

# AN OPEN SOURCE HEAD-EYE TRACKING SUITE FOR IMMERSIVE ENVIRONMENTS: POINT OF REGARD

David Morris, Karina Rodriguez-Echavarria, David Arnold  
*Faculty of Management and Information Science, University of Brighton  
Watts Building, Moulsecoomb, Brighton, U.K.*

## ABSTRACT

Immersive environments offer significant potential for engaging users in new ways of receiving information and learning. Although, commercial head-eye trackers offer advantages when compared to Open Source Software (OSS) solutions, their use for research presents the same problem as many proprietary software and hardware: restricted access to data and protocols. To overcome these problems, this paper describes the development of OSS which was designed to be used with either commercial or open source hardware. Particular focus was on designing the software from the ground up, to work with immersive environments. As a result, the **Point of Regard (PoR) head-eye tracking software** uses an open source library which takes as data input a users' head position, head orientation and eye data from any device giving the freedom to the developer for manipulating the resulting combined data.

KEYWORDS: Head-Eye Tracking, Immersive 3D Environments, Open Source

## 1. INTRODUCTION

Results from previous eye tracking studies (Outing S. and Ruel L. 2004) have highlighted the fact that users' can recall information more accurately when this is received in a multimedia format. Hence, immersive environments offer potential for engaging users in new ways of receiving information and learning. Some well known examples are the 180°-360° screens and the CAVE which are increasingly being adopted as a medium for transmitting information, specially in museums and other learning focused institutions.

Despite their popularity, there is a real need for more research on user interaction within this type of interfaces. Information on the user's gaze or "point of regard" in an immersive 3D environment offers great potential for: i) understanding how users behave and experience this transmission medium as well as ii) enhancing the human-computer interaction with it.

Within the realm of head and eye tracking technology costs for hardware and software required for tracking is prohibitively expensive. There are a few exceptions to this, such as the open eye tracker project: openeyes (Dongheng Li & Parkhurst 2006). This system provides both an open-hardware design and a set of OSS tools to support eye tracking using consumer grade hardware; for example Wiimotes (Lee 2008).

On the other hand, there have been recent efforts on lowering the cost of commercial eye tracking products (Kumar M. 2006). In addition, commercial companies still offer better functionalities as a result of their investment in R&D. For example, the ASL EYEHEAD™ Integration (Engineering Systems Technologies GmbH & Co. KG 2008) allows for the tracking of a users point of regard across 20 different surfaces within a 3D space. However, this feature is not standard within all head and eye tracking products.

Yet commercial head-eye trackers' obvious advantages, their use for research present the same problem as many proprietary software and hardware: restricted access to data and protocols. As a result, data is often hard to retrieve and use in real time. Since the software is closed source, adding support for different data to be outputted is prevented since extensions cannot be written.

To overcome these problems, this paper describes the development of OSS which was designed to be used with either commercial or open source hardware. Particular focus was on designing the software to work with immersive environments such as screens in arbitrary configurations. As a result, the **Point of Regard (PoR) head-eye tracking software** uses an open source library which takes as data input a user head position, head orientation and eye data from any device to produce either a 2D reference on a plane or provide the 3D vector. This gives the freedom to the developer for manipulating the resulting combined data.

The paper is organized as follows. Section 2 describes background information for head and eye tracking, followed by an overview of the development of the software in section 3. Section 4 presents an application to demonstrate it's use. Finally, section 5 discusses conclusions and further work.

## 2. BACKGROUND

Head-eye tracking technology has been used for over a century to study the cognitive processes of humans when performing different tasks. Although, it was not until around 20 years ago that the first PC based head-eye tracker solutions started emerging in the consumer market. Duchowski (2002) presents a comprehensive overview of the use of this technology. Broadly, head trackers can be split into different categories according to the type of sensor they use:

**Magnetic Tracker:** This type tends to be electro-magnetic trackers with a sensor and a transmitter station. The range of them varies depending on the strength of the transmitter. Magnetic trackers provide more accuracy than optical cameras as they are position tracking devices which also provide 3D orientation. The disadvantage of this type of trackers is that they are connected to the user via a trailing wire and the user has to wear a sensor, generally on the user's head. Figure 1 shows the Polhemus Fastrak™ magnetic tracker.



Figure 1: Magnetic Tracker device

**Optical Tracker:** This type functions by tracking the user's head within the images taken from optical cameras. This combines well with optical-camera eye tracking and has the advantage of been wireless. The disadvantage is that it is less accurate than electro-magnetic position trackers. Also they require line of sight between the camera and the subject been tracked.

**Ultrasonic trackers:** This type utilizes a transmitter of ultrasonic sound waves generally from three speakers towards the receiver. The receiver has three microphones to detect the ultrasonic wave, so that the position can be triangulated. This is done by using the known position of the transmitter, the time the waves are sent out and the time they reach each of the microphones on the receiver. On the contrary to magnetic trackers, the accuracy of ultrasonic devices are not affected by displays or ferrous materials nearby. However, they require a line of sight between the transmitter and the receiver.

Eye tracking has been performed with various technologies over the years such as magnetic induction (Joseph Wilder & Kaur 1999), corneal reflection (Monty 1975), contact lenses (Ditchburn & Ginsborg 1953), Purkinje reflection tracking (Joseph Wilder & Kaur 1999) and limbus eye tracker (Joseph Wilder & Kaur 1999). However, some of these methods are quite intrusive involving the attachment of nodes to the eyes, chin rests to insure the head stays still and large heavy head mounted camera. Obviously these invasive techniques can quickly become tiresome or uncomfortable for the user (Dongheng Li & Parkhurst 2006).

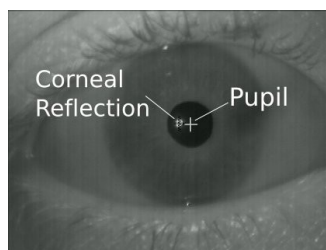


Figure 2: Eye tracker output

In recent years, small lightweight cameras have allowed for the introduction of relatively cheap eye trackers which track either the reflections of the objects found in the eye, known as “Purkinje” images or the border of the cornea and sclera known as the “Corneal Limbus” image. Attention has also been paid to the use of web cameras as cheap and easy means for eye tracking which do not require users to wear any equipment. Eye trackers usually output information related to the pupil position within the eye, using different methods to calculate. For example, the ISCAN ETL-600™ utilizes the corneal reflection to calculate the pupil position as shown in figure 2.

## 3. OVERVIEW OF POINT OF REGARD SOFTWARE

An open and object oriented architecture was designed for the Point of Regard software suite. This was based mainly on two requirements: Enabling tracking for any type of immersive environment as well as, taking into account the data outputs that head-eye trackers - despite their type - can provide.

Figure 3 illustrates an overview of the Point of Regard (PoR) architecture. This architecture was designed so any head-eye tracker can be used with it, as it utilizes a higher level library which interfaces with low-level libraries to drive the hardware.

The PoR suite consists of:

1. Head-Eye tracker low level libraries:
  - Head tracker library: FreeTrack library for the Polhemus Fastrak USB™.
  - Eye tracker driver: EyeTrack utility library to convert data from the ISCAN ETL-600™ on the serial bus, into the format required by the main program.
2. LibPoR: The main library which calculates the point of regard.
3. Applications: These make use of the LibPoR library to provide different functionalities, such as controlling, viewing or using data. Examples of these are:
  - gPoR: GTK GUI which allows for ease of configuring, calibrating and controlling libPoR.
  - ViewPoR: Provides a representation of the data.
  - Graphic Application: These can be any 3D graphic application which makes use of the libraries.

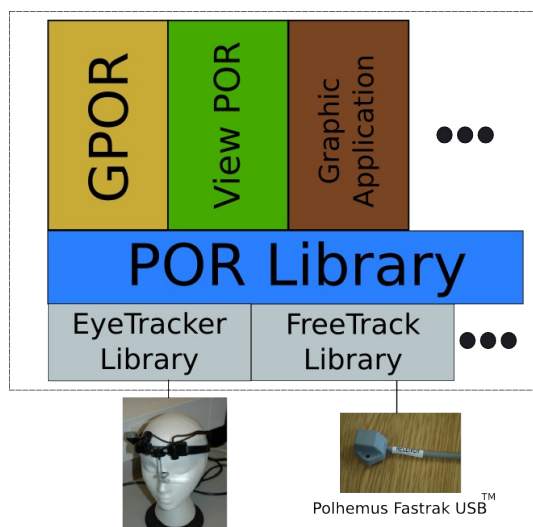


Figure 3: Overview of POR software architecture

Initially, the tracker that was chosen to integrate in the PoR software suite was the Polhemus Fastrak USB magnetic head tracker and the ISCAN ETL-600™ lightweight optical camera eye tracker. The Polhemus Fastrak was selected as the hardware provided the position of the users head in 3D space as well as the yaw, pitch and roll. Moreover, its communication protocol has already been published (InterSense Inc. 1999) making it a suitable device to use for data capture. The ISCAN ETL-600™ eye tracker provides software which outputs via a serial port the movement of the user's eye horizontally and vertically. As previously mentioned, the architecture was designed so the software drivers can be interchanged with other head and eye trackers drivers, as they simply provide the data to the PoR library. The following subsections will describe in more detail each of the elements of the architecture.

### FreeTrack Library

The FreeTrack library is a user space device driver which interacts with the Polhemus Fastrak. It was developed in C and makes use of the libusb library for access to the USB subsystem. The main functionalities of the library are reading from the device: i) the users' head position and ii) head orientation. This is done by writing a data packet to the USB down endpoint associated with the device requesting the transmission of the users' information. The requested information is returned via the USB up endpoint, where it is parsed and stored in a C data structure.

The FreeTrack library allows the polling of single data records or an auto poll mode. The latter provides upto 60 data sets per second. The current limitation of the library is that it requires the loading of firmware for the Fastrak via a Windows XP machine for legal reasons.

### EyeTrack Library

The software provided by the ISCAN ETL-600™ eye tracker was used for accessing the eye data. This data, which is outputted via serial I/O, includes both the horizontal and vertical users' pupil position minus the corneal reflection position. The EyeTrack library constantly reads from the device through a POSIX thread, and stores the latest data within a C data structure. This allows calling the program to get a near to live record of the eye movement.

Since the data is simply read off the serial buffer, the application supplying the data can easily be changed to other manufacturers' software suites. Alternatively another library could be created to act as an adapter which could be dynamically linked to other eye tracking software, such as the open source solution provided in the OpenEyes project (Dongheng Li & Parkhurst 2006).

## PoR Library

The PoR library is where the data collected from the FreeTrack and EyeTrack libraries are combined to calculate the point of regard. The PoR library utilizes the user's head position and orientation to create a vector and uses a ray-tracing algorithm to determine the point of intersection with the previously defined display area. For this, it is necessary to calibrate the head-eye tracker as well as define the space on which the immersive environment will be displayed before making any tracking. This space can have any configuration ranging from planar screen, to 180° screen, or CAVEs.

The eye calibration requires the use of a test card of five points with a known distance from the user. The distance between the users pupil and corneal reflection on both horizontal and vertical directions (P-CRH and P-CRV) is recorded for each data point, along with the distance from the user. For this, the eye tracking has been divided into a grid consisting of four zones. This is because the corneal reflections moves a different amount in relation to the pupil in each of these zones when the users is looking around. Therefore, it is required to calculated and store a different angle for each zone. The formula for calculating the angle is:

$$a = \frac{(PCR1 - PCR2)}{(\text{inv}\tan(g/d))}$$

Where:  
**a** is the calculated angle for each zone,  
**PCR1** is the point in the corresponding segment of the eye,  
**PCR2** is the center point,

**g** is the gap between the points on the axis been calculated, and

**d** is the distance from the user to the center point.

The correct **a** is selected depending upon the zone which the user corneal reflection has moved into, and is multiplied against the eye data to produce a vector. The resulting vector is added to the vector from the user's head data and used in the ray tracing algorithm.

The space which the ray intersects is in fact the screens which have been loaded into the program. The 3D positions of each screen's corners is stored within an XML data structure, as if in a 3D model - with (0,0,0) been the transmitter. Each of these positions are then parsed and loaded into the program as two triangular planes. Two assumptions are made at this point, although they could easily be changed i) each screen has a rectangular shape and ii) each screen is flat. Having screens represented as planes allow tracking across CAVEs, video walls and arbitrary planes scattered around the user.

During the tracking, the point of regard is converted into an X,Y position on the screen based on its resolution. The library produces as output an X,Y position along with the ID associated with the screen. The data generated from the PoR library can be logged into a plain text format for analysis at a later stage. Alternatively, it can be fed into another system to provide optimizations e.g. level of detail or culling.

## gPOR

This GTK GUI application provides an easy to use interface for controlling the underlying PoR functionality. The application functionalities include: i) loading the XML configuration file of the space, ii) user calibration, iii) configuring the angle which the head tracking equipment sits on a subjects head, iv) controlling the mode of operation of the head-eye tracker and v) specifying the file to output the data.

## viewPoR

This basic application provides a visualisation of the head-eye tracking data from one or more users. It offers the functionality of overlapping the data from users over a still image or a video.

## 4. CASE STUDY

In order to demonstrate the use of the PoR software suite we experimented with an interactive cinema experience called "Crossed Lines" (see figure 4). This experience consists of a user interacting with nine independent, but interlinked, video streams displayed on a 60" plasma using a telephone device. As such, each of the video streams plays a segment of video when selected, sometimes triggering other segments. The requirements were: i) track the users



Figure 4: "Crossed Lines" application

attention to each video, and ii) determine if users' watched the video they had selected or were distracted by other videos.

Initially, the XML file defining the plane was created with one single square in space. Using gPoR the eyes of the user are calibrated, the XML file loaded. The data collected indicates which video the user watched for each second. This is used to determine if the user was looking at the video they had selected.

Analysing the data proved to be difficult since different users would take a different route through the experience. Instead, the data was broken down into section surrounding the start of each video. ViewPoR then highlights which video(s) distracted the user or if they paid attention to the current video.

## 5. CONCLUSION AND FURTHER WORK

This paper describes ongoing research to provide an open source head-eye tracking software solution for immersive environments. As such, the PoR software suite provides a solid foundation for providing a feature rich head and eye tracking solutions to user-interaction researchers. The software suite is open and flexible as it has been designed so multiple hardware configurations can be used with the core library. This provides several advantages, such as:

- Using the hardware which is most suitable for the environment of operation rather than the hardware which works with a preferred software product.
- Accessing and manipulating the data rather than being restricted by the proprietary software.
- Easily extending the software for additional functionality.

The suite has been released as OSS which allows anyone to make changes and extend the product.

Further work will involve calculating and improving the accuracy level. Another improvement will involve better analysis and visualisation of the results.

## ACKNOWLEDGEMENT

This work has been conducted as part of the EPOCH Network of Excellence (IST-2002-507382). Our thanks to: Sarah Atkinson for providing a case study application: "Crossed Lines" as well as Aidan Delaney for his contribution on the viewPoR application.

## REFERENCES

- Engineering Systems Technologies GmbH & Co. KG, (2008) ASL Eyehead Integration. <http://www.est-kl.de/hardware/eyetracking/asl/eyeheadintegration.html>. Last accessed: February 2008
- Ditchburn, R. W. & Ginsborg, B. L., 1953. Involuntary eye movements during fixation. *Journal of Physiology* Volume 119, pp 1–17.
- Dongheng Li, J.B. & Parkhurst, D. J., 2006. openeyes: a low-cost head-mounted eye-tracking solution. *Proceedings of the 2006 symposium on Eye tracking research & applications*. New York, NY, USA, pp. 95-100.
- Duchowski, A. 2002. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments and Computers* 34, 4, 455–470.
- InterSense Inc. (1999) "User Manual for IS-300 and IS-300 Pro Systems " <http://mvl.mit.edu/ISS/IS-300Manual.pdf>
- Joseph Wilder, George K. Hung, M. M. T. & Kaur, M., 1999. Eye tracking in virtual environments, <http://vehand.engr.ucf.edu/handbook>
- Kumar M. (2006) Reducing the Cost of Eye Tracking Systems, <http://citeseer.ist.psu.edu/757849.html>
- Monty, R. A., 1975. An advanced eye-movement measuring and recording system. *American Psychologist* Vol 30, pp 331-335.
- Outing S. and Ruel L., 2004. The Best of Eyetrack III: What We Saw When We Looked Through Their Eyes, <http://www.poynterextra.org/eyetrack2004/main.htm>
- Young, L. R. & Sheena, D., 1975. Eye movement measurement techniques. *American Psychologist* Vol 30, pp 315-330.