5% for Real AdaBoost and 92.1% for Gentle AdaBoost, which was also biased towards the TNR. The AdaBoost classifiers also had very little trouble in correctly classifying actual non-eyes, but performed much better (more than 10% for Discrete AdaBoost) in correctly classifying actual eyes when compared

to both the neural network and the RBF SVM classifiers.

5.2.4 Eye Detection Results

From the final classification results presented in Table 5.9, it was evident that the Discrete AdaBoost classifier provided the best generalization results for eye detection and was consequently chosen as the classifier for the final system. Given all of the possible eye candidates due to the bright/dark pupil effect, the Discrete AdaBoost classifier would therefore be used to discriminate between eyes and non-eyes during operation of the system. Although this proved to be a highly effective method of detecting eyes, it was observed that a few false positives occurred from time to time (i.e. 'eyes' were detected that were not actually eyes). This implied that the eye tracking phase would then be initialized with a non-eye, which was fortunately usually detected within a couple of frames, only to re-initialize the eye detection phase. As a result a lot of processing time was wasted due to incorrect classification and therefore this had to be improved.

In order to verify the correct classification of eyes, geometric constraints were used in the form of the distance-to-angle relationship between two detected eyes. The use of such a geometric constraint was motivated by the observation that incorrect classification typically only occurred for one of the actual eyes, which in most cases implied that one eye was successfully detected and the second incorrectly detected 'eye' was situated either far away from the actual eye or at a large angle with respect to the actual eye. In either case the distance or angle (or both) between the actual eye and the incorrect eye was much larger than it would have been if both the detected eyes were actual eyes.

Since an image is essentially a 2-dimensional projection of the 3-dimensional space, the distance between the eyes will not remain fixed from frame-to-frame as it would in the 3-dimensional space. For example, the distance between the eyes for frontal face images with no rotation will be larger when compared to frontal face images with sideways head rotation. In order to get a feeling for these distances, some measurement experiments were performed on a single subject. The Discrete AdaBoost classifier was used to detect the eyes and the correct classification was then manually verified. In cases where the eyes were correctly detected, the distance between the eyes were measured together with the angle between the eyes (with respect to the horizontal plane) by using the following formulas:

$$\text{Eye distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad \text{[pixels]} \tag{5.2.1}$$

$$\text{Eye angle} = \cos^{-1}\left(\frac{|x_2 - x_1|}{\text{Eye distance}}\right) \quad \text{[degrees]} \tag{5.2.2}$$

where point (x_1, y_1) is the center of the left eye in the image (i.e. the actual right eye) and point (x_2, y_2) is the center of the right eye in the image (i.e. the actual left eye). The eye distances and angles were then measured for a single subject for various head poses to obtain a metric to verify the correct classification of the eyes. The eye distances were categorized according to the eye angle range it fell into (approximately 10 measurements per range), and for each range the minimum and maximum values were determined, which are shown in Table 5.11.

Table 5.11: The minimum and maximum distances between two detected eyes, for a given range of angles between the eyes (with respect to the horizontal plane).

Eye Angle Range [degrees]	**Min Distance [pixels]**	**Max Distance [pixels]**
0 - 10	111.5	190.1
11 - 20	136.9	168.4
21 - 30	133.0	165.1
31 - 40	144.0	162.2
41 - 50	130.1	160.6
51 - 60	157.1	173.8

From Table 5.11 it can be seen that the largest variation in the distance between the eyes, occurred in the range 0 to 10 degrees. These large variations in distance for this small range of angles, was mainly due to out-of-plane (i.e. sideways) head rotations. It was observed that in most of the cases when false positives occurred, the other detected eye was an actual eye and at least either the distance or the angle between the false positive 'eye' and the other actual eye was much larger than the typical distance or angle between two correctly classified eyes. This was of course dependent on the background and this might not always be the case.

Nevertheless, the very simple verification metric that was formulated from this data, was that two detected eyes were only considered to be actual eyes if the angle between them was smaller than 60 degrees and the distance between them was within the range 100 to 200 pixels. This is admittedly a rather crude approach since it depends on the image resolution, the actual

Figure 5.12: Eye detection results for a subject without wearing glasses, for various head poses, some of which caused significant occlusion. In each example image the distance between the detected eyes as well as the angle between the eyes (with respect to the horizontal plane) are also given.

distance that the subject is away from the camera and to a lesser extend the subject itself. However, by employing this simple metric in the test setup, it made the occurrences of false positives extremely rare. It should be noted that false negatives (i.e. an actual eye classified as a non-eye) was not much of a concern, since the system would just keep on attempting to detect both eyes until it succeeded.

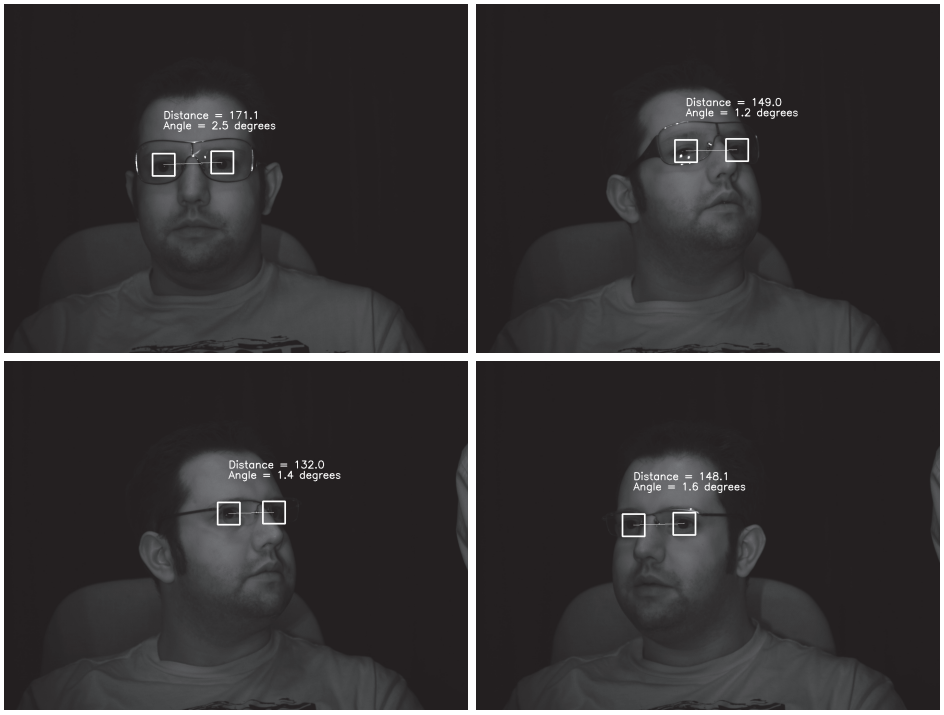

Figure 5.13: Eye detection results for a subject wearing sunglasses (top) and normal reading glasses (bottom), for various head poses. In each example image the distance between the detected eyes as well as the angle between the eyes (with respect to the horizontal plane) are also given. Note that the subject used for these images was not used in the training set of the classifier.

A few eye detection results for a subject (without wearing glasses) are shown in Figure 5.12, after employing this simple verification metric, which ranged from simple frontal eye detection to eye detection for large head movements, which could cause significant occlusions. In all of the images the angle between the detected eyes, as well as the distance between them are also shown, and the reader will notice that in every case the conditions of the simple verification metric were met.

Examples for eye detection with a subject, which was not in the training set of the classifier, wearing glasses are also shown in Figure 5.13. The top two examples are for sunglasses, while the bottom two examples are for normal reading glasses. It is evident that the system was still able to accurately detect the eyes while the subject was wearing glasses, even under different head poses. The reader will again notice that in every case the conditions of the simple verification metric were met. The system would be able to detect the eyes as along as the glasses were transparent to the NIR light and the reflections of the light from the glasses were not too severe. The system would, for example, fail when the subject was wearing polarized

sunglasses, since the eyes would in such cases not be visible at all. Also under certain head poses the reflected light from the glasses or the frame of the glasses would result in large occlusions, which made eye detection quite difficult.

From the results shown in Figure 5.12 and Figure 5.13 it is apparent that the presented eye detection approach was very robust in accurately detecting the eyes under various head poses, significant occlusion, varying ambient light and also in situations where the subject wore glasses. Also by using the trained Discrete AdaBoost classifier from Table 5.9, resulted in the robust detection of unseen subjects, although in certain cases a number of attempts were necessary before the eyes were successfully detected.

Since the proposed eye detection process was heavily dependent on the specifically designed hardware, it was not possible to directly compare it was other eye detection techniques in the literature, which used readily available datasets. It would of course be possible to generate a number of bright/dark pupil image pairs and manually mark the eyes in each dark pupil image and then execute this eye detection process on it and count in how many bright/dark pupil image pairs the eyes were successfully detected. However, the author believed that this would be a pointless exercise for the following reasons:

1. At the time of writing this thesis, there was no standard benchmark for the difficulty of the captured bright/dark pupil image pairs, e.g. the author could have chosen 100 bright/dark pupil image pairs of predominantly pure frontal head poses in which it would be relatively easy to detect the eyes and conclude that the eye detection approach was highly robust, which may perhaps not be the case.

2. The system was designed to be persistent, so even if the eyes were not detected with the first attempt, the system would keep trying until it succeeded. This implied that even if the system only successfully detected the eyes once in every third bright/dark pupil image pair (i.e. a successful detection rate of 33.3%), it might still be more than sufficient for a given application. The reader is reminded that the classification results from Table 5.9 suggest that the Discrete AdaBoost classifier is far more likely to misclassify an actual eye image than an actual non-eye image.

5.2.5 Discussion

This section presented the detailed results of the proposed eye detection approach followed in this research and started out by describing how the bright/dark pupil effect could be used to robustly and with minimal processing, obtain eye candidates. This was arguably the most efficient method of detecting the eyes, since the entire image does not have to be scanned in order to detect the eyes. If the image background contained little or no objects that reflected the NIR light in a similar manner than the pupils, the resulting eye candidates were almost limited to just the actual eyes, making this process even more efficient. However, a potential major limitation of this approach is the fact that the bright pupil effect was not consistent across ethnic groups, which implied that there could potentially be no major differences between the bright and dark pupil images in a subject, which in turn means that the resulting eye candidates might not even contain the actual eyes. In particular, this was observed for African and Indian subjects.

Given a number of eye candidates, the performance of three types of classifiers were evaluated to determine which type of classifier would be the best at discriminating between eyes and non-eyes. Preliminary experiments were performed on a single Caucasian subject in order to get an idea of how well these classifiers could perform at the specific task of eye classification. From the preliminary performance results, it was already apparent that the AdaBoost classifier (and its variants) would most likely be the best suited for this particular application. One of the goals of this research was to develop an eye tracker that worked with various subjects, without any prior calibration and therefore the training and testing datasets were extended to contain a total of 17 individual subjects from different ethnic groups.

The best performing classifiers from the preliminary classification results were again trained on the extended dataset, but were this time tested with subjects that were not present during training. Again the AdaBoost classifiers, and in particular Discrete AdaBoost, produced superior results when compared to artificial neural networks and SVMs, with a TPR of 85.0% and a TNR of 95.5%. Artificial neural networks were only used as a baseline for comparison, and it was clear from the start that it would not be used in the final configuration, but it should be noted that in practice the neural network might actually require more training examples than were used in this research, in order to obtain optimal results. It was interesting to note that the Discrete AdaBoost classifier significantly outperformed the SVM classifier

with a RBF kernel, something that was not completely expected.

This result can perhaps be explained in terms of the complexity of the actual underlying eye model that was being learned. When only considering a single subject, the appearance of the eye can change drastically with different head poses, as well as when the subject was wearing glasses. This large variability in eye appearance already makes the underlying eye model rather complex, but this eye model becomes much more complex when introducing inter-subject variability (which is known to be very large). Since the underlying eye model is in reality so complex, overfitting becomes a problem and therefore the classifier that can resist overfitting the best, will produce the best results. From the literature it is well known that AdaBoost is in general particularly good at resisting overfitting, which is therefore considered the most likely explanation for its superior performance at eye classification.

In any case, from a practical and real-world point of view, simpler is almost always better and the Discrete AdaBoost classifier is yet another example thereof (for eye detection at least), which obtained highly accurate results at a fraction of the computational complexity of both artificial neural networks and SVMs.

Another somewhat unexpected result was that all of the classifiers performed exceptionally well at correctly classifying the non-eyes of the extended dataset. It was expected that some non-eye examples would eventually have a very similar intensity distribution when compared to actual eye examples and consequently incorrectly classify non-eyes as eyes (i.e. false positives). In fact the opposite was more prevalent, which made it more likely for an eye to be classified as a non-eye (i.e. false negative). Fortunately, this was actually a favorable result, since false negatives will create much less of a negative perception (from the user's point of view) of the system's accuracy when compared to false positives.

However, false positives did occur occasionally, which were addressed by using a simple geometric constraint (based on measurements of the angle and distance between the eyes) to verify the correct classification of eyes. By using this simple metric, the false positive rate dropped drastically but did still occur in very rare instances, which were fortunately quickly detected by the eye tracking phase. False positives typically only occurred when both detected eyes were not actually eyes and were situated close enough to each other to satisfy the geometric constraint.

The results from this section have revealed that there are significant inter-subject variations

and that some form of system calibration was inevitable, either for the training of the classifier or for the thresholds for the geometric constraints.

5.3 EYE TRACKING

5.3.1 Frame Pre-processing

The first step of the eye tracking phase was to define the ROI, which had to be large enough so that the movement of the eyes would not result in the eyes in the next frame to be outside of the ROI, and also had to be small enough to still significantly speed up the image processing. The resolution of the images produced by the camera was 1360x1024 pixels and after some rule-of-thumb experimenting, it was determined that a ROI of 600x300 was reasonable since large sideway head movements were more likely than large up and down head movements (i.e. its physically easier for a subject to make sideway head movements).

The placement of the origin (the top left pixel coordinates) of the 600x300 ROI was calculated according to the location of the left detected eye in the image as follows:

$$ROI_x = LDE_x - ROI_{width} * 0.25 \quad (5.3.1)$$

$$ROI_y = LDE_y - ROI_{height} * 0.25 \quad (5.3.2)$$

where $LDE_{x,y}$ is the top left coordinates of the bounding window of the left detected eye, with ROI_{width} and ROI_{height} being equal to 600 and 300, respectively. If one of the corner coordinates of the ROI fell outside of the image bounds, its particular coordinate would simply be assigned to the bound in question. By only processing the much smaller ROI, implied that a significant amount of processing time was saved by only focusing on the immediate region around the eyes.

As discussed in the design of the eye tracking algorithm (Chapter 4), once the eyes were successfully detected, the bright pupils were modeled as white flood-filled circles which were then redrawn on a black ROI and then tracked in this manner from frame-to-frame. An important step in this approach was to detect the edges in the ROI, which can then in turn be used to detect the circles corresponding to the pupils. This was achieved by first smoothing the ROI and then applying the Canny edge detector to the smoothed ROI. A few examples of the resulting edges after applying Canny edge detection are shown in Figure 5.14, with the

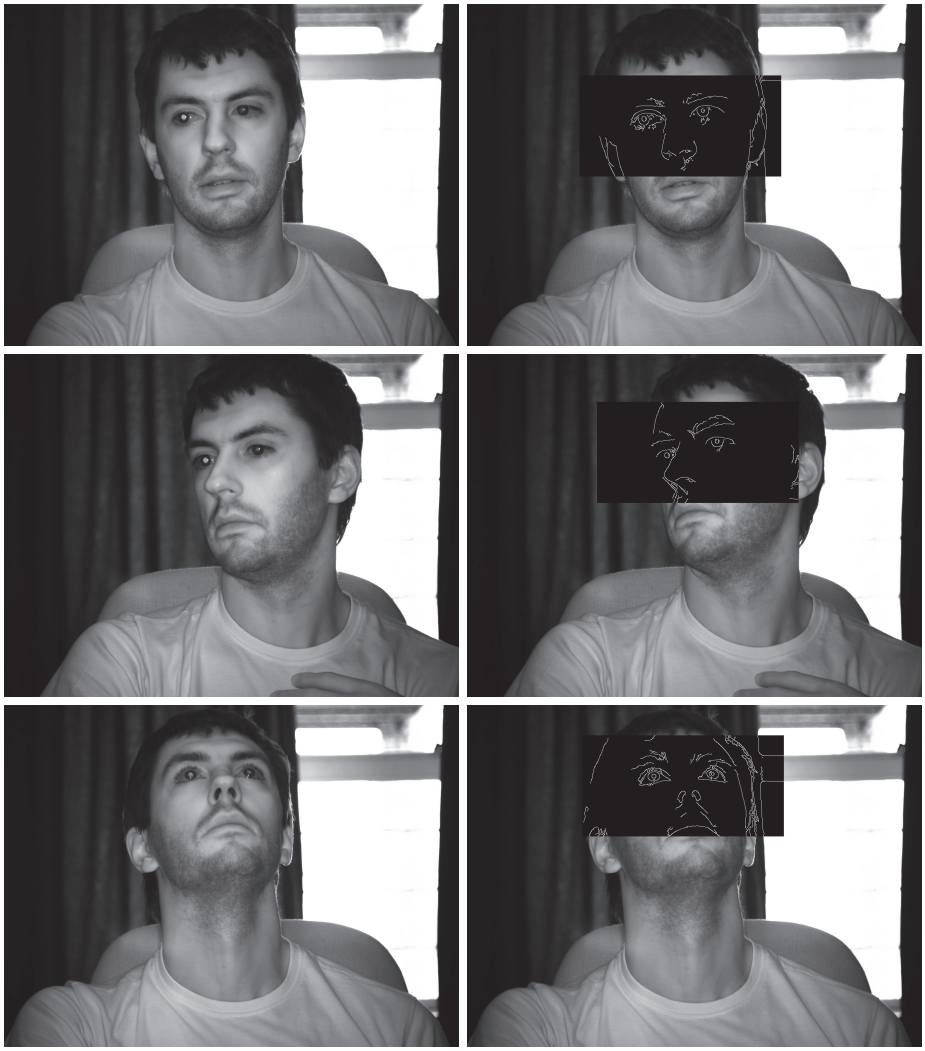

Figure 5.14: A few examples of the original images (left) and its corresponding ROI (right) after smoothing (using a Gaussian kernel) and applying the Canny edge detector. From these examples it is evident that the pupils can be modeled as circles.

left-hand images being the original images and the right-hand images indicating the detected edges in the ROI.

Equations 5.3.1 and 5.3.2 were used in Figure 5.14 to calculate the ROI and it was evident that it was much more efficient to only process the ROI, as opposed to the entire image. From

Figure 5.14 it can be seen that other than the eyes themselves, the eyebrows as well as the nose also produced strong edges in the region of the eyes. It was also evident that modeling the pupils as circles was a valid assumption, but required a relative high level of contrast between the pupil and the cornea of the eye, which was exactly achieved by the bright pupil effect.

Figure 5.15: An example image in which the Canny edge detector did not produce a proper circle for tracking the right eye in the image (i.e. the actual left eye).

The implication of modeling the pupils as circles was that this approach was quite dependent on the bright pupil effect being present. An example of how the lack of the bright pupil effect sometimes made it very difficult to model the pupils as circles, are shown in Figure 5.15. Although the bright pupil effect was in general strong with this subject, in this particular case the head pose made the bright pupil effect dissapear in the right eye in the image and the resulting semi-circle was actually due to the bottom edge of the cornea. In African and Indian subjects where the bright pupil effect was in general weak, it became cumbersome to detect the pupils as circles due to the lack of the required contrast.

Also in outdoor scenarios where the ambient light during daytime is very bright, the pupils would contract to limit the amount of light entering the eye (an evolutionary design) and therefore the area of the pupil that can reflect the NIR light becomes very small and the bright pupil effect would almost be non-existent in any human subject. These are distinct limitations of using the bright pupil effect as a feature for eye tracking, something that is seldomly mentioned in the literature. Nevertheless, in the many situations where the bright

pupil is in fact strong, it would indeed be a very robust feature to track and remains the reason why most current eye tracking systems are based upon it.

Figure 5.16: An example image in which the Hough transform was used to detect circles, after applying Canny edge detection. The left-hand image was the original image, whereas the right-hand image was the detected circles redrawn (as the same size) on the cleared ROI. Note that two of the detected circles were at the locations of the eyes.

Given the detected edges, as illustrated in Figure 5.14, the circles could then be detected using the Hough transform, which is an effective technique for detecting imperfect instances of objects (i.e. circles in this case). An example of how the Hough transform detected circles, is shown in Figure 5.16. The left-hand image shows the original image and the right-hand image shows all the detected circles for this particular case, but redrawn as the same size in the cleared ROI. The reader will notice that there were two detected circles at the locations of the eyes, which corresponds to the circles that were formed by the edges of the bright pupils. The circles due to the bright pupils were consistently detected from frame to frame, as long as reasonable head movements were followed, i.e. no sudden or erratic head movements.

5.3.2 Kalman Filtering Combined with Mean-shift Tracking

The detected and redrawn circles, representing the bright pupils, were ultimately the features that were tracked. At this point, both the Kalman trackers and the mean-shift trackers were oblivious of what they were actually tracking and it might as well have been any underlying object being represented as a circle. After the eyes have been successfully detected and the

ROI has been pre-processed to only contain solid circles, both the Kalman trackers and the mean-shift trackers were initialized at the center of the detected eye windows. These starting points were in almost all cases located close to the centroids of the redrawn circles that corresponded to the actual eyes. Eye tracking was therefore achieved by Kalman predicting the coordinates of the centroid, *"measuring"* the coordinates of the centroid with the mean-shift tracker and finally Kalman correcting the prediction with the *"measurement"*, which was then repeated for each frame.

Experiment 1: Consecutive Eye Tracking Frames

The first experiment that was conducted to determine how well the eye tracker worked, was to count the number of consecutive frames that the system was able to track both eyes. The parameters of the experiment were as follow:

- **Subjects:**
  - Caucasian
  - Strong bright pupil effect
  - Did not wear any glasses
- **Distance from camera:**
  - 1400mm to 1600mm
- **Head movements:**
  - Relative smooth head movements, with a few exceptions
  - Random free head movements
  - Constant head movement
- **Conditions:**
  - Indoors
  - Relatively low ambient light

The subjects were asked to sit in front of the camera and to look directly at the camera, at which point the eyes were detected and the eye tracking initialized. From this point onwards

Table 5.12: The experimental results for counting the number of consecutive frames for which the system was able to successfully track the eyes of subjects.

Experimental run	**Number of consecutive frames**	**Comments**
1	251	
2	128	Sudden change of direction
3	126	Blinking during motion
4	279	
5	225	
6	124	Fast nodding
7	162	
8	128	
9	422	Relative smooth, in-plane head movements
10	354	
11	342	
12	234	
13	126	
14	457	Particular strong bright-pupils
15	230	
Average	**239.2**	

the number of consecutive frames were counted until the geometric measurement determined that the system had lost track of the eyes, which was also manually verified. The subjects were asked to maintain constant head movements, so that the head was never stationary, since it would of course be trivial to track the eyes in such cases. The subjects were also asked to perform relatively smooth head movements, but still keep the movements as natural as possible. There was no pattern in the head movements and the subjects could change direction whenever they felt like it. The subjects were also allowed to vary their distances from the camera, but still had to remain in the range of 1400mm to 1600mm. It was estimated that the frame rate during this experiment was in the region of 15 frames per second (FPS) at full resolution.

The results for 15 runs of this experiment are shown in Table 5.12, with the last column providing a brief explanation of the circumstances under which the system lost track of the eyes in outlying cases. Given that the frame rate was approximately 15 FPS and the average

number of consecutive tracked frames was 239.2, this amounted to an average duration of almost 16 seconds of successful tracking, under constant head movements, before the system lost track. In general, when the system lost track, eye re-detection and tracking initialization took under 2 seconds, which was more a limitation of the hardware setup then the eye detection algorithm. Example frames for two of the eye tracking sequences are shown in Figure 5.17.

It was difficult to exactly quantify these results, since there was no similar baseline for comparison found in the literature with regards to eye tracking. For example, Zhu and Ji [19] tested their eye tracking system under different eye conditions (i.e. open, closed and occluded) and manually marked the locations of the eyes and then counted the number of frames in which their system successfully tracked the eyes. Their measurements therefore give the accuracy percentage for a given amount of frames, but this does not give an idea of how long the system would be able to track the eyes before requiring re-initialization. Although accuracy is indeed an important requirement, from a practical point of view it is also important to have a system that does not too frequently have to be re-initialized, since eye detection is typically much more computationally intensive than tracking.

Figure 5.17: A few consecutive eye tracking example images of a Caucasian female under reasonable head movements. The frame numbers in the sequence are also shown in each image.

The eye tracking system in this research relied on feedback and could quickly detect when tracking had been lost and was also highly accurate in the detection of the eyes, so therefore it was argued that the amount of frames accurately tracked out of some total amount of frames was not a very useful metric to quantify performance, mainly due to the dynamic nature of this system. Another problem with quantifying eye tracking in general was that even if there were similar results available in the literature, there was no universal definition for "reasonable" head movements and it is therefore rather subjective from research to research.

Nevertheless, when considering the results from Table 5.12 the author believes that an average of 239.2 consecutive successful tracking frames, under constant head movements, was a good indication of robustness since in many real-world applications it could be assumed that the subject would only occasionally perform large head movements, in which case the system would have very little difficulty in tracking the eyes for extended periods.

The main situations under which the system lost track occurred when the bright pupil disappeared during head movements, e.g. long blinking periods, sudden change of head direction and occlusion of the eyes due to out-of-plane head rotations. Tracking, in particular the mean-shift tracker, also drifted off in situations where the circles (resulting from frame pre-processing) were incorrectly detected close to the actual eyes (e.g. the eyebrows or forehead) by means of the Hough transform.

Experiment 2: Occlusion Handling

Eye occlusion during tracking was inevitable and it was therefore important that the system would be capable of handling this to a certain extent. The eye tracking approach followed in this research was heavily dependent on the presence of the bright pupils and therefore it was expected that eye tracking would fail if the bright pupils were not present for relative long periods during head movement.

However, the system is in most cases capable of handling brief disappearances of the bright pupils by means of the Kalman prediction phase. If the mean-shift tracker was the only means of tracking that was employed, the tracking window would remain stationary in the pre-processed ROI if the bright pupil disappeared, since there will be no circle present at the particular location of the eye and the search window would therefore be completely black, i.e. the gradient of the pixel intensity distribution would already be zero. By combining Kalman tracking with mean-shift tracking resulted in the locations of the eyes being predicted before

being measured, which implied that the tracking windows would keep on moving in the estimated direction of motion, even if no eyes (i.e. white circles) were measured. If the actual motion of the eyes continued in the same direction after the bright pupils disappeared momentarily, the tracking windows would at least be over part of the actual eyes by the time bright pupils reappeared again, which would enable the mean-shift trackers to easily *"lock"* onto the eyes again. If the bright pupils disappeared for too long relative to the motion, the tracking windows essentially only relied on the Kalman predictions and by the time the bright pupil reappeared, the tracking windows would typically be too far away from the actual eyes for the mean-shift tracker to make the necessary corrections.

Figure 5.18: Four eye tracking example images of a Caucasian female for eye occlusions due to sideway head rotation and blinking. For these examples, it is apparent that the system could handle these types of eye occlusions.

In this light, the aim of this experiment was to determine how well the system could handle eye occlusions and it was argued that the only way that the results could be quantified was with examples. The first example from Figure 5.18 shows four examples of typical occlusion scenarios during a sequence of frames, with the frame number indicated in each image. As can be seen in Figure 5.18, the typical causes of eye occlusions were sideway head rotations and blinking. The system was able to handle these types of occlusions under relative slow

head motion or under faster head motion, provided that the occlusion did not last too long.

For example, consider the top left and bottom left images of Figure 5.18, where the head movements were purely sideway head rotation. In this case the bright pupil located at the direction of rotation eventually dissapeared as the head rotated, which implied a zero measurement. This meant that the tracking window remained relatively stationary and did not drift away too easily. This was due to the bright pupil gradually moving slower and then disappearing, which meant that the Kalman prediction phase did not expect large motion. When the subject then rotated her head back to its original position, the bright pupil reappeared and the mean-shift tracker could then easily *"lock"* onto the eye again.

A similar occlusion situation occurred when the subject blinked while the head remained stationary, as shown in the top right and bottom right images of Figure 5.18. In this case the bright pupils would actually disappear abruptly, but since the head was relatively stationary, large motion was again not predicted by the Kalman prediction phase. The tracking window would therefore initially jump due to the first Kalman prediction right after the bright pupil disappeared, but would then settle and remain relatively stationary close to the eyes. When the subject opened her eyes again, the bright pupils reappeared and the mean-shift trackers could then easily *"lock"* onto the eye.

Figure 5.19: An example of how the system was capable of recovering from a relatively slow eye closure during head movements. In the left image the eyes were closed and the eye tracker mistakenly tracked the eyebrows. In the right image the eyes were opened again and the bright pupils re-appeared, which enabled the system to again *"lock"* onto the eyes.

The system was also capable of handling relatively short eye blinks during head motion as shown in Figure 5.19. In this case the subject were rotating her head upwards and was then requested to perform a short blink, simulating involuntary blinking. The bright pupils would then abruptly disappear but because there was motion before this disappearance, the Kalman prediction would let the tracking window jump in the direction of the expected motion (left image of Figure 5.19). In this case the predicted location was close enough to where the bright pupils actually reappeared when the eyes were opened again, which enabled the mean-shift tracker to *"lock"* onto the eyes (right image of Figure 5.19).

Experiment 3: Eye Tracking with Glasses

The final experiment that was conducted, was to determine how well the system could track the eyes when a subject wore glasses. The number of consecutive frames in which the eyes could be successfully tracked was again considered to be the most appropriate metric to quantify the performance. This experiment was therefore essentially the same as the first experiment, but with the following differences:

- Only a single subject were tested, in which case he wore reading glasses in half of the sequences and sunglasses in the other half of the sequences.
- Head movements were much more restricted and slower when compared to the first experiment.

The results for the number of consecutive frames tracked for 10 runs of this experiment are shown in Table 5.13. When considering the average number (125.7) of consecutive tracked frames from Table 5.13, this amounts to just over 8 seconds of consecutive tracking at 15 FPS. This is not a particularly impressive result, especially when taking into account that the head movements were much more constrained when compared to the first experiment.

The eye detection phase still accurately initialized the eye tracking phase when the subject wore glasses and in cases which the subject's head remained essentially stationary, the system at least had no difficulty in tracking the eyes. Examples for which the system was able to accurately track the eyes under relative stationary conditions are shown in Figure 5.20 and Figure 5.21 for reading glasses and sunglasses, respectively. Note in Figure 5.21 that the sunglasses actually seemed to enhance the bright pupil effect.

Table 5.13: The experimental results for counting the number of consecutive frames that the system was able to successfully track the eyes of a subject wearing glasses.

Experimental run	**Number of consecutive frames**
1	198
2	47
3	101
4	244
5	94
6	185
7	70
8	74
9	160
10	84
Average	**125.7**

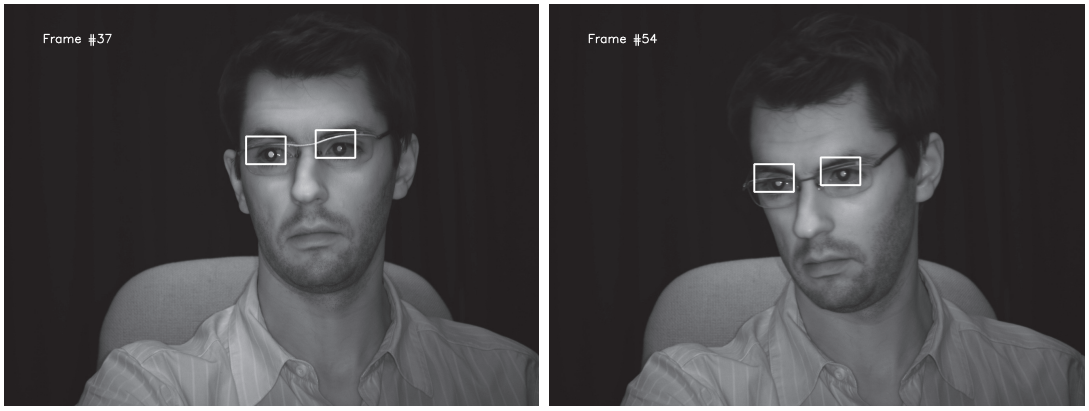

Figure 5.20: Two examples of how the system was able to track the eyes when a Caucasian subject wore normal reading glasses. The system was able to track the eyes reasonably well, but much worse when compared to the subject not wearing glasses.

As soon as the subject started with head movements, the eye tracking windows would typically "jump" around and would be distracted with relative ease. Upon further inspection of the processed ROI it was found that other reflections due to the NIR light from the glasses itself, would exhibit similar behavior than the actual bright pupils and were therefore prone to be

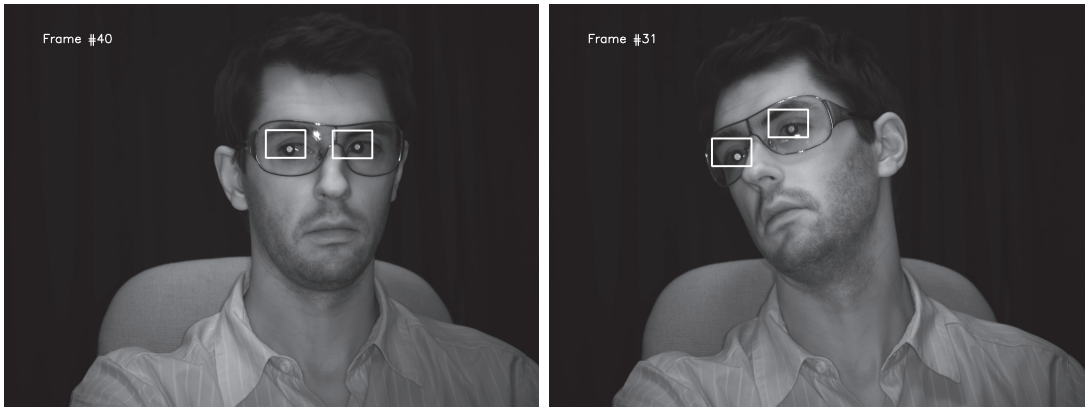

Figure 5.21: Two examples of how the system was able to track the eyes when a Caucasian subject wore sunglasses. The system was able to track the eyes only when the subject's head was stationary, and in general performed much worse than for reading glasses. It was interesting to note that in this specific case, the sunglasses actually enhanced the bright pupil effect.

mistaken for eyes. Examples for which the system failed to track the eyes when the subject wore glasses, are shown in Figure 5.22 and Figure 5.23 for reading glasses and sunglasses, respectively.

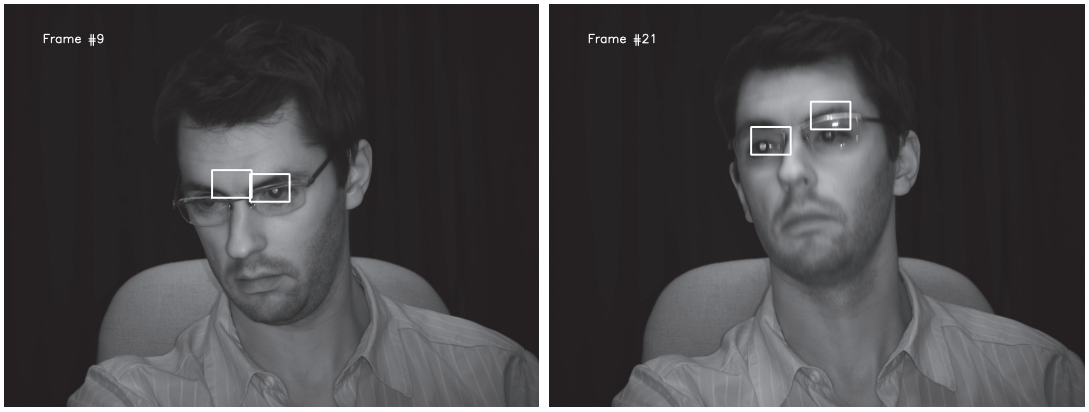

Figure 5.22: Two example of how the system failed to track the eyes when a Caucasian subject wore normal reading glasses. It was observed that the reflection from glasses themselves were a significant source of noise, which was the main reason for failure.

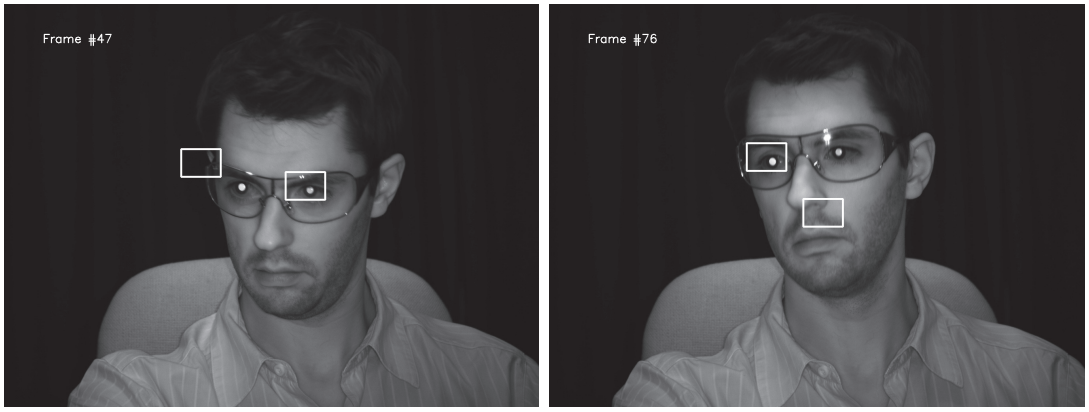

Figure 5.23: Two example of how the system failed to track the eyes when a Caucasian subject wore sunglasses. It was again observed that the reflection from glasses themselves were a significant source of noise, which was particularly severe for this pair of sunglasses.

These results may perhaps not be exactly what were hoped for, but does not necessarily mean that the eyes cannot be tracked when a subject wears glasses. It simply implies that the eyes should perhaps be modeled as a slightly more complex object, rather than just circles. Since the bright pupil effect still remained strong in this particular case, it might be worth while to consider incorporating adaptive thresholding, but this will be left as future work.

5.3.3 Discussion

The final approach used for eye tracking, as discussed in this section, was the result of numerous informal trial-and-error experiments performed to find a way to accurately segment the eyes for their background. Provided that the bright pupil effect was present, it was found to be a very robust feature to track, in particular for Caucasian subjects for which the bright pupil effect was in general strong. As necessary for the mean-shift tracker, the bright pupils were segmented by first detecting the edges in the ROI, using the Canny edge detector. Secondly the pupils were modeled as circles and the Hough transform was used to detect circles from the edges resulting from Canny edge detection. Finally the ROI was cleared and the detected circles were redrawn, which were then used for taking “measurements" (via the mean-shift tracker) for the Kalman trackers.

The developed eye tracking algorithm was able to accurately and with relative ease, track

the eyes for frontal face head movements (perhaps more appropriately described as head translations) for extended periods, provided that the head movements were not too fast and erratic. In fact, the performance of the eye tracking algorithm actually exceeded expectations and could handle much faster head movements than anticipated. The system also handled occlusions, due to out-of-plane head rotations, well and could in such cases still track the eyes even when the bright pupils were not present for short periods. Eye occlusions due to blinks could easily be handled when the subject's head was stationary and could also be handled during head movements, provided that the blinks were not too long.

Since only the ROI and not the entire image was processed for each frame, the eye tracking algorithm was executed at high speed to enable real-time performance, which is of course an important requirement for real-world applications. The relatively low computational requirements for only processing the ROI also means that it would be possible for this eye tracking algorithm to be implemented on an embedded hardware platforms for applications in which it would not be possible to use a normal PC.

Although the developed eye tracking system was capable of robustly tracking the eyes, the following limitations were identified:

- The robustness of the system was dependent on the bright pupil effect to provide enough contrast to detect the edges of the pupil as a circle. As mentioned before, it was observed that the bright pupil effect varies among different ethnic groups, and it was in general weak in African and Indian subjects, which limits the usefulness of the system for these groups of people.
- It was observed that the NIR illumination had a tiring effect on the eyes after constant exposure to it for an extended period of time. The intensity of the NIR illumination was measured at DPSS and found it to be within current eye safety regulations. However, in recent years many questions were raised on the constant exposure to NIR illumination and its effect on the human eye, with the final answer being inconclusive at the time of writing this thesis. Division 6 of the International Commission on Illumination (CIE) formed a technical committee (TC6-64) to provide new safety regulations for exactly this scenario and when these regulations are published, the amount of NIR illumination used for this research might potentially not be considered safe over the long run.
- A related limitation that was observed, was that it appeared that there was still a slight

mismatch between the camera's sensor and the 780nm NIR illumination. To enhance the bright pupil effect and limit the amount of daylight entering the camera's lens, an optical bandpass filter was used. This worked well, but in situations of very low ambient light the exposure time of the camera's sensor had to be significantly increased. The implication of an increased exposure time was that the resulting images would easily blur during motion, which in turn made it very difficult to detect the pupil edges and ultimately track the eyes. A likely solution to this problem is to use a camera that is specifically designed for the NIR illumination. By using such a camera, it might actually result in the first limitation being less profound and potentially solve this particular limitation as well as the second limitation, since such a sensor would be more sensitive to 780nm illumination, which in turn implies that less 780nm illumination might be required to achieve the same effect.

- The system struggled a lot with tracking the eyes when the subject wore glasses, which was mainly due to NIR light being reflected off the glasses, which appeared very similar than the actual bright pupil.

5.4 APPLICATIONS OF EYE TRACKING

5.4.1 Driver Fatigue Detection

In recent years driver fatigue related accidents, especially in the heavy vehicle industry, have become a major concern and highlighted a need for some system that would be capable of monitoring the fatigue level of a driver and pro-actively warn the driver in advance or perhaps some control room in critical conditions. A very important requirement for such a system is that it has to be completely non-intrusive and should not disturb the driver in any way. For this reason monitoring the driver by means of a camera would be an ideal method for monitoring fatigue. In particular by tracking the eyes, various accurate fatigue metrics can be calculated, such as the percentage of eye closure for certain period of time or the blink duration in general.

The ambient light will vary drastically for driving conditions, and it is therefore sensible to use an appropriate camera with active infrared illumination. The theory is that the active infrared light is not visible to the driver and should therefore not interfere with the task of driving. However, following this research, the issue can be raised that even though the infrared

light may not be visible to the driver it might still have a tiring effect on the eyes, which would then ultimately defy the whole purpose. At this point in time, monitoring the driver with a camera is in many circles considered an invasion of privacy and has therefore still not been widely accepted. Nevertheless, it still remains an accurate method for monitoring fatigue and have been proposed by numerous authors including [67], [68], [69], [50], [51] and [48]. Commercial camera-based fatigue detection systems include Smart Eye Pro 2 by Smart Eye AB [70] and the DSS system by Seeing Machines [71].

5.4.2 A User Interface for the Severely Disabled

Another popular application of eye tracking systems, is to provide severely disabled people with a computer interface to interact with the world. These people are typically not mentally disabled, but physically disabled in which case it is impossible from them to use their body parts to physically control a pointing device such as a computer mouse, for example. Therefore an eye tracking, but more specifically a gaze tracking system can be used to pinpoint the position on a screen at which the user is staring. For example, a keyboard can be drawn on a screen and the user would then stare at a specific key, which can then be virtually clicked by blinking. The most frequent use for these eye tracking systems is to move the cursor on a computer screen, based on where the user is looking on the screen. Authors that proposed such systems include [72], [38] and [73]. Eyegaze Edge by LC Technologies [74] is a commercial eye gaze tracking system, specifically designed for physically disabled people.

5.4.3 Human Behavior Research

Perhaps one of the first applications of eye tracking was for research in a wide range of human behavior studies. Some of the research fields that use eye tracking systems include:

- Cognitive and behavioral psychology, e.g. how experts and novices interpret dynamic stimuli, eye gaze behavior in children with dyslexia, etc.
- Medical research, e.g. estimating depth perception for improving minimal invasive surgery, lung nodule detection, etc.
- Ophthalmology and vision science, e.g. out-of-focus blurring for displays to improve visual comfort, determining the regions of interest in videos , etc.

- Marketing research, e.g. how people scan websites on the Internet, how people perceive advertisements, etc.

Very popular commercial eye tracking products developed specifically for these types of research fields are the T60 and T120 eye trackers from Tobii [75].

5.5 CONCLUDING REMARKS

In this chapter the detailed performance results of the eye tracking system were presented, which started out with results for achieving the bright pupil effect with the developed hardware. Since the bright pupil effect was a critical aspect of both the eye detection and tracking phase, it was important to achieve a strong bright pupil effect. As was shown, a strong bright pupil effect was indeed obtained, but it was found to be inconsistent among people from different ethnic groups, which is perhaps a limiting factor of this approach.

Following the results of the bright pupil effect, the results for the eye detection phase was presented. These results indicated that the proposed eye detection approach was very robust in detecting the eyes under various head poses as well as when the subject wore glasses. The conclusion that was drawn, was that AdaBoost is very effective in classifying eyes and could also generalize well on subjects that were not used during the training phase of the classifier. However, although the AdaBoost classifier performed well on unseen subjects, it was probably not good enough for *all* unseen subjects and therefore some form of calibration would be necessary for practical purposes.

Finally the eye tracking results were presented in the form of three conducted experiments. The results from these experiments have shown that the system was very effective in tracking the eyes for reasonable free head movements and was also capable of handling occlusions in various forms. The system did however struggle to track the eyes when the subject wore glasses, but it is believed that this can be improved in future work.

The chapter also presented some applications of eye tracking, which was by no means exhaustive, but does illustrate the usefulness of an accurate eye tracking system in many real-world problems. Therefore the work presented in this thesis may not merely be of academic interest, but can also prove to be valuable in practice.

Chapter 6

CONCLUSION

Although the aim of the academic research presented in this thesis was concerned with improving the machine vision task of eye tracking, significant effort went into developing the hardware platform to enable this research. This research therefore had a very strong practical aspect and solved a number of real-world problems in the process of developing the eye tracking system. This implies that this system can potentially be very useful for a number of other research activities and perhaps even become a part of a commercial product.

The results from the eye detection phase illustrated just how robust the application of the bright/dark pupil effect can be in detecting the eyes. By combining the bright/dark pupil effect with a strong classifier such as AdaBoost and using geometric constraints, the occurrences of false positives were extremely low. However, the classification results also revealed that there was significant inter-subject variability even for the relatively small dataset of 17 subjects, which implies that for a practical eye tracking system, some form of subject calibration would be inevitable. Another concern was that the bright pupil effect varied significantly among ethnic groups, being particularly weak for African and Indian subjects. This of course raised the question of just how applicable bright/dark pupil based eye detection (and in effect eye tracking) would be for the general public, especially in an ethnic diverse country such as South Africa. Nevertheless, when the bright/dark pupil effect is present, there is arguably not a more efficient and robust method for eye detection.

The approach followed for eye tracking was also heavily dependent on the appearance of the bright pupil, but the way in which the eyes were tracked decoupled the Kalman tracking and mean-shift tracking from what was actually being tracked. This is a nice feature, which implied that the way in which the eyes were modelled can be changed with relative ease,

without affecting the actual eye tracking procedures. Although the proposed approach for eye tracking could robustly track the eyes as long as the bright pupils were present, there were two main concerns.

The first concern that was observed, was that constant exposure to the NIR illumination for an extended period of time (2 to 3 hours) had a tiring effect on the eyes. As mentioned before, this might be due to additional NIR illumination used to compensate for the slight mismatch of the camera's sensor and the particular wavelength of the NIR light. If the current research performed by the CIE shows that long term exposure to NIR light can be harmful, this might significantly limit the use of any active IR-based eye tracking system (almost all commercial eye tracking systems are based upon active IR light).

The second concern was that the system struggled a lot with tracking the eyes when the subject wore glasses, even though the sunglasses used for the experiments actually enhanced the bright pupil effect. This was mainly due to the reflections that the glasses made (which were similar to bright pupils), as well as the strong edges resulting from the frame of the glasses. This is a common problem with current eye tracking systems and some more research is still necessary to compensate for this.

Overall the performance of the system was very satisfying and all of the initial goals were achieved and even exceeded, with the exception of eye tracking when the subject wore glasses. However, there is always room for improving the system, but this is left as future work as discussed next.

6.1 FUTURE WORK

Due to the difficult nature of eye tracking under practical conditions, there are still a number of unresolved issues and aspects than can optimized, including the following:

- A better understanding of the inter-subject variability for the bright pupil effect is required, such as at what wavelengths of NIR light the reflection from the pupil is the strongest. The optimal wavelengths might well be significantly different for people, depending on their ethnic group.
- Currently the feature vector for eye classification is just the raw pixel intensities, which becomes cumbersome for very large datasets. As an optimization, some form of feature extraction (e.g. SURF) or compact descriptors (e.g. Fourier descriptors) can be used

to significantly reduce the size of the feature vectors.

- The combination of Kalman tracking with mean-shift tracking was found to be very robust for tracking, but it would probably even be more robust if some other feature for tracking (besides the bright pupil) could additionally be incorporated. It would also be sensible to use a completely different eye tracking technique for comparison. Currently particle filters for tracking have become quite popular and it would be interesting to see its performance for eye tracking.

- The system struggled to track the eyes when a subject wore glasses, and considering a different approach for handling glasses would definitely be worth while.

Bibliography

[1] R. Coetzer, "Driver fatigue detection : A literature survey," University of Pretoria (for Eskom), Tech. Rep., 2009.

[2] D. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 478–500, 2010.

[3] R. Coetzer and G. Hancke, "Driver fatigue detection : A survey," in *AFRICON, 2009. AFRICON '09.*, September 2009, pp. 1 –6.

[4] R. Coetzer, "Driver fatigue detection based on eye tracking (wip paper)," in *SATNAC 2010*, September 2010.

[5] R. Coetzer and G. Hancke, "Eye detection for a real-time vehicle driver fatigue monitoring system," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, june 2011, pp. 66 –71.

[6] T. Hutchinson, "Human-computer interaction using eye-gaze input," *IEEE Transactions on systems, man, and cybernetics*, vol. 19, no. 6, pp. 1527–1534, November 1989.

[7] ——, "Eye movement detection with improved calibration and speed," Patent US 4 950 069, 08 21, 1990.

[8] Y. Ebisawa and S. Satoh, "Effectiveness of pupil area detection technique using two light sources and image difference method," *Proceedings of the Annual Conference on Engineering in Medicine and Biology*, vol. 15, no. 3, pp. 1268–1269, October 1993.

[9] Y. Ebisawa, "Improved video-based eye-gaze detection method," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, no. 4, pp. 948–955, 1998.

[10] C. Morimoto, D. Koons, A. A., and M. Flickner, "Pupil detection and tracking using multiple light sources," *Image and vision computing*, vol. 18, no. 4, pp. 331–335, March 2000.

[11] K. Nguyen, C. Wagner, D. Koons, and M. Flicker, "Differences in infrared bright pupil response of human eyes," *Proceedings ETRA 2002: Eye Tracking Research and Applications Symposium*, pp. 133–138, March 2002.

[12] F. Mulvey, A. Villanueva, D. Sliney, R. Lange, S. Cotmore, and M. Donegan, "D5.4 Exploration of safety issues in eyetracking," Communication by Gaze Interaction (COGAIN), Tech. Rep., 2008.

[13] V. Vapnik, *The nature of statistical learning theory*. Springer: New York, 1995.

[14] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[15] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, Müller, E. Säckinger, P. Simard, and V. Vapnik, "Learning algorihms for classification: A comparison on handwritten digit recognition," *Neural Networks*, pp. 261–276, 1995.

[16] H. Drucker, D. Wu, and V. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[17] S. Bengio and J. Mariéthoz, "Learning the decision function for speaker verification," *IEEE internation conference on acoustics, speech and signal processing - proceedings*, vol. 1, pp. 425–428, 2001.

[18] B. Heisele, P. Ho, and T. Poggio, "Face recognition with support vector machines: global versus component-based," *8th International conference on computer vision*, vol. 2, pp. 688–694, July 2001.

[19] 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*, vol. 98, pp. 124–154, 2005.

[20] K. Bennett and O. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods Software*, vol. 1, pp. 23–34, 1992.

[21] R. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, pp. 197–227, 1990.

[22] X. Li, L. Wang, and E. Sung, "Adaboost with svm-based component classifiers," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 5, pp. 785–795, August 2008.

[23] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and computation*, vol. 121, pp. 256–285, 1995.

[24] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Science*, vol. 55, no. 1, pp. 119–139, August 1997.

[25] H. Drucker and C. Cortes, "Boosting decision trees," *In advances in neural information processing systems 8*, pp. 479–485, 1996.

[26] J. Quinlan, "Bagging, boosting and c4.5," *In proceedings of the thirteenth national conference on artificial intelligence*, pp. 725–730, 1996.

[27] L. Breiman, "Arcing classifiers," *The annals of statistics*, vol. 3, no. 26, pp. 801–849, 1998.

[28] R. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 5, no. 26, pp. 1651–1686, October 1998.

[29] J. Friedman, H. T., and R. Tibshirani, "Special invited paper: Additive logistic regression: a statistical view of boosting," *The annals of statistics*, vol. 2, no. 28, pp. 337–407, 2000.

[30] G. Welch and G. Bishop, "A introduction to the Kalman filter," *UNC-Chapel Hill, TR 95-041*, July 2006.

[31] H. Sorenson, "Least-squares estimation: from Gauss to Kalman," *IEEE Spectrum*, vol. 7, pp. 63–68, July 1970.

[32] R. Brown and P. Hwang, *Introduction to random signals and applied Kalman filtering*, 2nd ed. John Wiley and Sons Inc, 1992.

[33] O. Jacobs, *Introduction to control theory*, 2nd ed. Oxford University Press, 1993.

[34] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, March 2004.

[35] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with application in pattern recognition," *IEEE Transactions on infomation theory*, vol. 21, no. 1, pp. 32–40, January 1975.

[36] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.

[37] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[38] K. Kim and R. Ramakrishna, "Vision-based eye-gaze tracking for human computer interface," *IEEE SMC99 Conference Proceedings 1999 IEEE International Conference on Systems Man and Cybernetics*, pp. 324–329, 1999.

[39] D. Young, H. Tunley, and R. Samuels, "Specialised Hough transform and active contour methods for real-time eye tracking," School of Cognitive and Computing Sciences, Tech. Rep., 1995.

[40] A. Yuille, P. Hallinan, and D. Cohen, "Feature extraction from faces using deformable templates," *Internationl journal of computer vision*, vol. 8, no. 2, pp. 99–111, 1992.

[41] G. Feng and P. Yuen, "Variance projection function and its application to eye detection for human face recognition," *Pattern recognition letters*, vol. 19, pp. 899–906, 1998.

[42] S. Kawato and J. Ohya, "Real-time detection of nodding and head-shaking by directly detecting and tracking the "between-eyes"," in *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, 2000, pp. 40–45.

[43] ——, "Two-step approach for real-time eye tracking with a new filtering technique," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 2, 2000, pp. 1366–1371.

[44] C. Morimoto, T. Santos, and A. Muniz, "Automatic iris segmentation using active near infra red lighting," *Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on*, pp. 37–43, October 2005.

[45] K. Grauman, M. Betke, J. Gips, and G. R. Bradski, "Communication via eye blinks - detection and duration analysis in real time," in *Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. 1010–1017.

[46] F. Samaria and S. Young, "Hmm-based architecture for face identification," *Image Vision Computing*, pp. 537–543, 1994.

[47] J. Huang and H. Wechsler, "Eye detection using optimalwavelet packets and radial basis functions (rbfs)," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 13, no. 7, pp. 1009–1026, 1999.

[48] T. Ishikawa, S. Baker, I. Matthews, and T. Kanade, "Passive driver gaze tracking with active appearance models," in *In Proceedings of the 11th World Congress on Intelligent Transportation Systems*, October 2004.

[49] D. Hansen, J. Hansen, M. Nielsen, A. Johansen, and M. Stegmann, "Eye typing using markov and active appearance models," in *Applications of Computer Vision, 2002. (WACV 2002). Proceedings. Sixth IEEE Workshop on*, 2002, pp. 132 – 136.

[50] Q. Ji and X. Yang, "Real-time eye, gaze and face pose tracking for monitoring driver vigilance," *Real Time Imaging*, vol. 1, no. 8, pp. 357–377, Febuary 2002.

[51] Q. Ji, Z. Zhu, and P. Lan, "Real-time nonintrusive monitoring and prediction of driver fatigue," *IEEE transactions on vehicular technology*, vol. 53, no. 4, pp. 1052–1068, July 2004.

[52] K. Talmi and J. Liu, "Eye and gaze tracking for visually controlled interactive stereoscopic displays," *Signal Processing: Image Communication*, vol. 14, no. 10, pp. 799–810, 1999.

[53] T. Morris, P. Blenkhorn, and F. Zaidi, "Blink detection for real-time eye tracking," *Journal of Network and Computer Applications*, vol. 25, no. 2, pp. 129–143, 2002.

[54] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th international joint conference on Artificial intelligence*, vol. 2, 1981, pp. 674–679.

[55] A. Haro, M. Flickner, and I. Essa, "Detecting and tracking eyes by using their physiological properties, dynamics, and appearance," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, 2000, pp. 163–168.

[56] D. Hansen and A. Pece, "Eye tracking in the wild," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 155–181, 2005.

[57] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[58] A. Vision Technologies, "Prosilica GC1380 data sheet," Taschenweg 2a, 07646 Stadtroda, Germany, Accessed January 2010. [Online]. Available: http://www.alliedvisiontec.com

[59] ——, "Prosilica GC1600 data sheet," Taschenweg 2a, 07646 Stadtroda, Germany, Accessed January 2010. [Online]. Available: http://www.alliedvisiontec.com

[60] ——, "Prosilica GE1900 data sheet," Taschenweg 2a, 07646 Stadtroda, Germany, Accessed January 2010. [Online]. Available: http://www.alliedvisiontec.com

[61] O. Opto Semiconductors, "SFH4232 data sheet," Accessed Febuary 2010. [Online]. Available: http://catalog.osram-os.com/

[62] A. Vision Technologies, "Prosilica GC1380 user manual," Taschenweg 2a, 07646 Stadtroda, Germany, Accessed January 2010. [Online]. Available: http://www.alliedvisiontec.com

[63] National Semiconductor, "WEBENCH Designer for LED Drivers," Accessed July 2011. [Online]. Available: http://www.national.com/en/led/index.html

[64] Microchip, "PICKit 2 User Guide," Accessed July 2011. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/51553e.pdf

[65] ——, "PIC16F887 datasheet," Accessed July 2011. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/41291e.pdf

[66] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 1, pp. 511–518, 2001.

[67] D. Dinges, M. Mallis, and J. Powell, "Evaluation of techniques for ocular measurement as an index of fatigue and the basis for alertness management," *Department of transport safety*, vol. 808, no. 762, April 1998.

[68] A. Albu, B. Widsten, T. Wang, J. Lan, and J. Mah, "A computer vision-based system for real-time detection of sleep onset in fatigued drivers," *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 25–30, June 2008.

[69] M. Eriksson and N. Papanikolopoulos, "Driver fatigue: a vision-based approach to automatic diagnosis," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 6, pp. 399–413, December 2001.

[70] S. Eye. (2011, October) Smart eye pro 2. [Online]. Available: http://www.smarteye.se/products/vehicle-eye-trackers

[71] S. Machines. (2011, October) Driver State Sensor (DSS). [Online]. Available: http://www.seeingmachines.com/product/dss/

[72] A. De Santis and D. Iacoviello, "Robust real time eye tracking for computer interface for disabled people," *Comput. Methods Prog. Biomed.*, vol. 96, pp. 1–11, October 2009.

[73] A. Kaufman, A. Bandopadhay, and B. Shaviv, "An eye tracking computer user interface," in *Virtual Reality, 1993. Proceedings., IEEE 1993 Symposium on Research Frontiers in*, October 1993, pp. 120–121.

[74] L. Technologies. (2011, October) Eyegaze edge. [Online]. Available: http://www.eyegaze.com/content/assistive-technology

[75] Tobii. (2011, October) Tobii T60 and T120. [Online]. Available: http://www.tobii.com/

Appendix A

SERIAL COMMUNICATION PROTOCOL

The serial communication protocol between the PC and the embedded system (i.e. the PIC16F887) is presented in this Appendix. This very simple 4-byte protocol was developed under the assumption that the communication will be error free in the controlled lab environment. Therefore in practice this protocol will not be robust enough for reliable communication. The small amount of commands required was also not enough to justify an elaborate protocol. The general packet structures of a command and a response is shown in Table A.1 and Table A.2, respectively.

Table A.1: The general packet structure for commands from the PC to the embedded system.

Packet fields	$< STX >$	$< Command >$	$< Parameters >$	$< ETX >$
Value	0x02	See Table A.3	See Table A.3	0x03

Table A.2: The general packet structure for a response from the embedded system to the PC.

Packet fields	$< STX >$	$< Response >$	$< Command\ Type >$	$< ETX >$
Value	0x02	ACK = 0x06 NACK = 0x15	Command value as defined in Table A.3	0x03

The commands and their parameters that have been implemented as well as a description of each command can be found in Table A.3.

The embedded system will always respond when issued with a command, in the form of either an acknowledgement (ACK) or a negative acknowledgment (NACK). A NACK response will only be issued when the embedded system does not recognize the given command. The response to a command will be of the form as described in Table A.2, with the *Command*

Table A.3: The available commands in the simple 4-byte serial protocol

Command Name	**Value**	**Parameters**	**Description**
Ping	0x50 / 0x70 ('P'/'p')	None (Set to 0x00)	Used to check connectivity between the PC and the embedded system.
Detect Eyes	0x42 / 0x62 ('B'/'b')	None (Set to 0x00)	Used to capture a bright and dark pupil image pair for eye detection.
Set Inner LEDs	0x49 / 0x69 ('I'/'i')	OFF = 0x00 ON = 0x01	Used to switch the inner set of LEDs on and off.
Set Outer LEDs	0x4F / 0x6F ('O'/'o')	OFF = 0x00 ON = 0x01	Used to switch the outer set of LEDs on and off.
Capture Video	0x43 / 0x63 ('C'/'c')	STOP = 0x00 START = 0x01	Used to continuously capture images for eye tracking.

Type indicating which command has been ACKed or NACKed.

Appendix B

PROSILICA GC1380 TRIGGERING

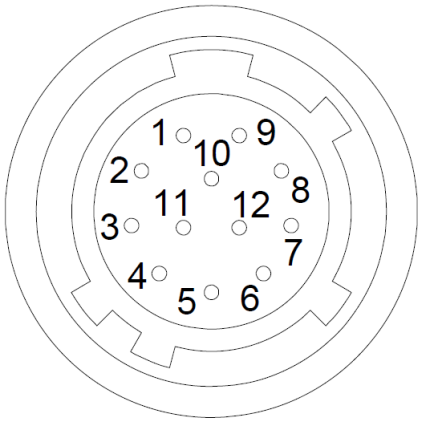

Figure B.1: The general purpose I/O port of the Prosilica GC1380 camera (see the technical manual [62]).

The Prosilica GC1380 camera provides a general purpose I/O (GPIO) port which allows for additional control of the camera, in particular external triggering for image acquisition. On the camera side a Hirose HR10A-10R-12PB connector is used and in order to get access to this GPIO port, the mating connector that was used, was the Hirose HR10A-10P-12S.

The pin out of the GPIO can be found in Table B.1. From this table it can be seen that there are two triggering inputs, one opto-isolated input and one non-isolated input. Since the fastest possible triggering time was required and the controlled lab environment has very little noise, the non-isolated triggering input was the best suited. As a result the Sync 2 Input (camera GPIO pin 11) was connected to port RD1 (PIC16F887 pin 20) and the Sync 2 Output (camera GPIO pin 12) was connected to port RD0 (PIC16F887 pin 19), both through a buffer to protect the PIC microcontroller.

Table B.1: The general purpose I/O port pin out of the Prosilica GC1380 (see the technical manual [62])

Pin	Function
1	Power Ground
2	12V Power
3	Sync Input 1 - isolated
4	Sync Output 1 - isolated
5	Isolated Ground
6	Video Iris
7	DNC
8	RS-232 TXD
9	RS-232 RXD
10	Signal ground
11	Sync Input 2 - non-isolated
12	Sync Output 2 - non-isolated

The camera also had to be correctly configured over the GigE interface to enable external triggering. Table B.2 shows all the relevant parameters that had to be configured, as well as their values for this particular setup.

Table B.2: The relevant Prosilica GC1380 camera parameter configuration for externa triggering.

Parameter	Value
AcquisitionMode	SingleFrame
FrameStartTriggerMode	SyncIn2
FrameStartTriggerEvent	EdgeRising
SyncOut2Invert	Off
SyncOut2Mode	FramerTriggerReady

Once the the camera has been correctly configured for external triggering, the PIC microcontroller has to provide the necessary timing signals as shown in Figure B.2. The Sync 2 Output pin of the camera produces the *Trigger Ready* signal, which the PIC microcontroller will first read and then subsequently produce the *User Trigger* signal to initiate the sensor exposure. Once image acquisition have been completed, the image will automatically be transmitted to the PC over the GigE interface, where the camera's software development kit (SDK) will present the image to the developed application software.

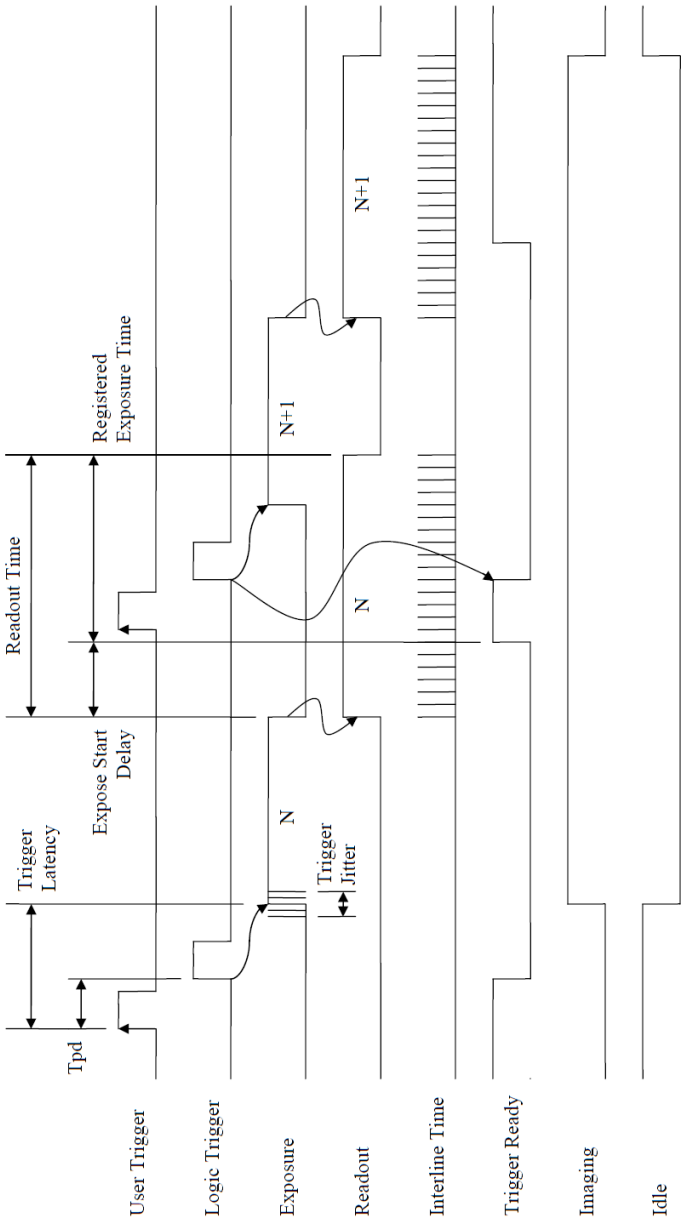

Figure B.2: The timing diagram necessary for external triggering of the Prosilica GC1380 camera (see the technical manual [62]).