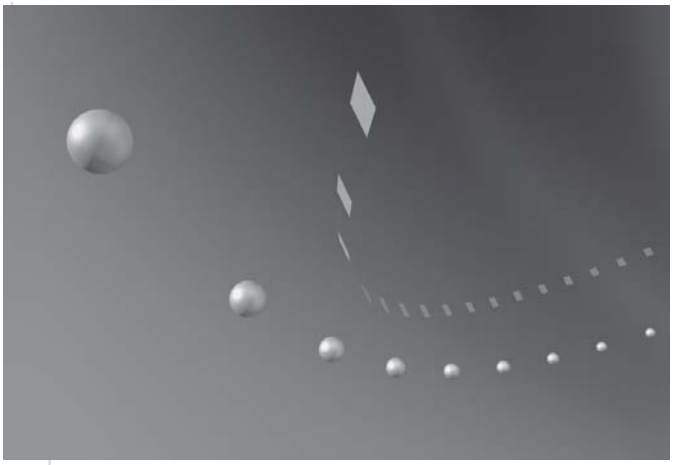
VTI notat 12A-2008
Published 2008

www.vti.se/publications



# Automatic analysis of eye tracker data

Martin Bergstrand

vti
FINDING A BETTER WAY

**Automatisk analys av ögonrörelsedata**

av Martin Bergstrand
VTI
581 95 Linköping

## Sammanfattning

En persons ögonrörelser är en intressant faktor att studera i olika forskningsområden där uppmärksamheten är viktig, till exempel vid bilkörning. År 2004 invigdes Simulator III på Statens väg- och transportforskningsinstitut (VTI). Simulator III är den tredje generationen av körsimulatorer som utvecklats på VTI. I simulatorn finns ett kamerabaserat system för ögonrörelsemätning. För att vara användbart måste rådata från systemet analyseras och koncentreras till ett antal mätvärden som kan användas i forskningen på VTI.

I detta examensarbete utvecklas metoder för att analysera data från ögonrörelsemätnings-systemet och omvandla det till någonting mer användbart.

Ett världskoordinatsystem definieras, vilket kopplar samman ögonrörelsemätningssystemet med verkligheten. Ett antal mätvärden samlas in, framförallt från ISO- och SAE-standards, för att användas som utdata från analysen. Slutligen utvecklas en applikation som utför analysen.

Applikationen läser in data från ögonrörelsemätningssystemet och simulatorn, analyserar data, samt producerar ett antal mätvärden som kan användas i forskningen på VTI.

**Automatic analysis of eye tracker data**

by Martin Bergstrand
VTI (Swedish National Road and Transport Research Institute)
SE-581 95 Linköping Sweden

## Summary

The movements of a person’s eyes are an interesting factor to study in different research areas where attention is important, for example when driving. In 2004 the Swedish National Road and Transport Research Institute (VTI) introduced Simulator III – their third generation of driving simulators. Inside Simulator III a camera based eye tracking system is installed that records the eye movements of the driver. To be able to be used, the raw data from the eye tracking system needs to be analyzed and concentrated into a number of measures relevant for the research at VTI.

This thesis presents methods to analyze the data from the eye tracker and to transform it into something more useful. A world coordinate system is set up to connect the eye tracking system with the real world in a consistent way. A set of measures is collected, mainly from ISO and SAE standards, to be used as output from the analysis. Finally an application is developed for performing the analysis. The application reads the data from the eye tracker and the simulator, analyzes the data, and outputs a set of eye movement measures usable for the researchers at VTI.

## Quality review

Review seminar was performed on 29 February by Johan Östergaard at Linköping University, Campus Norrköping. Martin Bergstrand has made alterations to the final manuscript of the report. The research director of the project manager Lena Nilsson examined and approved the report for publication.

## Kvalitetsgranskning

Granskningsseminarium har genomförts fredagen den 29 februari 2008 av Johan Östergaard vid Linköpings universitet, Campus Norrköping. Martin Bergstrand har genomfört justeringar av slutligt rapportmanus. Projektledarens närmaste chef Lena Nilsson har därefter granskat och godkänt publikationen för publicering.

VTI dnr: 2007/0253-26

# Automatic Analysis of Eye Tracker Data

Martin Bergstrand

1st March 2008

VTI notat 12A-2008

# Abstract

The movement of a persons eyes is an interesting factor to study in different research areas where attention is important, for example driving. In 2004 the Swedish national road and transport research institute (VTI) introduced Simulator III – their third generation of driving simulators. Inside Simulator III a camera based eye tracking system is installed that records the eye movements of the driver. To be useful, the raw data from the eye tracking system needs to be analyzed and concentrated into a number of measures relevant for the research at VTI.

This thesis presents methods to analyze the data from the eye tracker and transform it into something more useful. A world coordinate system is setup to connect the eye tracking system with the real world in a consistent way. A set of measures is collected, mainly from ISO and SAE standards, to be used as output from the analysis. Finally an application is developed for performing the analysis. The application reads the data from the eye tracker and the simulator, analyzes the data, and outputs a set of eye movement measures usable for the researchers at VTI.

# Acknowledgements

A number of persons have been involved in the process of creating this thesis. I would like to thank Mikael Nordenrot, also a thesis student and my office roommate at VTI, for the many discussions, ideas and good company, making my time at VTI more interesting and gratifying. My supervisor Tania Dukic for all the valuable and rewarding feedback, and for engaging me to put in that little extra effort in reaching the deadlines. Mikael Adlers for all support and technical discussions. Katja Kircher for much helpful input and ideas. All the other persons at VTI – you know who you are. Last but not least I would like to thank my academic supervisor Matt Cooper for all useful feedback and guidance in the process of completing this thesis.

# Contents

# Chapter 1

# Introduction

Visual attention and eye movements have been studied for over a century [5]. Eye motion is considered to be closely interlinked with attention [14], and is therefore an interesting variable to study in many different research fields where attention is an important factor. These areas of research include reading, scene perception, marketing, computer science, aviation and driving [4].

Since the mid-1970s equipment for measuring eye movements has become more accurate and technically advanced [14]. This has made it possible to use computers to collect and analyze large amounts of data from eye-tracking systems. A wide variety of eye tracking devices and applications now exist, and the increasing availability of fast image processing hardware in recent years has led to the development of non-intrusive video-based real-time eye tracking systems [5].

VTI is the Swedish national road and transport research institute, an independent, internationally established research institute engaged in the transport sector [26]. VTI is a world leader in several research areas, including simulator technology, and in 2004 VTI introduced their third generation of driving simulators. Simulator III uses a real vehicle body and an advanced motion system to make the driving experience as realistic as possible. Inside Simulator III a camera based eye tracking system is installed. The system automatically records several parameters regarding the eye motions of the test subject, for example the position of the head and the eyes, the direction of the gaze, and the opening and closing of the eyelids.

So far the data generated from the eye tracking system installed in the simulator has mostly remained unanalyzed. This is mainly due to the lack of existing applications for this purpose. To be able to use the data from the eye tracker in the research at VTI, an application has to be developed in which it is possible to analyze the data and perform the calculations needed to derive relevant measures and information.

## 1.1 Objective

The objective of the thesis can be divided into three parts, where the first two make up the preliminary work needed as a base for the third and main objective.

1. Define a world coordinate system (WCS) for the vehicle body in the simulator and use this in the eye tracking system. This connects the data from the eye tracker to the physical world of the driving compartment.

2. Find a set of eye movement measures that are relevant for the research at VTI. These measures are to be used as output from the application described in the next objective.

3. The main objective is to develop an application for automatic analysis of eye movement data from the simulator. The application is to be used off-line, analyzing data collected in different trials in the simulator. With the application, analysis of where in the physical world the driver is looking should be easy, fast and intuitive.

## 1.2 Limitations

Smart Eye Pro is the eye tracking system installed in the simulator at VTI, and thus it is the eye tracking system to be used in the thesis. The objective does not include any comprehensive evaluation and testing of the Smart Eye system. The system is assumed to work properly and deliver data with reasonable quality. The task of improving the quality of the eye tracker system and the data received from it is left to the developers of the system.

Matlab is to be used for implementing the application, this is to allow for easy extension and future maintenance of the application at VTI. Using Matlab poses some limitations. Matlab does not provide a very powerful programming language, and there are a number of limitations in GUI (graphical user interface) development. However, Matlab provides numerous functions for analyzing and presenting data, which are suitable for the analysis to be carried out. The rapid and straightforward analysis and development environment provided by Matlab offsets, at least partially, the downsides of the programming language and GUI limitations.

## 1.3 Outline of the report

Chapter 2 gives the reader a review of the background and theory forming the base of the following chapters. The subjects reviewed are eye movements and attention, different eye tracking systems and processing of eye movement data.

Chapter 3 describes the preliminary work before the application can be developed. This chapter corresponds to the first two parts of the objective. This chapter also gives an overview of the functionality of the eye tracking system used in the thesis.

Chapter 4 goes through all the different parts of the application and the algorithms used. This chapter corresponds to the main part of the objective.

Chapter 5 presents a simple validation procedure performed on the application, and the results from it.

Chapter 6 is the final and concluding chapter of the thesis. This chapter contains a discussion of the results of the thesis, what was good and what could have been done differently, and what can be done in the future to improve and extend the outcome of the thesis.

# Chapter 2

# Background

This chapter presents the background of the thesis. VTI and Simulator III will be presented in the first section, then comes a theoretical section describing eye movements and why they are studied. Different eye tracking systems are described in the next section, followed by a review of different methods of analyzing eye movement data. The last section reviews some previous work that has been done on the subject.

## 2.1 VTI and the simulator

VTI, the Swedish National Road and Transport Research Institute, is an independent research institute within the transport sector [26]. The research includes safety, economy, environment, traffic and transport analysis, public transport, behavior and human-vehicle interaction, as well as road design, operation, and maintenance.

VTI provides services in research and development, investigations, studies, expert statements, and project management for authorities, universities, and businesses [26]. The number of clients is about 200 each year, and the largest one is the Swedish Road Administration.

### 2.1.1 Simulator III

VTI has been building and using driving simulators since the end of the 1970s [6]. From the very start, with their Simulator I, VTI acknowledged the need for an elaborate motion system for a more realistic driving experience. After many years of working with driving simulators, the new Simulator III was introduced in 2004 (see fig. 2.1, 2.2). Simulator III has an improved linear motion system compared with that of the earlier Simulators I and II. The new moving base


Figure 2.1: VTI Driving Simulator III

allows a lateral movement of ±3.75 m, with an acceleration of, at most, 0.8 g. The lateral motion utilizes a steel tape drive to allow smoother motion compared to earlier chain transmissions. In addition to this, the vehicle body is attached to a vibration table allowing pitch, roll and heave motion in relation to the screen. Furthermore the whole setup, including the screen, can be pitched and rolled, and also rotated 90° to allow longitudinal motion instead of transverse.

Eye movements have been studied at VTI for several years, using a number of different systems. In the simulator, studies of attention and drowsiness have been carried out with the assistance of eye tracking. In Simulator III, a video based eye tracking system (Smart Eye Pro) is installed, which will be described in detail later on in the report.

## 2.2 Eye movements

This section presents a background of why eye movements are studied at VTI, their connection with attention and cognition, and also describes the physiological properties of different kinds of eye movements that can be observed in humans.

### 2.2.1 Eye Movements and Driving

Eye movements are interesting to study in research regarding transportation and driving because of their connection with attention. Two areas studied at VTI, where eye movements are of great interest are *distraction* and *sleepiness*.

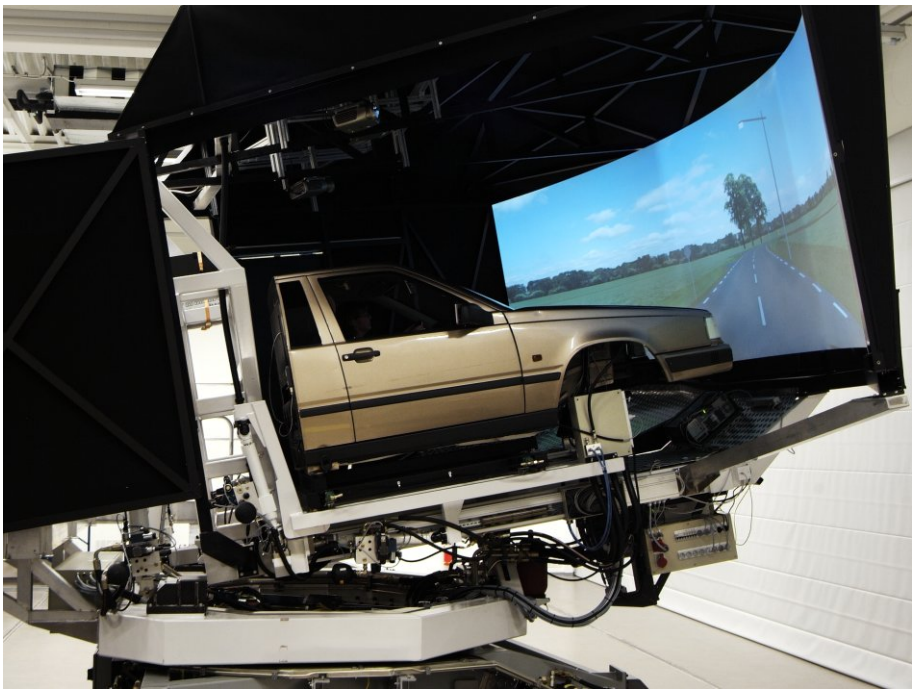


Figure 2.2: Drivers cabin of Simulator III

### Visual and cognitive distraction

In recent years, the range of technology and equipment available to the drivers of passenger vehicles has increased, and more devices may appear in the near future [12, 22]. Navigational systems, route guidance equipment, real time information systems and the like are becoming common, and these devices may pose considerably larger distractions than those posed by, for example, CD players, climate controls, and mobile phones.

It is widely believed that a driver has only one visual resource which has to be shared between different visual tasks [21]. To manage multiple tasks at the same time, for example the main task of driving and a secondary task of handling the radio or a mobile phone, a time-sharing strategy is often observed. This means that the drivers one and only visual resource takes turns on gathering information for the different tasks. Typically the driver looks away from the forward roadway for 1–1.5 seconds and then looks back. This is repeated until enough information is gathered from the secondary task and focus lies solely on the main task of driving. There seems to be an upper bound for the duration of these off road glances, most drivers seem reluctant to look away from the road for longer than 2 seconds at a time. This figure may vary a bit with the road geometry and the traffic situation, but it also depends on the complexity of information displays and devices. This is a clear sign that badly designed in-car devices may pose a

significant distraction for the driver and thus seriously affect traffic safety.

*Visual* distractions like in the examples above can quite easily be observed by measuring how the driver moves his gaze around while performing different tasks, and this is one of the main reasons why traffic researchers, vehicle manufacturers and developers of in-car devices might be interested in eye movement data.

There also exist purely mental, *cognitive* distractions, where an overload of information can cause problems while driving [22]. An intense business conversation is different from a regular social conversation in terms of the cognitive workload it poses on the driver. Using hands-free phones while driving decreases the drivers cognitive load but does not eliminate it completely. Some researchers have indicated that the risk of driving while on the phone can be compared to that of driving while intoxicated. Purely cognitive distractions are harder to observe using eye tracking, but there are certain eye movement patterns that can be connected to a high cognitive workload. For example, an overall reduction of eye movements is often seen while the driver performs a mental task.

#### Sleepiness

Many researchers consider sleepiness to be the greatest identifiable cause of traffic accidents, even greater than alcohol [23]. The actual number of accidents caused by drowsy drivers is hard to estimate, but in different surveys, as much as 40–50 % of drivers admit to have been driving while drowsy or fallen asleep at the wheel. It would be of great interest to be able to automatically detect symptoms of sleepiness in the driver.

Two of the most usable indications of sleepiness are the blink duration and blink amplitude (the level of eyelid opening) of the driver [23]. Another drowsiness measure is the level of gaze activity of the driver, where an alert driver scans the environment and frequently looks in the mirrors. Low gaze activity on the other hand indicates drowsiness.

### 2.2.2 Eye movements and attention

Eye movements are very informative as a data source for analysis of human cognition [16]. The reason for this is that eye movements give an indication of the focus of visual attention. The link between eye movements and attention is strong and provides researchers with a window into a persons thought process. Attention can be divided into two types, *overt* attention where the eye movements brings the fovea onto a subject of interest, and *covert* attention where mental focus is placed on peripheral objects not fixated in the fovea, by moving our "inner eye" [25]. Attention and eye movements are not only functionally related, but also share the same anatomical areas in the brain, so there is clearly a strong relationship between them.

Overt attention, where you fixate a subject on the fovea, is the primary method of paying attention, and is supported by covert attention for deciding where in the periphery to direct the gaze next [25]. One important argument why overt attention is the predominant method of attention is that it involves high resolution foveal vision, which gives rise to a much larger proportion of processing in the brain than covert attention does. The magnitude of the effects of covert attention are relatively small compared to overt attention. In research where eye movements are measured, the assumption is often made that attention is linked to foveal gaze direction, since an eye tracker can only measure the overt movements of the eyes [5]. Thus the covert part of attention is usually ignored, since there is no easy way of detecting it by external observation.

### 2.2.3 The different types of eye movements

A number of different eye movements can be observed in humans, in this section a few of them are presented and described.

#### Saccades

*Saccades* are the fast eye movements used to bring a new part of the visual field into the fovea [3]. They are mostly voluntary, and they are in fact the only voluntary eye movements that one can make without special training. Saccades are so fast that no visual processing can be done while executing them, and there is normally no way for the brain to guide the eyes to the right position once a saccade have been launched. The visual system must calculate in advance how to activate the eye muscles to throw the eye into the desired position. The movement is thus considered to be preprogrammed or *ballistic*.

The reason we make saccades as often as we do is that acuity is good only in a very limited region of the visual field [14]. We see clearly only in the *foveal* region which consist of the central 2° of vision. Acuity is not nearly as good in the *parafoveal* region (5° on either side of the fovea), and it is even worse in the *peripheral* region (the rest of the visual field). Thus we make saccades to place the interesting part of the visual field at the fovea.

When performing a saccade the eye moves very fast, often at more than 700°/s for large amplitudes [3]. Large amplitude saccades are fairly uncommon however, and more than 85% of saccades under natural conditions have amplitudes of less than 15°. The duration of saccades larger than 5° is approximately 20–30 ms plus 2 ms for every degree of amplitude. The total duration of most saccades ranges between 10–100 ms [5].

#### Fixations and smooth pursuits

*Fixations* take place between the saccades, where the eyes lock onto a target while the brain encodes the image [16]. Fixation durations range from about 100 ms to over a second, depending on the given task.

Another type of eye movement where information processing occurs are *smooth pursuits*, where the eyes match the velocity of a moving target (depending on the range of target motion) [5]. Fixations can intuitively seem to be a special case of smooth pursuits, where the target is standing still, but this is probably not the case. Fixations are instead characterized by the *miniature eye movements* described below.

#### Other types of eye movements

*Tremor*, *drift*, and *microsaccades* are the three components of the miniature eye movements that occur during a fixation [3]. The effect of these very small eye movements (in the region of a few minutes of arc) is to move the retinal image about at random, thus constantly triggering different foveal receptors. Without this re-triggering of the receptors, vision would soon fade. *Vergence* is the eye movement used to rotate the eyes inward to be able to focus the pair of eyes on targets at different distance [5]. *Nystagmus* is a sawtooth-like eye movement pattern consisting of a slow following movement interrupted at intervals by a fast saccadic movement [3]. One use of nystagmus is to stabilize a background moving continuously in one direction.

Neither miniature eye movements, vergence, nor nystagmus will be further examined in the thesis, instead focus lies on fixations and saccades, and different composite measures of the two.

## 2.3 Eye tracking systems

Systems for recording eye movements have existed for about a century [5]. With the progress of the technology, more elaborate eye tracking systems have continued to evolve. There now exist a number of eye tracking systems using different techniques. Some of the most common types of systems are discussed below.

### 2.3.1 Electrooculography

Electrooculography, or EOG, was the most widely used eye movement recording method in the 1970s, and is still in use today [5]. EOG uses the fact that there is a difference in electrical potential between the front and the back of the eyeball [23]. Electrodes that detect changes in the potential are placed around the eyes in a pattern to best measure the eye movements of interest (see figure 2.3). One major advantage of EOG is the possibility to use high sample frequencies. This

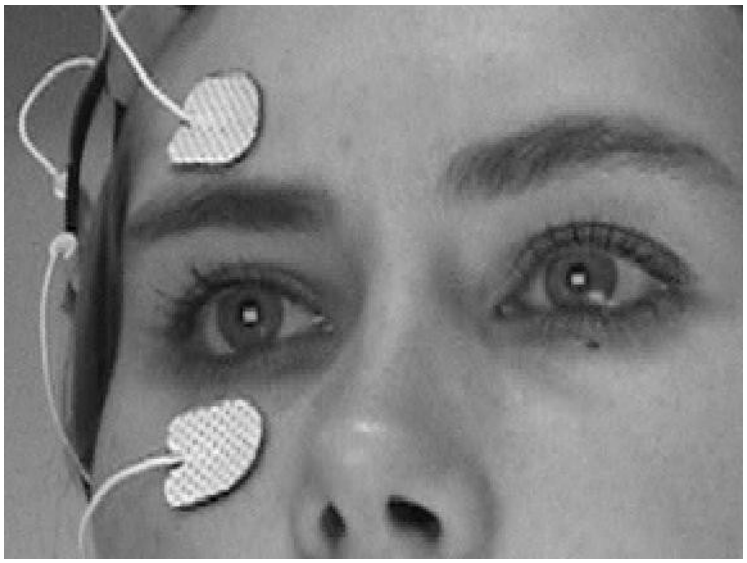


Figure 2.3: Electrooculography – EOG

means that very fast eye movements, like saccades and blinks, can be recorded and analyzed in detail.

### 2.3.2 Scleral search coils

One of the most precise eye tracking methods involves physically attaching a reference object to the eye, using a contact lens [5]. The reference object in most cases is a small wire coil embedded in the contact lens (see figure 2.4), which makes it possible to measure its movements in an electromagnetic field. Although scleral search coils is a very accurate method of eye movement measurement, it is highly intrusive and causes discomfort for the wearer.

### 2.3.3 Photo and video based systems

A wide variety of eye tracking systems exist that measure some distinguishable feature of the eyes using photo sensors or video [5]. Features that can be measured are the shape of the pupil, the position of the boundary between iris and sclera, and corneal reflections of some fixed, often infrared, light source. The measurements of these features are often made automatically by the eye tracking system, but may also involve manual inspection of the recorded eye movements, a process that can be extremely tedious and error prone. Some systems require the head to be fixed, which can be accomplished either by using some kind of chin rest or bite bar, or by wearing the system on the head (see fig. 2.5). Other systems are totally non-intrusive and pose no requirements of the head to be

Figure 2.4: Scleral search coil

fixed or anything to be worn. One example of such a system is the Smart Eye Pro system installed in the VTI Simulator III.

## 2.4 Eye movement data analysis

The raw eye movement data received from an eye tracking system can be both complex and of substantial size. In some eye tracking systems there is also a lot of noise present in the data. This means that the data may have to go through some processing steps, in order to better be able to derive relevant information from it.

### 2.4.1 Noise reduction

To reduce the noise that may be present in the data from the eye tracker, filtering of the data with some suitable filter can be performed. To remove high frequency noise, a low pass filter can be applied to the data. One way of achieving this is by using a moving average. For eye movement data, using a median filter may be more suitable than other smoothing procedures, since it is effective in preserving sharp edges and smoothing spiky noise [8].


Figure 2.5: iView X HED – head mounted eye tracker

### 2.4.2 Identification of fixations and saccades

To reduce the size and complexity of the eye tracker data, some form of fixation identification can be performed [17]. This means that saccade data points are removed and fixation data is collapsed into tuples, one for each fixation. This results in some loss of information, but it also means a significant reduction in size and complexity of the data. During saccades, no visual processing is carried out in the brain, so the actual paths the eyes make during saccades are seen as irrelevant for many research areas. The miniature eye movements (tremor, drift and microsaccades), which are also lost, also have very little impact on higher level analysis. Fixation identification thus leaves you with the most essential information regarding cognitive and visual processing.

#### Fixation identification algorithms

Salvucci & Goldberg [17] defines a taxonomy for fixation identification algorithms. The criteria used to classify different algorithms are the following:

- Spatial
  - *Velocity-based* algorithms are based on the fact that fixation points have low velocities, while saccade points have high velocities.
  - *Dispersion-based* algorithms assume that fixation points occur near one another.

  - *Area-based* algorithms identify points within given areas of interest (targets).

- Temporal

  - *Duration sensitive* algorithms use the fact that fixations most of the time have a duration above 100 ms.
  - *Locally adaptive* algorithms allow the interpretation of a given data point to be influenced by adjacent points.

A given fixation identification algorithm can incorporate one or more of the above criteria in a number of different ways. For example, velocity-based algorithms can use a velocity threshold or hidden Markov models, while dispersion-based algorithms can use a dispersion threshold or minimum spanning trees to identify fixations. Later in this thesis, a velocity-based and duration sensitive algorithm for identifying fixations will be described.

## 2.5 Previous work

Analysis of eye movements has been carried out for many years and there are many different types of eye tracking systems, as described above. To process and analyze the raw data from an eye tracking system, some computer software is often used. Development of such an application is the main objective of this thesis. Some previously developed software applications for eye movement analysis are presented here.

### 2.5.1 Applications for 2D

With many eye tracking systems and applications, the test subject looks at a plane surface or a computer screen. The output from the eye tracker is 2D coordinates for the surface or screen. These types of systems are very common and are often used for analyzing reading patterns, perception of images, advertisements, websites, etc.

*Tobii* is a Swedish company that develops 2D eye trackers and applications where the test subject looks at a computer screen [24]. Tobii has developed a number of applications with different purposes, for example assessment of web pages and advertisements, psychology and vision research, and eye control and interaction for people with limited mobility.

*SensoMotoric Instruments* (SMI) develops a range of eye tracking systems and softwares for analysis in 2D [18]. A head mounted system, similar to the one in figure 2.5 has been used at VTI. Two dispersion based fixation identification algorithms were compared using Matlab and data from the SMI tracker, in a masters thesis by Bjällmark and Larsson [1].

*ILAB* is an open source project for eye movement analysis, developed in Matlab [7]. ILAB is a free-standing application that can analyze data from a number of different eye trackers. The input must be two-dimensional however, which is not the case for the Smart Eye Pro system installed in VTI Simulator III.

### 2.5.2 Applications for 3D

In recent years eye tracking systems have emerged that measure the position and direction of the eyes in three dimensions. The eye tracker in the simulator at VTI is such a system, thus the analysis application must work in 3D.

The Swedish car manufacturer Volvo uses FaceLAB for research, a similar system to the Smart Eye Pro eye tracker installed in the simulator at VTI. Algorithms for fixation identification and calculation of eye movement measures using FaceLAB data were developed at Volvo in a masters thesis by Larsson [11], and in a PhD thesis by Victor [25]. An application for visualizing gaze direction was developed in a masters thesis by Blissing [2].

Autoliv, a worldwide leader in automotive safety, has constructed a test vehicle equipped with a Smart Eye Pro system. Different applications have been developed for analyzing gaze data from the test vehicle. No detailed information is available about these applications however, since the research is not public.

# Chapter 3

# Preliminary work

This chapter describes the preliminary work needed to develop the eye movement analysis application. The first section reviews the Smart Eye system and the problems associated with it. The following two sections correspond to the first two parts of the objective – defining a world coordinate system and finding a set of relevant eye movement measures.

## 3.1 The Smart Eye Pro system

Smart Eye Pro is a non-intrusive video based head and gaze tracking system [19, 20]. Smart Eye uses 2–6 video cameras at different locations to measure both the subjects head position and gaze direction.

The system installed in VTI Simulator III is a Smart Eye Pro 3.5 system, using 3 cameras. The cameras are placed on the dashboard of the cabin, in front of the driver (see figure 3.1).

### 3.1.1 Operation of the system

For the Smart Eye system to function properly, a number of steps must be performed [19]:

- The cameras must be positioned properly and focused correctly. The test subject should be clearly visible in all cameras.
- The cameras must be calibrated to detect their positions and orientations with respect to each other. This is done semi-automatically by holding a predefined chessboard pattern in front of the camera. The software calculates camera positions and rotations based on this pattern.


Figure 3.1: Smart Eye cameras in VTI Simulator III

- A profile must be built for each test subject, with a number of images of the persons face from different cameras and rotations of the head. Features like nostrils, ears, corners of the eyes and mouth, and centers of the eyes, are marked in the images to form a 3D head model.
- A world coordinate system must be setup. This is described further in section 3.2.1.

### 3.1.2 Data from the system

Smart Eye outputs a number of variables regarding head position, gaze direction, eye closure, quality measures, etc. The data can be logged to a text file or sent over a network using TCP or UDP. The maximum sample frequency is 60 Hz, which may or may not be sufficient for analysis, depending on the purpose of the analysis. To analyze saccades and blinks in detail, for example, you need much higher sample frequencies. To see this, consider the duration of a saccade, which is about 10–100 ms, compared to the time step of the eye tracker, $T = 1/f$, which in this case is about 17 ms. It is easy to understand that a detailed analysis of fast eye movements like saccades require a much shorter time step than this. However, a sample rate of 60 Hz may be totally acceptable for slower eye movements like fixations.

The data from the Smart Eye system is rather noisy, and due to the complexity of the system it is not always easy to determine the cause of the noise. All the different elements in the system may introduce errors, which will be propagated

throughout the system. Consequently there are many sources of error in the system – camera setup, camera calibration, profiles, world coordinate system, lighting conditions, facial features of the test subject, etc.

It is mainly the variables regarding the eyes (i.e. gaze direction, eye closure) that exhibit a lot of noise. The head position and rotation are often more stable. The reason for this may be that there is only a very limited amount of pixels to base the eye calculations on, while the head is represented by a much greater number of pixels, and is thus more accurately tracked. This source of error is, however, hard to influence as a user of the system, since it is inherent in the system itself.

### 3.1.3 Accuracy

The accuracy of the system depends a lot on the placement of the cameras. Since the cameras are normally spread out horizontally, accuracy is better horizontally than vertically. Accuracy is also better for straight ahead gaze directions than in the periphery. This is because all cameras can track the eyes when looking straight ahead, while some or all cameras may loose tracking for large gaze angles.

The internal calculations of gaze directions in Smart Eye are based on identifying the edge between the iris and sclera*. Thus accuracy should be better for situations where as much as possible of this edge is seen by the cameras. Examples where this is not the case are when the test subject squints, glances to the sides, looks down (partially closing the eyes), or sits at a large distance to the cameras with fewer pixels representing the iris/sclera edge as a result.

## 3.2 World coordinate system

The first part of the objective is to connect the Smart Eye data to the physical world of the simulator by defining a WCS and then use this in the Smart Eye system. When this is done, the Smart Eye data can be put in relation to the objects of interest in the simulator. Section 3.2.1 and 3.2.2 was carried out in collaboration with Mikael Nordenrot, who also did his masters thesis at VTI in 2007.

### 3.2.1 Setting up the coordinate system in Smart Eye

To be able to use Smart Eye's measured positions and directions consequently throughout different trials and test subjects in the simulator, the origin and directions of the coordinate axes must not change from one session to another. The Smart Eye system has three ways of defining a WCS [19]:

*In Smart Eye Pro version 4, released in June 2007, corneal reflection is also used to calculate the gaze direction.

1. Use one of the cameras as the origin, with x- and y-axes parallel with the image sensor plane and z-axis pointing in the camera view direction.

2. Mark a number of known points in the simulator to use as reference points. These points must then be marked in the Smart Eye software, and their coordinates entered.

3. Use a predefined chessboard-pattern as reference (the same pattern that is used for camera calibration). The origin and direction of the axes are automatically determined by the software using the position and direction of the chessboard-pattern.

Since the positions and directions of the Smart Eye cameras can change, on purpose or unintentionally, the first method is not appropriate for defining a consistent WCS. If the camera that was used to build the WCS is slightly shifted, the whole coordinate system is shifted along with it. As the cameras are very small, this method also makes it hard to measure exact positions of objects in the simulator in WCS coordinates.

The second method produces a fixed WCS which will not change from one trial to another as long as the reference points do not move. It also lets the user place the origin and direction of the axes freely. It does however require the user to mark all the points in the software and input their coordinates each time the system needs to be re-calibrated. This could become a time-consuming process if the system is re-calibrated on a regular basis.

Using the chessboard-pattern is a very fast and easy way of defining the WCS, and it is the method chosen for the thesis. No points need to be marked and no coordinates need to be entered into the software. The origin is automatically set to the middle of the pattern, the x-axis pointing to the right from the cameras view, the y-axis pointing up, and the z-axis pointing toward the cameras (see figure 3.2). For the WCS to be fixed between trials, the chessboard pattern must however be held in exactly the same spot each time the system is to be re-calibrated. Previously this method had been used at VTI with the pattern placed on the head restraint of the car seat. Since the driver can adjust the seat, this placement of the pattern is not very consistent. Therefore the WCS of previous trials may differ a great deal between subjects, which makes it hard to use the Smart Eye data from those trials for comparative purposes.

### 3.2.2 Calibration pattern fixture

To hold the chessboard pattern fixed in an exact position, some sort of mechanical fixture must be used. The best placement for the pattern is approximately at the drivers head position, since there it can be viewed from all cameras, at a distance allowing the camera-image of the pattern to be sufficiently large. This makes the

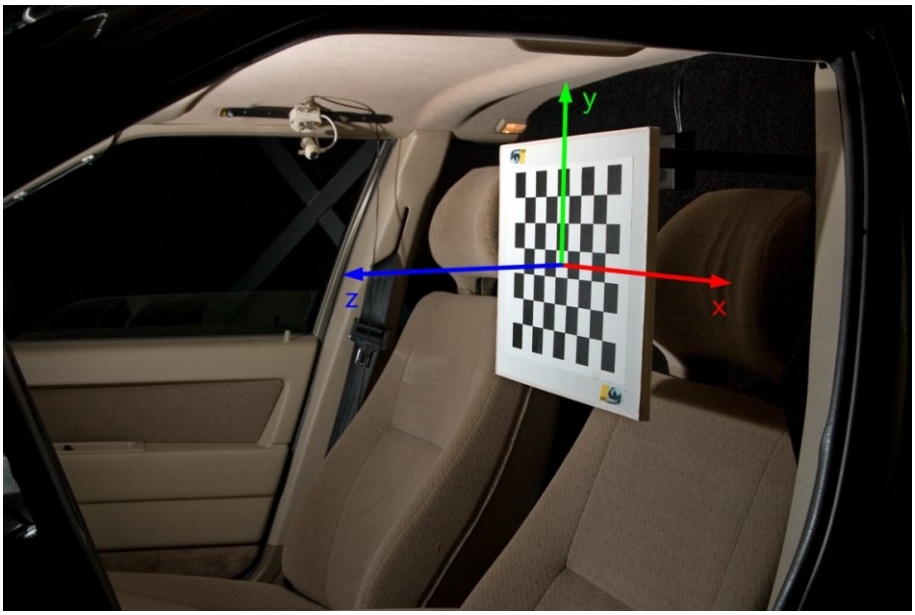



Figure 3.2: Origin of WCS in the center of the calibration pattern

accuracy better than had it been placed further away from the cameras. Of course if the pattern is permanently placed at the head position of the driver it would be very obstructing for the driver, therefore the fixture of the pattern must allow it to be removed easily and to be replaced in the exact same position over and over again. A fixture consisting of a welded aluminum profile with a quick-release screw-mount was constructed at VTI's metal workshop. The mounting-bracket for the fixture is placed at the back wall in the simulator, a structure that is rigid and does not move between trials. This makes setting up the WCS an easy procedure which can be repeated with great consistency:

1. Mount the fixture with the chessboard pattern in the bracket and tighten the screw.

2. Automatically set up the WCS by press-of-a-button in the Smart Eye software.

3. Remove the fixture and continue with the trial.

### 3.2.3 Measuring coordinates in the simulator

For the output positions and directions from the Smart Eye system to be meaningful, positions of the objects of interest (for example mirrors, instrument panel, etc.) in the simulator must be established in WCS coordinates. In this way the output from Smart Eye can be compared with the positions of the objects in the simulator. The origin and direction of the coordinate axes is given by the position

and orientation of the chessboard pattern as described above. Measuring coordinates of an object is thus done by simply measuring the distance from the origin in the x, y and z directions. The measurements in this thesis was performed using a tape measure. For better accuracy, more elaborate methods may be used, such as laser devices or digitizing arms.

## 3.3 Eye movement measures

The second objective of the thesis is to find a set of eye movement measures to use as output from the analysis application. The measures need to be relevant for the research at VTI. This section describes a number of standardized measures and some additional ones, all which are used in the thesis.

### 3.3.1 Standardized measures

The ISO 15007 and SAE J2396 standards [9, 10, 15] define a number of eye movement measures. These measures are defined for research regarding visual behavior of road vehicle drivers, and should thus be very suitable for the research at VTI. The ISO and SAE measures used in the thesis are defined as follows:

**Dwell time** – the sum of consecutive fixation and saccade times to a target in a single glance.

**Glance duration** – the time from the moment at which the direction of gaze moves toward a target to the moment it moves away from it.

**Glance frequency** – the number of glances to a target within a predefined time period, or during a predefined task, where each glance is separated by at least one glance to a different target.

**Glance location probability** – the probability that the eyes are fixated at a given target during a sample interval.

**Link value probability** – the probability of a glance transition between two different locations.

**Transition time** – the duration between the end of the last fixation on a target and the start of the first fixation on another target.

The glance duration can be examined over the length of an entire trial, rendering three more measures [9, 21]:

**Peak glance duration** – the time of the longest glance at a target area during the performance of a task.

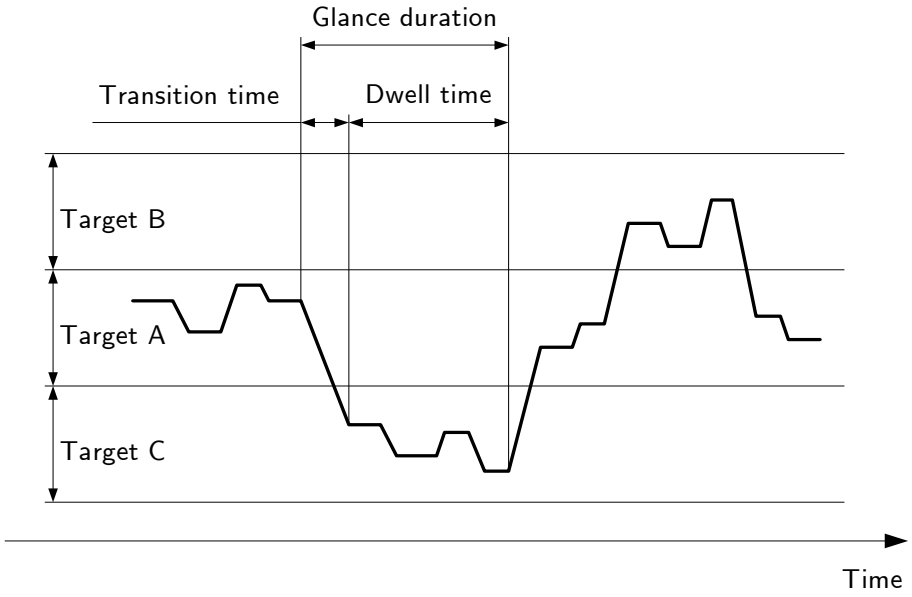



Figure 3.3: A glance as defined in the ISO standard

**Mean glance duration** – the mean amount of time of all the glances at a target area during the performance of a task.

**Total glance duration** – the cumulative time elapsed for all glances at a target area during the performance of a task.

These measures can for example be used in visual distraction research [21]. A result from that kind of research is for example that the peak glance duration away from the road scene ahead is rarely longer than 2 seconds.

### 3.3.2 Glances and targets

As one may have noticed in the list above, most of these measures are based on *glances*. The word glance is often used to refer to a single glance duration, which is defined as the transition time to a target and the dwell time within that target [10] (see figure 3.3). Another term that keeps occurring in the list of measures is *targets*, which are defined as predetermined areas within the visual scene. Targets of a drivers environment are typically rear view mirrors, instrument panel, side windows, windshield, etc. The targets may also be subdivided into smaller sections or subtargets to increase the level of detail in the measurements. The windshield is, for example, a very large target and it may be possible to gain more information if it is divided into a midsection and two side sections. The size of the targets must of course correspond to the quality and accuracy of the data from the eye tracker, there is no use of having targets smaller than the

error of the tracking system, since no statistically significant information could be derived from such a setup. Targets used for eye tracking in driving situations tend to be quite few and sometimes a bit oversized, to take into account possible errors and problems with accuracy in the tracking.

# Chapter 4

# Automatic Eye Movement Analysis

The main objective of the thesis is to build an application for automatic eye movement analysis. The application is to be used on data collected in different trials using VTI's driving simulator. This section describes the application – the algorithms and different parts used, how they are put together, and how it is supposed to be used.

## 4.1 Overview of the application

The application, called VTEye (as a reference to VTI and Eye movements), consists of a GUI and a number of functions that operate on the data. In this section an overview of the application and a description of the program structure is given. Following sections explains the different parts of the application in greater detail.

### 4.1.1 User interface

The GUI is constructed of Matlab figures, each containing a number of components such as buttons, texts, menus, axes, lines, panels, sliders, etc. The main window (see figure 4.1) lets the user load files, look at data, filter noise, select subsets of the data, calculate eye movement measures, and output these measures to file.

From the main window, the user can open the target editor (see figure 4.2), where targets can be defined, edited, deleted, and sets of targets can be loaded and saved. The targets are drawn in 3d, and can be rotated using the mouse.

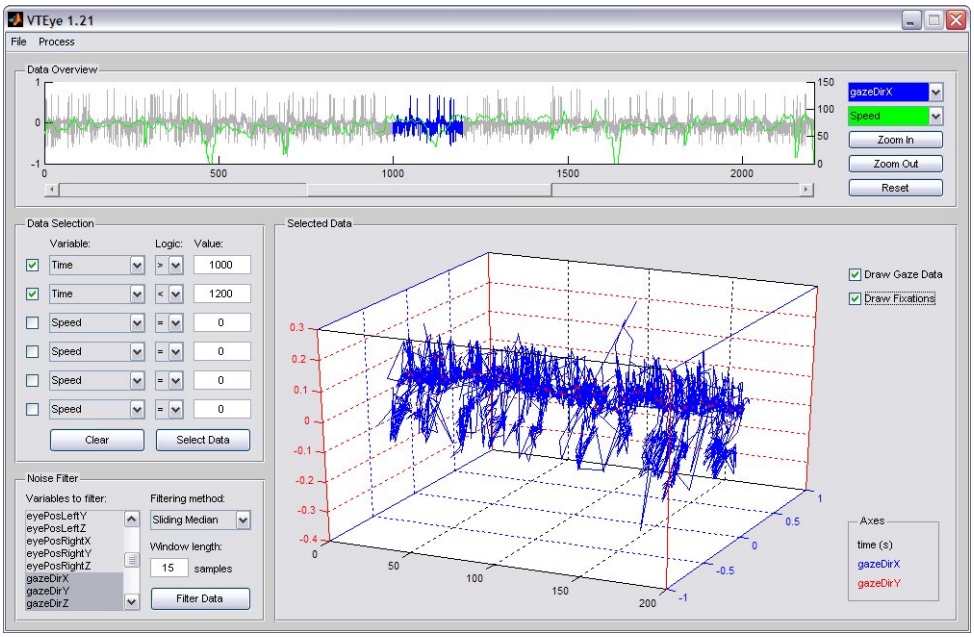



Figure 4.1: VTEye main window

The user can also inspect the gaze directions of the loaded data directly within the targets.

The settings window (see figure 4.3) is also opened from the main window. Here the user can select which of the Smart Eye variables to use as gaze direction and eye position. Settings regarding the fixation identification can also be edited in this window.

### 4.1.2 Program structure

There are several ways to structure a program. Matlab is mainly a procedural programming language, but also incorporates some elements of object oriented programming. However, object oriented programming in Matlab is not very common, and most Matlab users are unfamiliar with it. Since VTEye is supposed to allow for further development and added functionality, a procedural approach was chosen. Procedural programming can also be more straightforward when it comes to smaller projects, where the overhead cost of producing an object oriented approach from scratch may be larger than its gains.

Some of the components in the GUI use callbacks to invoke one or more functions when the component is activated, for example when a button is pushed. Many of the functions analyze simulator or Smart Eye data, data about fixations or glances, data about targets, etc. This application data needs to be stored somewhere so that the callback functions can access and modify it. All GUI components have a property named `UserData` where you can store data that

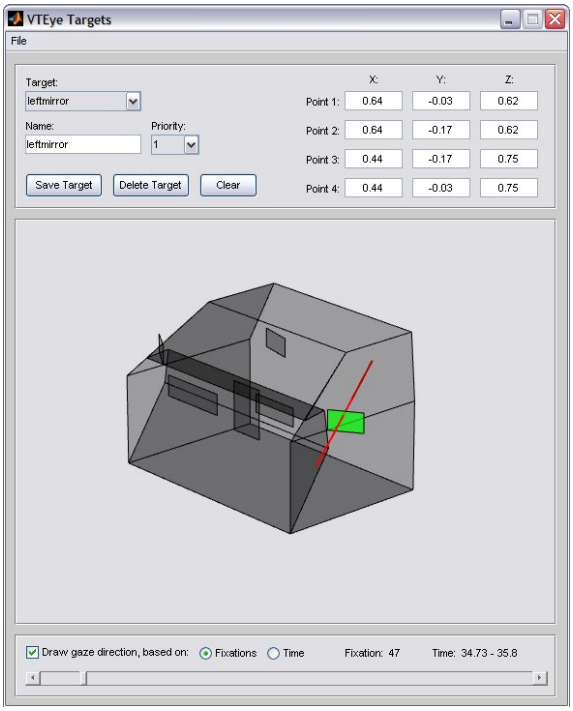



Figure 4.2: VTEye target editor

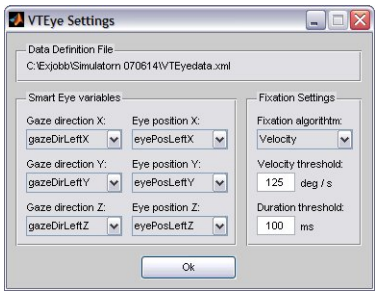



Figure 4.3: VTEye settings window

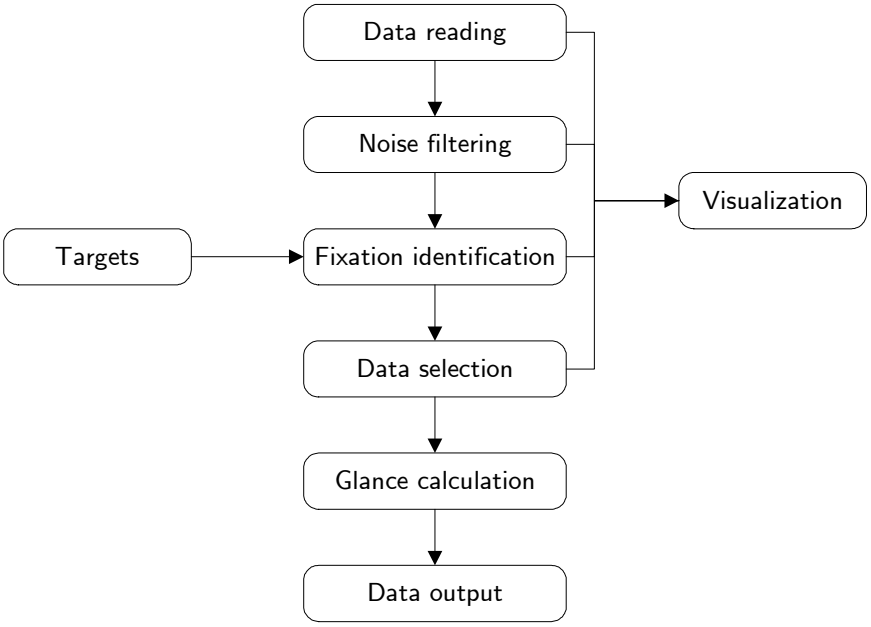



Figure 4.4: Data flow in VTEye

needs to be accessed and shared by different functions. Storing the application data in the `UserData` property of the main program window lets all functions access and modify this data using Matlab's `get` and `set` functions. This solution allows for great flexibility, and eliminates the need for passing function arguments back and forth between the callback functions. The downside is that all functions can access and modify all data, which is not as safe as only allowing the functions to have access to a limited set of data.

The code for all functions of the application is placed in one m-file (the Matlab file format). This means that you only have to deal with one file instead of having a separate m-file for each function, something that will quickly clutter your program directory with a large number of files. When a callback is executed from the GUI, the main `vteye` function is called with an argument specifying which action is to be taken. The main function then calls one or more functions to perform the requested action. Calling the `vteye` function without any arguments initializes the GUI and sets up the application to be ready for action. This is how you start the application from the Matlab prompt.

### 4.1.3 Data flow

Figure 4.4 gives an overview of how the different components interact in the application. Sections 4.2 to 4.8 describe these components in detail.

## 4.2 Data reading

The first step of analyzing the eye movement data is to read the data into the memory of the application. The data to be read consist of binary files produced by the simulator program. These files do not have any built in description of their structure, so data description files also have to be supplied to the application.

### 4.2.1 Binary data files

The binary files produced by the simulator program contain both simulator and Smart Eye data. The number of simulator and Smart Eye variables can vary from trial to trial, depending on the purpose of the trial. For example, all thinkable variables regarding speeds, accelerations, times, distances, states of the controls of the car, physics of the simulation, etc. can be chosen to be included in the data. It is also possible to construct index variables to mark data where something of interest is happening, for example when the road is slippery, a moose is crossing the road, or an instrument in the car is diverting the drivers attention. The Smart Eye variables output in the data can be chosen from the ones available in the Smart Eye software. The set of Smart Eye variables must at least contain the position of the eye and the direction of the gaze, since these are needed for the analysis in the application.

The update frequencies of the simulator and Smart Eye data often differ. Smart Eye can be updated at 30 or 60 Hz (version 3.5), while the simulator update frequency can be freely chosen. Typical frequencies for previous simulator trials are 25 or 50 Hz. A data package from the simulator simply consists of the values of all the variables, each represented as 32 bit floating point data. A Smart Eye package is constructed in exactly the same way but is preceded by a delimiter element (currently the value -88888.0), to identify the following data as Smart Eye data. The simulator program writes a new simulator data package to the binary file when it is due, and a Smart Eye package when it has received one from the Smart Eye system. Thus there is no given order of the packages.

### 4.2.2 Data description files

For the application to understand the structure of the binary data files, which contain nothing but a large number of floats, it must be supplied with a data description file. The description files are written in xml-format and contain information about the number of simulator and Smart Eye variables, variable names, update frequencies, Smart Eye delimiter element, and positions of gaze direction and eye position variables. Below is an example, showing the structure of the data description files.

```
<dataset fortran_mode="true">
   <simulator freq="25">
      <data name="Speed" />
      <data name="Distance" />
      <data name="Time" />
      <...>
   </simulator>
   <smarteye delimiter="-88888." freq="60" gx="9" gy="10" gz="11"
         ex="19" ey="20" ez="21">
      <data name="frameNr" />
      <data name="headPosX" />
      <data name="headPosY" />
      <...>
   </smarteye>
</dataset>
```

The attribute `fortran_mode` tells the application whether it needs to remove Fortran block headers and sentinels (the simulator control program is written in Fortran), or if they have already been removed. The simulator and Smart Eye subsets both contain the attribute `freq`, which specifies the update frequencies. The attribute `delimiter` is the Smart Eye delimiter element. Gaze direction and eye position variables are pointed out by `gx`, `gy`, `gz`, and `ex`, `ey`, `ez`.

### 4.2.3 Separation and synchronization of data

All the values in the binary file are read into a Matlab vector. The data is then separated using the number of simulator and Smart Eye variables and the delimiter element obtained from the data description. The data is placed in

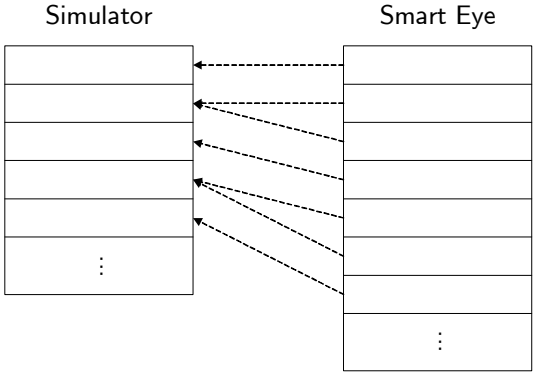



Figure 4.5: Synchronization of data where Smart Eye has a higher update frequency than the simulator

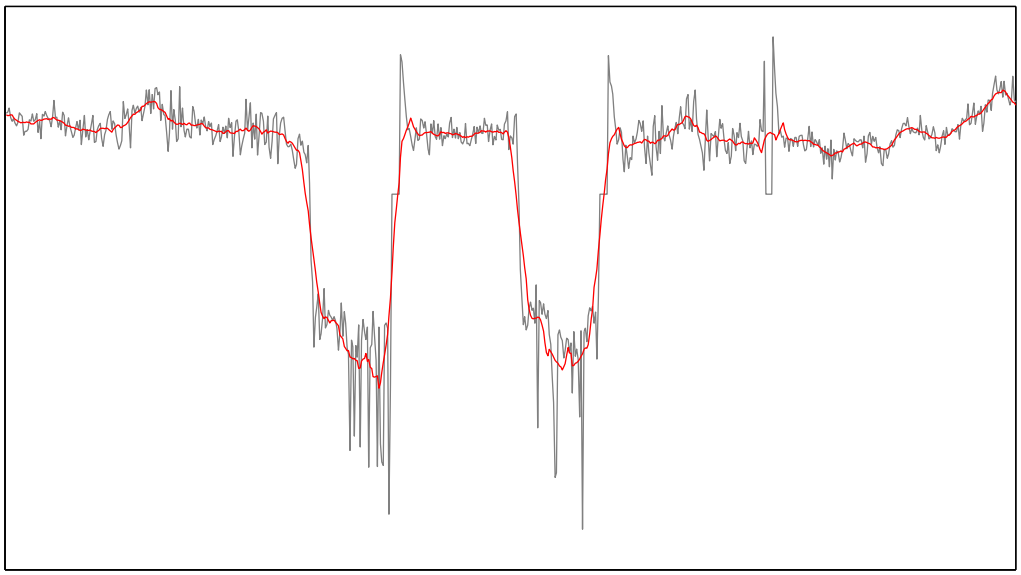


Figure 4.6: Mean filtered Smart Eye signal

two matrices, one with simulator data and one with Smart Eye data, with one column for each variable and one row for each data package. A reference variable is added to the Smart Eye matrix, which, for each Smart Eye row, points to the corresponding simulator row (the previously read one), see figure 4.5. In this way the simulator and Smart Eye data is synchronized with a maximum error of $f_s^{-1}$, where $f_s$ is the simulator update frequency.* In this way it is easy to point out which Smart Eye row(s) corresponds to a given simulator row.

## 4.3 Noise filtering

The gaze direction from the Smart Eye system contains noise in the order of a few degrees visual angle. To suppress this noise, some kind of smoothing can be performed on the data. One common way to achieve this is by using a moving average. Figure 4.6 displays a typical noisy section of eye movement data (in this case the y-component of the gaze direction), with a 15-sample central moving average applied to it.† In this case the arithmetic mean was used as the average, and we clearly see that the noise is reduced. Eye movement data can contain a lot of sharp edges where the saccades take place. By using the arithmetic mean to average out the noise, these edges are also somewhat smoothed. To

*The total error also includes the delay caused by the internal calculations in the Smart Eye system, and the delay caused by the graphics engine in the simulator (you see things on the screens some 40–50 ms after they actually happen).

†The length of 15 samples was chosen here as a compromise between achieving sufficient smoothing and preserving small fixations. 15 samples is 0.25 s with a sample frequency of 60 Hz, which preserves fixations down to 0.125 s long.

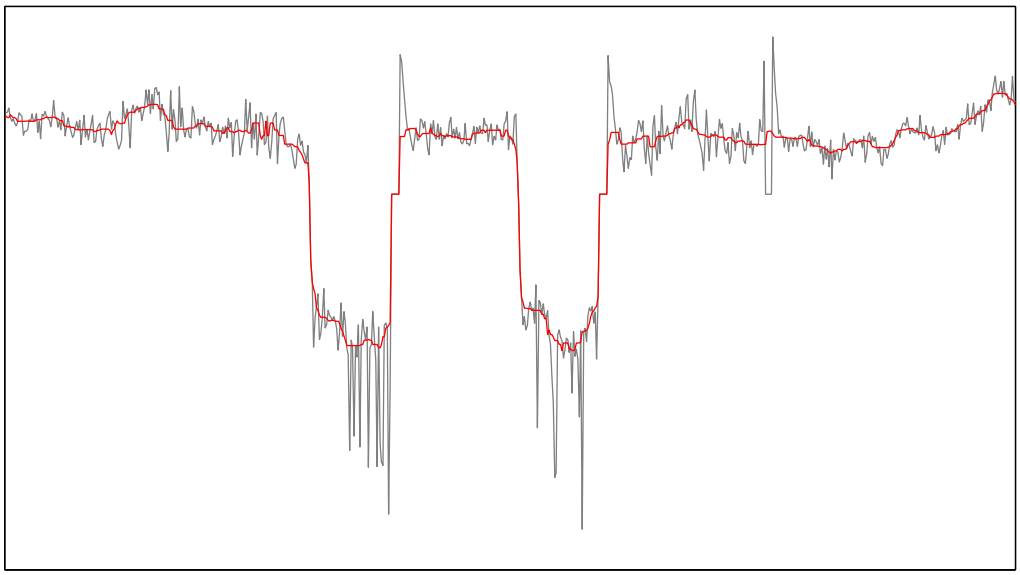


Figure 4.7: Median filtered Smart Eye signal

represent the eye movements better, it may be a good idea to use an algorithm that preserves edges. By using the median instead of the arithmetic mean, the edges are preserved [8], which can be seen in Figure 4.7. In the application, the user can choose between median and mean filtering, and can also specify how many samples the average is calculated on.

## 4.4 Fixation identification

The eye movement measures to be used as output from the application are all based on glances, as defined in the ISO 15007 standard [10]. Some sort of analysis must be performed on the raw eye movement data to identify the glances. Since glances are defined in terms of fixations and saccades, it seems reasonable to use an algorithm on the raw data that finds the fixations and saccades, and then calculate the glances.

### 4.4.1 The algorithm used in the application

Salvucci & Goldberg [17] do not refer to glances as defined in the ISO-standard in their paper. They do however touch the subject by indicating that area-based algorithms can also take fixations as input, instead of raw eye movement data. These types of area-based algorithms cannot be seen as fixation identification algorithms, but rather as tools to explain higher level visual behavior, using groups of fixations in different visual targets. The purpose of the application is to perform exactly this kind of higher level analysis, and output glance based

measures calculated from raw eye movement data. Thus the concept, as described by Salvucci & Goldberg, of using a fixation identification algorithm as input to an area-based algorithm for further higher level analysis, can be used in the application.

The fixation identification algorithm used in the application is velocity-based and duration sensitive. For all Smart Eye data points an angular velocity is calculated using the point before and the point after. Points with velocity higher than a certain threshold, which can be set in the application (125°/sec as standard), are marked as saccades. Remaining points are thus fixations. Consecutive fixation points in the raw data are then collapsed into single fixation tuples containing start time, stop time, average gaze direction, and average eye position. The durations are then calculated, and fixations with duration lower than the duration threshold (which can also be set in the application, 100 ms as standard) are removed.

One problem with pure velocity threshold algorithms occurs when the velocities stay near the threshold for some period of time [17]. If the velocities cross the threshold value several times during this time period, a number of fixations containing only one or a few consecutive points are identified, resulting in a flicker between fixations and saccades. This problem is however solved, partly by the duration threshold removing short fixations, and partly by the fact that the fixations are the input to the higher level calculation of glances. Two consecutive fixations identified near the velocity threshold then simply become part of the same glance (given that they pass the duration threshold and occur in the same target). Calculation of glances and glance based measures are described further ahead.

## 4.5 Targets

Targets are important in the definition of glances and related measures. A target is defined as a predetermined area within the visual scene [10] and typical targets are road scene ahead, mirrors, instrument panel, information displays, etc. The targets of interest may differ from trial to trial, the purpose of one trial may be to evaluate the distraction caused by a new type of information display situated on the dashboard, while another trial aims to measure visual scanning patterns for sleepy drivers. Different trials require different sets of targets, and these targets need to be defined somewhere. The Smart Eye software allows the user to define a world model with a set of targets. Intersections of the gaze direction with these targets are then output in the data from the system. However, this does not allow one to change the targets or add new ones once the trial has been run, which is a serious limitation. Therefore VTEye includes functionality for definition of targets and calculation of intersections between targets and gaze directions.

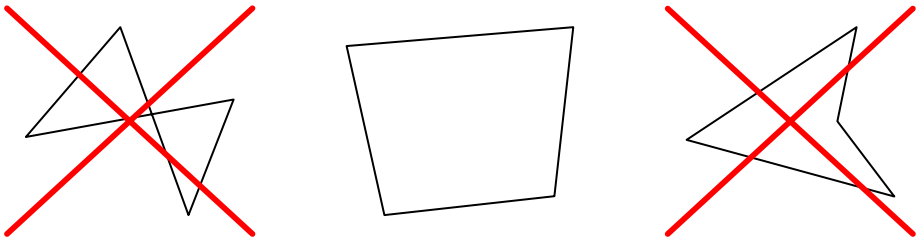


Figure 4.8: Simple convex quadrilateral

### 4.5.1 Definition of targets

The targets used in the application are simple convex quadrilaterals, i.e. non-self-intersecting polygons with four sides (see figure 4.8). The user defines a target simply by inputting the $x$, $y$, and $z$ coordinates of its four vertices. The coordinates are given in the coordinate system used by Smart Eye, with its origin in the center of the chessboard-pattern described in section 3.2.1. No check is made by the program that the targets are correctly input, but this is easily inspected by the user, as the targets are drawn on the screen and can be rotated using the mouse (see figure 4.2). An improperly defined target (i.e. concave, self-intersecting, or non-planar) can result in incorrectly calculated intersections for that target.

For each target, a priority value must also be assigned. This priority is used when intersections with multiple targets occur. A fixation can have only one target assigned to it, since it is unlikely that a person has his or her attention at two locations at the same time. If for example the center mirror and windshield are defined as targets, both these targets will be intersected by the gaze of the test subject when he or she looks in the center mirror, but the attention of the test subject lies solely on the mirror. One way of dealing with this situation would be to simply consider the closest target as the focus of attention, but this may not be suitable for other situations. Consider for example when the subject looks through the side window target into the side mirror target. In this case attention does not lie on the closest target. This is the reason why the user must specify a priority for each target.

### 4.5.2 Intersections

For each fixation, the application checks for intersections of the gaze direction with all of the defined targets. The algorithm used for this is a ray/triangle intersection algorithm developed by Möller & Trumbore in 1997 [13]. The algorithm was originally developed for the C programming language, so some adaptation was required for it to work in Matlab. Matrix and vector operations are used as much as possible instead of loops, to speed up the calculations (Matlab is not

optimized for using loops). Furthermore, as pointers cannot be used in Matlab, some variables are handled differently from the original algorithm. Since the algorithm checks for intersections with triangles, each target is divided into two triangles. If the gaze intersects any of the two triangles, an intersection with that target is marked in the data.

The algorithm calculates the barycentric coordinates $(u, v)$ of the intersection between a ray and a triangle [13]. A point, $T(u, v)$, on the triangle is given by:

$$T(u, v) = (1 - u - v)V_0 + uV_1 + vV_2, \tag{4.1}$$

where $V_0$, $V_1$ and $V_2$ are the three vertices of the triangle. A ray, $R(t)$, with origin $O$ and normalized direction $D$ is defined as:

$$R(t) = O + tD. \tag{4.2}$$

In this case $O$ is the eye position and $D$ is the gaze direction for the fixations. The intersection between a ray and a triangle is given by $R(t) = T(u, v)$. Combining (4.2) and (4.1) then gives us:

$$O + tD = (1 - u - v)V_0 + uV_1 + vV_2. \tag{4.3}$$

To check whether the ray intersects the triangle or not, we need the barycentric coordinates $(u, v)$ and the distance $t$. These can be found by rearranging the terms of (4.3) and solving the resulting linear system of equations:

$$\begin{bmatrix} -D, & V_1 - V_0, & V_2 - V_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - V_0. \tag{4.4}$$

The barycentric coordinates must fulfill $u \geq 0$, $v \geq 0$ and $u + v \leq 1$ for the intersection to be within the triangle. Since we only consider intersections in the positive gaze direction, $t \geq 0$ must also be fulfilled. If these conditions are all satisfied, we have an intersection.

Each fixation is checked for intersections with all targets, and the result of this is a vector of booleans indicating an intersection with one or more targets. The intersections are sorted according to their priority value and only the intersection with highest priority (lowest value) is saved. The index value (starting at one, as most index values do in Matlab) of the intersected target is stored alongside each fixation. If no target is intersected the value stored is zero.

## 4.6 Data selection

The user of the VTEye application might want to select subsets of the data that has been read. If there is, for example, a certain situation in the trial

where something interesting happens, the user can select only that section of the data to inspect and analyze. Since the variables regarding the trial itself (times, distances, situations, etc.) reside in the simulator data, the selection is based on this data. The user can choose any variable from the simulator set and, using the logical operators $=$, $<$, $>$, and $\neg$ (not), form an expression to select a subset of the data. Up to six logical expressions can be used for selection, and expressions are combined using logical AND. If the user for example wishes to select the data where the variable `time` has a value between 50 and 100, two logical expressions are required: `time` $> 50$ and `time` $< 100$.

When the user has formed one or more expressions to select a subset of the data, the simulator data is run through to find all rows where the logical expressions are true. The indexes for these rows are saved in a vector, which is then compared with the reference column in the Smart Eye dataset, to find the corresponding Smart Eye rows. These rows constitute the selected data, and the calculations of glances and related measures are based only on this data. If no subset of data is selected, the whole dataset is used.

## 4.7 Calculation of glances and eye movement measures

The final step of analyzing the data consists of calculating glances from the fixations, and calculating the eye movement measures.

### 4.7.1 Calculation of glances

A glance duration is defined as the time from the moment at which the gaze direction moves toward a target to the moment it moves away from it [10] (see section 3.3.1 for details). Thus one glance includes the transition, or saccade, to a new target, and all consecutive fixations (and implicitly the saccades between them) within that target.

To calculate glances from the fixations, the entire list of fixations is looped through. When the application encounters a fixation in a different target than the last one, a new glance is defined. The start time of the new glance is set to the start time of the first fixation in the new target (note that this is not the start time of the glance according to the ISO definition, but for practical reasons this is used as the internal representation of a glance). Likewise, the stop time of a glance is set to the stop time of the last consecutive fixation within the target. The target index of a glance is of course the same target index as that of all the fixations within the glance.

If a subset of data has been selected by the user, the glance calculation must be based only on this data. This gives rise to a problem with fixations spanning the borders of the selected data. Depending on how the application handles these

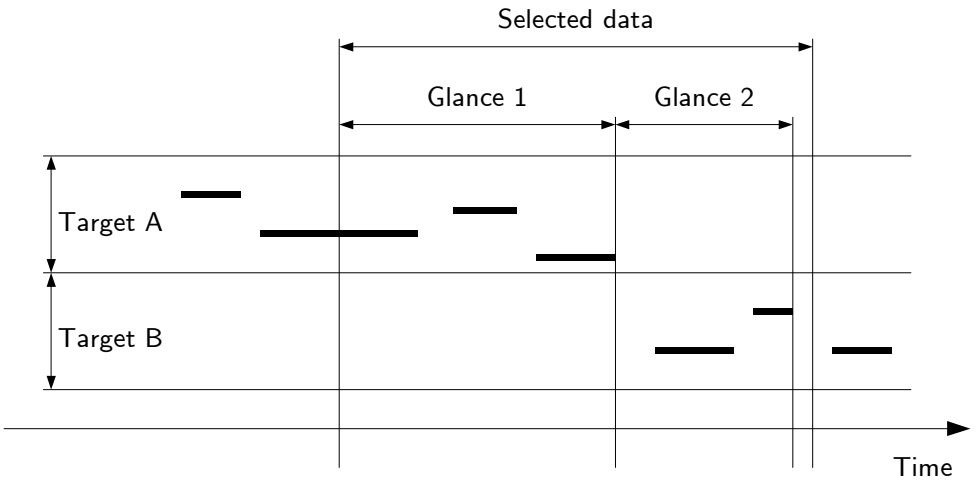



Figure 4.9: Glances with fixation spanning the border of selected data

fixations (if any), the resulting glances and measures may differ quite a lot. There are three ways to deal with these fixations:

- include them in the glance calculations,
- exclude them from the glance calculations, or
- cut them at the borders and include only the parts within the selected data.

To include or exclude fixations at the borders may skew the calculated glances and measures, especially for long fixations and a small set of selected data. Therefore the third option is used in the application. Fixations at the borders of selected data are thus cut, and the parts within the selected data are included in the glance calculations (see figure 4.9).

### 4.7.2 Calculation of measures

When start and stop times for the glances have been identified, the different eye movement measures can be calculated. Dwell times for each glance are calculated as the difference between the start and stop times. Transition times are given by the difference between the stop time of the last glance and the start time of the current glance. The start and stop times of the glances are given in samples, so to get the dwell times and transition times in seconds, they have to be multiplied with the sample frequency. The glance duration is then simply the sum of the dwell time and transition time.

Glance durations are summed for each target. The distribution of glance durations between the different targets is calculated as percentages for each target.

Max and mean glance durations are also calculated for each target. Glance frequencies for the targets are simply the number of glances in each target, and glance location probabilities are the glance frequencies divided by the total number of glances for all targets. An important extra target called *no target* is added. This target corresponds to all glances with target index zero, indicating no intersection with any of the defined targets. Not displaying calculated measures for these no-target glances could skew the data seriously.

## 4.8 Data output

When glances and related measures have been calculated, they need to be easily viewed and stored. The glance data is in tabular form and there are standard office applications that are very good at displaying, processing and graphing this type of data. To not invent the wheel over again, VTEye outputs data in MS Excel format. This allows for easy viewing and, if needed, further processing of the data.

To output an Excel-file, VTEye creates a COM server that returns a COM (Component Object Model) application object representing the entire MS Excel application. Interface objects can then be returned from the application object, and different methods can be invoked on and properties set for those objects. Some interface objects used by VTEye are: `workbook` representing an Excel-document, `sheets` representing the worksheets of an Excel-document, `range` representing a range of cells in a sheet, and `value` representing the content of a cell.

VTeye writes two worksheets to the Excel-document, one containing data based on glances, and one containing data based on targets. The glance sheet contains the target, transition time, dwell time, and glance duration for all glances. The glance sheet also contains a percentage measure of how much of the data was lost, due to Smart Eye loosing the tracking of the test subject. The target sheet contains the glance durations (total in seconds, total in percent, max, and mean), glance frequencies, and glance location probabilities for all targets. The target sheet also contains cross-tables of the number of transitions between targets and link value probabilities for pairs of targets.

## 4.9 Data visualization

### 4.9.1 Line plots

There are two different line plots in the application (see figure 4.1). The first one gives an overview of the data in 2D, plotting one Smart Eye variable and one simulator variable against time in the same plot. The variables to be plotted are chosen from two pop-up-boxes. The left y-axis shows the values of the selected

Smart Eye variable and the right one corresponds to the simulator variable. The user can zoom in and out, and navigate from side to side using a slider. The plots are composed of standard Matlab graphing objects – two `axis` objects and two `line` objects. Since the simulator and Smart Eye data may be of different size, it is important to scale the plots correctly, so that the time values on the x-axes of the two lines correspond.

The second line plot graphs the selected data, if any data has been selected. This plot draws the x and y components of the gaze direction as a line in 3D, which can also be rotated to give a more spatial feel for the data. To know which variables to plot, the application retrieves the gaze direction variables from the settings window. In this plot, the identified fixations can also be drawn. This gives the user some feedback to how the fixation identification has performed, and if the fixation settings need to be changed. Drawing and rotating 3D-plots use considerably more computing power than 2D-plots, so both the gaze direction and the fixation plots can be turned off using check-boxes.

### 4.9.2 Targets plot

Targets are plotted in the target editor using an `axis` object and a number of `patch` objects, one for each target. The targets are plotted in 3d and can be rotated (see figure 4.2). In the target plot the user can also choose to plot the gaze direction directly among the targets. The gaze direction is drawn as a line with origin at the eye position and extending out a distance in the gaze direction. The gaze direction can be either the average gaze direction for a fixation, or the gaze direction for any given time step in the raw data. This is selected using radio-buttons, and the fixation or time step is selected using a slider. This gives the user an intuitive tool to inspect the gaze direction in relation to the defined targets, for any given time step or fixation in the data.

# Chapter 5

# Validation

To check if the application functions as intended, a simple validation procedure was executed in the simulator. In this way both the VTEye application and the coordinate system setup was checked at the same time.

Data from an eye tracking system is not easily validated. The original undistorted signal, i.e. the exact direction the subject is looking in, is very hard to obtain. This makes validation hard, since you cannot simply compare the resulting signal with the original one and analyze the difference.

## 5.1 Method of validation

One way of getting a rough estimate of the original signal is by telling the test subject too look at a number of known positions at certain times. In this way you know where the subject is supposed to look, but whether he or she actually looks there is not certain. When a person is told to look at a specific target for a period of time, will the gaze be firmly fixed on the target or will it drift around? Because vision and attention are partly automatic systems a person may perceive that he or she is fixating at exactly the same spot, when in reality the eyes are moving around unintentionally.

This way of obtaining an original signal to compare the results with was used in a simple validation procedure in the simulator. The test subject was told to fixate the middle of a specific target area in the cabin and hold the gaze there until told to fixate at the next target. The five targets to look at were the left mirror, instrument panel, center console, center mirror, and right mirror. The targets were fixated for ten seconds each. The data was then run in the VTEye application. Because of limited access to the simulator at the time (a new driving compartment was fitted in the simulator), the validation was performed with only two test subjects.

The targets used in VTEye are a set of targets measured by hand in the cabin of the simulator. Since no advanced measuring equipment has been used, the positions of these targets may contain errors in the region of a few centimeters. Furthermore, the sizes of the targets used in the tests are the actual sizes of their corresponding objects in the cabin (approximated as quadrilaterals), no oversize targets are used.

## 5.2 Results of the validation

Running the data from the trials in VTEye results in the glances shown in table 5.1 and 5.2. Glances resulting from the unfiltered signal are shown to the left, and using a 15 sample median filter to the right. A velocity threshold of 125°/s and a duration threshold of 100 ms was used for the fixation identification. The correspondence with the original signal is shown as a percentage at the bottom of the tables. The correspondence is calculated in a binary fashion, either it is a hit or a miss. A miss of a target by merely a few millimeters is thus treated as a total miss.

Noise filtering produces a more stable signal and reduces the number of glances identified. This is most likely due to the fact that noise filtering causes a reduction in flickering between two targets, which can clearly be seen in the data. For example, glance 9–14 in the left part of table 5.1 is replaced with one single glance when noise filtering has been performed. The correspondence between the original and resulting signal does not seem to be much affected by noise filtering, but a bigger statistical material would be needed to establish this clearly.

The correspondence values (mean correspondence is 76 %) indicate that the setup of the coordinate system and the functionality of the VTEye application are not totally off. The errors can have a number of different sources. One likely source of error is the Smart Eye system itself, where accuracy is highest when the test subject is facing straight ahead and decreasing with increasing gaze angles. Another likely error is the position of the targets, where especially the position of small targets like mirrors can have a great effect on the result. Higher accuracy in measuring target positions may eliminate this error source. The size of the targets also have effect on the result, where smaller targets means bigger risk of missing them.

Table 5.1: Glances for test subject 1

| No noise filter | | | Median 15 samples | | |
|---|---|---|---|---|---|
| No. | Dur. | Target | No. | Dur. | Target |
| 1 | 0.52 | left mirror | 1 | 7.53 | left window |
| 2 | 4.03 | left window | 2 | 2.80 | left mirror |
| 3 | 0.25 | left mirror | 3 | 10.18 | speedometer |
| 4 | 2.25 | left window | 4 | 10.22 | middle console |
| 5 | 3.27 | left mirror | 5 | 9.53 | center mirror |
| 6 | 10.20 | speedometer | 6 | 10.82 | right window |
| 7 | 10.22 | middle console | | | |
| 8 | 9.53 | center mirror | | | |
| 9 | 1.28 | dashboard | | | |
| 10 | 1.12 | right window | | | |
| 11 | 1.83 | dashboard | | | |
| 12 | 0.58 | right window | | | |
| 13 | 0.88 | dashboard | | | |
| 14 | 3.70 | right window | | | |
| Correspondence: 68 % | | | Correspondence: 65 % | | |

Table 5.2: Glances for test subject 2

| No noise filter | | | Median 15 samples | | |
|---|---|---|---|---|---|
| No. | Dur. | Target | No. | Dur. | Target |
| 1 | 10.35 | left mirror | 1 | 10.52 | left mirror |
| 2 | 10.27 | speedometer | 2 | 10.18 | speedometer |
| 3 | 2.03 | middle console | 3 | 2.07 | middle console |
| 4 | 7.77 | dashboard | 4 | 7.78 | dashboard |
| 5 | 0.32 | windshield | 5 | 10.23 | center mirror |
| 6 | 9.92 | center mirror | 6 | 10.03 | right mirror |
| 7 | 0.30 | right window | | | |
| 8 | 9.78 | right mirror | | | |
| Correspondence: 85 % | | | Correspondence: 86 % | | |

# Chapter 6

# Discussion

## 6.1 WCS and eye movement measures

The first part of the objective was to set up a WCS for the cabin, and in that way connect the Smart Eye system with the real world in a consistent way. With the developed calibration pattern fixture, the procedure of setting up the WCS has improved greatly. Data can now easily be compared between different trials as the coordinate system remains stable and consistent. Positions of objects in the drivers cabin can be measured relatively easily using a simple tape measure. For better accuracy however, more sophisticated measuring equipment is recommended. At Autoliv, for example, a digitizing arm is used for the measurements, resulting in extremely accurate positions.

A set of highly usable eye movement measures are defined in the ISO 15007 standard. These measures, which are based on the concept of glances, are used as output from the VTEye application. The measures can be used in different research areas, including attention and drowsiness studies, two important areas studied at VTI. This fulfills the second part of the objective. However, some commonly used measures are not addressed in the thesis. Blinks for example are very informative in drowsiness studies, but are not analyzed at all in the application.* One reason for this is that because of the update frequency of the Smart Eye system it is hard to measure the duration of blinks in an exact way. Fixations are identified, but only used for calculating the glances whereas some researchers may be more interested in analyzing the fixations. Blinks and in-depth analysis of fixations are left as possible future extensions.

*Even though blinks are not used in the application, drowsiness can be studied using other measures. For example glance frequencies to different targets can be used to study whether the driver is actively searching the environment or is less active – which can be a sign of sleepiness.

## 6.2 VTEye

The main objective of the thesis was to develop an application for analysis of eye movement data from the simulator. This has resulted in the VTEye application. VTEye reads files from the simulator program and outputs a number of eye movement measures. The application can be used for different purposes, studying attention, distraction and sleepiness for example.

### 6.2.1 Programming environment

Matlab was used for programming VTEye, and this turned out to work very well. Matlab is a very high level programming language and does not allow certain things you may find in a lower level language like C++. Pointers and memory management cannot be used in Matlab for example. Despite this, Matlab is actually quite powerful and flexible, and most things can be done with it. Reading binary files, parsing xml-trees, setting up GUIs and handling COM objects works very well in VTEye for example.

One downside with using Matlab for programming is that the user needs Matlab to run the program. A Matlab program can actually be compiled and run on a computer with no Matlab license, but this requires the quite extensive *Matlab Component Runtime* to be packaged with the program and installed on the computer. A C++ program would probably have a much smaller footprint, but also require much more work when it comes to visualizing and analyzing data, areas where Matlab's numerous built-in tools are very useful.

Object oriented programming (OOP) has not been used in VTEye. This may be a downside when it comes to the possibility of future extensions and improvements, where OOP is generally a good thing. Though Matlab claims to be an object oriented programming language, it is not used in this way very often. C++ or Java would probably be a better choice when it comes to OOP.

### 6.2.2 Usability

VTEye can be used to analyze eye movement data from trials in the VTI Simulator III. The program is built around the structure of the binary files produced in the simulator program. These files contain data from two different sources – the simulator itself and the Smart Eye system. It would be a good idea to also be able to use different types of files as input. For example, some user might want to analyze Smart Eye data directly from the Smart Eye system and not combined with simulator data. This cannot be done at the moment.

The application is fast and easy to use in most cases. Entering targets can be perceived to be a bit tricky as the coordinates must be entered in the right order to produce a valid target. This is however easily inspected in the targets plot and can be redone if needed. Selecting the wrong data description file does

not generate any errors, but makes the data look abnormal, which can be seen in the data overview plot. Since the data is visualized in a number of way it is often easy to see if something is wrong.

As there is an absence of similar applications, a comparison of VTEye and other applications is hard to do. One major thought in developing the application has been to create a GUI that feels familiar and similar to other Windows programs. Loading and saving files and targets use standard dialog boxes for example. No knowledge of Matlab is required to use the program.

## 6.3 Conclusions

The coordinate system defined in the simulator, the measures selected as output, and the VTEye application fulfills the three objectives of the thesis. The validation procedure shows a mean correspondence of 76 % which is not a great number, but video based eye tracking is not an exact science today. The application is merely a tool to analyze data, if the source data itself does not contain enough information, the application will have to make do with what is there.

With VTEye researchers at VTI have a tool for analysis of eye movement data, something that they did not have before. Hopefully the application will be used and extended, making eye movements and visual behavior an important factor in the research using Simulator III. My time at VTI has been very interesting and meaningful, and I sincerely hope that my work can come to some use.

## 6.4 Future work

There are a number of possible extensions and future improvements that can be made to the VTEye application. Some I have already thought of myself, but not been able to implement due to the ever present lack of time. Here follows a list of some possible future work extending the work that has been done in this thesis.

- *Analysis of blinks.* Being able to analyze blink duration and frequency would make the application more usable in research regarding sleepiness. Blink durations are hard to analyze with the sample frequency of the Smart Eye system, but the blink frequency should be possible to obtain.

- *Extended file format support.* Extended support for different file formats would make the application more flexible. Reading and analyzing eye tracker data separately, without any simulator data would make the application usable in other areas than the research in the simulator at VTI.

- *More sophisticated fixation identification algorithms.* To make fixation analysis more exact, more sophisticated algorithms could be used. There are a

number of different algorithms for fixation identification (see [17] for example), and a selection of those could be implemented in the application.

- *Batch processing of files.* When a trial is run in the simulator, many test subjects may be used. The possibility of batch processing files from the same trial with the same settings, would speed up the analysis of a whole trial. Aggregation of measures into measures regarding the whole test population could also be a usable function.

- *Using an object oriented programming language.* If the application would be used a lot and extended with new functionality, migrating from Matlab to a more powerful and scalable programming language like C++ would probably be a good idea. This would make it possible to use all the advantages of object oriented programming when it comes to modularity and scalability for example.

- *Scanpaths in 3D.* One request that has not been implemented is the use of so called scanpaths. The path the eyes of a test subject takes during an entire trial could be drawn on the screen together with the targets. This could be a good way of getting an overview of where the test subject has looked during the trial.

# Bibliography


[1] Anna Bjällmark and Matilda Larsson. Mjukvaruutveckling inom eye tracking och visuell perception hos personer med asperger syndrom. Master's thesis, Linköping University, 2005. LiTH-IMT/BIT20-EX–05/395–SE.

[2] Björn Blissing. Visualising the visual behaviour of vehicle drivers. Master's thesis, Linköping University, Norrköping, 2002. LITH-ITN-MT-EX–2002/19–SE.

[3] Roger H. S. Carpenter. *Movement of the Eyes*. Pion, London, 2nd edition, 1988.

[4] Andrew T. Duchowski. A breadth-first survey of eye tracking applications. *Behavior Research Methods, Instruments, and Computers*, 34(4):455–470, 2002.

[5] Andrew T. Duchowski. *Eye Tracking Methodology - Theory and Practice*. Springer, London, 2003.

[6] Staffan Nordmark et al. The new vti driving simulator. multi purpose moving base with high performance linear motion. In *DCS 2004 Europe - driving simulator conference*, pages 45–55. Inrets-Renault, 2004.

[7] Darren R. Gitelman. Ilab: A program for postexperimental eye movement analysis. *Behavior Research Methods, Instruments, and Computers*, 34(4):605–612, 2002.

[8] J. Gu, M. Meng, A. Cook, and M G. Faulkner. Analysis of eye tracking movements using fir median hybrid filters. In *ETRA '00: Proceedings of the 2000 symposium on Eye tracking research & applications*, pages 65–69, New York, NY, USA, 2000. ACM Press.

[9] ISO. *Road vehicles – Measurement of driver visual behaviour with respect to transport information and control systems – Part 2: Equipment and procedures*. 2001. ISO 15007-2:2001(E).

[10] ISO. *Road vehicles – Measurement of driver visual behaviour with respect to transport information and control systems – Part 1: Definitions and parameters*. 2002. ISO 15007-1:2002(E).

[11] Petter Larsson. Automatic visual behavior analysis. Master's thesis, Linköping University, 2002. LITH-ISY-EX-3259-2002.

[12] John Lee, Michelle Reyes, and Daniel McGehee. A literature review of cognitive distraction. Technical report, University of Iowa, 2004. SAVE-IT Phase I Final Reports, Task 5.

[13] Tomas Möller and Ben Trumbore. Fast, minimum storage ray-triangle intersection. *Journal of graphics tools*, 2(1):21–28, 1997.

[14] Keith Rayner. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422, 1998.

[15] SAE. *Definitions and Experimental Measures Related to the Specifications of Driver Visual Behavior Using Video Based Techniques*. 2000. SAE J2396.

[16] Dario D. Salvucci. *Mapping eye movements to cognitive processes*. PhD thesis, 1999. Chair-John R. Anderson.

[17] Dario D. Salvucci and Joseph H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the Eye Tracking Research and Applications Symposium*, pages 71–78, New York, 2000. ACM Press.

[18] Sensomotoric instruments website. `http://www.smi.de`, July 2007.

[19] Smart Eye AB, Göteborg. *Smart Eye Pro 3.5 User Manual*, 2006. Revision 175.

[20] Smart Eye AB, Göteborg. *Smart Eye Pro 4.0 technical whitepaper*, 2007.

[21] Matthew Smith and Harry Zhang. A literature review of visual distraction research. Technical report, Delphi Electronics & Safety, 2004. SAVE-IT Phase I Final Reports, Task 7.

[22] Manbir Sodhi, Bryan Reimer, and Ignacio Llamazares. Glance analysis of driver eye movements to evaluate distraction. *Behavior Research Methods, Instruments, and Computers*, 34(4):529–538, 2002.

[23] Birgitta Thorslund. Electrooculogram analysis and development of a system for defining stages of drowsiness. Master's thesis, Linköping University, 2003. LIU-IMT-EX-351.

[24] Tobii technology website. `http://www.tobii.com`, July 2007.

[25] Trent Victor. *Keeping Eye and Mind on the Road*. PhD thesis, Uppsala University, 2006.

[26] Vti website. `http://www.vti.se`, June 2007.

www.vti.se
vti@vti.se

VTI är ett oberoende och internationellt framstående forskningsinstitut som arbetar med forskning och utveckling inom transportsektorn. Vi arbetar med samtliga trafikslag och kärnkompetensen finns inom områdena säkerhet, ekonomi, miljö, trafik- och transportanalys, beteende och samspel mellan människa-fordon-transportsystem samt inom vägkonstruktion, drift och underhåll. VTI är världsledande inom ett flertal områden, till exempel simulatorteknik. VTI har tjänster som sträcker sig från förstudier, oberoende kvalificerade utredningar och expertutlåtanden till projektledning samt forskning och utveckling. Vår tekniska utrustning består bland annat av körsimulatorer för väg- och järnvägstrafik, väglaboratorium, däckprovnings-anläggning, krockbanor och mycket mer. Vi kan även erbjuda ett brett utbud av kurser och seminarier inom transportområdet.

VTI is an independent, internationally outstanding research institute which is engaged on research and development in the transport sector. Our work covers all modes, and our core competence is in the fields of safety, economy, environment, traffic and transport analysis, behaviour and the man-vehicle-transport system interaction, and in road design, operation and maintenance. VTI is a world leader in several areas, for instance in simulator technology. VTI provides services ranging from preliminary studies, highlevel independent investigations and expert statements to project management, research and development. Our technical equipment includes driving simulators for road and rail traffic, a road laboratory, a tyre testing facility, crash tracks and a lot more. We can also offer a broad selection of courses and seminars in the field of transport.

vti

HUVUDKONTOR/HEAD OFFICE
LINKÖPING
POST/MAIL SE-581 95 LINKÖPING
TEL +46(0)13 20 40 00
www.vti.se

BORLÄNGE
POST/MAIL BOX 760
SE-781 27 BORLÄNGE
TEL +46 (0)243 446 860

STOCKHOLM
POST/MAIL BOX 6056
SE-171 06 SOLNA
TEL +46 (0)8 555 77 020

GÖTEBORG
POST/MAIL BOX 8077
SE-402 78 GÖTEBORG
TEL +46 (0)31 750 26 00