5-2007

LIMBUSTRACK: STABLE EYE-TRACKING IN IMPERFECT LIGHT CONDITIONS

Wayne Ryan
Clemson University, wryan@clemson.edu

LIMBUSTRACK: STABLE EYE-TRACKING IN IMPERFECT LIGHT CONDITIONS

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Science

---

by
Wayne Ryan
May 2007

Accepted by:
Dr. Andrew Duchowski, Committee Chair
Dr. Brian C. Dean
Dr. Wayne Madison

Abstract

Eye tracking research appears to have reached critical mass in terms of technological advances as well as sustained theoretical, applied, and even mainstream interest.There is now a wealth of interesting empirical results, performed by researchers in numerous domains, e.g., Psychology, Industrial Engineering, Marketing, and Computer Science [Duchowski 2003], to name a few. Most studies, however, are often conducted in the laboratory under various environmental constraints. Meanwhile, relatively little work has been done to collect eye movement data during performance of *natural tasks*, i.e., outside the lab, not to mention outside the building. Indeed, due to reliance on Infra-Red (IR) illumination of the eyeball for eye tracking precision purposes it is often difficult to track the eyes in the presence of sunlight.

Some work on collecting eye movements during dynamic activities exists, such as driving, basketball foul shooting, golf putting, table tennis, baseball, gymnastics and other situations [Rayner 1998]. However, such results come at a price, literally and figuratively. Wearable eye trackers are scarce, expensive, and/or uncomfortable (due to clumsy or heavy headgear or heavy backpacks containing laptops, cameras, etc.). We are aware of only one serious effort at development of a cheap, accurate, wearable eye tracker: the open source openEyes project [Li et al. 2006], motivated by the wearable eye tracker developed by Pelz et al. [2000]. The openEyes wearable eye tracker has matured to allow operation in the visible spectrum [Li and Parkhurst 2006]. However, its method of ocular feature detection is such that it is prone to failure in variable lighting conditions. To address this deficiency, we have developed a cheap (< \$1,000) wearable eye tracker that is as comfortable to wear as a pair of plastic safety glasses. At the heart of our development are novel techniques that allow operation under variable illumination. Our wearable eye tracker provides acceptable accuracy (about 2° visual angle) at suitably fast processing speed (close to video frame rates) allowing eye movement analysis during natural tasks.

Acknowledgments

I can do nothing without God.

These people have been particularly helpful:

Andrew T. Duchowski : Writing and background research.

Stan Birchfield : Understanding of image processing.

Lea Benson : Was there when I needed...

Table of Contents

Page

Title Page . . . . **i**

Abstract . . . . **iii**

Acknowledgments . . . . **v**

List of Figures . . . . **ix**

1 INTRODUCTION **1**

2 BACKGROUND **5**
- 2.1 Eye Tracking Technological Advancements 8

3 APPARATUS DESIGN **13**
- 3.1 Hardware Design 14
- 3.2 Software Design 16
  - 3.2.1 Video Pre-Processing 16
  - 3.2.2 Calibration 20
  - 3.2.3 Visualization 22
- 3.3 Limbus Tracking 24
  - 3.3.1 Image Pre-Processing 24
  - 3.3.2 Feature Detection 25
  - 3.3.3 Ellipse Fitting 26

4 EXPERIMENTAL VALIDATION **29**

5 FUTURE WORK **35**
- 5.1 Proposed improvements 35
- 5.2 View Stitching 37
- 5.3 Object Registration 38

6 CONCLUSION **41**

Bibliography **43**

List of Figures

Figure Page

1.1 Our DIY wearable eye tracker 3

3.1 Illustration of ocular anatomy 13
3.2 Eye tracker assembly 15
3.3 Software architecture 17
3.4 Image pre-processing 19
3.5 Using the gradient direction 20
3.6 Automatic calibration 21
3.7 Sample frames from video 23
3.8 Limbus tracking strategy 24
3.9 Basic emulation of Li et al.'s [2006] *Starburst* algorithm. 25
3.10 Ellipse fitting 28

4.1 Calibration Accuracy 30
4.2 Accuracy of calibration re-use 32
4.3 Sample frames from video 33

5.1 Artistic conceptualization of view stitching 37
5.2 Automatic object registration 38

Chapter 1

INTRODUCTION

Eye tracking technology has advanced significantly, especially in recent years. Classical techniques involve the use of electromechanical and electromagnetic apparatus that are highly invasive, difficult to use, and applicable only to laboratory research.

The predominant concerns in the development and evaluation of an eye tracker are: cost, invasiveness, ease of use, availability, accuracy, and versatility [Morimoto and Mimica 2005]. Accuracy is possibly the most significant. The use of a contact lens fitted with a magnetic coil provides accuracy of approximately 0.08° visual angle. For interactive use, accuracy of 1° is desirable and 2° is often acceptable.

Through recent developments in digital video technology, minimally invasive eye trackers have introduced the possibility of mainstream use. Currently, the most popular eye tracking techniques involve the use of IR cameras. Developments in image processing, including those introduced here, are making possible the use of eye trackers easily constructed with off-the-shelf visual spectrum video cameras, thus reducing the cost while simultaneously increasing ease of use, availability, and versatility.

Video based eye tracking approaches can be classified by the range of illumination in which they operate, i.e., infra-red or the visible spectrum. Hardware setups may also be divided into two categories, head mounted, and remote.

Remote eye trackers use a remote camera, often built into a monitor. They use image processing to locate the subject's head, eyes, and pupil. After a short calibration the software can use this information to determine the participant's gaze point with respect to screen coordinates. The primary limitation is that head movement distorts the calibration. The stationary nature of these systems restricts them to situations involving television or computer monitor use.

Head mounted eye trackers, though inherently mobile, introduce a different set of problems. The primary concern is intrusiveness. Historically video cameras have been rather large and heavy. Recently digital video cameras have become much smaller, lighter, and more affordable, increasing the feasibility of the head mounted design.

Infrared eye trackers exploit the ability to provide light sources invisible to the user thus constraining the problem domain and improving eye tracker accuracy. The primary limitation of these systems is the high cost of custom hardware and restriction to indoor applications where IR interference is minimal.

Visible spectrum eye trackers do not require special circuit boards or IR cameras, and may be functional in outdoor lighting. This reduces the cost of the hardware, and improves versatility.

A head mounted eye tracker that remains accurate in the presence of drastic light changes is desirable. As a person walks from room to room in a building, or simply turns the head, shadows across the face and reflections in the eye change drastically. Such changes in brightness, shadows, and specular reflections can prevent an eye tracker form yielding useful data. Li et al. [2006] developed openEyes, an open-hardware design for a digital eye tracker that can be built from low-cost, off-the-shelf components, and a set of open-source software tools for digital image capture, manipulation, and analysis. Concomitantly, they developed the *Starburst* algorithm for tracking the pupil in the visible spectrum. A description of how it performs in the presence of drastic light changes was not provided in their evaluation.

Chapter 2 provides a survey of previous eye tracking research, establishing Li et al.'s design as the most suitable platform for wearable eye tracker development, as shown in Figure 1.1. Chapter 3 continues with a detailed description of our hardware and software implementation contrasting it with that of Li et al. Chapter 4 provides objective results of our implementation indicating the viability of our design. Our novel techniques improve the performance of the *Starburst* algorithm in the presence of variable illumination.

In spite of our improvements many challenges remain. Through our experience in this project we have gained significant insight. Chapter 5 discusses remaining problems and proposes potential solutions.

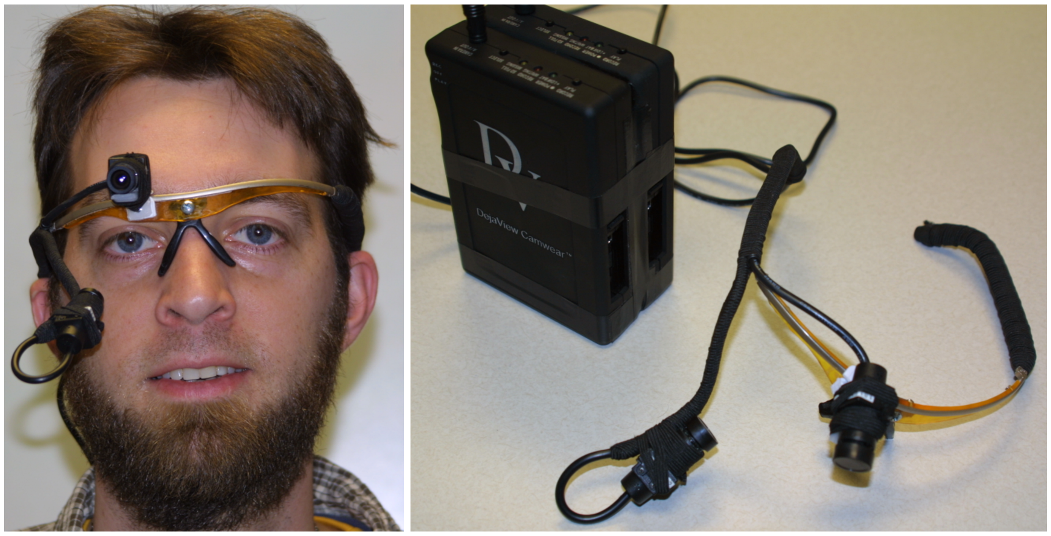


Figure 1.1: Our “DIY” wearable eye tracker from relatively inexpensive (~$700) off-the-shelf components.

Chapter 2

BACKGROUND

In 1935 Guy T. Buswell published *How People Look at Pictures: A Study of The Psychology of Perception in Art* [Buswell 1935]. This work is important to the history of eye tracking because it is the first demonstration of eye gaze recordings over complex scenes, e.g., photographs of paintings, patterns, architecture, and interior design. However, viewing of pictures projected on a laboratory display still constitutes something of an artificial task.

Wearable eye trackers allow collection of eye movements in more natural situations, usually involving the use of generally unconstrained eye, head, and hand movements. Classic work in this area was performed by Land et al. [1999]. The aim of the study was to determine the pattern of fixations during the performance of a well-learned task in a natural setting (making tea), and to classify the types of monitoring action that the eyes perform. Results indicate that even automated routine activities require a surprising level of continuous monitoring. A head-mounted eye movement video camera was used that provided a continuous view of the scene ahead, with a dot indicating foveal direction with an accuracy of about 1°. Foveal direction is always close to the object being manipulated, and very few fixations are irrelevant to the task. Roughly a third of all fixations on objects can be definitely identified with one of four monitoring functions: *locating* objects used later in the process, *directing* the hand or object in the hand to a new location, *guiding* the approach of one object to another (e.g., kettle and lid), and *checking* the state of some variable (e.g., water level). Land et al. conclude that although the actions of tea-making are ‘automated’ and proceed with little conscious involvement, the eyes closely monitor every step of the process. This type of unconscious attention must be a common phenomenon in everyday life.

Investigating a similar natural task, Land and Hayhoe [2001] examined the relations of eye and hand movements in extended food preparation tasks. The task of tea-making was compared to the task of making peanut butter and jelly sandwiches. In both cases the location of foveal gaze was monitored continuously using a head-mounted eye tracker with an accuracy of about 1°, and the head was free to move. In the tea-making study three subjects moved about the room to locate objects required for the task; in the sandwich-making task seven subjects were seated in one place, in front of a table. The eyes usually reached the next object in the sequence before any sign of manipulative action, indicating that eye movements are planned into the motor pattern and lead each action. The eyes usually fixate the same object throughout the action upon it, although they often move on to the next object in the sequence before completion of the preceding action. Specific roles of individual fixations were found to be similar to roles in the tea-making task. Land and Hayhoe argue that, at the beginning of each action, the oculomotor system is supplied with the identity of the required objects, information about their location, and instructions about the nature of the monitoring required during the action. Eye movements during this kind of task are nearly all to task-relevant objects, and thus their control is seen primarily 'top-down', and influenced very little by the 'intrinsic salience' of objects. According to Land and Hayhoe, the eyes provide information on an 'as needed' basis, but relevant eye movements usually precede motor acts they mediate by a fraction of a second. Eye movements are thus in the vanguard of each action plan, and are not simply responses to circumstances.

Ballard et al. [1995] investigated the use of short-term memory in the course of a natural hand-eye task. They focused on the minimization of subjects' use of short-term memory by employing deictic primitives through serialization of the task with eye movements (e.g., using the eyes to "point to" objects in a scene in lieu of memorizing all of the objects' positions and other properties). The authors argue that a deictic strategy in a pick-and-place task employs a more efficient use of a frame of reference centered at the fixation point, rather than a viewer-centered reference frame which might require memorization of objects in the world relative to coordinates centered at the viewer.

A head mounted eye tracker was used to measure eye movements over a three-dimensional physical workplace block display, divided into three areas, the *model*, *source*, and *workspace*.

The task assigned to subjects was to move and assemble blocks from the source region to the workspace, arranging the blocks to match the arrangement in the model area. By recording eye movements during the block pick-and-place task, the authors were able to show that subjects frequently directed gaze to the model pattern before arranging blocks in the workspace area. This suggests that information is acquired incrementally during the task and is not acquired *in toto* at the beginning of the tasks. That is, subjects appeared to use short-term memory frugally, acquiring information just prior to its use, and did not appear to memorize the entire model block configuration before making a copy of the block arrangement.

In a similar block-moving experiment, Smeets et al. [1996] were able to show that horizontal movements of gaze, head, and hand followed a coordinated pattern. A shift of gaze was generally followed by a movement of the head, which preceded the movement of the hand. This relationship is to a large extent task-dependent. In goal-directed tasks in which future points of interest are highly predictable, Smeets et al. hypothesize that while gaze and head movements may decouple, the actual position of the hand is a likely candidate for the next gaze shift.

A more recent example of such intentional visual attention was demonstrated by Pelz et al. [2000]. Using a commercially available wearable eye tracker from ASL, Pelz et al. were able to show intentionally-based eye movements, which they termed "look-ahead" eye movements. In this experiment, a subject donned the wearable computer and eye tracking rig (the computer was worn in a backpack). During a simple hand-washing task, recorded eye movements showed that gaze moved to a location (soap dispenser) prior to the movement of the hands to the same location.

Following their work identifying look-ahead eye movements, Babcock and Pelz [2004] built their own wearable eye tracker from off-the-shelf components. Their aim was to spark an open-source movement to develop practical eye tracking software and hardware. They provided practical tips on constructing the tracker, opening the door to open-source software development. This corneal reflection eye tracker, mainly constructed from inexpensive components (a Radio Shack parts list was made available at one time), was one of the first Do-It-Yourself eye trackers.

Babcock and Pelz's early wearable eye tracker suffered from two significant problems. First, it required the inclusion of one expensive video component, namely a video multiplexer, used to synchronize the video feeds of the scene and eye cameras. Second, the system relied on somewhat dated (by today's standards) video recording equipment (a Sony DCR-TRV19 DVR). Nevertheless, the system served its purpose in fostering a nascent open-source approach to (wearable) eye tracking that is still influential today. The next section reviews the state-of-the-art of open source eye tracking software and hardware.

2.1 Eye Tracking Technological Advancements

Eye trackers can be categorized by the same classification used to describe motion capture ("mocap") technology used in the special effects film industry (e.g., see Menache [2000]). Similarities between the two applications are intuitive since the objective of both is recording the motion of objects in space, relative to a fixed camera. In eye tracking, the object measured is the eye, while in mocap, it is (usually) the joints of the body.

Motion capture/eye tracking devices can be classified by four technology types:

- Electromechanical. In mocap applications, sensors are placed on the skin or joints. In eye tracking, sensors are placed on the skin around the eye cavity. This type of eye tracking is known as electro-oculography (EOG).

- Electromagnetic. Mocap sensors of this type are often composed of orthogonal wire coils measured in a surrounding electromagnetic field. In Virtual Reality applications, for example, such sensors are attached to the limbs and head. Electromagnetic eye trackers function similarly—a metallic stalk is fixed to a contact lens worn by the viewer.

- Video-oculography (VOG). These types of motion trackers are based on optical motion analysis of photo sequences (video frames) and are widely used in special effects film, video, and game production. A camera is used to record raw motion, which is subsequently processed by manual or digital means. To delineate certain types of eye movements (e.g., fixations, saccades, smooth pursuits), manual VOG analysis relied on trained observers advancing the video, frame by frame, labeling the observed motion of the eye. This rather tedious approach is today being replaced by automatic, signal processing means whose objective is to calculate the motion path of the imaged pupil.

- Video-based, IR reflection. Relying on reflective markers (worn by actors), this particularly popular type of mocap is free of cable tethers that encumber electromagnetic and electromechanical trackers. IR light is usually used since it is invisible to the human eye and hence non-distracting. Eye trackers can benefit from one of four types of IR reflections, two from the front and rear surface of the cornea, two from the front and rear surface of the crystalline lens. All four reflections are known as the Purkinje images [Crane 1994]. Most video-based IR reflection eye trackers use the first Purkinje image—the corneal reflection—as a fixed reference point for calculation of the Point Of Gaze (POG) (see below).

EOG tracker accuracy is limited to the number of sensors placed on the skin. These trackers are invasive, may be difficult to use if sensors are not placed consistently, and tend to be noisy due to the nature of the signal being measured. Although this type of eye tracker has been used effectively in situations where the placement of a camera may not have been possible, (e.g., see Kaufman et al. [1993]), it has generally been superseded by the camera-based tracker mainly due to the latter's improved positional accuracy (cf. Hayhoe et al. [2002]).

Early gaze recordings, dating as far back as the late 19^{th} century (see Duchowski [2003]), used a plaster of paris ring attached directly to the cornea connected by mechanical linkages to recording pens (a basic mechanical device). This technique evolved to the use of a contact lens to which a mounting stalk is attached. Various mechanical or optical devices have been placed on the stalk attached to the lens. The most accurate method employs a wire coil that is measured as it moves in an electromagnetic field (an electromagnetic tracker). Although highly accurate, this method is of course also highly intrusive.

Modern eye tracking technology relies on cameras. Image processing methods are used to locate and track the pupil (and sometimes the corneal reflection of IR LEDs). If localization of the eye in the image frame is problematic, the frame difference method may be used for finding the eyes in the eye camera image(s). This technique, introduced by Ebisawa and Satoh [1993] and popularized by Morimoto et al. [2000], takes advantage of the retro-reflection property of the eyes. When light from a nearby IR light source enters the eye through the pupil, part of it is reflected back, in a narrow beam of light directed back toward the light source. An *on-axis* light source, located near the optical axis of the camera, produces a bright pupil image, commonly known as the "red eye" effect in photography. An *off-axis* light, located away from the camera's lens, produces a similar image, except that the pupils are dark. After taking one video frame with each light, the pupils show up clearly in the difference image. The difference image has in general a high signal to noise ratio and is easy to segment.

In addition to tracking of the pupil, the corneal reflection (glint) of the auxiliary (IR) light source allows computation of the user's gaze point in the scene, if it can be reasonably assumed that the corneal glint is fixed with respect to the translating pupil. This is the technique employed by Babcock and Pelz's [2004] wearable eye tracker (an IR LED is positioned off-axis just to the left of the eye imaging camera). A laser diode is used to project points in the user's visual field to enable calibration of the tracker to the individual and subsequent POG calculation.

During calibration, the user is prompted to look at a series of points usually laid out in a grid pattern. Pupil and corneal glint features are recorded at each calibration point and used to fit a mapping from these features to the point of gaze in the scene. The modeling of this feature-to-gaze mapping is usually not a true 3D model, and hence confounds the geometry of the user's eye with head location, if the head is free to move with respect to the camera's reference frame (as in a remote eye tracking setup). As a result, calibration tends to deteriorate when the user's head deviates from the 3D position where it was initially calibrated. Many eye tracking practitioners will perform a calibration for every user session, sometimes with multiple calibrations for a longer session. One popular remote eye tracker, the Tobii 1750, allows head motion only inside a $30 \times 15 \times 20$ cm volume. Wearable eye trackers tend to assume a stable eye image frame, provided the headgear is immobile.

Infra-red illumination offers certain benefits due to the availability of an additional trackable feature (the corneal reflection). However, it may simultaneously pose a restriction on where the tracker can be used. For example, operation in sunlight has been problematic. To allow operation in the visible spectrum, and to simplify complexity while reducing cost, Li and Parkhurst [2006] adopted the technique of limbus tracking—a modernization of traditional video-oculography, or VOG. The limbus is defined as the iris/sclera boundary. Elimination of IR illumination simplifies the tracker's design by removing several components: the IR LED as well as a voltage regulator which required additional circuitry. Provided the headgear is sufficiently stable, any fixed point in the eye image frame can be used as an acceptable reference point for POG estimation.

Li and Parkhurst's VOG tracker is limited in several ways. First, being a limbus tracker, it may not function particularly well under variable light conditions, i.e., in bright light when the contrast of the iris/sclera boundary is masked by specular reflections. Second, manual intervention is required at startup, e.g., initial eye location is entered manually. Third, being worn by an individual, human variability poses a significant problem to eye movement (scanpath) comparison between individuals. The last problem is not necessarily restricted to a VOG limbus tracker, rather it falls under the more general problem of scene registration (e.g., a well known problem in Augmented Reality). We address the first two deficiencies through novel enhancements to Li and Parkhurst's VOG tracker. By doing so, and by maintaining the cost of the tracker under $1,000, we further the applicability of this technology to the exploration of a variety of natural tasks. We propose two methodological paradigms: view stitching, and object registration to address the third problem.

Chapter 3

APPARATUS DESIGN

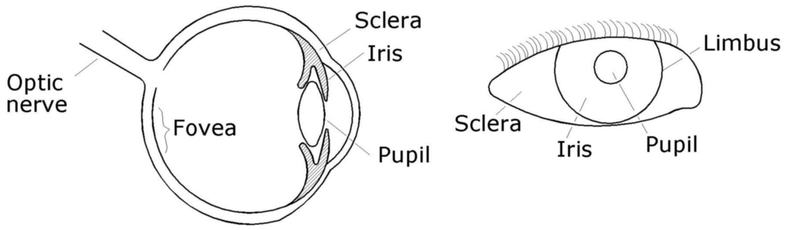


Figure 3.1: Illustration of ocular anatomy: the cross-section of the eye at left (Gullstrand's model of the eye is similar to this illustration but specifies various physical properties such as dimensions of the optical features and their indices of refraction); the idealized eye as seen by a camera at close proximity.

There are currently two basic approaches to POG calculation. The first is based on a 3D model of the eyeball—often Gullstrand's (see illustration in Figure 3.1)—and the computation of a ray emanating from a central point within the eye. The POG is then calculated (parametrically) as the intersection of the ray and some surface of interest in the environment, i.e., $p = c + t \cdot L$, where p is the 3D POG, c is the center of the cornea (modeled by a sphere), t is the parametric distance along the ray L coinciding with the optical axis. This approach requires both a geometric model of the eye (e.g., the corneal radius must be defined), and a basic (planar) model of the environment (for example, see Hennessey et al. [2006]).

The second approach, based on traditional video-oculography (VOG), was taken by Pelz et al. [2000], continued by Li et al. [2006], and is adopted as our starting point. As such, the goal is to estimate the center of the limbus (defined by an ellipse fit to the iris/sclera boundary; see Figure 3.1) and define (x, y) as the vector from some static reference point in the eye image to the limbus center (this will vary as the eye rotates).

To estimate the POG, a calibration is required where the viewer is asked to sequentially view a set of spatially distinct calibration points (markers; 9 is common) with known scene coordinates, (s_x, s_y). The POG at any point in the scene image frame is then obtained by calculation of a mathematical mapping from (x, y) to (s_x, s_y).

This method requires the collection of two synchronized video sequences, one of the eye, the other of the scene being viewed. We use the same hardware design described by Li [2006], with minimal modifications as necessary to facilitate material availability. Section 3.1 provides all specifications required for hardware replication.

There are many components to any eye tracking application. The interaction of each component can be quite confusing. Section 3.2 provides a complete picture of every component, describes its function, and discusses specific challenges encountered in the implementation of each. Two key refinements are discussed in this section. First, localization of the initial eye location is performed automatically and does not require user intervention. Second, calibration requires that scene coordinates of the calibration points are known. Unlike remote camera systems, these coordinates change with head movement. We developed an image processing technique that identifies the location of the calibration dot with respect to the scene camera.

It is important to note that both calibration and tracking contain a common component (limbus tracking). Limbus tracking is the most challenging component of the entire application. Estimation of the limbus center follows Li et al.'s [2006] *Starburst* algorithm but introduces a critical refinement. Ellipse fitting considers both the pupil/iris boundary as well as the iris/sclera boundary. This refinement enhances operation in variable light conditions. Section 3.3 provides a detailed description of our limbus tracking implementation.

3.1 Hardware Design

The apparatus is constructed entirely from off-the-shelf (OTS) inexpensive commercially available components. The simplicity of the design facilitates easy construction requiring only a minimal amount of expertise.

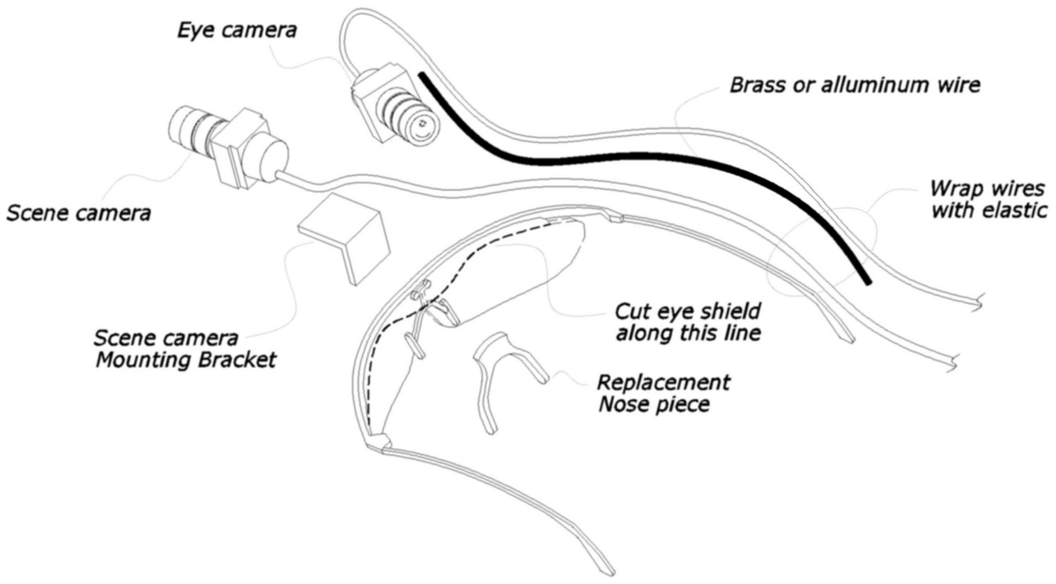


Figure 3.2: Eye tracker assembly.

Parts List:

- 1 pair of safety glasses (AOSafety X-Factor XF503) www.aosafety.com : These safety glasses were chosen over others because the one piece shield and metal frame remained rigid enough to support the cameras after lens modifications were made.
- 1 pair plastic sunglasses (AOSafety I-Riot 90714) : The thin wire nose piece of the XF503 is not adequate for our purposes. It is removed and replaced with the more comfortable nose piece of the 90714.
- 3/8 inch by 2 yards black polyester braided elastic : Wires were wrapped with elastic rather than conventional wire wrap. This elastic is available in small quantity at any local fabric store.
- 2 Phillips Pan head #8 by 1/2″type A screws : One screw was used to connect the scene camera bracket. The other was used to connect the nose piece.
- Plastic mounting bracket : This bracket was cut from scrap plastic, and used to mount the scene camera.
- 1/8″×10″round aluminum or brass rod : This rod may be purchased at any local hardware store. It was used to mount the eye camera. Temporarily attach rod, and

eye camera with zip-ties. Test and adjust camera positions. Cut rod to proper length, then wrap wires with elastic strap.

- 2 digital video minicams (Camwear Model 200) http://dejaview.no-ip.org: These inexpensive cameras are commercially available from Deja View [Reich et al. 2004]. Each Deja View wearable digital mini-camcorder uses the NW901 MPEG-4 CODEC from Divio, Inc. The NW901 single-chip CODEC enables the design and manufacture of a new class of solid-state camcorders that combine 30 fps MPEG-4 video, high fidelity AAC 2-channel audio, and up to 4-mega-pixel digital still camera functionality. The Deja View Camwear product offers continuous video capture using a solid state buffer. Video is recorded on memory sticks for offline processing.

In the next sections, we describe our eye tracking algorithm in detail. Contrasting with prior work, specifically Li et al.'s [2006] *Starburst* algorithm, we discuss our improvements toward automatic estimation of the initial eye location, automatic calibration dot detection, and enhanced tolerance to adverse lighting.

3.2 Software Design

Our algorithm is composed of three main parts: video preprocessing, calibration, and visualization. Each of these main components may be decomposed into several sub-components (see Figure 3.3). During preprocessing the two videos are converted to a usable format, synchronized, and the initial pupil location is found. The calibration process computes the coefficients of a transform function that maps eye coordinates into gaze data. The remainder of the video may then be processed allowing useful inferences to be made.

3.2.1 Video Pre-Processing

There are four steps to video preprocessing: conversion, segmentation, synchronization, and template search. We utilized the most expedient approach to conversion and segmentation. A description is provided to facilitate replication of our results. Exploration of more effective approaches is recommended. Synchronization was achieved by a method Li and

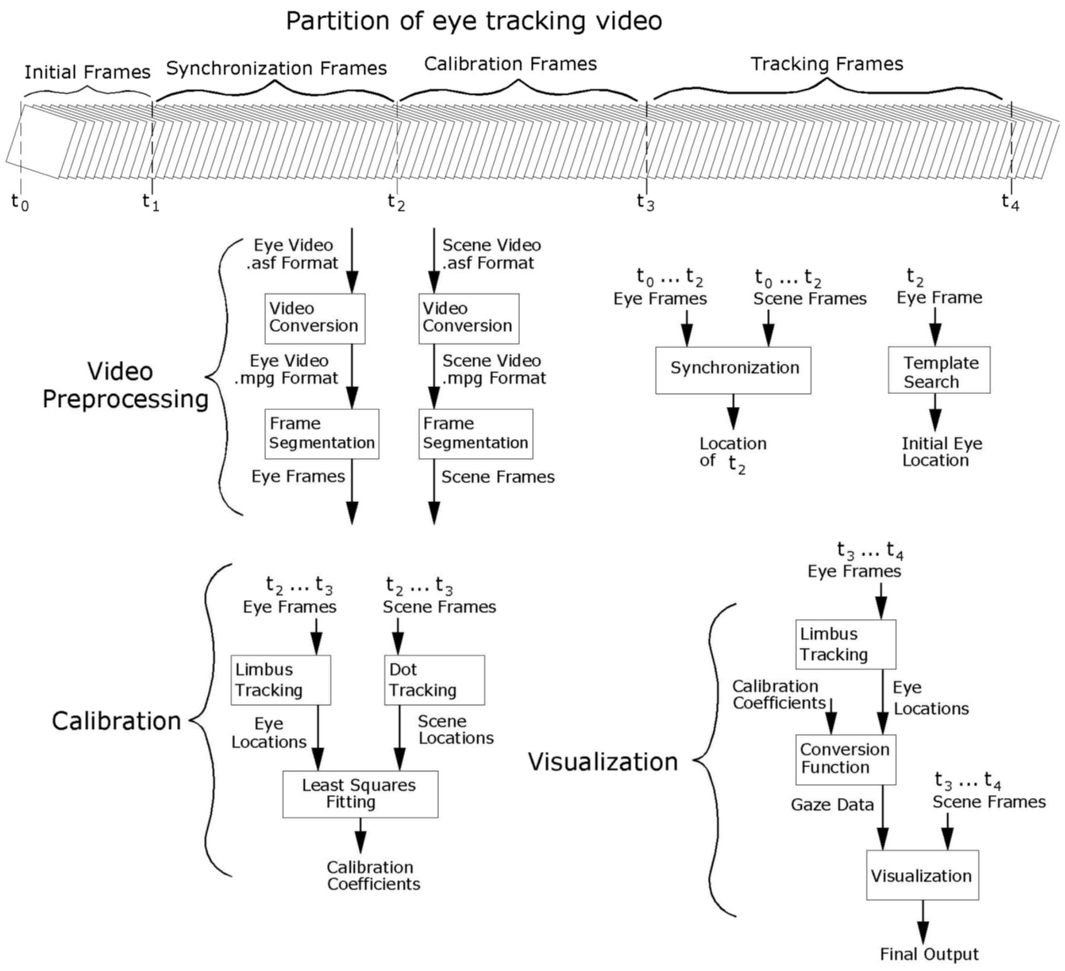


Figure 3.3: Software architecture

Parkhurst [2006] suggested. We employ a novel variant of the Borgefors' [1988] template search algorithm to identify the initial pupil location.

Conversion is necessary because the Deja View recorder stores the video in `.asf` format. This video is highly compressed, and does not facilitate convenient manipulation. Ideally we would like to access each frame individually in RGB format. We use `libmpeg2dec` to parse individual frames from the video file. Unfortunately, this library, and all others we found, is specific to `.mpg` files. We tried to find an application to convert directly from `.asf` to `.mpg` but all converters failed to make this conversion reliably. We eventually settled upon the Apex Video Converter to convert from `.asf` to `.wmv` and `ffmpeg` to convert `.wmv` to `.mpg`.

If the entire uncompressed video is stored on disk, it requires hundreds of megabytes for a video lasting only a few minutes. Such a strategy is hampered by slow disk access. Fortunately the `libmpeg2dec` application provides the (`-p`) option that facilitates the use of named pipes in the Linux environment. By using script files we were able to pipe the output of two simultaneous invocations of `libmpeg2dec` into our application. We were then able to handle the named pipes within our application as if they were regular files while preventing any uncompressed data from being written to disk.

It cannot be assumed that the eye and scene cameras begin recording at exactly the same time. It is therefore necessary to synchronize the video streams of both cameras. As suggested by Li and Parkhurst [2006], a flash of light visible in both videos is used as a marker. We have employed an LCD monitor to display calibration dots, we therefore found it convenient to flash the monitor in order to create the short burst of light necessary for synchronization. We flash the monitor again to signal the end of calibration. The light from the monitor is sufficiently bright to be automatically identified in both video streams.

Even though the *Starburst* algorithm is able to accurately find the pupil center, it only preforms a local search. It therefore needs a good starting point from which to begin searching. For frames in the middle of the video we may simply use the result from the previous frame, but for the first frame some other method must be devised. Li et al. [2006] developed an application to facilitate manual entry of the initial start point. In order to make our eye tracker easier to use we opted for a more automated solution.

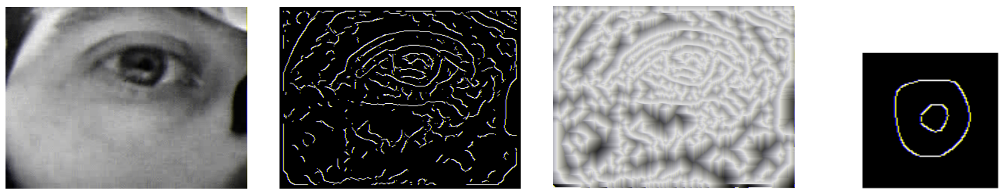

(a) raw image frame (b) Canny edges (c) chamfer image (d) limbus pattern

Figure 3.4: Pre-processing steps for pupil/limbus pattern matching.

Our automatic localization of the pupil is based on template matching as described by Borgefors [1988]. Her algorithm begins by using the Canny [1983] edge detector on both the image to be searched and a template of the object to be searched for. It then creates a chamfer image for which each pixel value is set to its distance from the nearest edge. A search map is then created by convolving the edge detected template with the chamfer image. The pixel in the search map with lowest value is the most likely location for the object.

The underlying idea of Borgefors's algorithm is computation of the average edge distance at every possible location. It finds the location at which the edges of the template are most closely aligned with edges in the image. Only edge distance is considered. The algorithm does not consider edge direction. An extreme case where this algorithm fails is in the search for a dark circle on a gray background among many bright circles. The gradient of the dark circle is directed toward the center while the gradient of the bright circle is directed away from the center. This information is however discarded after edge detection. Such loss of edge direction prevents the algorithm from distinguishing between bright and dark circles.

Our innovation derived from Birchfield's [1997; 1998] elliptical head tracking work allows the algorithm to utilize edge direction information in addition to the edge distance information. One component of the Canny [1983] edge detector computes the x and y components of the gradient for each pixel. Since gradient information is therefore already available, and knowing that the dot product of two unit vectors yields a value indicating the extent to which they are aligned, the modification is almost trivial.

First we invert the chamfer image. Next we normalize the gradients for both the template and the search image. We then augment the convolution operation with the dot product of the normalized gradient. Finally we consider the highest value in the search map to be the most likely location rather than the least. The proposed pupil localization algorithm is given in Listing 3.1.

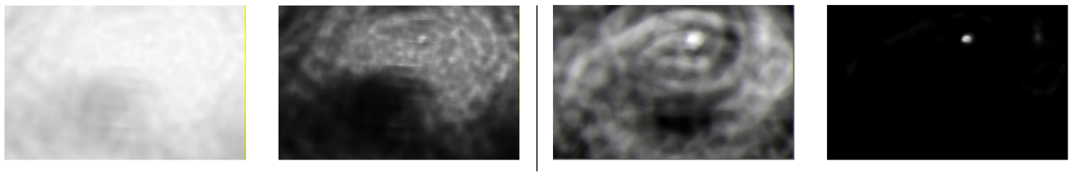


Figure 3.5: Effect of using the gradient direction to locate pixels associated with the pupil; the image pair at left considers only edge distance, the pair at right also considers gradient direction.

```
MakeSearchMap(Gradient image $G$, Chamfer image $C$,
              Template image $T$, Template gradient $GT$)
{
  for all template locations $x,y$ {
    sum = 0;
    for all locations $dx,dy \in T \neq 0$
      sum += $G(x+dx,y+dy) \cdot GT(dx,dy) \times C(x+dx,y+dy)$;
    $M(x,y)$ = sum; // $M$ is the search map;
  }
  return maximum pixel in $M$;
}
```

Listing 3.1: Algorithm for finding initial seed point for Starburst algorithm.

3.2.2 Calibration

Calibration requires mapping of the position of the pupil center (x, y) in the eye frame to the location of a calibration point in the scene frame with coordinates (s_x, s_y), under the assumption that the eye gazes at the given calibration point at the same instance. We developed an image processing technique that automatically identifies the location of calibration dots.

The coordinates of calibration points displayed on a computer monitor are obtained in the scene image by appropriate thresholding of the image, as shown in Figure 3.6. As there are many small bright spots in the scene, we must distinguish the small bright spot that is the calibration dot from all others.

The calibration dot is clearly the only bright spot on the computer screen, so we first distinguish the screen from its surroundings then distinguish the calibration dot from the screen. We do this in a five step process.

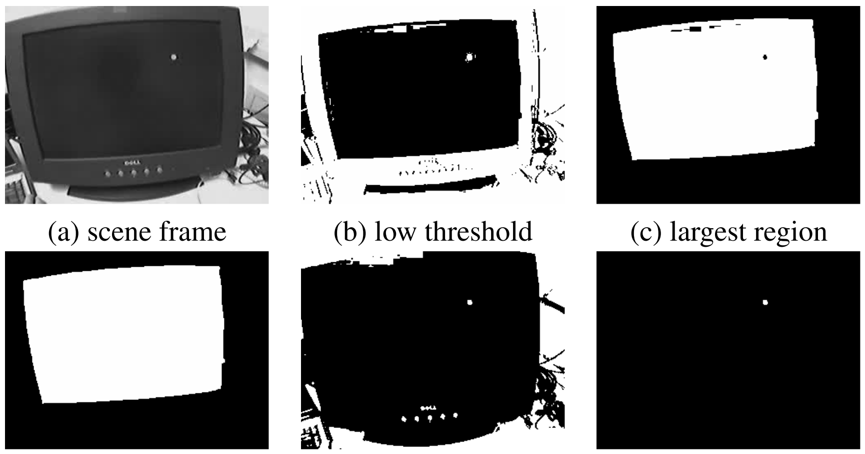
1. Apply a low level threshold to the scene frame.

2. Identify the largest dark region (the computer screen).

3. Eliminate holes in the largest region using a modified flood fill.

4. Apply a high level threshold to the original scene frame.

5. Combine results from 3 and 4 above with logical & operation.



(a) scene frame (b) low threshold (c) largest region

(d) after flood fill (e) high threshold (f) final result

Figure 3.6: Automatic calibration depends on synchronization of eye and scene camera images and automatic detection of calibration dots based on thresholding.

The viewer is assumed to fixate upon these points as they are displayed. Treating the position of the pupil center as the (x, y) vector from a static reference point in the eye image we may compute a transform function that maps coordinates of the pupil obtained in the eye image into coordinates of gaze points with respect to the scene image.

Morimoto and Mimica [2005] suggest using this second order polynomial:

$$s_x = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2, \quad s_y = b_0 + b_1 x + b_2 y + b_3 xy + b_4 x^2 + b_5 y^2. \quad (3.1)$$

The unknown parameters a_k and b_k can be computed via Lagrange's method of least squares (e.g., see Lancaster and Šalkauskas [1986]).

3.2.3 Visualization

Visualization is composed of three parts: limbus tracking, mapping, and analysis. Limbus tracking is discussed in section Section 3.3. Mapping is a straightforward application of Equation 3.1 following least-squares minimization.

Analysis is the process of converting raw gaze data into a form that allows useful inferences to be made. There has been much study as to how raw gaze data should be processed. These techniques employ knowledge from the field of signal processing as well as knowledge of the human visual system.

As with any discrete representation of a continuous phenomenon, aliasing is of significant concern. Finite impulse response (FIR) filters are commonly used as a preprocessing step to smooth the noise that exists in raw gaze data. Careful analysis of eye movements indicates that the human eye tends to remain fixed for short periods of time, termed fixations, and moves rapidly between fixations, performing quick eye movements called saccades. It is well accepted that the human visual system does not process information gathered during saccades. For this reason procedures have been developed that group gaze points pertaining to the same fixation while discarding those that correspond to saccades. This is usually done through the use of convolution filters that detect velocity and acceleration.

The spatial relationship between fixations is used to determine regions of interest ROI. Sometimes ROIs may also be manually predetermined. Once this is done we may determine which ROIs were fixated upon, and for how long. String matching algorithms have been used to compare the order in which different people fixate upon the same ROIs, scan-path analysis.

With a head-mounted eye tracker, we would also like to compare analyzed eye movements among individuals. The free movement allowed by the head-mounted design introduces sufficient variability between individuals' scene video that scene registration registration becomes a significant problem.

We are unaware of any automated technique to handle this situation. In Section 5 we propose two methodological paradigms to alleviate this issue: view stitching and object registration.

As implementation of these techniques is beyond the scope of this project we opted to use a simple 5-tap FIR filter to smooth the gaze data, and display a simple cross hair at the location of gaze as illustrated in Figure 3.7. Our analysis is therefore limited to the display of the smoothed Point Of Gaze, or POG.

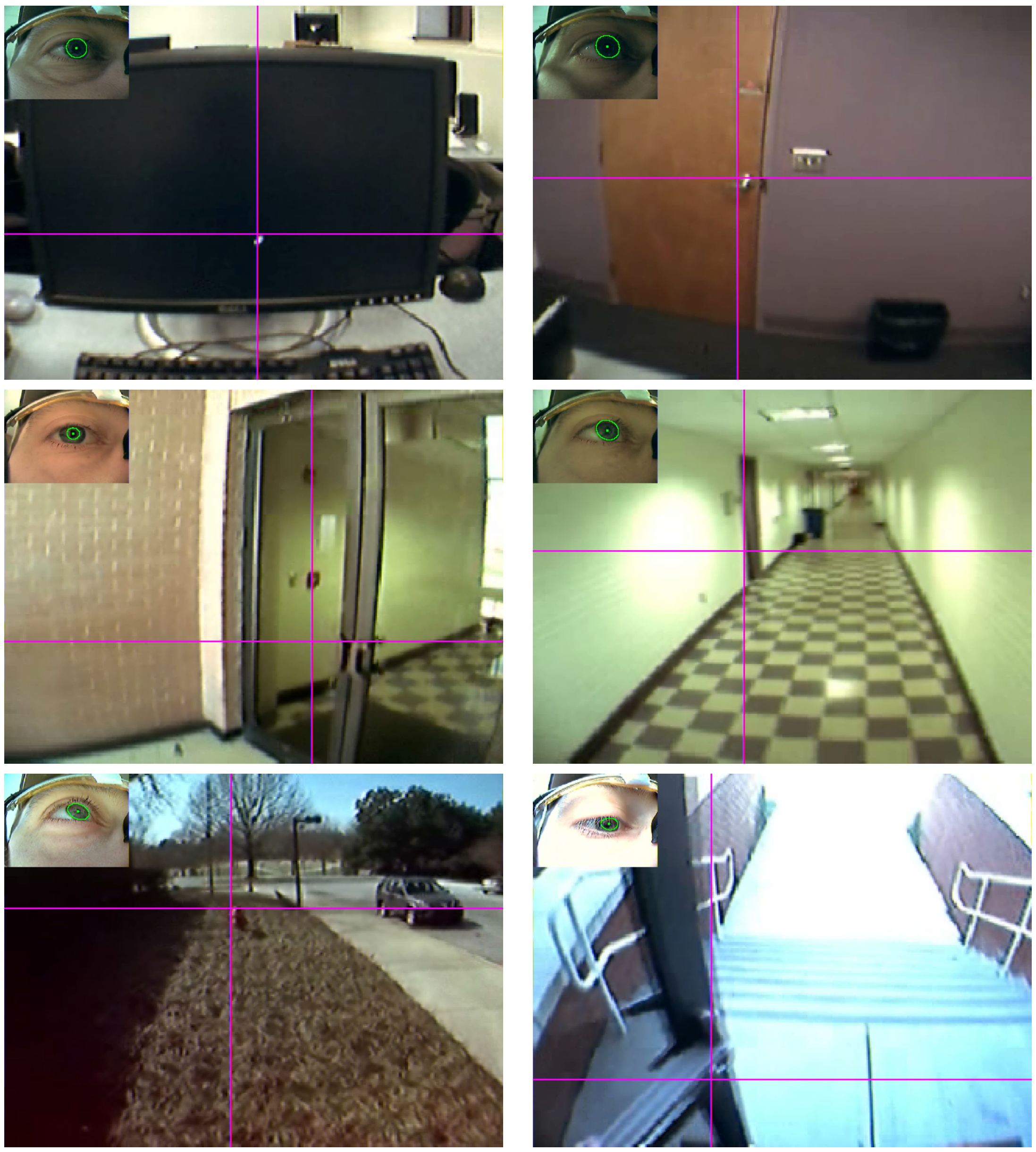


Figure 3.7: Cross hairs indicate point of gaze.

3.3 Limbus Tracking

Limbus center estimation is composed of three basic steps performed on each eye image frame (see Figure 3.8): image preprocessing, feature detection, and ellipse fitting.

During image pre-processing, the image is filtered to reduce noise and the gradient is calculated. Feature detection is a local search resulting in a set of points on the limbus or around the pupil. Ellipse fitting uses the set of feature points to define an ellipse whose center is assumed to coincide with the center of the pupil.

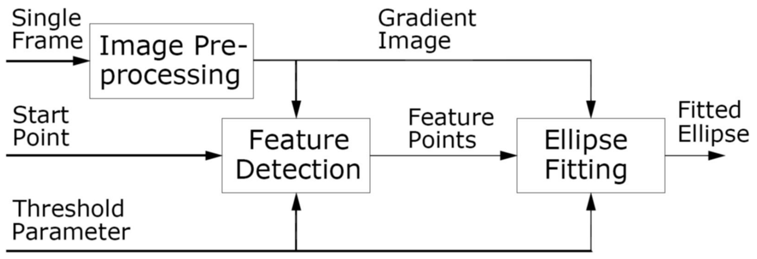


Figure 3.8: Basic strategy behind proposed limbus tracking.

3.3.1 Image Pre-Processing

It is necessary to pre-process each frame in such a way that noise in the image is reduced while retaining as much information as possible. Like the *Starburst* algorithm, we begin by convolving the eye camera image with a Gaussian filter to reduce camera shot noise. Gaussian filters are effective in reducing noise problematic to any algorithm that utilizes gradient information. Furthermore, the separable nature of the 2D Gaussian enables efficient implementation.

Most of the noise in video produced using OTS cameras is due to lossy compression. Any lossy compression algorithm can be expected to produce some problematic artifacts in the video. As more artifacts are permitted, the videos can become more compressed. The algorithms used have been specifically tuned to allow artifacts so long as they are just barely noticeable. This is commonly known as the just noticeable difference. Unfortunately image processing techniques are much more sensitive to these artifacts than the human eye.

Most compression schemes divide the image into small blocks, and preform a conversion on each block individually. The values within a block are rounded. As adjacent blocks are rounded differently, we are left with edges along block boundaries. These undesirable edges produce sharp gradients that mask the thick gradients along the edges of desired features. The Gaussian filter smooths the image eliminating sharp highly localized gradients. Thicker gradients are blurred but not eliminated. In *Starburst* the gradient of the entire image is not precomputed but rather computed during feature detection. It was speculated that doing so reduced computation. However gradient estimation is simply a convolution operation identical to Gaussian smoothing except that rather than convolving with the Gaussian, we convolve with the derivative of the Gaussian. It is therefore no more computationally expensive than noise reduction. We opted to simplify implementation by pre-computing the gradient. We speculate that our decision to do so has no significant effect upon performance.

3.3.2 Feature Detection

Limbus feature points are found in two steps. As in *Starburst*, candidate feature points are found in the first step by casting rays out from an initial seed point and terminating the ray as it exits a dark region. We determine if the ray is exiting a dark region by checking the gradient magnitude and direction (see Figure 3.9).

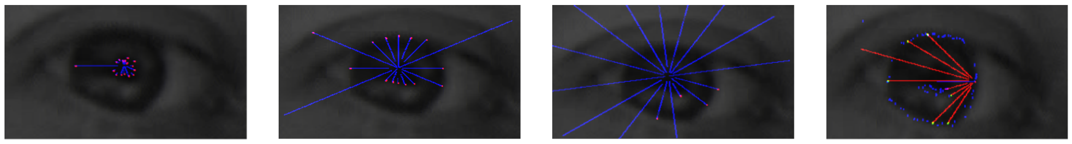


Figure 3.9: Basic emulation of Li et al.'s [2006] *Starburst* algorithm where rays are cast outward from an initial seed point assumed to lie within close proximity to the pupil.

In the second step the feature detection process is repeated with rays cast back from candidate boundary features toward the seed point. The second sub-step tends to increase the ratio of the number of feature points on the pupil contour over the number of feature points not on the pupil contour. The ray does not terminate until the magnitude of the gradient component collinear with the ray exceeds some predetermined threshold. Although this technique is quite effective and efficient in some lighting conditions it is quite sensitive to the threshold chosen. Identification of an ideal threshold is confounded by the fact that higher thresholds are more effective in bright light, giving way to lower thresholds in dim light. Our algorithm iterates through multiple thresholds to average the best results. The basic feature detection algorithm is given in Listing 3.2.

```
FeatureDetect(Gradient image $G$, Threshold $T$, Start point $S$)
{
  for $\theta$ from $0$ to $2\pi$ { // phase 1
    $L$ = $S$; // $L$ is the current location
    $\Delta_x = \cos(\theta)$; $\Delta_y = \sin(\theta)$;
    while $G(L) \cdot \Delta$ < $T$ { $L(x)$ += $\Delta_x$; $L(y)$ += $\Delta_y$; }
    push $L$ on $P$; // $P$ is a point stack
  }
  for $\theta$ from $2\pi$ to $0$ { // phase 2
    pop $L$ off $P$; add $L$ to $P'$; // $P'$ is a point set
    for $\alpha$ from $-(\theta + \frac{\pi}{6})$ to $-(\theta - \frac{\pi}{6})$
      while $G_L \cdot \Delta$ < $T$ { $L(x)$ += $\Delta_x$; $L(y)$ += $\Delta_y$; }
    add $L$ to $P'$;
  }
  return $P'$;
}
```

Listing 3.2: Algorithm for finding feature points.

3.3.3 Ellipse Fitting

The *Starburst* algorithm uses the Random Sample Consensus (RANSAC) paradigm to fit ellipses to the feature points. RANSAC was chosen due to its tolerance to outliers; outliers are common in the observed feature sets. Five points are selected at random from the feature set. These points are used in conjunction with the standard equation for a conic section as suggested by Fitzgibbon et al. [1999]: $F(\mathbf{a}, \mathbf{x}) = \mathbf{a} \cdot \mathbf{x} = ax^2 + by^2 + cxy + dx + ey + f = 0$ with $\mathbf{a} = [a\ b\ c\ d\ e\ f]^T$ and $\mathbf{x} = [x^2\ y^2\ xy\ x\ y\ 1]$ where the system of linear equations is solved for $\mathbf{a}$. We may then quickly verify that the resulting coefficients do indeed describe an ellipse. Many such ellipses are created, and the quantity of feature points within some small epsilon counted. The ellipse with the highest count is retained.

We generate random ellipses in a similar manner, but rather than evaluating them based upon a characteristic of the feature set, evaluate them based upon characteristics of the original image. Ellipse fitting itself is accomplished via the proposed algorithm given in Listings 3.3 and 3.4. We label each pixel that the ellipse passes through as bad or good depending upon the magnitude and direction of the gradient at that pixel. The percentage of good pixels is computed. The ellipse with the highest ratio is retained. Our modification allows the algorithm to be more tolerant to poorly localized feature sets.

```
EllipseFit(Gradient image $G$, Threshold $T$, Set of points $S$)
{
  repeat $C_1$ times {
    repeat $C_2$ times {
      fill $S$ randomly from $P$; // $S$ is a set of 5 points
      make $EQ$ from $S$; // $EQ$ is a 5 × 5 system of equations
      solve $EQ \rightarrow E'$; // $E'$ is a candidate ellipse
      if $E'$ is not an ellipse continue;
      if Fit($G$,$T$,$E'$) > Fit($G$,$T$,$ET$)
        replace $ET$ with $E'$ // $ET$ is a temporary ellipse
    }
    add coefficients of $ET \rightarrow E$; // $E$ is the ellipse
  }
  divide coefficients of $E$ by $C_1$;
  return $E$;
}
```

Listing 3.3: Algorithm for fitting ellipse to limbus feature points.

```
Fit(Gradient image $G$, Threshold $T$, Ellipse $E$)
{
  for every $P$ that $E$ crosses { // $P$ is a pixel in $G$
    $CT$++; // $CT$ is the count of total pixels
    let $V$ point from $CE$ to $P$; // $CE$ is center of ellipse $E$, $V$ is unit vector
    if $G_P \cdot V > T$ $CG$++; // $CG$ is the count of good pixels
  }
  return $CG$ / $CT$;
}
```

Listing 3.4: Counting number of feature points used in ellipse fit.

Starburst's ellipse fitting is further complicated by its inability to distinguish ellipses that partially span the pupil and those that partially span the limbus from those that exclusively adhere to one or the other. We suggest that the feature points be split into two groups based upon pixel luminance. Pixel luminance is used to allocate feature points into one of two equally sized bins. This form of feature point splitting allows the creation of two sets of ellipses, one corresponding to the iris/sclera limbus, the other to the pupil/iris boundary.

Figure 3.10 (a) shows the result of feature detection: points on both pupil and iris boundaries are identified. Without luminance-based delineation, average ellipses that span both pupil and iris are fit erroneously, as shown in Figure 3.10 (d), resulting from the generated ellipses fit to sets of 5 points, randomly chosen and shown in Figure 3.10 (b). Splitting into two sets of boundary points improves pupil/iris classification, as shown in Figure 3.10 (c) allowing us to properly detect the erroneous ellipse segments in Figure 3.10 (d).

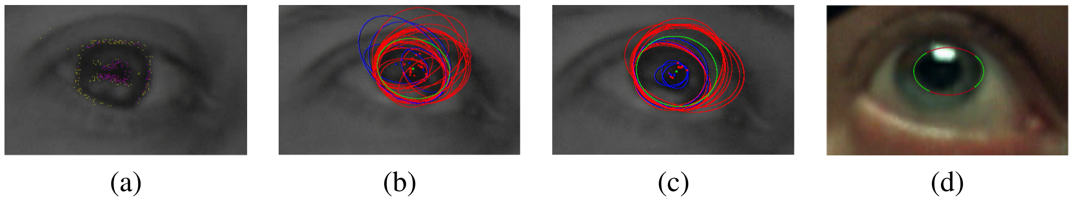


Figure 3.10: Ellipse fitting: (a) result of feature detection: points on both pupil and iris boundaries are identified; (b) ellipses generated by fitting sets of 5 randomly selected points; (c) luminance-based delineation of feature points into two sets; (d) proper detection of erroneous ellipse spanning both pupil and iris.

Chapter 4

EXPERIMENTAL VALIDATION

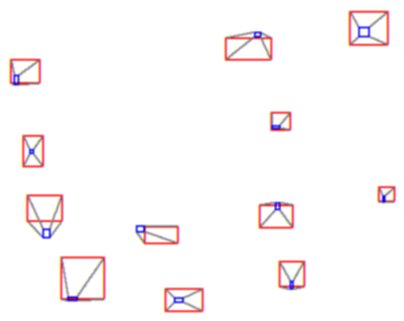
We evaluate our implementation by first calculating calibration coefficients as discussed in Section 3.2.2. We then reprocess the same video, tracking the disparity between mapped gaze points and tracked calibration dots. We calculate the Euclidean distance between the mean calibration dot center and the mean gaze point over the period of gaze dot display. This distance is our error measurement given in pixels. The Deja-View camera has a 60° field of view, with video resolution of 320×240. Therefore a simple multiplication by 0.1875 converts our units of measurement from pixels to degrees visual angle. Using this metric, we were able to track the eye in three separate videos each with an average error less than 2° (see Table 4.1).

Video No.	Average Error	Standard Deviation
#1	1.625	0.986
#2	1.803	1.066
#3	1.601	0.698

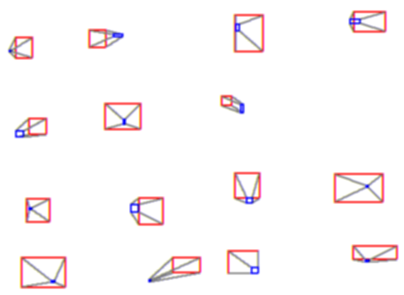
Table 4.1: Error in degrees visual angle.

For each video we displayed calibration dots at 16 locations on the computer LCD. Each dot remained stationary on the computer display for about 1.5 sec before moving to the next location. We tracked the dot with respect to the scene camera and the eye with respect to the view camera at a rate of 30 samples per second. Figure 4.1 provides a visual representation of our results. The dot tracking algorithm is sometimes unable to locate the calibration dot. In such cases the corresponding eye data is discarded.

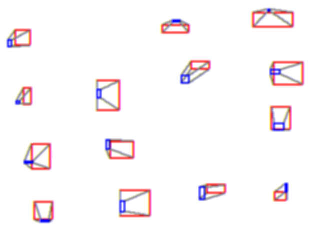
An eye tracker that allows reuse of calibration coefficients between sessions rather than recalibrating for each session is desirable. We therefore also evaluated calibrations across different sessions (i.e., using calibration coefficients from video 1 to evaluate video 2). The



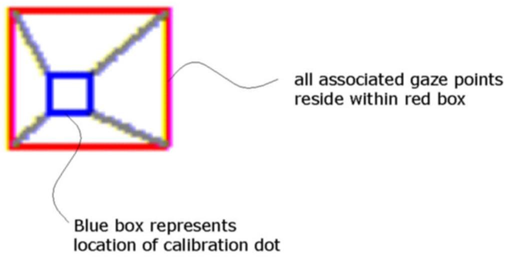
(a) video #1



(b) video #2



(c) video #3



(d) legend

Figure 4.1: A visual representation of calibration accuracy.

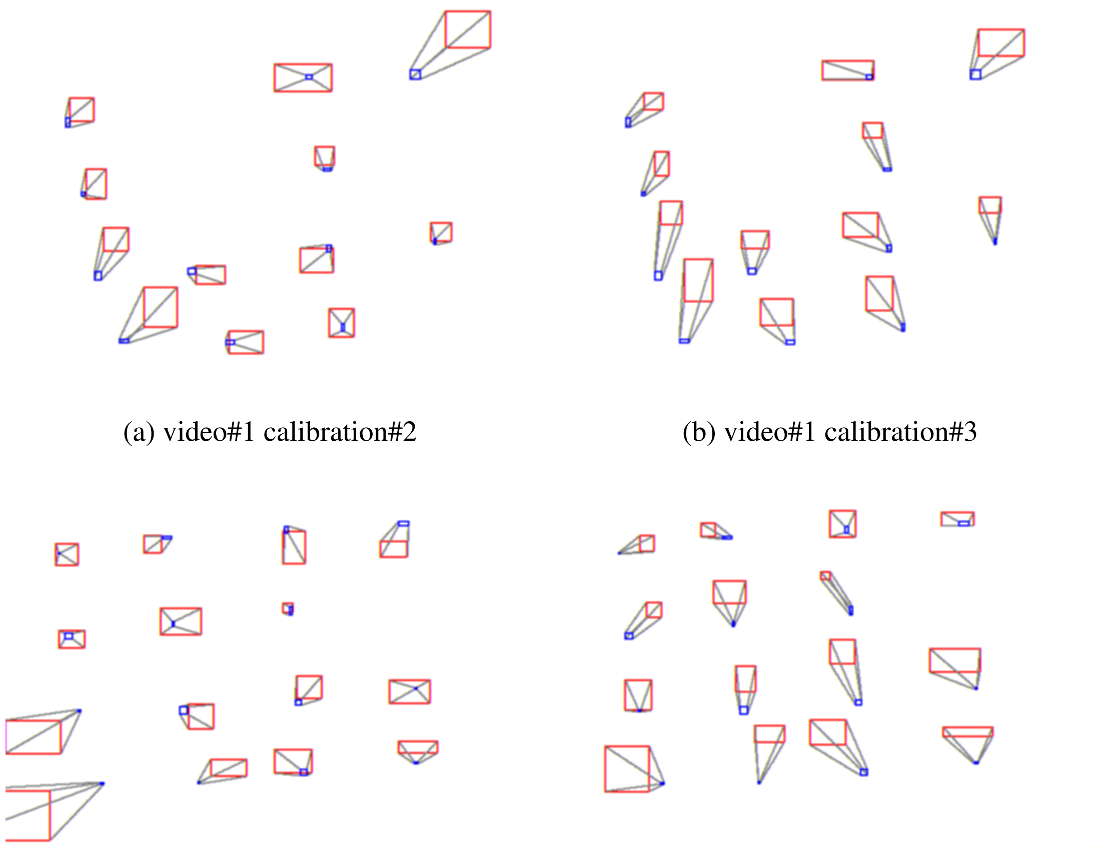
results were not quite as good but still encouraging. Using this method we were able to repeatedly track the eye with an average error less than 5° visual angle (see Table 4.2 and Figure 4.2).

Video No.	Calibration No.	Average Error	Standard Deviation
#1	#2	2.899	2.196
#1	#3	4.499	1.616
#2	#1	2.932	3.271
#2	#3	3.633	1.748
#3	#1	4.327	1.377
#3	#2	3.808	1.330

Table 4.2: Error in degrees visual angle.

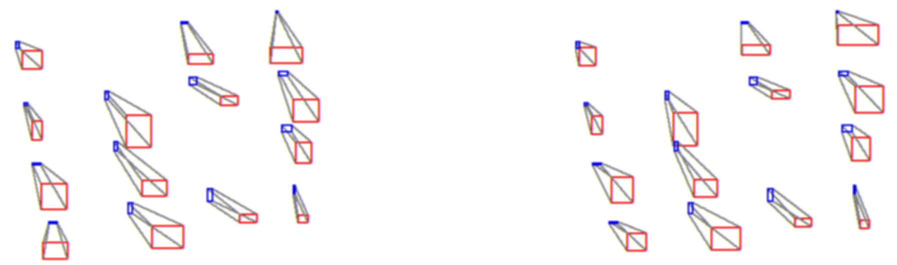
Although dynamically changing light does have a negative effect on our ability to track the eye, we were able to obtain usable data even in bright sunlight. This was primarily due to our luminance based delineation of feature points. With this additional feature our eye tracker was able to automatically switch from tracking the limbus in dim light to tracking the pupil in bright light as illustrated in Figure 4.3.

Even though we have made significant advances in the development of a visual spectrum head mounted eye tracker, we have observed several sources of error in our current implementation. Section 5 discusses these sources and suggests possible solutions.



(a) video#1 calibration#2 (b) video#1 calibration#3

(c) video#2 calibration#1 (d) video#2 calibration#3



(e) video#3 calibration#1 (f) video#3 calibration#2

Figure 4.2: A visual representation of calibration accuracy.

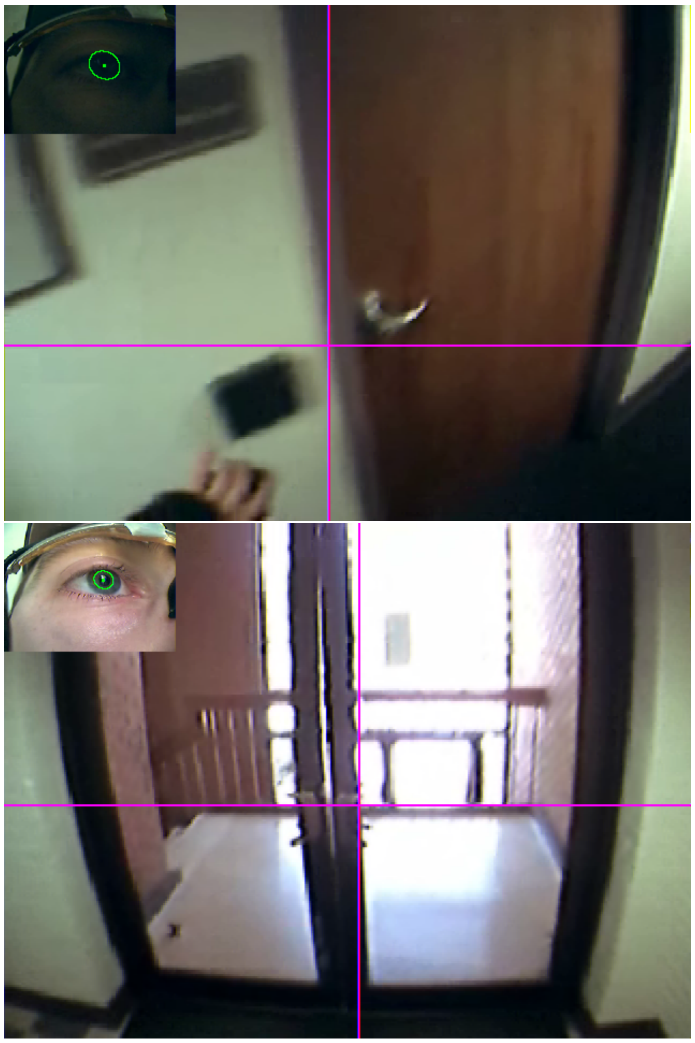


Figure 4.3: Cross hairs indicate point of gaze.

Chapter 5

FUTURE WORK

Future work may be divided into two main categories, building a better eye tracker, and constructing better analysis tools. When building a better eye tracker one can either improve the hardware or the software. Our focus is on improving the software. The accuracy may be improved in software either by improving the calibration or by more accurately tracking the limbus. In this section we will not discuss calibration. Section 5.1 will discuss sources of error in our eye tracker and propose potential solutions.

Due to the high variability between sessions introduced by the use of a head mounted eye tracker it is difficult to compare eye movements among individuals. We therefore present View Stitching (Section 5.2) as a potential solution. For dynamic object tracking, Object Registration is proposed (Section 5.3).

5.1 Proposed improvements

Specular reflections continue to be a problem. Whenever a specular reflection is present in the general vicinity of the pupil, the algorithm detects feature points near the border of the reflection. These erroneous feature points surround neither the pupil nor the iris. From the scene camera we can locate light sources that cause these reflections. Using Gullstrand's model of the eye we may render the specular component under these light conditions. Knowing the approximate location of the specular reflection we may then adjust pixel values in the eye image to compensate.

Although our implementation uses multiple threshold values and averages the best results, it is still accurate only within a small range of lighting conditions. In a recent publication Li and Parkhurst [2006] alter the feature detection algorithm. Rather than terminating rays at the first location where the gradient exceeds the threshold the ray continues so long as the gradient magnitude monotonically increases. This is analogous to the non-maximal suppression technique used in Canny's [1983] edge detector. Due to time constraints we have not implemented this feature. Upon implementation we expect some performance improvement.

Many erroneous feature points were detected along the eye lashes, and in the dark region near the nose. Both of these regions lay beyond the eye lid. By tracking the eye lid we could filter out these points. Eye lid tracking also holds promise as an additional source of information for eye tracking practitioners, may help us implement a more accurate ellipse evaluation function, and allow us to eliminate gaze data during blinks.

Our current method of converting the video files into individual frames introduces a number of harmful artifacts. Some frames are dropped and others are duplicated. This reduces the accuracy of our synchronization. On average the two videos remain synchronized and never differ by more than two frames. The MPEG-4 standard however requires that each frame has a time stamp. If we can obtain this time stamp information we can improve synchronization.

The Gaussian is commonly used as a general purpose noise filter in many image processing applications. It is well known that noise filters specifically designed for a particular type of noise can be much more effective. For example median filters are much more effective than the Gaussian at removing salt and pepper noise. Most of the noise in video obtained using off the shelf cameras is due to lossy compression. This noise has a specific structure, and we believe that development of a specific filter targeting compression artifacts will significantly improve the performance of the *Starburst* algorithm.

The quality of feature detection depends upon the quality of the start point. As the start point drifts away from the pupil center the feature set becomes unbalanced. For each frame we used the result from the previous frame as our start point. As the eye moves rapidly this start point may be significantly off center. We may use a Kalman filter to better estimate the start point for the next frame. We may also use our knowledge of typical eye movement, extreme locations, saccade velocity, to mitigate errors.

If we calibrate under ideal light conditions and retain the information of every ellipse generated during calibration we will have a population of about 800 ideal ellipses. Using principal component analysis on this population we may generate a function that evaluates the probability of a given ellipse belonging to the ideal population. This probability could be used as an additional factor in our ellipse evaluation method.

5.2 View Stitching

To compare eye movements among individuals, we propose to customize match moving algorithms to stitch a field of view *mosaic* over which we will project multiple scanpaths, as illustrated in Figure 5.1.

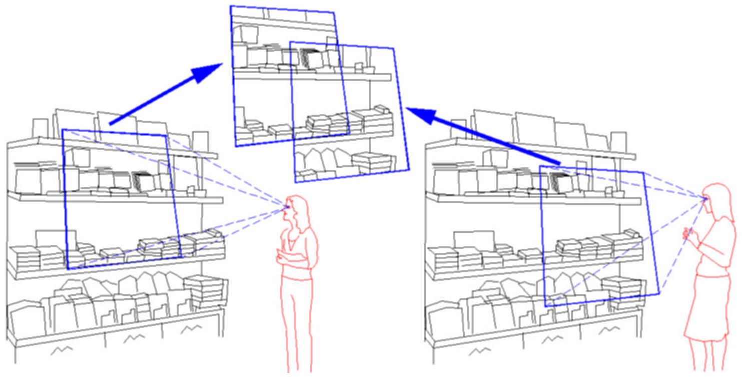


Figure 5.1: Artistic conceptualization of view stitching: assembly of stable image *mosaic*.

The standard procedure for creating a mosaic from individual images involves detecting and matching interest points in neighboring images, estimating the relative camera orientations, and stitching the images. Borrowing ideas from Steedly et al. [2005], we may use the inherent ordering of the video frames to constrain the search for matches. To reduce the computational burden, a recursive clustering algorithm may be employed to compress the measurements by replacing groups of interest point matches with *hallucinated* matches obtained from the centers of the clusters. This procedure increases speed with only a slight decrease in accuracy, since matches that span a small angular window poorly constrain the non-translational components of the camera motion. Camera orientations will be estimated using a second-order non-linear technique such as Newton-Raphson or Levenberg-Marquardt to minimize an objective function for P image pairs and M_i measurements between pair i: $\sum_{i \in P} \sum_{j \in M_i} e_{ij}^T \Sigma_{ij}^{-1} e_{ij}$, where e_{ij} is the 2D measurement error due to match j in pair i, and Σ_{ij} is the measurement covariance. Stitching of the images may be performed using bilinearly weighted contributions to reduce visible artifacts [Szeliski 1996]. Interest points may be detected by the 2×2 gradient covariance matrix [Shi and Tomasi 1994], Lowe's [2004] shift-invariant feature transform (SIFT), or Multi-scale oriented patches (MOPs) [Brown and Winder 2005].

5.3 Object Registration

Automatic identification of landmarks in the video scene during human navigation of natural environments is a problem related to scene registration in Augmented Reality. We may provide signal processing means to identify known or artificially placed landmarks in the environment, as suggested in Figure 5.2. This would allow comparison of the number of fixations devoted by individuals to navigational landmarks (such as signage).

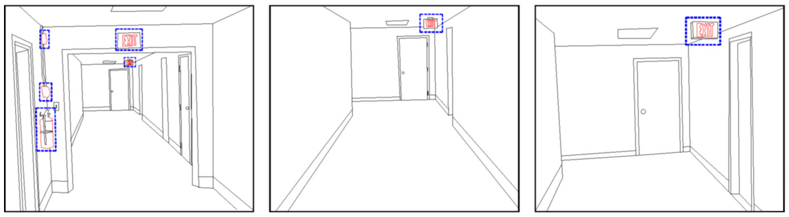


Figure 5.2: Automatic object registration during navigation in natural environment. Such landmark registration will allow comparison of individuals' fixation counts within these automatically detected Regions Of Interest (ROIs).

The well-known face detector of Viola and Jones [2004] is effective at classifying patterns in an image with accuracy comparable to other more computationally expensive techniques. The speed of the approach is due to its use of simple features resembling Haar wavelets computed using an efficient integral image computation, along with a cascade architecture that quickly discards image locations whose appearance is drastically different from the target. This algorithm can be quite effective at detecting other types of objects, such as automobiles, hospital stretchers, and the torsos of people.

To train the classifier, one only needs to provide hand-labeled positive and negative training examples during the one-time training phase, and the rest of the processing is fully automatic. We may apply this same procedure to detect natural or artificial landmarks of interest in eye tracking scenarios in real time.

Chapter 6

CONCLUSION

The predominant concerns in the development and evaluation of an eye tracker are: cost, invasiveness, ease of use, availability, accuracy, and versatility. Recent development in video technology has allowed video based eye tracking to become predominant over all older techniques. Video based approaches may be classified by range of illumination: visual spectrum, or infrared. Hardware may be classified as head mounted or remote. Visual spectrum outperforms infra-red in terms of cost, availability, and versatility. Head mounted eye trackers are more versatile than remote. We opted to use the openEyes project [Li et al. 2006] as the foundation for our own research, as it is the main open sourced development of a cheep, accurate, wearable eye tracker we are aware of.

We have made three significant improvements upon their work. The first two, initial pupil location detection, and calibration dot tracking improve ease of use by reducing the necessity for user intervention.

At the heart of our initial pupil location detection is a novel innovation to Borgefors' [1988] template matching algorithm. We achieve higher reliability by incorporating edge direction information along with the edge distance information. This technique may be further improved through the incorporation of a variable threshold mechanism.

Tracking of the calibration dot is an essential component of automated calibration of a head mounted eye tracker. We are unaware of any previous publication describing such a process. We have implemented a simple and effective thresholding procedure that achieves calibration dot tracking. Gradient information may be utilized in a more sophisticated more reliable and efficient local search strategy.

Our third improvement, pixel luminance delineation, improves eye tracking accuracy and versatility by allowing the algorithm to better distinguish the pupil boundary from the limbus. Through our work we have gained valuable insight as to how further improve eye tracking accuracy.

Bibliography

Babcock, J. S. and Pelz, J. B. 2004. Building a Lightweight Eyetracking Headgear. In *Eye Tracking Research & Applications (ETRA) Symposium*. ACM, San Antonio, TX, 109–114.

Ballard, D. H., Hayhoe, M. M., and Pelz, J. B. 1995. Memory Representations in Natural Tasks. *Journal of Cognitive Neuroscience 7,* 1, 66–80.

Birchfield, S. 1997. An Elliptical Head Tracker. In *Proceedings of the 31st Asilomar Conference on Signals, Systems and Computers*. IEEE, 1710–1714.

Birchfield, S. 1998. Elliptical Head Tracking Using Intensity Gradients and Color Histograms. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 232–237.

Borgefors, G. 1988. Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10,* 6, 849–865.

Brown, R. S. M. and Winder, S. 2005. Multi-image matching using multi-scale oriented patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Buswell, G. T. 1935. *How People Look At Pictures*. University of Chicago Press, Chicago, IL.

Canny, J. F. 1983. Finding Edges and Lines in Images. Tech. Rep. AI-TR-720, MIT Artificial Intelligence Laboratory.

Crane, H. D. 1994. The Purkinje Image Eyetracker, Image Stabilization, and Related Forms of Stimulus Manipulation. In *Visual Science and Engineering: Models and Applications*, D. H. Kelly, Ed. Marcel Dekker, Inc., New York, NY, 13–89.

Duchowski, A. T. 2003. *Eye Tracking Methodology: Theory & Practice*. Springer-Verlag, Inc., London, UK.

Ebisawa, Y. and Satoh, S. 1993. Effectiveness of pupil area detection technique using two light sources and image difference method. In *Proc. 15th IEEE Int. Conf. on Medicine and Biology Society*. 1268–1269.

Fitzgibbon, A., Pilu, M., and Fisher, R. 1999. Direct Least Square Fitting of Ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 21,* 5 (May), 476–480.

Hayhoe, M. M., Ballard, D. H., Triesch, J., Shinoda, H., Rogriguez, P. A., and Sullivan, B. 2002. Vision in Natural and Virtual Environments (Invited Paper). In *Eye Tracking Research & Applications (ETRA) Symposium*. ACM, New Orleans, LA.

Hennessey, C., Noureddin, B., and Lawrence, P. 2006. A Single Camera Eye-Gaze Tracking System with Free Head Motion. In *ETRA '06: Proceedings of the 2006 Symposium on Eye Tracking Research & Applications (ETRA)*. ACM Press, New York, NY, 87–94.

Kaufman, A., Bandopadhay, a., and Shaviv, B. 1993. An Eye Tracking Computer User Interface. In *Research Frontiers in Virtual Reality Workshop*. IEEE Computer Society Press, 120–121. URL: <http://www.cs.sunysb.edu/~vislab/projects/eye/Papers/short.paper.pdf>.

Lancaster, P. and Šalkauskas, K. 1986. *Curve and Surface Fitting: An Introduction*. Academic Press, San Diego, CA.

Land, M., Mennie, N., and Rusted, J. 1999. The Roles of Vision and Eye Movements in the Control of Activities of Daily Living. *Perception 28,* 11, 1307–1432.

Land, M. F. and Hayhoe, M. 2001. In What Ways Do Eye Movements Contribute to Everyday Activities. *Vision Research 41,* 25-26, 3559–3565. (Special Issue on *Eye Movements and Vision in the Natual World*, with most contributions to the volume originally presented at the 'Eye Movements and Vision in the Natural World' symposium held at the Royal Netherlands Academy of Sciences, Amsterdam, September 2000).

Li, D. 2006. Low-Cost Eye-Tracking for Human Computer Interaction. M.S. thesis, Iowa State University, Ames, IA. Techreport TAMU-88-010.

Li, D., Babcock, J., and Parkhurst, D. J. 2006. openEyes: A Low-Cost Head-Mounted Eye-Tracking Solution. In *Eye Tracking Research & Applications (ETRA) Symposium*. ACM, San Diego, CA.

Li, D. and Parkhurst, D. 2006. Open-Source Software for Real-Time Visible-Spectrum Eye Tracking. In *Conference on Communication by Gaze Interaction (COGAIN)*. COGAIN, Turin, Italy, 18–20. URL: <http://www.cogain.org/cogain2006/COGAIN2006_Proceedings.pdf>.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60,* 2, 91–110.

Menache, A. 2000. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann, (Academic Press), San Diego, CA.

Morimoto, C., Koons, D., Amir, A., and Flickner, M. 2000. Pupil detection and tracking using multiple light sources. *Image and Vision Computing 18,* 4, 331–336.

Morimoto, C. H. and Mimica, M. R. M. 2005. Eye Gaze Tracking Techniques for Interactive Applications. *Computer Vision and Image Understanding 98*, 4–24.

Pelz, J. B., Canosa, R., and Babcock, J. 2000. Extended Tasks Elicit Complex Eye Movement Patterns. In *Eye Tracking Research & Applications (ETRA) Symposium*. ACM, Palm Beach Gardens, FL, 37–43.

Rayner, K. 1998. Eye Movements in Reading and Information Processing: 20 Years of Research. *Psychological Bulletin 124,* 3, 372–422.

Reich, S., Goldberg, L., and Hudek, S. 2004. Deja View Camwear Model 100. In *CARPE'04: Proceedings of the 1st ACM Workshop on Continuous Archival and Retrieval of Personal Experiences*. ACM Press, New York, NY, 110–111.

Shi, J. and Tomasi, C. 1994. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 593–600.

Smeets, J. B. J., Hayhoe, H. M., and Ballard, D. H. 1996. Goal-Directed Arm Movements Change Eye-Head Coordination. *Experimental Brain Research 109*, 434–440.

Steedly, D., Pal, C., and Szeliski, R. 2005. Efficiently registering video into panoramic mosaics. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 1300–1307.

Szeliski, R. 1996. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications 16,* 2 (Mar.), 22–30.

Viola, P. and Jones, M. J. 2004. Robust real-time face detection. *International Journal of Computer Vision 57,* 2, 137–154.