Universitat Autònoma de Barcelona

Master in Computer Vision and Artificial Intelligence

Report of the Master Project

Option: Computer Vision

Eye-Tracking with Webcam-Based Setups: Implementation of a Real-Time System and an Analysis of Factors Affecting Performance

Author: **Onur Ferhat**

Advisor: **Fernando Vilariño**

Acknowledgements

I would like to thank first my supervisor Dr. Fernando Vilariño for his continuous trust, support and supervision throughout the master year and final project. I am very grateful to all friends who willingly participated in our experiments and shared their valuable time with us. Also, I want to thank all my colleagues from Centre de Visió per Computador for the comfortable working environment, their friendship and all their help during the last year. I appreciate very much the help of Dr. Oleg Komogortsev who shared his valuable Neural Network Eye Tracker software with us. Later on, I would like to thank my flat-mates Aslı, Jonatan and Hannah for providing me with the warmth and calmness of a home which made this work possible. Lastly, I send my gratitude to my family whose teachings and cultural heritage I feel deep inside and whose support could not be blocked despite distances.

ABSTRACT

In the recent years commercial eye-tracking hardware has become more common, with the introduction of new models from several brands that have better performance and easier setup procedures. A cause and at the same time a result of this phenomenon is the popularity of eye-tracking research directed at marketing, accessibility and usability, among others. One problem with these hardware components is scalability, because both the price and the necessary expertise to operate them makes it practically impossible in the large scale. In this work, we analyze the feasibility of a software eye-tracking system based on a single, ordinary webcam. Our aim is to discover the limits of such a system and to see whether it provides acceptable performances. The significance of this setup is that it is the most common setup found in consumer environments, off-the-shelf electronic devices such as laptops, mobile phones and tablet computers. As no special equipment such as infrared lights, mirrors or zoom lenses are used; setting up and calibrating the system is easier compared to other approaches using these components. Our work is based on the open source application Opengazer, which provides a good starting point for our contributions. We propose several improvements in order to push the system's performance further and make it feasible as a robust, real-time device. Then we carry out an elaborate experiment involving 18 human subjects and 4 different system setups. Finally, we give an analysis of the results and discuss the effects of setup changes, subject differences and modifications in the software.

Keywords: *Computer vision, eye-tracking, gaussian process, feature selection, optical flow*

Contents

Chapter 1

Introduction

From a computer scientist's aspect, human beings are machines which receive input from their sensors such as ears, eyes, skin and which interact with the world they live in through their actuators, which are their hands, feet, and so on. Therefore, as in the case of robots, one can understand the basis of their reasoning by inspecting the input they receive and also how they direct the attention of their sensors, for instance by looking at specific locations or inspecting unknown objects by touching or smelling.

Eye-tracking studies build upon this insight and analyze the relation between the movements of a person's eyes and their attention, reasoning and feelings. These studies make use of special equipment called eye-trackers, which either calculate the direction of a person's gaze [1] or a point in the surrounding area where the gaze is fixed at. This point may be located on a 2D plane [2] (i.e. the display of an electronic device) or a 3D volume [3] and is also called the point of regard (PoR) [4]. There exist several commercial models of eye-trackers [5, 6] which come with their software platforms to carry out eye-tracking research, also expertise for calibration and actual usage is provided through documentation and support.

Although the commercial components provide satisfactory performance for marketing or scientific research, the scalability of this methodology is problematic because these products require trained operators and their price (starting from several thousands of Euros) makes them not suitable for this purpose. Therefore, the availability of a cheap, easy to setup alternative which does not require special hardware and which provides comparable performance is a necessity.

Building this alternative constitutes the problem we address in this work, and it consists of understanding where a subject is looking at using cheap components such as light sources, cameras and a computer to run the eye-tracker software. Actually, these requirements are already met in many consumer electronic devices such as laptops, smart-phones, tablet computers. We believe that the availability of a system running on these machines will provide basis for a variety of applications. For example, the eye-tracker can be used as a new input method for the electronic device just like a mouse [7, 8], or it can be used as a tool to enable remote marketing and usability research [9, 10]. The system can replace

commercial eye-tracking equipment to the extent allowed by its performance, and it can enable mobile applications of these techniques. The uses of such a system is only limited by the imagination of software developers, who will surely come up with many usage areas that we don't discuss here in order not to get off our topic.

1.1 Brief information on human visual system

In order to be able to understand the methodologies behind eye-trackers, one should be familiar with the human visual system (HVS), in particular with the human eye. Figure 1.1 shows the parts of the eye which are of interest for eye-trackers. The white part of the eye which is seen from outside is called sclera, and in the front part of the eye there exists cornea which is a transparent layer. Cornea lets the light inside of the eye, which passes through an opening called pupil. Around pupil, we find the iris which is the colored part of the eye as seen from outside. Iris controls the amount of light which passes through pupil, and in a sense acts like the diaphragm of a camera. The light that enters the eye is refracted by the eye lens, and then it falls onto the retina layer which covers the inner surface of the eyeball, forming the image of what is seen through the eyes. Here lies the fovea, which is the area where eye's light sensing cells are accumulated in huge numbers. The part of the image that corresponds to this area is special, in the sense that the human visual attention is mostly directed here [11]. In the figure, two axes are shown: the optical axis passes through the eyeball center (E) and the pupil center, whereas the visual axis passes through the fovea and a special point called corneal center (C). This difference is caused by the fact that fovea does not lie on the optical axis.

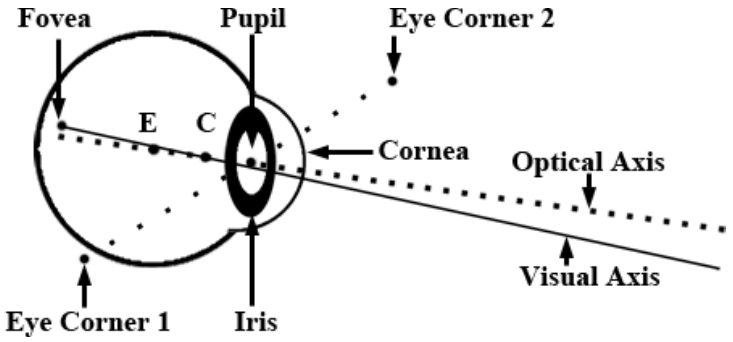


Figure 1.1: Structure of the human eye

Two other points important for software implementations of eye-trackers are the eye corners. These are the extreme points of sclera that are visible from the outside. As the iris and pupil change location due to the rotation of the eye, the corner points become important for the software because they are mostly stable and help understand the relative movement of the eyeball.

Another important topic about HVS that remains is related to the movements of the eye. As noted before, the visual attention of human beings are mostly directed towards the objects that lie on the visual axis and whose image drops onto the fovea. Therefore, the movements of the eye should be seen as a change of attention point. The eye movements can be mainly classified into three categories [11]:

Saccades: These are rapid eye movements that are aimed at focusing the visual attention to a different object. For example, when a computer user jumps from one menu item on the display to another, the eye moves rapidly in order to focus on the next target. The duration of these movements is between 10 ms and 100 ms, and during this short period the visual system becomes practically blind as the attention cannot be directed at a particular object.

Smooth Pursuits: These happen when the eyes are following a moving target. In order to achieve this movement, the speed of the target should be in the range that can be handled by the human eye.

Fixations: These are the movements which stabilize the retina on the object of interest. This is particularly interesting for our work because during fixations, the subject directs their attention on the object that lies on the visual axis and calculating the gaze point makes sense.

Apart from these main movements, blinks which are involuntary movements to keep the eyes moist, cool and clean [4] should be considered. For the correct operation of an eye-tracking system, these movements should be detected and excluded from training and testing processes.

1.2 Current implementations of eye-trackers

Now that we have familiarized with the structure of the human eye, we can continue with analyzing the current implementations of eye-trackers. In the last twenty years, there has been ongoing research in this field where many techniques have been proposed and explored. These techniques can be classified in several categories depending on the property of interest. These classifications mainly are:

According to camera type: IR spectrum systems, visible light spectrum systems.

According to light source setup: Systems using additional lights, systems relying on ambient light.

According to camera count: Systems using a single camera, systems using multiple cameras.

According to placement: Head-mounted systems, remote systems.

According to methodology: Appearance based systems, model based systems.

IR systems are also called active light systems, because they require carefully placed IR light sources as part of their algorithms. The placement of these lights is part of the calibration procedure and their positions w.r.t. the camera is known, therefore enabling 3D information deduction from the reflections of these lights on subject's cornea. The use of IR light enables better segmentation of the pupil in these systems because the IR light passing through this opening is mostly reflected back [4], resulting in a

large intensity difference. The light wavelength used in IR setups can be captured in many commercial cameras, although some modifications in the hardware might be necessary. As this work focuses on visible light spectrum systems, we leave out a more detailed overview of IR methods, and a good survey including these methods can be found from Hansen and Ji [4].

Visible light methods are also called passive light methods, because it is not feasible to use additional light sources in these setups. Such a setup would discomfort the subjects and cause the pupil of their eyes to shrink. However, lamps that are used to increase the ambient light should not be considered as a light source, as they are not used actively as part of the algorithm. The methods referenced below are examples to visible light methods unless stated otherwise.

Often, multiple cameras are used in order to have 3D information about tracked feature points [12, 13]. In some cases, extra cameras may be used for different purposes such as head-pose estimation [14]. At this point, it should also be noted that some applications also make use of zoom-in cameras [14] and mirrors in their setups, in order to direct the camera onto the area of interest (e.g. tracking the subject's head or eyes). These techniques require extra calculations in order to make up for the use of these equipments and techniques.

Another classification criteria for eye-trackers is the type of placement that is employed in the building phase. There exist systems that are placed on top of the subject's head with a head band or which are placed inside of eyeglasses. However, as this equipment is sometimes heavy and may cause discomfort on the side of the subject, remote systems pose an alternative. These systems may either be built into special monitors or they may be separate components that are placed near the monitor.

According to the techniques used in the software, we can classify the methods as appearance based systems and model based systems. In model based systems, the aim is to fit a 2D or 3D model onto the eyes or face of the subject, and then calculate the gaze point making use of these models. In some works, the iris boundaries are modeled as circles in 3D space and after the model is fitted, the normal vectors passing through the center of these circles are used to calculate gaze direction [14, 1]. In such methods, a zoom-in camera can be used to achieve a higher quality image of the eyes and thus, a better performance [14]. These types of techniques are also called ellipse algorithms, because the projection of circles on the image are ellipses and the algorithms may fit an ellipse on the iris boundaries in the image, then calculate the 3D positions from these projected ellipses. In these techniques, focal length of the camera should be known for the calculations of back projection.

Other model based techniques involve fitting a known 2D or 3D mesh structure onto subject's face. In one implementation, a generic 3D face model consisting of 7 points (four for eye corners, two for nose tips and one for the corneal center of an eye) is fitted on the image and the gaze direction is computed as the vector connecting the corneal center and the pupil center [2]. In more flexible methods, a unique face model is calculated for the subject by asking them to move their head around for a while [15]. There exists a method involving flexible 3D face models called Active Appearance Models (AAMs) [16], which model the whole face and allow linear shape changes [17].

On the other hand, appearance based models focus on the goal of determining the gaze point, without calculating any models for eye or face features. One technique starts with detecting the face, eyes and iris boundaries, and the image region around the iris is fed to a neural network to map the image to the point of regard [18]. In an open source alternative, the eye region image is mapped to the gaze point through the use of Gaussian process interpolation [19]. There are also earlier records of the use of neural network interpolation for this purpose [20]. It should be noted that these methods' performance generally decreases when there is a change in head pose. However, when the assumption of fixed head pose holds, these methods give comparably better results.

We have described the problem and discussed the existing approaches to solve it. In Chapter 2 we continue with the approach we propose in this work. Chapter 3 dives into the detail of our work and Chapter 4 proposes a setup to experimentally test the performance of this system. Chapter 5 provides the results of these experiments, followed by a discussion in Chapter 6. Lastly, we will conclude with final remarks and point out directions for future research in this area.

Chapter 2

An Eye-Tracking System for a Single Webcam Setup

We have seen several approaches which differ in the equipment and operation conditions they require, their performance and usage areas. Now, we are presenting an overview of the system that we try to build in this work and our motivations for making these decisions among others possibilities.

Most state-of-the-art examples use infrared (IR) methods where the subject's face is actively illuminated with IR light sources. These systems provide great performance values and this methodology is also almost exclusively used in the commercial eye-tracker models. However, our aim in this work is to present a system which works in common setups, without requiring hardware such as IR lights and IR cameras that are not easily found in consumer environments. Therefore we have constrained ourselves in the visible light methodologies in the rest of our work.

Another important feature that we wanted to have in our system is the ease of setup. Although the commercial models have become more and more user-friendly, easy to setup recently, they still require certain expertise and training in order to get up and running. Moreover, they have their own development environments which may make it harder to integrate into custom software applications. Considering the non-commercial IR eye-trackers which are results of ongoing research, we see that these are generally hard to setup because the placement of the light sources requires expertise and there is a special calibration procedure before starting the actual usage.

In order to have an idea about the current situation of open source eye-tracker implementations, we carried out a state-of-the-art research of these tools. The three applications that we have found are:

Opengazer: It is based on OpenCV [21] and written in C++ language. Supports Linux and Mac OS X operating systems. Has been developed by a research group at the University of Cambridge; however, is lastly updated three years ago. Uses a setup consisting of one webcam and has the infrastructure to communicate the results to other client applications [19].

ITU Gaze Tracker: Based on .NET Framework and written in C# language, this software is still developed by a research group from IT University of Copenhagen. Uses IR cameras and provides a user-friendly interface to setup the system. Also provides the infrastructure to send the gaze estimation results to other programs [22].

TrackEye: Based on .NET Framework and OpenCV, it is developed as a personal project and published on the Internet. It is not under development and calculates only the angular gaze direction, but not the gaze point [23].

2.1 Hardware baseline

As a result of our priorities and analysis of the state-of-the-art in free eye-tracker software tools, we have decided on the following setup as the required hardware baseline for our system:

A regular webcam: With the increased availability of broadband Internet and popularity of online video communication tools, webcams have become a standard component of home computer setups. Moreover, high Internet speeds enabled the use of cameras with higher resolutions and better image qualities. Modern video chat applications and cameras support high-definition (HD) images with resolutions up to 1920×1080 pixels. Two other types of consumer electronic device that have gained popularity in the recent years are smart-phones and tablet computers. These gadgets often contain a built-in camera facing the user, which is intended for 3G video calls. Although they might not always come with HD capabilities, we believe they still provide the necessary baseline to feed the system we propose. This baseline is defined by resolutions of 1280×720 and 640×480, which are the resolutions of iPad FaceTime camera and the common VGA cameras, respectively. We consider the differences in the camera placements and camera qualities of these equipments, therefore the effects of these variances will be analyzed in the coming chapters.

No special lighting: As we aim at working with visible spectrum cameras, usage of light sources is completely out of question. However, we make the assumption that the testing environment is well-lit so that the camera can provide an acceptable image quality at an acceptable frame rate (≥ 10 fps).

A display: The display is unquestionably another compulsory component of these systems. During our experiments, we have placed the display at known distances away from the subject's face, however in real world situations the distance is only limited by the camera's point-of-view. In our setup we only make two assumptions about this: that subject's face is completely included in the camera image and that it covers around half of the camera image vertically. This is the minimum constraint of environment we have assumed in our setup.

2.2 Software architecture

After stating our assumptions about the required hardware setup, we continue with describing the general architecture of the proposed system.

Among the open source eye-tracker software that we analyzed, Opengazer stood out as the system that mostly meets the requirements in a baseline. Although the codebase is not well-commented, its modular and simple structure enables building additional work upon the existing system. It has the feature of transmitting the estimated gaze points through network, which enables accessing these results from other programs without dealing with application's codes. It requires a manual calibration procedure and it does not handle head pose variations, which constitute its main flaws.

Table 2.1 shows the components of the Opengazer application that we take as basis for our work. We describe the original components and the improvements we propose to make in this work. In the last column, our motivations for implementing these changes are briefly explained. With these improvements we believe that the proposed eye-tracking system can provide acceptable performance values and as a result, many usage areas will arise such as mobile applications of eye-trackers and remote eye-tracking research with decreased cost.

Component	Original Opengazer	Proposed Improvements	Justification
Point Selection:	-Manual selection of feature points	-Automatic selection of points	Initialization of the system should be easier and independent of the operator.
Point Tracking:	-Combination of optical flow (OF) and head pose based estimation	-A different combination which can recover from mistrackings -Apply a median filter before OF	The system's recovery after a mistracking should be improved.
Eye Image Extraction:	-Extract the image of a single eye	-Extract the image of both eyes	Second operation does not require much computational power and enables us to have a second estimator.
Blink Detection:	-A non-working detector	-Get the detector running	Blink detector is a crucial part of the system.
Calibration:	-Calculate average grayscale eye images for each target point and train a Gaussian process (GP) interpolator	-Do not include blinking frames in average images -Calculate and remove the calibration errors caused by GP training	Blinking frames during training are considered noise and should be removed. GP training errors cause shifts in estimations during the testing phase.
Gaze Estimation:	-GP estimator -Calculate estimate for a single eye	-Calculate estimates for both eyes -Implement a neural network (NN) estimator	Average of both estimates will perform better. NN estimators are shown to have good performances [18].

Table 2.1: Components and methodology of original Opengazer application, shown side-by-side with the proposed improvements and the justifications of these choices.

Chapter 3

Implementation Details of the Eye-Tracking System

In this chapter, we describe in detail the work we have done for this thesis. We start from the first step of choosing suitable cameras for use in our systems, then, we continue with the improvements made in each component of the proposed system.

3.1 Search for a suitable camera

In order to find a suitable camera for our needs (high frame rate, high resolution, good image quality) we have tested several cameras: Logitech c615 (1080p), Logitech ... (720p), Microsoft HD-3000 (720p) and several other cheaper models (from brands such as Trust, NIUM, Hercules) [1]. We have seen that with the more expensive models we can get good HD quality images at acceptable frame rates ($\geq$ 10 fps). The c615 model contained some flickering effect in the live preview, whereas the lower 720p model did not have this defect. However, we have seen that in better lighting conditions (ground floor of CVC) this effect is not perceived in a relevant way.

The UVC specification is a communication standard which regulates the control of webcams in a variety of operating systems such as Linux, Windows and Mac OS X. Linux contains the required drivers for controlling cameras which support this framework.

As we want to have full control of the camera so that we can change the frame rate and image resolution, or remove the automatic adjustments made on the image before serving it to the client application, we analyzed the necessary tools that enable us to use control our cameras to the limits. These applications are:

[1] Furthermore, we tried to convert a regular webcam into an IR camera. The details of this experiments are explained in Appendix A.

guvcview: It allows changing the parameters of the camera via a GUI, where the image format, frame rate, resolution, automatic image normalizations (white balance, etc.) can be specified. Also it acts as a photo taking or video recording application.

v4l2: This command line application can be used to determine which features a UVC compatible camera supports and to change the supported settings. Some features which cannot be changed using the guvcview application (e.g. auto-exposure) can be changed in this program, thus allowing even further control over the camera.

Using these programs, we could compare the frame rates and extra features of the cameras we had provided for this study. Moreover, we analyzed the effects of different image formats of webcams on the frame rate. We observed that although the frame rate can be set manually, high frame rates cannot be set or achieved when using image formats with low compression. Using these tools, we could unlock a setting in Logitech c615 camera related to exposure and it allowed us to select from a larger range of exposure values.

After the comparison of the cameras using these tools, we chose the Logitech c615 camera as the component of our system as it provided good frame rates and has a sensor with higher resolution capabilities. We identified 1280×720 and 640×480 as the baseline resolutions that we are going to compare in the experiments, therefore the camera will support a higher frame rate compared to the full resolution case.

3.2 Improvements in the components

3.2.1 Point selection

In the original Opengazer application, a set of feature points including the outer eye corners and at least two other facial feature points should be selected by the user as part of the initialization routine. As this constitutes an inconvenience on the side of the user, and as it poses problems for the experimental setup that is described in the next chapter, we propose an automatic point selection method as our contribution which is described below.

As seen in Figure 3.1(d), we decided that 8 points are needed to be selected during initialization. In choosing these points, we considered how easily they can be selected in an automatic way and how stably they can be tracked over time. The first condition is related to the tools for feature detection. Regarding the condition of stability, we argue that stable points decrease the mistracking and thus have a direct effect on the overall performance of the system.

The first two points that are essential for our system are the outer eye corners. The system uses these points to locate the image region containing the eyes, with the help of geometrical constraints of the face which will be explained below. The rest of the points are used to obtain 3D pose information of the face and to correct the tracking estimations in case some points are lost during tracking. Therefore, a higher number of well-tracked feature points is expected to improve the tracking performance. We chose the

two inner eyebrow corners, the two nose tips and the two mouth corner points as the remaining 6 points. These points are good candidates because there is more texture in these areas of the face compared to other parts (i.e. cheeks, forehead) and also they are not occluded at times like the ears.

The list of steps to select the feature points is as follows (see Figure 3.1). The methods used in each step will be explained afterwards.

Step 1: As shown in Figure 3.1(a), the system first locates the region containing both of the eyes. The result is the green rectangular area.

Step 2: Then the region containing single eyes are found in the right and left halves of the green rectangle. The results are the blue rectangles inside the green rectangle.

Step 3: Finally, the outer side center points of these single eye regions are selected as the eye corner points (red dots).

Step 4: In order to locate the nose, a square region starting from the eye corners' level and expanding towards the lower part of the face is selected as the search area (green square, second image). The length of the sides of this square is determined by the distance between two eye corner points selected before.

Step 5: Nose detector is then run (blue rectangle) and nose tips are selected using geometrical constraints (red dots).

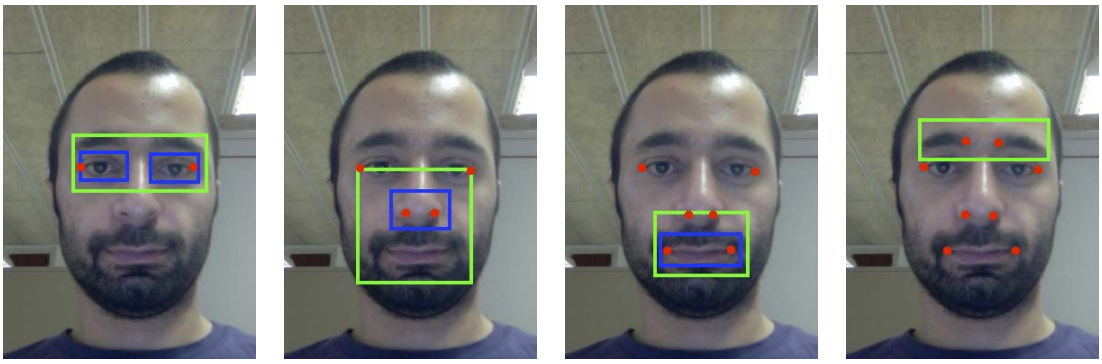
Step 6: In order to locate the mouth, a rectangular region starting from the extracted nose points' level and expanding down is selected as the search area for the mouth (green rectangle). This region's width is equal to the distance between the extracted eye corners, and the height is half of this value.

Step 7: Next, a mouth detector is run (blue rectangle) and mouth corners are selected by geometrical constraints (red dots).

Step 8: Eyebrow corners are selected in the same manner as nose tips and eye corners. This time, a rectangular search area above the extracted eye corners is considered. The width of this area is equal to the distance between the extracted eye corners, and the height is one third of this value. In this case there is not a Haar cascade detector that works well, therefore we use corner point selection method to choose the most important two corner points.

Here, point selection using geometrical constraints refers to a technique where the points are assumed to be located at a certain coordinate in the searched area. For example, mouth tips are assumed to be located on the central horizontal line of the detected mouth rectangle. The detectors used to locate the region containing two eyes, individual eyes, nose and mouth make use of Haar cascades [24, 25] and are based on OpenCV [21].

This initial methodology does not work perfectly, because it is limited by the performance of its component detectors. For instance, the single eye detector does not perform very well and causes the



(a) Eye corners (b) Nose tips (c) Mouth corners (d) Eyebrow corners

Figure 3.1: Sequence of control point selection

system to halt the selection process at this point. Therefore, we have analyzed other methods to replace this module. The state-of-the-art methods for eye corner detection that we have encountered can be grouped in four categories:

Template matching methods: An eye corner image template is prepared and this is matched to the eye region image using correlation. However, it is argued that this is not accurate all the time [26]. A variation which includes using deformable templates is proposed, however tuning the parameters is reported to be problematic.

Variance projection methods: Methods involving calculation of variance projections (the intensity variance of horizontal and vertical pixels around an interest point) are also proposed [26]. This variance feature is combined with Harris response for the pixels in order to weigh the effects of pixels accordingly.

Semantic feature based methods: Another method is based on detecting corners using Harris and then choosing the best corners using semantic features [27]. In order to choose among the candidates, lines are fitted for the eyelids in each candidate, then other discriminative features which capture the intensity difference between eyelids-sclera and sclera-iris are calculated w.r.t. the fitted lines.

Other types of semantic features have been proposed to make the choice among the candidates [28]. These features include ratio of other candidates below and above the analyzed candidate, the distance of the analyzed candidate to the intersection point of the curves fitted to the eyelids and so on. This method is shown to perform better than template matching algorithms and it shows more invariance to rotation and focus.

Active appearance models: A generalized appearance model is created for eyes or whole face, then

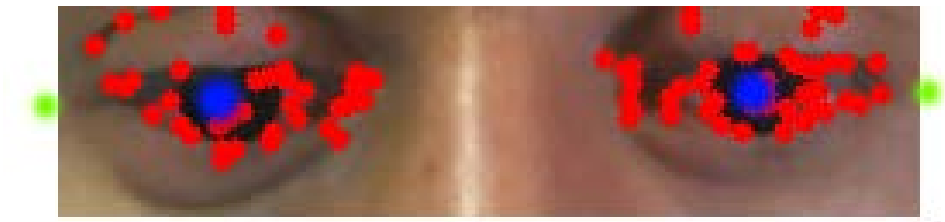


Figure 3.2: Selection of the left and right eye corner points using geometrical heuristics after Harris corner detection

this is deformed to fit the current image. The model contains a representation of the shape (vertices and edges) and the gray level appearance [16].

These state-of-the-art methods are results of work which pushes the limits to locate the eye corners as accurate as possible. Therefore, because of the complexity of the methods they are left out of scope of this work. Instead, we propose a few simpler methods based on Harris corner detectors [29] which work well for our purpose. It should also be noted that in our case, we do not necessarily need the localization of the exact eye corner because this point is easily mistracked during blinks and other situations. The point that is expected to work better for our system is a little further away from the eyes. The system allows this point to have a small vertical offset because it does not matter for the eye image extraction very much.

Corner selection based on eye center points

In this method, the same image region containing both of the eyes is processed and Harris corners are detected as shown in Figure 3.2. At this point, the algorithm separates the points into two groups, according to which half of the image they are located at. The coordinates of the ones in the left part of the image are averaged to calculate the left eye center point, and the right center point is calculated from the remaining ones (blue points). Then, a geometrical calculation is employed and eye corner points are selected on the line that passes through these two center points (green points). In this calculation, first the distance between the two eye centers is calculated (d) and the points that are located $d/3$ away from the eye centers are chosen as the corner points. Here, the ratio of $1/3$ is used in both vertical and horizontal coordinate calculations.

We implemented this method after validating that simpler methods such as choosing one of the points resulting from Harris corner detection does not perform very well.

3.2.2 Point tracking

The point tracking component of the original system uses a combination of optical flow (OF) and 3D head pose based estimation. Optical flow calculations are done between the last camera image and the previous image, and also between the last image and the first image where the feature points were selected. However, in the original version the first OF calculation is overwritten by the second one and is practically not used.

As mistracking of feature points proved to be one of the leading causes of estimation errors, we have focused on this component in order to sort out these problems.

One idea that we try in the OF component is running a median filter on the images before passing them to the calculations. We experimented with several sizes of filters and we observed that 5 pixel sized filter works well after visually controlling from a resulting image. In parallel, we changed the parameters of OF function and verified that using a larger pyramid level (6, instead of the original value of 2) yielded better results and allowed larger head movements.

Our most detailed experiments in this part of the system are aimed at achieving the best combination of the three methodologies used: OF w.r.t. previous frame, OF w.r.t. initial frame and 3D head pose based estimation. We tried several combinations and compared them qualitatively:

Only OF w.r.t. previous frame + 3D for mistracked points: System cannot recover from mistracking.

OF w.r.t previous frame + OF w.r.t. initial frame + 3D for mistracked points: Here we get the results of the first calculation and supply them as the prior values to the second calculation. The mistracked points are updated with the 3D based estimations. In this update, the 3D based estimation accounts for 90% of the final value, whereas the rest is determined by the OF estimation. This setup gives good results for small head movements and rotations.

Previous combination + 50% update: This combination is similar to the previous one, only the update ratio of 3D estimations is changed to 50%. This change did not yield better results compared to the previous setup.

Previous combination + 30% update: Here we try using 30% update rate. We did not achieve any visible difference in this method.

Only OF w.r.t. initial frame + 30% update: In this method we completely ignore the OF calculation w.r.t. the previous frame and we only consider the initial frame. This method yielded the best results in our qualitative analysis.

As seen in these combinations, we also come to the conclusion that OF w.r.t. previous frame may be ignored without any performance decrease. The major difference in our system comes from the weighting differences and the mistracking recovery strategy. In such case, our system uses the estimations calculated from head pose directly and omits the OF calculations for mistracked points. The other difference is the change in the pyramid levels as described above.

3.2.3 Eye image extraction

Originally, Opengazer tracks the outer corners of both eyes, however only a single eye is used for the gaze estimation. We tested using the second eye just like the other eye to come up with another estimation in order to decrease the uncertainty of the final estimation.

We applied the same method for single eye extraction and only modified it to extract the region which contains the second eye. The resulting image is processed to train and use another gaze point estimator using the method explained below.

3.2.4 Blink detection

Opengazer program that we chose as our base system is an open project, therefore it contains several parts which are in draft version and not completed for use in the final version. The blink detector is an unfinished component and as we consider it a crucial part of our system, we continue with analyzing it and making the necessary modifications to get it running.

The blink detector is designed as a state machine with states representing the initial state, blinking state and double blinking state. The system switches between these states depending on the changes in eye images that are extracted as described in the previous section. These changes are obtained by calculating the L2 norm between the eye images in consecutive frames. When the difference threshold for switching states is exceeded during several frames, the state machine switches to the next state and the blink is detected. However, as noted above, this method was left in a draft version and required additional work.

We built on this structure and completed the rules for the state switching mechanism. Moreover, we observed that a state reset rule is required so that whenever the threshold criteria is not met at a certain frame, the system is reset to the initial state. We implemented this improvement in the final version.

3.2.5 Calibration

The calibrator component of our system works by displaying markers on several locations on the screen until 20 frames can be captured from the camera and then training the estimator component using the eye images extracted during this time period. For each marker location, these corresponding eye images obtained from several frames are averaged to create a single grayscale image. This image represents the average appearance of the eye while the marker was displayed. Lastly, a Gaussian process based gaze estimator is trained using the supplied grayscale average images and the corresponding positions on the screen.

We propose several improvements in the calibrator component in order to increase the gaze estimation performance during testing. These contributions are:

1. Fix a small problem in the display of markers
2. Exclude the blinking frames from calibration

3. Apply a drift correction algorithm to correct the errors introduced by calibration

The first improvement is related to a minor defect that we encountered in the Opengazer application. We observed that the pointer which is shown on the display during calibration and testing is shown in the wrong coordinates on the display. This is because the coordinates of the top-left corner are used in setting the location of the window on display. In the correct case, as the subject directly looks at the window's center point, the images of user's eyes should be mapped to this point's coordinates. For testing purposes, this does not pose a huge problem because the same problem exists in the gaze estimation component and they cancel each other. However, when another program uses the estimate of Opengazer, the resulting coordinates will include a small shift which is half the height of the window vertically and half the width of the window horizontally. We fixed this issue in our version.

Later on, we made the modifications in the calibration code so that the frames which contain blinks are no longer included in the calibration procedure. This is crucial because previously these frames could alter the average eye images calculated during calibration and therefore were reflected as noise in the calibration procedure. However; as these frames are no longer available for calibration, we increased the number of frames that the pointer stays in the same location from 20 frames to 30 frames. With the new, higher number of frames, we expect to provide the system with enough samples during calibration.

The last improvement that we propose for this component is the correction of calibration errors. Figure 3.3 illustrates this problem.

Here, each colored symbol corresponds to a target point displayed on the screen and the corresponding gaze estimations of our system, one for each frame. The larger symbols which are arranged in a 3×5 grid denote the actual targets, whereas the smaller signs of the same symbol type and color are the estimations of the gaze position. For each symbol group, a line of the same color connects the average estimation and the target location. Therefore, the length and direction of this line gives us the magnitude and direction of average testing error for each test point location. Apart from these colored symbols, 15 blue lines that start from each target denote the direction of the calibration error for them. However, it should be noted that in order to easily observe the direction, the magnitude of the calibration errors are increased by a factor of 5. In this figure, we see that for 9 points (1, 2, 3, 4, 10, 11, 13, 14, 15) there is a clear correlation between the calibration error and average testing error. Since this correlation is present, we propose to apply a drift correction.

In order to calculate the calibration errors, we tried two different methods:

Method 1: In this algorithm, we store the average grayscale eye images that are calculated during calibration for each training target. Then, after calibration is finished, we pass these eye images to the gaze estimator component and calculate the gaze estimations that correspond to them. We observed that this method does not work very well in capturing the correlation between training and testing errors.

Method 2: Differently from the previous method, here we store the individual grayscale images which are used to calculate the average images during calibration. Therefore, we save several images

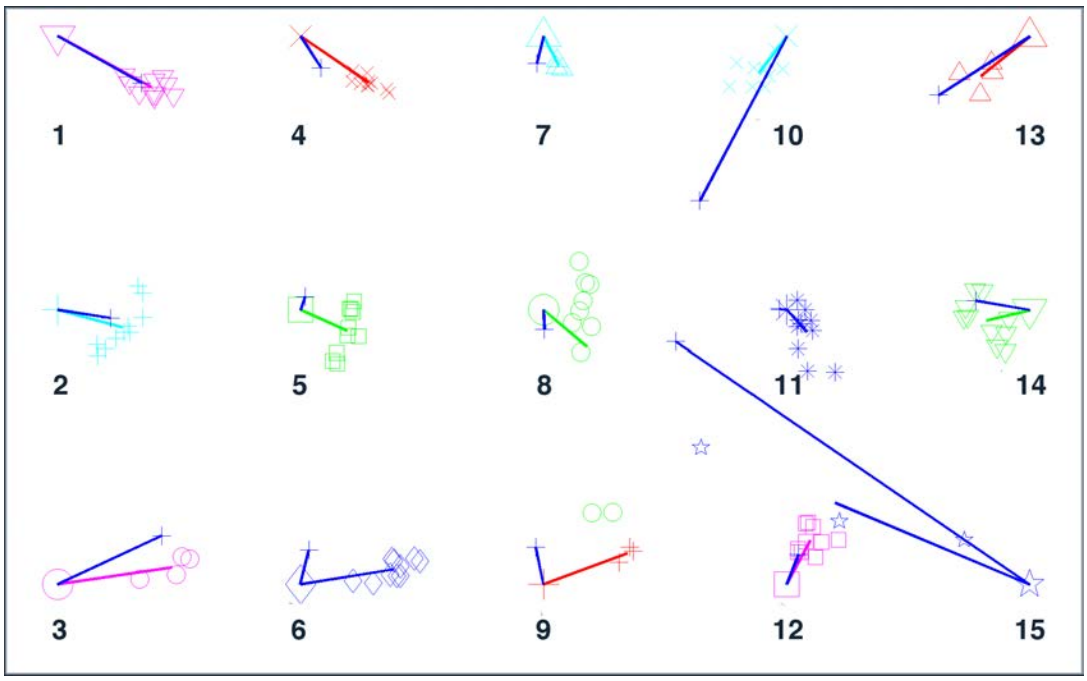
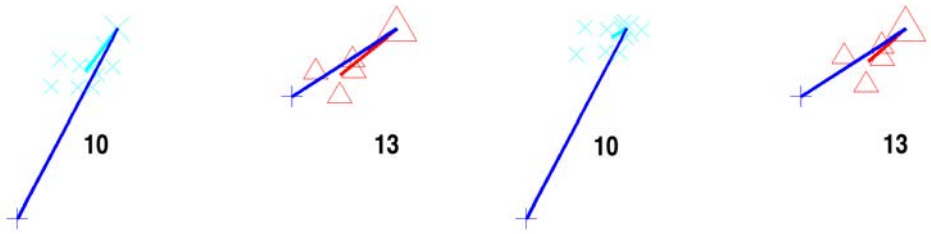


Figure 3.3: Calibration errors and corresponding testing errors. The symbols with the same shape and color show the gaze point estimations for the same target point. For each target, the blue lines show the calibration error direction and the line which has the target color shows the test error direction.

corresponding to different frames for each calibration point. Similar to the previous case, after calibration the gaze estimations for these individual images are calculated and then they are averaged to obtain the average estimation.

A qualitative analysis of these two methods showed that the second method is more successful in calculating the calibration error directions. After the calibration errors are calculated, we continue with removing these errors during testing. For this purpose, we employ a multivariate interpolation method [30] which we train by passing the average gaze estimations calculated as described above and the actual targets they belong to. An open source implementation of this method is available online [31]. After the interpolation method is trained, we use it during testing to remove the effects of calibration errors. We pass the calculated gaze point estimation of the current frame to the trained interpolator and use the output as the corrected gaze point estimation. The results of this correction is presented in Figure 3.4, where the estimations are clearly drawn towards the actual targets.



(a) Before correction (b) After correction

Figure 3.4: The drift correction moves the estimations towards the actual target.

3.2.6 Gaze estimation

The last component in the pipeline of our system is the gaze estimator, where the Gaussian process (GP) interpolator trained during calibration is used to estimate the gaze position. Here, we tested two different approaches to improve the system:

1. A straightforward averaging of the gaze position estimations provided by both eyes
2. Implementation of a neural network based interpolator

In the first contribution, we simply use both of the extracted eye images to calculate two gaze point estimations. Then, we combine these estimations by averaging.

Apart from this simple idea, we also consider the case where the Gaussian process interpolator used here is completely substituted. Neural network (NN) methods constitute a popular alternative for this purpose. There exists recent implementations of this technique [18]. In the work that is mentioned, an eye tracker using NNs to map the eye image to gaze point coordinates is implemented and is made available [32]. However author's permission must be requested to be able to access the archive contents which are protected by a password.

We incorporated the NN method in our system by making use of the Fast Artificial Neural Network (FANN) library [33] and created a similar network structure, and a similar input-output system as the original work. Our neural network had 2 levels where the first level contained 128 nodes (1 for each pixel of 16×8 eye image) and the second level contained 2 nodes (one each for x and y coordinates). We scaled the pixel intensities to the interval [0, 1] because of the chosen sigmoid activation function.

Other changes that we implemented consist of normalizing the histogram of eye image before feeding it to NN, trying different training methods (incremental, batch, RPROP), trying different stopping conditions for training (maximum iterations, upper limit for mean square error (MSE)), and testing different learning rates. After these changes, the NN component successfully replaced the GP estimator of our system. However, the GP interpolator showed better qualitative results that would have to be tested in a quantitative analysis.

Chapter 4

Experimental Setup

4.1 Setup of the experiment environment

In this section, we describe the experimental setup we have created to test the performance of our application. Variations in the setup are made to create separate experiments which allow us to see how the system performs in different conditions. These experiments are carried out on 18 subjects during a period of four days. Figure 4.1 shows how the components of the experimental setup are placed in the environment.

On the left, we see the operator's computer where the application is running. This computer is rotated in order not to distract the subject. The stimuli display faces the subject and it is raised by a support which enables the subject to face the center of the display directly. The camera is placed at the top of this display at the center (A), and it has an alternative location which is 19.5 cm towards the left from the central location (B). An optional chinrest is placed at the specific distance of 80 cm away from the display, acting as a stabilizing factor for one of the experiments. Behind this we see the rigid chair placed for the subject. Depending on which experiment is carried out at that time, the chair position is adjusted so that the distance between subject's face and test display stays at the predefined value.

By introducing variations in this placement, we achieve several setups for different experiments. These have specific purposes in testing a different aspect of the system. These setups are:

Standard setup: Only the optional chinrest is removed from the setup shown in Figure 4.1. Subject's face is 80 cm away from the display. The whole screen is used to display the 15 target points one by one.

Extreme camera placement setup: This setup is similar to the previous one. The only difference is that the camera is placed at its alternative location which is 19.5 cm shifted towards the left. The purpose of this setup is to test how the position of the camera affects the results.

Chinrest setup: A setup similar to the first one. The only difference is that the chinrest is employed.

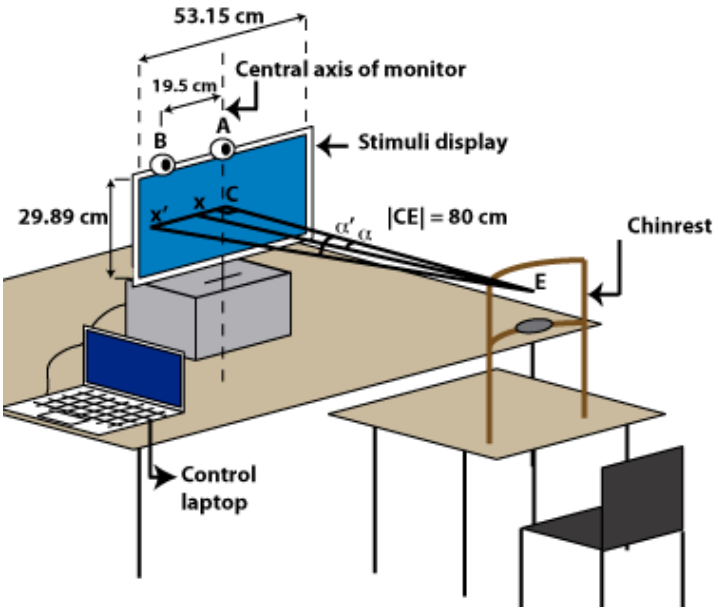


Figure 4.1: Placement of the components in the experimental setup and the geometry involved in error calculation

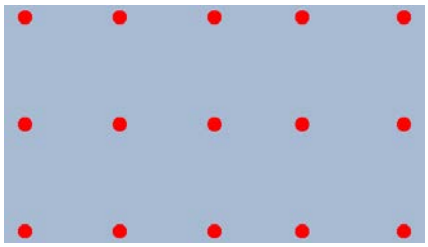


Figure 4.2: Test point distribution for standard, extreme camera and chinrest setups

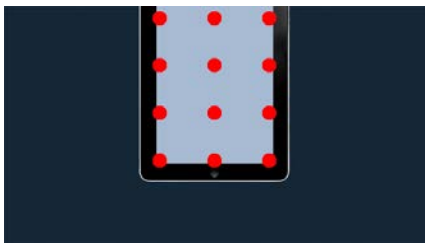


Figure 4.3: Test point distribution for iPad setup

This experiment is aimed at testing the effects of head pose stability in the performance.

iPad setup: This setup is used to test the performance of our system simulating the layout of an iPad on the stimuli display. This background image contains an iPad image where the iPad screen corresponds to the active area in the experiment and is shown in a different color (see Figure 4.3). The distance of the subject is decreased to 40 cm, in order to simulate the use-case of an actual iPad. The camera stays in the central position; and it was tilted down as seen necessary in order to center the subject's face in the camera image.

We chose this set of experiments in order to see how our system performs in real situations. For example, in the case of carrying out a remote usability research of a given website over the Internet, we do not have the control over the camera position for all the subjects in the study. The second setup addresses this issue and is designed to find an answer to whether this constraint constitutes a problem for our program. The third setup tests the performance of our system in a mobile environment where the screen size is smaller and the user watches the screen from a shorter distance. The last setup is intended to evaluate how successful the subjects are in keeping their heads still and as a result it will tell us whether head pose variance is a must-have feature in applications like this one.

We also analyze the effect of different camera resolutions in these setups. This is done in an offline manner by resizing the webcam video stream. The original 1280×720 high-definition image, which corresponds to the iPad's FaceTime frontal camera resolution, was downsampled to 640×480 (VGA

quality).

Several constraints made us decide on this specific setup. For example, the usage of two cameras in parallel was considered; however, the complexity of synchronizing the camera control and setting up the system was considered out of the scope of this work. The alternative location of the camera was previously closer to the corner of the display; but as the geometrical problems prevented the automatic point selection component from functioning correctly in this case, this final position has been decided.

4.2 Building a complete experiment suite

Preparing for the experiments meant that we had to make some improvements in our eye-tracking system so that it would enable offline processing. Moreover, we developed several tools for the analysis of results and in the end we created a complete eye-tracking experiment suite which is ready for use in eye-tracking research. The key properties of this suite are:

1. Video recording and replaying the experiment from video by simulating mouse clicks
2. Completely automatical offline analysis of large experiment datasets
3. Outputting a video file showing the target point and the estimation for easy analysis and detection of problems
4. Outputting calibration errors and gaze estimation results as a text file
5. Reporting tools which enable fast analysis of data
6. Automatic visualization tools that make the visual results ready as soon as the experiment is finished (see Section 5.5)
7. Uncluttered stimuli display for distraction-free execution of experiments

We made this experiment suite available online for the interested readers [1].

The stimuli used in this experiment suite is a red circular marker which moves to several positions on the stimuli display. This marker is shown on the stimuli display at the positions listed in Figure 4.2 and Figure 4.2. The second figure corresponds to the positions in the iPad setup where only part of the display is used.

4.3 Error calculation methodology

In Figure 4.1, the geometry involved in the error calculations can be seen. E stands for the midpoint of two eyes, C marks the center of the screen, x shows the actual horizontal coordinate of the tracked target and x' is the estimation. For simplicity, only horizontal error calculation is explained here.

[1] http://mv.cvc.uab.es/eyetracker/eye-tracking-experiment-suite.zip

The angular error is defined as the difference between the actual α angle (the angle formed by x, E and C) and the estimated angle α' (the angle formed by x', E and C). To calculate these angles, two distances should be calculated first:

D_{xC}**:** Distance between x and C (in cm)

$D_{x'C}$**:** Distance between x' and C (in cm)

These distances are converted from pixel units to cm units using the information already gathered about the display: width and height in cm, and resolution in pixels which was 1920×1080 for our display. Using these distances and distance of the eye center (E) to the display center point (C), the error in degrees is calculated as follows:

$$Err = abs(arctan(D_{xC}/D_{EC}) - arctan(D_{x'C}/D_{EC}))$$

Chapter 5

Results

In order to carry out our experiments on the setup described before, we recruited 18 subjects and conducted our experiments in 4 days. The subjects were presented a short informative document about the experiment they took part in (see Appendix B), and they were requested to sign a consent form for their participation in this experiment (see Appendix C). This document document asks for basic contact information, and permissions to share the recorded videos and other non-personal information publicly. Moreover, the conditions of the experiments and their rights are explained here. Another document that was created for these experiments is a methodology document which was intended for our usage (see Appendix D).

5.1 Effects of implemented modifications

In this section, we present the results which show the effects of the proposed changes on the performance. In order to achieve this, we reflect our changes on the original Opengazer code one by one and compare the results for all four experiments. Here, we compare 6 different versions of the system which reflect certain phases of our system over time:

1. **[ORIG]** Original Opengazer applications with automatic point selection and blink detection during testing applied
2. **[2-EYE]** Previous case + estimation using 2 eyes
3. **[TRACK]** Previous case + tracking changes
4. **[BLINK]** Previous case + excluding blinks during calibration
5. **[CORR]** Previous case + calibration error correction
6. **[NN]** Previous case + neural network estimator

	Horizontal err.		Vertical err.	
	$^\circ(\sigma)$	Px	$^\circ(\sigma)$	Px
ORIG	**1.82** (1.40)	95	**1.56** (0.81)	80
2-EYE	1.56 (1.28)	82	1.48 (0.86)	75
TRACK	1.73 (1.37)	91	1.58 (0.78)	81
BLINK	1.73 (1.37)	91	1.58 (0.76)	80
CORR	**1.62** (1.40)	85	**1.50** (0.77)	76
NN	5.12 (2.26)	266	2.26 (0.92)	115

(a)

	Horizontal err.		Vertical err.	
	$^\circ(\sigma)$	Px	$^\circ(\sigma)$	Px
ORIG	**2.07** (1.69)	109	**1.73** (1.04)	88
2-EYE	1.94 (2.14)	101	1.67 (1.07)	85
TRACK	1.77 (1.52)	93	1.50 (0.80)	76
BLINK	1.78 (1.53)	93	1.49 (0.80)	76
CORR	**1.68** (1.56)	88	**1.43** (0.80)	73
NN	5.33 (2.83)	277	2.09 (0.89)	107

(b)

Table 5.1: Standard experimental setup results for (a) 480 and (b) 720 camera

	Horizontal err.		Vertical err.	
	$^\circ(\sigma)$	Px	$^\circ(\sigma)$	Px
ORIG	**2.02** (1.45)	106	**1.82** (0.73)	93
2-EYE	1.54 (0.99)	81	1.65 (0.57)	84
TRACK	2.02 (1.39)	106	1.90 (0.60)	97
BLINK	2.01 (1.39)	106	1.90 (0.59)	97
CORR	**1.88** (1.37)	98	**1.78** (0.58)	91
NN	4.59 (1.87)	239	2.35 (0.73)	120

(a)

	Horizontal err.		Vertical err.	
	$^\circ(\sigma)$	Px	$^\circ(\sigma)$	Px
ORIG	**2.22** (1.68)	116	**2.07** (0.77)	106
2-EYE	1.70 (1.30)	89	1.82 (0.75)	93
TRACK	1.88 (1.15)	99	1.86 (0.66)	95
BLINK	1.89 (1.14)	99	1.88 (0.68)	96
CORR	**1.76** (1.15)	92	**1.78** (0.67)	91
NN	4.93 (2.48)	257	2.15 (0.46)	110

(b)

Table 5.2: Extreme camera placement experimental setup results for (a) 480 and (b) 720 camera

In all versions of the application, gaze is not estimated during blinks detected through the state machine explained previously.

The resulting performances are grouped according to the experimental setups (standard, extreme camera placement, chinrest, iPad) and presented in the following tables. Moreover, for each case, we have run the experiment both using the original HD videos and using the resized VGA videos. For each case and camera resolution, average horizontal and vertical errors for all suspects and all readings are given both in degrees and in pixels. For the errors in degrees, the standard deviation is also supplied in parentheses.

In Table 5.1, the results for standard setup is given. Especially for the high definition case, the improvements we have made on the system have affected the performance. Table 5.2 shows the performance values of the system in the experimental setup where the camera is placed towards the extreme end of the display. Here, the performance is inferior to the standard setup as we expected.

The performance of the setup where the chinrest was employed can be seen in Table 5.3. In comparison with the standard setup, we can observe the stabilizing effect of the chinrest in the lower error rates.

	Horizontal err.		Vertical err.	
	$^{\circ}(\sigma)$	Px	$^{\circ}(\sigma)$	Px
ORIG	**1.29** (0.62)	68	**1.53** (0.62)	78
2-EYE	1.04 (0.64)	55	1.37 (0.47)	70
TRACK	1.37 (0.85)	72	1.64 (0.45)	84
BLINK	1.38 (0.85)	73	1.62 (0.44)	83
CORR	**1.25** (0.78)	66	**1.53** (0.44)	78
NN	4.06 (1.25)	212	2.04 (0.66)	104

(a)

	Horizontal err.		Vertical err.	
	$^{\circ}(\sigma)$	Px	$^{\circ}(\sigma)$	Px
ORIG	**1.35** (0.63)	71	**1.61** (0.71)	82
2-EYE	1.18 (0.82)	62	1.46 (0.54)	75
TRACK	1.27 (0.85)	67	1.48 (0.56)	75
BLINK	1.27 (0.85)	67	1.47 (0.56)	75
CORR	**1.14** (0.79)	60	**1.36** (0.53)	69
NN	4.19 (1.03)	218	2.12 (0.64)	108

(b)

Table 5.3: Chinrest experimental setup results for (a) 480 and (b) 720 camera

	Horizontal err.		Vertical err.	
	$^{\circ}(\sigma)$	Px	$^{\circ}(\sigma)$	Px
ORIG	**3.18** (2.13)	81	**2.64** (1.38)	68
2-EYE	2.88 (2.22)	74	2.43 (1.20)	63
TRACK	2.61 (1.63)	67	2.27 (1.03)	58
BLINK	2.60 (1.63)	66	2.25 (1.01)	58
CORR	**2.49** (1.64)	64	**2.08** (0.92)	54
NN	5.12 (2.26)	266	2.26 (0.92)	115

(a)

	Horizontal err.		Vertical err.	
	$^{\circ}(\sigma)$	Px	$^{\circ}(\sigma)$	Px
ORIG	**2.87** (1.66)	73	**2.49** (0.91)	64
2-EYE	2.54 (1.64)	65	2.46 (1.05)	63
TRACK	2.32 (1.29)	59	2.17 (0.75)	56
BLINK	2.31 (1.30)	59	2.14 (0.77)	55
CORR	**2.15** (1.32)	55	**2.02** (0.76)	52
NN	5.33 (2.83)	277	2.09 (0.89)	107

(b)

Table 5.4: iPad experimental setup results for (a) 480 and (b) 720 camera

	Horizontal err.		Vertical err.	
	°(σ)	Px	°(σ)	Px
Standard	1.62 (1.40)	85	1.50 (0.77)	76
Extreme	1.88 (1.37)	98	1.78 (0.58)	91
Chinrest	1.25 (0.78)	66	1.53 (0.44)	78
iPad	2.49 (1.64)	64	2.08 (0.92)	54

(a)

	Horizontal err.		Vertical err.	
	°(σ)	Px	°(σ)	Px
Standard	1.68 (1.56)	88	1.43 (0.80)	74
Extreme	1.76 (1.15)	92	1.78 (0.67)	91
Chinrest	1.14 (0.79)	60	1.36 (0.53)	69
iPad	2.15 (1.32)	55	2.02 (0.76)	52

(b)

Table 5.5: Overall results for the final application version in four experimental setups for (a) 480 and (b) 720 camera)

In Table 5.4 the iPad setup results are resumed. Although the pixel errors are smaller, the angular errors appear larger as the distance of the subject to the display is smaller.

5.2 Comparison of experimental setups

In this section, we gather and present the results of the application version labeled as CORR, which shows the best performance. The results for all four experiments are provided in Table 5.5. Again, the results achieved by using 480 videos and 720 videos are shown separately for easy comparison. Here, we can see that our expectations in choosing these setups come true: the extreme camera placement decreases system's performance, the chinrest decreases the error rate around 25%. The values are given in pixels; however, it is trivial to approximately obtain the errors in degrees. For iPad setup, each 25 pixels of error corresponds to 1° of angular error and for the standard setup, each 52 pixels correspond to 1° of angular error.

5.3 Error distributions

We analyze the distribution of system's errors w.r.t. two factors: the number of frames that have passed since a test marker is shown on the screen and the position of the marker. In the first case, we want to see how many frames should pass in order to let the system estimate the gaze point accurately. This time period also includes the interval during which the user settles their eyes towards the target, and is affected primarily by their reflexes to fixation on the stimuli.

During calibration and testing, the target pointers move to 15 different locations on the screen. These locations form a grid of 3 rows and 5 columns. In the iPad setups, only 12 locations are used which are arranged in a 4×3 grid. In Figure 5.1, each cell shows the mean error and the standard deviation (in parantheses) in pixels for each test pointer location as shown in Figure 4.1. The same information for iPad setup can be found in Figure 5.2. The positions having the smallest and largest errors have been marked in bold in each table. The general trend in the horizontal errors show that

error rates are generally higher along the edge of the display. However, we cannot come to the same conclusion in the iPad results as the points in the first and third columns do not have similar results.

118 (128)	76 (106)	52 (48)	**46 (54)**	135 (209)
115 (184)	73 (87)	54 (48)	57 (63)	61 (110)
178 (246)	129 (144)	84 (81)	68 (54)	104 (130)

(a) Horizontal

58 (70)	56 (65)	73 (137)	**43 (59)**	75 (136)
90 (63)	**95 (76)**	60 (57)	72 (78)	80 (61)
81 (68)	83 (106)	63 (76)	81 (170)	62 (103)

(b) Vertical

Figure 5.1: Standard experimental setup error distribution w.r.t. test point locations (pixels)

101 (78)	54 (49)	55 (67)
87 (88)	40 (40)	**25 (33)**
114 (138)	43 (29)	30 (50)
121 (85)	59 (44)	35 (44)

(a) iPad horizontal

60 (64)	**36 (51)**	**72 (102)**
71 (51)	53 (56)	54 (59)
72 (65)	54 (50)	54 (58)
41 (38)	47 (44)	67 (82)

(b) iPad vertical

Figure 5.2: iPad experimental setup error distribution w.r.t. test point locations (pixels)

In Figure 5.3, both of these aspects can be seen more easily. For each sub-chart, the horizontal axis shows the number of frames that has passed since the test point has been shown on the screen. The vertical axis shows the errors for both horizontal (blue) and vertical (red) directions. These charts show that only after the 10^{th} frame, the error becomes stable; meaning that the subject needs this amount of time to be able to focus on the displayed marker.

5.4 Inter-subject variances

In Figure 5.4, the inter-subject variances are plotted. Here, the medium red lines show the median and the box limits show the 25^{th} and 75^{th} percentiles for each case. The black lines above and below the boxes denote the extreme data points which are not considered outliers. The outliers are shown with a red plus sign. The plot contains 8 sub-plots: standard experimental setup (horizontal, vertical), chinrest setup (horizontal, vertical), extreme camera placement setup (horizontal, vertical), iPad setup (horizontal, vertical).

The figure shows that in the horizontal case, the boxes are mostly overlapping and they are not separable. Therefore, the errors in pixels follow the same trend. However, in the extreme camera placement setup, the variance is higher and the differences between subjects are larger. In the vertical errors, it can be argued that the extreme camera placement setup should be considered as an outlier group; however, it must be validated by further statistical analysis.

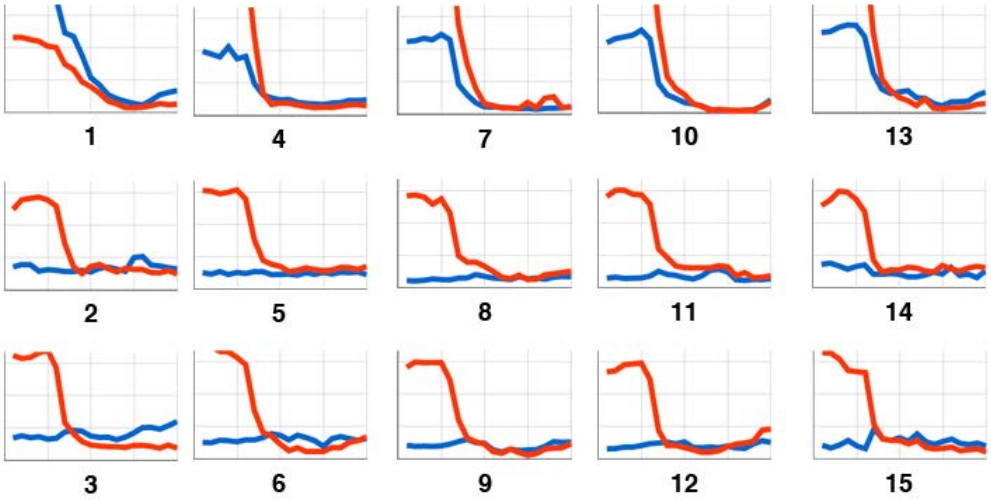
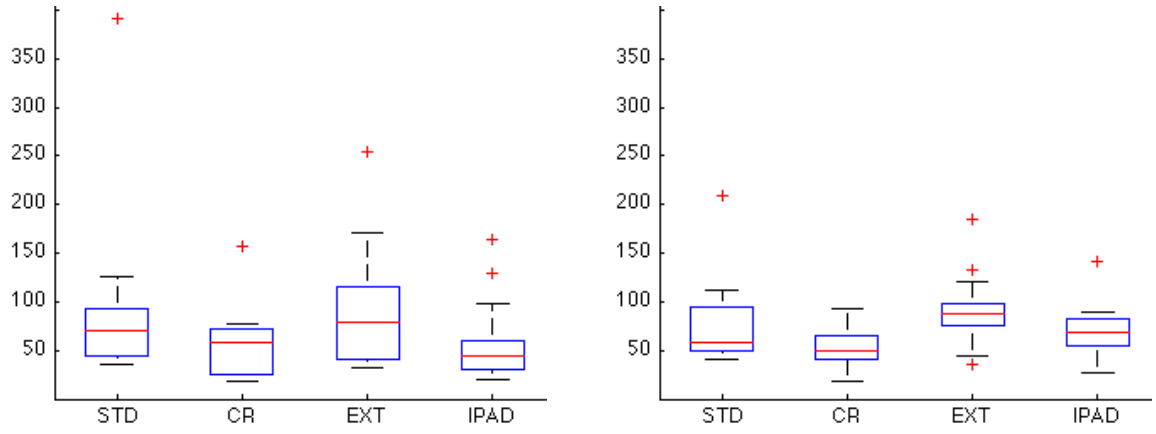


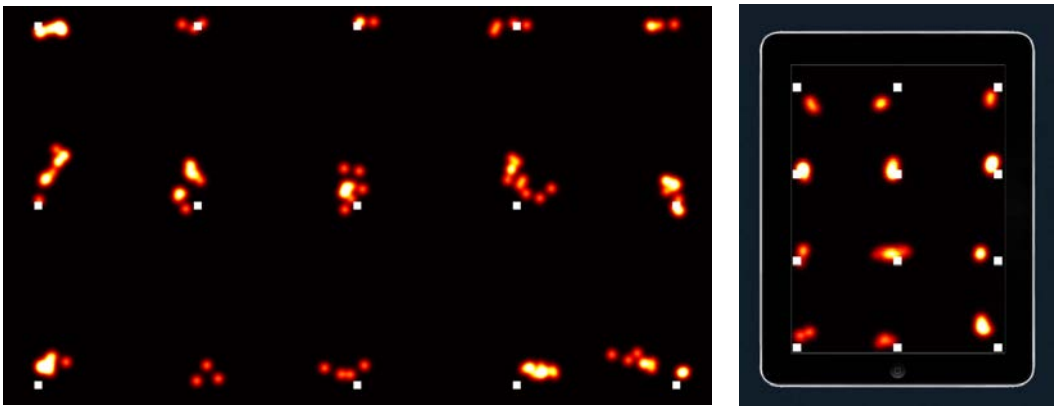
Figure 5.3: Error distribution w.r.t. frames and test point locations. Each sub-chart shows the distribution of horizontal (blue) and vertical (red) errors for the test point that is displayed at the corresponding location on the screen. Horizontal axis denotes the number of frame (1-20) and vertical axis denotes the error in pixels (max. 500 shown).



(a) Horizontal

(b) Vertical

Figure 5.4: Distribution of errors in pixels w.r.t. subjects shown on a box plot. The sub-plots show the errors of four experimental setups (standard, chinrest, extreme camera, iPad). The red lines denote the median and the blue box shows the 25^{th}-75^{th} percentile range. The outlier subject performances are shown with red plus signs.



(a) Heatmap for experiments using the entire screen area

(b) Heatmap for iPad setup

Figure 5.5: Heatmap visualization of results. Areas with high intensity show the accumulated gaze point estimations around the same point. Square white markers denote the target points.

5.5 Qualitative analysis

In the previous sections, numerical performance values of our system were provided. Here, we show some examples of the visualization techniques we have used to assess the performance qualitatively.

In Figure 3.3, the first technique that we employ to visualize the numerical performance values is shown. Here, every target point and the estimations belonging to it are displayed by a colored symbol (triangles, squares, etc.). The whole image represents the testing screen, and target points are shown at their corresponding locations on the image. The pointss can be differentiated from the estimates by their larger size. In results with good performances such as this one, the estimates are mostly accumulated near the target itself. Also, the magnitude of the error can be easily seen by checking the length of the line which connects the average estimation center to the target point. In experiments where the system cannot estimate the gaze very well, the estimate symbols may appear further away from the target point and large lines can be seen in the image. One last feature of this representation is the display of calibration errors, which are the blue plus signs located close to each test point. These signs show whether the calibration errors are correlated with the testing errors.

Another, simpler representation makes use of heatmaps as seen in Figure 5.5(a). Heatmaps were created in Matlab by first accumulating the gaze estimations on an image representing the display, and then filtering the resulting image with a Gaussian filter of 10×10 pixels. Final representation is created by adjusting the intensities so that the area with the highest intensity is shown by the white color. Therefore, the brighter areas in these images denote the locations where estimations are accumulated in higher number and we expect these locations to be close to the test targets in a well-performing experiment. The test targets are shown by small white dots for easy comparison and analysis of the visual.

Chapter 6

Discussion

The results of the experiments that we have carried out contain the answers we asked ourselves when starting this work. Is the performance level acceptable compared to commercial eye-trackers? Does the difference in setups change the performance of the system? Now, we will analyze the results and search for the answers to these questions and many others.

To begin with, considering the 1.68° horizontal and 1.43° vertical errors of the final system in the standard experimental setup, we can conclude that we have improved the original system by 18% horizontally and 17% vertically. The performance difference in the same experiment done with VGA cameras (11% horizontally, 4% vertically) is comparably lower than the first case, which shows us that our contributions in this work show their true performance with the increased image quality. From another aspect, it means that the future advancements in cameras will favor the methods we proposed.

Commercial remote eye-trackers have a performance of between 0.1 and 1°. One popular model, Tobii X1 Light Eye Tracker, is reported to have an accuracy of 0.5° under ideal conditions [34]. With up to 300 Hz sampling rates, these devices are crucial for research studies where these speeds are necessary. However, as even their rental prices are starting from thousands of euros, and as this sampling rate is not needed in all use cases, we believe that our system poses a good alternative with promising performance levels.

By analyzing the distributions of errors w.r.t. test point locations, we come to several conclusions. First, the horizontal errors are higher along the sides of the screen. In the vertical direction, the same is true for the lower parts of the screen. However, in the second case the difference is not as huge as in the first one. These observations should be taken into account when using the eye-tracker for practical purposes. For example, if the buttons of a graphical user interface (GUI) is to be controlled by gazing, the buttons should not be of the same size in all parts of the screen. Instead, the buttons closer to the sides should be given larger sizes. Our guess on this distribution of errors is that the Gaussian process (GP) estimator cannot generalize very well outside the screen area covered by the test points. For instance, the test pointers are located at least around 90 pixels away from the screen edges and it

is possible that the area between the pointers and the edges are not trained perfectly. In order to verify this, a new experimental setup may be prepared where pysical pointers are used to calibrate the system for out-of-bounds locations.

When analyzing the distribution of errors among the subjects, we see that the chinrest acted as a stabilizing factor and decreased the variance of errors. However, there is not a clear separation among the setups in the horizontal direction. In the vertical direction, the error rates of extreme camera placement setup is higher compared to others, and also the variance is larger. This is mostly caused by problems in detecting and tracking points from an angle.

Going into more detail, we see that in the tables where we evaluate the progressive improvements we have made on the system, we see that mostly we are pushing the performance in the positive direction. We observed that excluding the blink frames from the calibration process (application version labeled BLINK) did not have a perceivable effect on the performance. We argue that the averaging step in the calibration procedure already takes care of the outlier images. The other component that failed our expectations is the neural network estimator, which performed up to 2.5 times worse than the Gaussian process estimator. As we tried several methods while implementing this component, and as we had a sample implementation at hand, at this time we are not clear about the reasons of this failure.

An aim of this work was to analyze the factors which have an effect on the performance. Here, we conclude with a list of the factors that we have analyzed and our conclusions on their effects:

Camera resolution: Using a higher camera resolution improves the results between 6% and 14% in the last three setups. However, in the standard setup we observed a descrease in performance in all versions of the application. Still, comparing the effects of our contributions, we see that these the improved CORR version recovers most of the performance lost in the ORIG (see Table 5.1).

Camera position: Shifting the camera from the top center of the display decreased the performance by 5% horizontally and 24% vertically. Moreover, the error rate is higher in the horizontal direction in the setup where the VGA camera is used. Therefore, we conclude camera position affects the performance and that lower resolution cameras are affected more.

Stability: The use of a chinrest improves the performance by 32% horizontally compared to the standard setup. Also the variance among subjects is reduced, which increases the reliability of this setup for experimental purposes. We also observed that in the vertical errors, this difference is not as significant.

Subjects: As seen in Figure 5.4, there are not many outliers among our subjects. In most of the cases, the same subject is the only one for whom we have the higher error rates. Thus, we conclude that mostly the subjects will not have problems in taking part in this kind of experiments.

Lighting conditions: The ambient light is an important factor as it affects both the camera frame rate, and the image quality. During our experiments, we have observed that a change in this factor may effect especially corner detection algorithms and part detectors, as shadows and other noise appear in the webcam image.

Chapter 7

Conclusions

In this work, we built an eye-tracking system which is ready to use for experimental purposes and which has all the necessary tools to analyze the results, repeat the experiments and automate these procedures. Although the final performance is not on the same level as commercial eye-trackers, we have come a long way with our contributions. We believe that it poses a cheap alternative working in real-time which requires only a webcam (70 Euros in our setup) and we also believe that it has potential usage areas.

The fact that a simple improvement of estimating the gaze point for both eyes and then averaging them is left out of the base system shows us that there are many unexplored aspects of the system which are waiting for improvements. Therefore, we believe there is still much work to do and the achieved performance levels can be taken even further.

From the experiments, we learned that resolution, subject stability, camera position all have an effect on the performance. We believe that in practical applications of this system, these conditions should be changed to keep the performance as high as possible. Moreover; lighting conditions, which tend to vary during the day even at the same location, also affect the error rates and therefore constitutes a problem that should be addressed.

Our final system is not head-pose invariant; therefore, the changes in subject's head-pose are reflected in the errors directly. We believe that one of the next steps to take should be building a real-time face tracker and implementing one of the model-based methods analyzed in the Introduction. This improvement will remove many constraints from the system such as camera placement, subject stability and it will enable more practical uses.

Coming to an end of a study which continued for several months, we strongly believe that we have learned a lot and at the same time created valuable contributions. The experiment suite is available for download publicly, and a dataset gathered during the experiments will be published which will only contain the experiments of the subjects who willingly consented to their videos being shared.

Bibliography

[1] H. Wu, Q. Chen, and T. Wada, "Conic-based algorithm for visual line estimation from one image." in *FGR*. IEEE Computer Society, 2004, pp. 260–265.

[2] J. Chen and Q. Ji, "3D gaze estimation with a single camera without IR illumination." in *ICPR*. IEEE, 2008, pp. 1–4.

[3] S. M. Munn and J. B. Pelz, "3D point-of-regard, position and head orientation from a portable monocular video-based eye tracker." in *ETRA*. ACM, 2008, pp. 181–188.

[4] D. W. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 478–500, 2010.

[5] (2012, Sep.) Tobii eye-trackers. [Online]. Available: http://www.tobii.com/

[6] (2012, Sep.) SensoMotoric Instruments eye-trackers. [Online]. Available: http://www.smivision.com/

[7] K. Toyama, ""Look, ma–no hands!" Hands-free cursor control with real-time 3D face tracking," in *Workshop Perceptual User Interfaces*. IEEE Computer Society, 1998, pp. 49–54.

[8] E. Y. Kim, S. K. Kang, K. Jung, and H. J. Kim, "Eye mouse: mouse implementation using eye tracking," in *Consumer Electronics, 2005. ICCE. 2005 Digest of Technical Papers. International Conference on*, jan. 2005, pp. 207 – 208.

[9] (2012, Sep.) Gazehawk website. [Online]. Available: http://www.gazehawk.com

[10] (2012, Sep.) EyeTrackShop website. [Online]. Available: http://www.eyetrackshop.com

[11] A. T. Duchowski, *Eye tracking methodology – theory and practice.* Springer, 2003.

[12] R. Newman, Y. Matsumoto, S. Rougeaux, and A. Zelinsky, "Real-time stereo tracking for head pose and gaze estimation." in *FG*. IEEE Computer Society, 2000, pp. 122–128.

[13] Y. Matsumoto and A. Zelinsky, "An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement." in *FG*. IEEE Computer Society, 2000, pp. 499–505.

[14] J.-G. Wang and E. Sung, "Gaze determination via images of irises." in *BMVC*, M. Mirmehdi and B. T. Thomas, Eds. British Machine Vision Association, 2000.

[15] H. Yamazoe, A. Utsumi, T. Yonezawa, and S. Abe, "Remote gaze estimation with a single camera based on facial-feature tracking without special calibration actions." in *ETRA*. ACM, 2008, pp. 245–250.

[16] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 681–685, 2001.

[17] T. Ishikawa, S. Baker, I. Matthews, and T. Kanade, "Passive driver gaze tracking with active appearance models," in *Proceedings of the 11th World Congress on Intelligent Transportation Systems*, vol. 3, 2004.

[18] C. Holland and O. V. Komogortsev, "Eye tracking on unmodified common tablets: challenges and solutions." in *ETRA*, C. H. Morimoto, H. O. Istance, S. N. Spencer, J. B. Mulligan, and P. Qvarfordt, Eds. ACM, 2012, pp. 277–280.

[19] P. Zielinski. (2012, Sep.) Opengazer: open-source gaze tracker for ordinary webcams (software). [Online]. Available: http://www.inference.phy.cam.ac.uk/opengazer/

[20] L.-Q. Xu, D. Machin, and P. Sheppard, "A novel approach to real-time non-intrusive gaze finding." in *BMVC*, J. N. Carter and M. S. Nixon, Eds. British Machine Vision Association, 1998.

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

[22] J. S. Agustin, H. H. T. Skovsgaard, E. Møllenbach, M. Barret, M. Tall, D. W. Hansen, and J. P. Hansen, "Evaluation of a low-cost open-source gaze tracker." in *ETRA*, C. H. Morimoto, H. O. Istance, A. Hyrskykari, and Q. Ji, Eds. ACM, 2010, pp. 77–80.

[23] Z. Savas. (2012, Sep.) TrackEye: Real-Time Tracking of Human Eyes Using a Webcam. [Online]. Available: http://www.codeproject.com/Articles/26897/TrackEye-Real-Time-Tracking-Of-Human-Eyes-Using-a

[24] M. Castrillón-Santana. (2012, Sep.) Modesto castrillón-santana. [Online]. Available: http://mozart.dis.ulpgc.es/Gias/modesto.html?lang=0

[25] S. Hameed. (2012, Sep.) Haar cascade for eyes. [Online]. Available: http://www-personal.umich.edu/ shameem

[26] X. Haiying and Y. Guoping, "A novel method for eye corner detection based on weighted variance projection function," in *Image and Signal Processing, 2009. CISP '09. 2nd International Congress on*, oct. 2009, pp. 1 –4.

[27] C. Xu, Y. Zheng, and Z. Wang, "Semantic feature extraction for accurate eye corner detection." in *ICPR*. IEEE, 2008, pp. 1–4.

[28] G. Santos and H. Proenca, "A robust eye-corner detection method for real-world data," in *Biometrics (IJCB), 2011 International Joint Conference on*, oct. 2011, pp. 1 –7.

[29] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[30] Q. Wang, P. Moin, and G. Iaccarino, "A high order multivariate approximation scheme for scattered data sets," *Journal of Computational Physics*, vol. 229, no. 18, pp. 6343 – 6361, 2010.

[31] (2012, Sep.) MIR | Free software download at SourceForge.net. [Online]. Available: http://sourceforge.net/projects/mvinterp

[32] O. V. Komogortsev. (2012, Sep.) Identifying fixations and saccades in eye tracking protocols. [Online]. Available: http://cs.txstate.edu/ ok11/nnet.html

[33] S. Nissen, "Implementation of a Fast Artificial Neural Network Library (fann)," *Report, Department of Computer Science University of Copenhagen (DIKU)*, vol. 31, 2003.

[34] (2012, Sep.) Specification of Gaze Precision and Gaze Accuracy, Tobii X1 Light Eye Tracker. [Online]. Available: http://www.tobii.com/Global/Analysis/Marketing/Brochures/ProductBrochures/Tobii_X1_Light_Eye_Tracker_Technical_Specifcation_Leaflet.pdf

Appendix A

Building Steps of an Infrared Camera

(a) IR-only filter, camera lens and camera body

(b) Camera lens with IR blocking filter removed

Figure A.1: Parts of the built IR camera and the detail of the lens

Many webcams come with sensors which can capture infrared (IR) lights as well. However, as the lights in this wavelength are seen as noise in visible spectrum images, a filter which blocks IR light is used in these cameras. Based on this idea, we tried to build our own IR camera by removing the filter in the camera's lens system and placing another filter which only lets in IR lights in front of the camera.

We can group the cameras in two according to the structure of the IR filter: those that have the filter in the form of a coating attached to the lens, those that have a separate glass part which contains the filter. The filters in the second group are easier to remove, therefore we took the Trust Spotlight webcam which is of this structure and removed its filter. After placing the IR-only filter in front of it, we could take IR images with it.

Appendix B

Information for Subjects Document

Information for Subjects

You will take part in several experiments today. The purpose of these is understanding the point on the screen our subject (you) is looking at, at a given time using only our webcam stream. Please read the following instructions carefully and try to keep them in mind during the experiments. Also please pay attention to the corrections and warnings of the experiment conductor.

The expected length of the experiments is around 15-20 minutes.

Positioning Your Body

- You are not allowed to move the chair from the marked position.

- Sit in a comfortable position on the chair.

- Make sure you are facing the center of the monitor directly.

- Place your arms on the table in order to achieve a more steady posture.

- Try to keep as still as possible during experiments, you will be told when you can relax between experiments.

- We will place the monitor to a specific distance from you, therefore try not to get closer or further away from the monitor.

- If you feel uncomfortable at any point and lose your concentration, please inform the conductor in order to restart that experiment after a short pause.

- Please note that in the experiments carried out using the chin rest, most of these points will already be handled. In this case, try not to separate your head from the chin rest and try not to rotate your head.

Experiments
- In the beginning of each experiment, look steadily at the center of the monitor while the necessary adjustments are made in the application.

- **Training phase:** You will be asked to follow the red dot on the screen while it changes its location in intervals of ~1 seconds. If the errors in training are too high, you may be asked to repeat this process.

- **Testing phase:** After training, you will be asked to follow the dot once more while it goes through the same displacements on the screen.

- Remember that in the iPad experiments (those where the background contains an iPad image) the movement of the dot is confined to the area specified by the iPad screen.

Results
You will be presented with immediate results of the experiments after your part is completed. These results will show how accurate the system has tracked your gaze point. However, as the experiment will continue with other subjects, a more thorough explanation is left for later on.

Once the more detailed results of the overall experiments including other subjects are compiled, you will be informed of these results too.

The results will be used in the publications related to this research; information about the subject (gender, age, glasses/contact lenses usage, and other optical problems) may be included if they are relevant for explaining the results.

Appendix C

Subject Information and Conditions Document

Opengazer Experiments - Subject Information

Name Surname:

Email:

Gender: ☐Male ☐Female

Age:

Glasses/Contact Lenses: ☐Glasses ☐Contact lenses

Other Optical Problems:

☐ I give permission for the information above (excluding name, surname and email address) to be shared publicly (i.e. in publications, public datasets, on the Internet).

☐ I give permission for the recorded videos of my face to be shared publicly.

Experiment Conditions

The conductors of these experiments have the following responsibilities before the subject:

- Informing the subject about the experiments and clarifying any questions that may arise.
- Informing the subject about the usage and publication of recorded data and related results.

The subject keeps the following rights reserved:

- The right to cancel the participation in the experiments at any time.
- The right to be informed about the usage of recorded data.
- The right to revoke the permission for sharing recorded videos.

I agree to the terms above and I am willingly participating in the experiments as described in the document titled “Information for Subjects”. I confirm that a copy of this contract is provided to me and I have read the experiment methodology document.

Signed,

Appendix D

Experiment Description Document

Opengazer Experiments I

Goal

In these experiments, we will try to test the work we have done on top of the open-source Opengazer application. Our goal is to assess the change in performance that is due to the components we have worked on and modified. Furthermore, the performance differences that are caused by different subjects, different image resolutions, different camera placements, different test point locations on the screen, different screen sizes and different stability conditions (of the face) will be analyzed.

In order to achieve some of the goals above, we will record the videos of experiments programmatically and carry out the experiments in an offline manner.

Methodology

- Subjects will be presented the instruction sheet found at the end of this document.
- Subject will be seated in front of the monitor, with the monitor adjusted to be directly in front of the face in a centered position (i.e. the 3D line connecting the center of the eyes to the monitor center will be perpendicular to the monitor)
- The perpendicular distance to the monitor center will be fixed to the distance indicated in each setup below.
- During the calibration process, subject will be shown 15 training points on the screen and he/she will be asked to follow the dots while they are shown on the screen one after another.
- During the testing process, same set of points will be shown on the screen and again, the subject will be asked to follow them.
- After each experiment, if a modification in the setup is necessary, the experiments will be continued after the changes are made.

- Information about the subject that might be useful in understanding the results (contact lens or glasses usage, age, any other optical problems) will be collected for each subject.
- For each experiment, a video and a text file (containing the UI actions of the conductor such as starting testing, starting calibration, etc.) will be saved programmatically. These files will be processed in an offline manner to test different setups of the application.
- In order to simulate the effects of using a lower resolution camera, some of the offline experiments will be carried out by downsampling the recorded video to a resolution of 640x480.

Results

- Immediate results showing the heatmap of estimations and the symbol representation are generated automatically. In the second output, the estimations can be matched with the test point they belong to.

- Raw results (actual point coordinates and estimations) will be saved to the Excel format prepared beforehand. The following results will be prepared immediately:
    - Average errors for each frame (1-20) that the dots remain on the screen (also comparing the 1 eye and 2 eye approaches) (in pixels and degrees)
    - Average errors w.r.t. each point on the screen (in pixels and degrees)
    - Average horizontal errors for each column of points (5 columns) and average vertical errors for each row of points (3 rows) (in pixels and degrees)
    - All horizontal and vertical errors for each point and each frame of the testing phase (in pixels and degrees)

- Combining the results of the three experiments, further outputs will be achieved:
    - Whether the camera placement has a visible effect on the results
    - Whether the camera resolution has a visible effect on the results
    - Whether the results vary greatly among different subjects
    - Whether using chinrest has a visible effect on the results