Ubiquitous Head-Mounted Gaze Tracking

Dissertation zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Fakultät für Informatik und Mathematik
der Universität des Saarlandes

vorgelegt von
Christian Karl Lander, M.Sc.
Saarbrücken 2018

Dean:
Prof. Dr. Sebastian Hack

Head of Committee:
Prof. Dr. Jürgen Steimle

Reviewers:
Prof. Dr. Antonio Krüger
Prof. Dr. Andrew Duchowski

Committee member:
Dr. Gerrit Kahl

Day of defense:
January 24, 2019

Notes on Style
References to web resources (e.g., links to articles published on the Internet) are provided as URLs within footnotes with date of last access; longer URLs have been shortened. References to scholarly published resources (e.g., journal articles and conference proceedings) can be found in the Bibliography at the end of this thesis. When referring to unknown individuals in third person singular (e.g., users of a system or anonymous participants of a study) we alternate between the female and male pronouns between paragraphs for reasons of gender neutrality.

Acknowledgments

Over the last few years I had the pleasure to get into contact with a lot of great people who supported me in a variety of ways in writing this thesis. In the following, I want to use the possibility to thank some of them in particular. I apologize if anyone has been left out.

First of all, I deeply thank **Antonio Krüger** for the opportunity to work on this thesis. I would especially like to thank him for all the guidance and support he provided on this thesis and all related publications. Furthermore, I thank **Andrew Duchowski** for not only reviewing my thesis, but also for inspiring my overall work early on with his papers and research in related fields. I also thank **Markus Löchtefeld, Sven Gehring** and **Florian Daiber** for their valuable support, fruitful discussions and a lot of fun.

I thank **Margaret De Lap** who has proofread not only this dissertation, but also my publications in the last few years. I am sure that your efforts contributed significantly.

I also thank **Felix Kosmalla, Marco Speicher** and **Frederik Wiehr** for our collaborations during Software Campus. I will never forget our various business trips.
I had the pleasure to work in an inspiring and creative environment at DFKI. For this, I want to thank all my colleagues for the great time and working atmosphere. Furthermore, I thank all students who worked with me.

Most certainly, my deepest thanks go to my whole family. I especially want to thank my parents **Anke** and **Karl-Heinz** for supporting my dreams through all these years. My greatest thanks go to my lovely wife **Kristina**! I am eternally grateful for our two children. You were my anchor during all good and bad times while writing this thesis.

Für Marleen und Mats.

Abstract

Recently, gaze-based interfaces in stationary settings became a valid option, as remote eye trackers are easily available at a low price. However, making gaze-based interfaces ubiquitous remains an open challenge that cannot be resolved using remote approaches. The miniaturization in camera technology allows for gaze estimation with head-mounted devices, which are similar to a pair of glasses. A crucial step towards gaze estimation using such devices is calibration. Although the development is mainly technology driven, a hypothetical fully calibrated system in which all parameters are known and valid is affected by calibration drift. In addition, attributes like minimal intrusiveness and obstruction, easy and flexible setup and high accuracy are not present in the latest commercial devices. Hence their applicability to spontaneous interaction or long-lasting research experiments is questionable.

In this thesis we enable spontaneous, calibration-free and accurate mobile gaze estimation. We contribute by investigating the following three areas: *Efficient Long-Term Usage* of a head-mounted eye tracker; *Location, Orientation & Target Independent* head mounted eye tracking; and *Mobile & Accurate Calibration-Free* gaze estimation. Through the elaboration of the theoretical principles, we investigate novel concepts for each aspect; these are implemented and evaluated, converging into a final device, to show how to overcome current limitations.

Zusammenfassung

Heutzutage sind blick-basierte Schnittstellen zur Interaktion eine Alternative, da remote Eye Tracking Systeme leicht zugänglich geworden sind. Allerdings ist die Herausforderung diese Schnittstellen überall verfügbar zu machen nicht mit remote Ansätzen zu lösen. Die Miniaturisierung in der Kamera-Technologie erlaubt die Blickbestimmung durch am Kopf getragene Systeme, die ähnlich zu einer Brille sind. Ein entscheidender Schritt zur Blickbestimmung mit diesen Geräten ist eine Kalibrierung. Obwohl die technische Entwicklung weit voran geschritten ist, wird die Präzision eines voll kalibrierten Systems in dem alle Parameter bekannt und validiert sind durch den sogenannten calibration drift beeinträchtigt. Eigenschaften wie ein einfaches und flexibles Setup und hohe Präzision werden von aktuellen Geräten oft nicht erfüllt. Daher ist die praktische Anwendbarkeit von diesen Systemen in interaktiven Szenarien und Experimenten fraglich.

In dieser Arbeit ermöglichen wir spontane, kalibrierungsfreie und mobile Blickbestimmung mit hoher Präzision. Folgende drei Bereiche werden dazu untersucht: Benutzung von Eye Trackern über einen *längeren Zeitraum*; Eye Tracking *unabhängig von Ort, Orientierung oder Fokuspunkt*; *Mobile und präzise kalibrierungsfreie* Blickbestimmung. Durch das Erarbeiten von theoretischen Prinzipien werden neue Konzepte entwickelt, implementiert und evaluiert, die in ein finales System münden, um spontane Blickbestimmung zu jeder Zeit an jedem Ort zu ermöglichen.

Relevant Publications

The work presented in this thesis, including figures and text fragments have in some cases appeared in the following publications. The chapters of this dissertation are partly based on these publications.

Full conference papers:

[98] C. Lander, M. Löchtefeld, and A. Krüger. **Heyebrid: A hybrid approach for mobile calibration-free gaze estimation.** In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies Volume 1 Issue 4, December 2017*, IMWUT'18, pages 149:1-149:29, 2018. ACM. (appears in Section 6.3)

[94] C. Lander, S. Gehring, M. Löchtefeld, A. Bulling and A. Krüger. **Eyemirror: Mobile calibration-free gaze approximation using corneal imaging.** In *Proceedings of the 16th International Conference on Mobile and Ubiquitous Multimedia*, MUM'17, pages 279-291, 2017. ACM. (appears in Section 6.2)

[93] C. Lander, S. Gehring, A. Krüger, S. Boring and A. Bulling. **Gazeprojector: Accurate gaze estimation and seamless gaze interaction across multiple displays** In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology. ACM Symposium on User Interface Software and Technology*, UIST'15, pages 395-404, 2015. ACM. (appears in Sections 4.2.2, 5.2)

[101] C. Lander, M. Speicher, N. Coenen, S. Biewer, D. Kahl and A. Krüger. **Collaborative Newspaper: Exploring an adaptive scrolling algorithm in a multi-user reading scenario** In *Proceedings of the International Symposium on Pervasive Displays. International Symposium on Pervasive Displays*, PerDis'15, pages 163-169, 2015. ACM. (appears in Section 1.5)

Short papers and notes:

[**95**] C. Lander, F. Kerber, T. Rauber, and A. Krüger. **A time-efficient re-calibration algorithm for improved long-term accuracy of head-worn eye trackers.** In *Proceedings of the 9th ACM Symposium on Eye Tracking Research and Applications. Symposium on Eye Tracking Research & Applications*, ETRA'16, pages 213-216, 2016. ACM. (appears in Section 4)

Poster papers and Demos:

[**100**] C. Lander, M. Speicher, N. Coenen, S. Biewer, D. Kahl and A. Krüger. **Collaborative Newspaper Demo: Exploring an adaptive scrolling algorithm in a multi-user reading scenario.** In *Proceedings of the International Symposium on Pervasive Displays. International Symposium on Pervasive Displays*, PerDis'15, pages 271-272, 2015. ACM. (appears in Chapter 1)

[**92**] C. Lander, N. Coenen, S. Biewer, and A. Krüger. **Kollaboratives Text Lesen: Adaptive Text Scroll Geschwindigkeit.** In *Proceedings of Mensch und Computer 2015*, MuC'15, pages 437-438, 2015. ACM. (appears in Chapter 1)

[**102**] C. Lander, F. Wiehr, N. Herbig, A. Krüger, and M. Löchtefeld. **Inferring landmarks for pedestrian navigation from mobile eye tracking data and google street view.** In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA'17, pages 2721-2729, 2017. ACM. (appears in Section 5.6)

[**99**] C. Lander, M. Speicher, F. Kerber, and A. Krüger. **Towards fixation extraction in corneal imaging based eye tracking data.** In *Proceedings of the 2018 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA'18, pages 558:1–558:6, 2018. ACM. (appears in Section 6.5)

Workshops & Workshop papers:

[**1**] A. Bulling, E. Kasneci, and C. Lander. **"Proceedings of the 7th Workshop on Pervasive Eye Tracking and Mobile Eye-Based Interaction.** In *Proceedings of the 2018 Symposium on Eye Tracking Research and Applications*. ETRA'18, Petmei Workshop, ACM, 2018. (appears in Section 6.5)

[**96**] C. Lander, F. Kosmalla, F. Wiehr, and S. Gehring. **Using corneal imaging for measuring a human's visual attention** In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and*

Proceedings of the 2017 ACM International Symposium on Wearable Computers. UbiComp'17, UbiTtention Workshop, ACM, 2017. (appears in Section 6.5)

[97] C. Lander, A. Krüger, and M. Löchtefeld. **"The story of life is quicker than the blink of an eye": using corneal imaging for life logging.** In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. UbiComp'16, Petmei Workshop, ACM, 2017. (appears in Section 6.5)

Bachelor's and Master's theses:

[156] T. Rauber. **A time-efficient re-calibration algorithm for improved long time accuracy of head-worn eye trackers**. *Master Thesis*. Advisor: Christian Lander and Frederik Kerber, Saarland University, Department of Computer Science, 2015. (appears in Chapter 4)

Contents

1 Introduction **1**

1.1 Motivation 1
1.2 History of Eye Tracking 4
1.3 Diversity of Applications 6
1.4 Problem Statement 8
1.5 Scenario 12
1.6 Research Questions 17
1.7 Thesis Outline 18

2 Foundations & Background **21**

2.1 The Human Eye 21
2.1.1 The Physiological Eye Model 21
2.1.2 Eye Movements 26
2.2 Acquiring Eye & Gaze Position 28
2.2.1 Eye Tracking 28
2.2.2 Gaze Estimation 32
2.3 Summary 34

3 Related Work **37**

3.1 Application of Eye Tracking 37
3.1.1 State of the Art Devices 37
3.1.2 Gaze-based Human Computer Interaction 45
3.2 Eye Tracker Calibration 54
3.2.1 Notes on Calibration 54
3.2.2 Re-Calibration Strategies 58
3.2.3 Calibration-Less Approaches 61
3.3 Corneal Imaging 64
3.4 Summary 67

4 Investigating the Re-Calibration of Head-Mounted Eye Trackers 71
4.1 Introduction . . . 72
4.2 Studying the Long-Term Usage of Head-Mounted Eye Trackers . . . 73
4.2.1 Eye Tracker Calibration . . . 75
4.2.2 Time-Efficient Recalibration . . . 75
4.3 User Evaluation . . . 77
4.3.1 Results . . . 80
4.4 Discussion . . . 86
4.5 Application to research questions . . . 90
4.6 Summary . . . 92

5 Spontaneous Gaze Estimation 95
5.1 Introduction . . . 96
5.2 GazeProjector – Accurate Gaze Estimation and Seamless Gaze Interaction Across Multiple Displays . . . 97
5.2.1 Tracking Spatial Relationships of Users and Displays . . . 99
5.2.2 Enabling Gaze Interaction On Large Display . . . 100
5.2.3 Implementation and Applications . . . 104
5.3 Evaluation . . . 113
5.3.1 Experiment I: Gaze Estimation Accuracy . . . 113
5.3.2 Experiment I – Results . . . 116
5.3.3 Experiment II: Multiple Displays . . . 119
5.3.4 Experiment II – Results . . . 122
5.4 Discussion . . . 124
5.5 Application to research question . . . 127
5.6 Application of GazeProjector in a Real-World Setting . . . 130
5.6.1 Eye Tracking & Navigation . . . 130
5.6.2 Automatic Gaze Estimation In The wild . . . 131
5.6.3 Implementation . . . 133
5.6.4 Evaluation . . . 139
5.6.5 Discussion . . . 141
5.7 Summary . . . 142

6 Mobile and Calibration Free Gaze Estimation Approach 145
6.1 Introduction . . . 146
6.2 EyeMirror – Mobile Calibration-Free Gaze Approximation . . . 148
6.2.1 The EyeMirror System . . . 151

6.2.2 Experiment I – Gaze Estimation on a Display 159
6.2.3 Experiment II – Influence of display content 165
6.2.4 Discussion 168
6.2.5 Application Example 172
6.2.6 Summary of Findings 175
6.3 hEYEbrid – A Hybrid Approach for Mobile Calibration-Free Gaze Estimation 175
6.3.1 hEYEbrid System 177
6.3.2 Evaluation 186
6.3.3 Discussion 192
6.3.4 Making hEYEbrid Mobile 194
6.3.5 Overall Limitations 199
6.3.6 Summary of Findings 201
6.4 Application to research question 202
6.5 Attention Maps in Corneal Imaging using hEYEbrid 205
6.6 Summary 208

7 Conclusion **211**
7.1 Major Contributions 211
7.2 Future Work 215
7.3 Closing Remarks 217

List of Figures **i**

List of Tables **xi**

Bibliography **xiv**

Chapter 1

Introduction

In the first chapter, we motivate the work and present an introduction into the topic of this dissertation. First we will provide a brief historical recap of the evolution of eye tracking. The emerged challenges and problems will be discussed and serve as a basis to develop the research question, we address with this work. In the last part of this first chapter, we will give an overview on the structure of this thesis.

1.1 Motivation

Simply put, eye tracking or oculography is the process of recording and analyzing human eye activity. This includes all different types of eye movements as well as the direction and point of gaze. When talking about this topic, it is important to develop our understanding of the purpose of tracking a person's eye. The human eye is the organ of vision and one of the basic senses that we use to perceive our environment, as already defined by the Greek philosopher Aristotle back in 350 B.C. (original text translated to English [2]). In practical terms, we move our eyes to change the direction of gaze and see a specific point in the environment at high resolution. Usually this is coupled with drawing our attention to that object or region. Consequently, the opportunity to track a person's eye and thus the point of gaze may give us information about what is attracting him or her and might be of interest [196]. Nowadays, we can see that science (e.g., neuroscience, psychology and computer science), industry (e.g., aviation and driving) and marketing research (e.g., advertising, the web) make use of eye tracking [37].

The machines to record a person's eye movement and perform gaze estimation are divided into three main categories. The electro-oculography (EOG) based method

places electrodes around the eye to measure the skin potentials, as proposed by Kaufman et al. [79]. The scleral search coil method uses a small coil that is embedded into a contact lens. It measures the voltage caused by an external electro-magnetic field [161]. Camera-based (video-based) techniques use the images of the eye to detect its characteristics in combination with images of the field of view to realize gaze estimation. This method is mainly applied nowadays by using either a head-mounted/mobile (cameras are mounted on the head) or a remote (cameras are placed in the environment) system.

In this thesis we consider eye tracking from the perspective of human computer interaction, in particular the usage of eye tracking as a *Ubiquitous Computing* device. This term dates back to 1988, and was originally introduced by Mark Weiser [200]. This paradigm (often also *Pervasive Computing*), also known as the third wave in computing, will make computers invisible for us (calm computing). That is to say, one person is using many computers, as they are totally integrated in our everyday objects. The shift to post-desktop devices has already been reached, as the three device classes *Tab, Pad* and *Board* – described by Weiser – are already well established through mobile phones, tablets and large-size touch screens. Interestingly, Mark Weiser already had the vision of eyeglasses as a ubiquitous computing device in 1997 [201], as they are a seamless extension of the human. A person looks through them and they just work, while the fact of wearing the glasses will be forgotten quickly. Besides Weiser, Robert J. K. Jacob also describes the vision of using eye trackers as an input device through gaze-based interfaces or as an analytics tool [71] in the early 90s.

According to the visions by Weiser and Jacob from above, we define a *Ubiquitous Computing for Eye Tracking Continuum*. Figure 1.1 depicts our three-dimensional continuum, which shows the rising complexity of the number of devices, locations and users from a *1 : 1 : 1* to a *(K x L) : M : N* relation (K = number of digital objects, L = number of physical objects, M = number of locations, N = number of users): The simplest scenario (1 : 1 : 1) is constituted by a desktop setting, in which a single user interacts with one digital device (e.g., a single display) at one location. The complexity can be increased in three different directions: (1) A single user can interact with several digital devices (e.g., display and mobile phone) and physical objects at the same location ((K x L) : 1 : 1). (2) One person can interact with one display at different locations (e.g., information screens at different urban places, 1 : M : 1). (3) Multiple users can interact with the same display at the same location (1 : 1 : N).

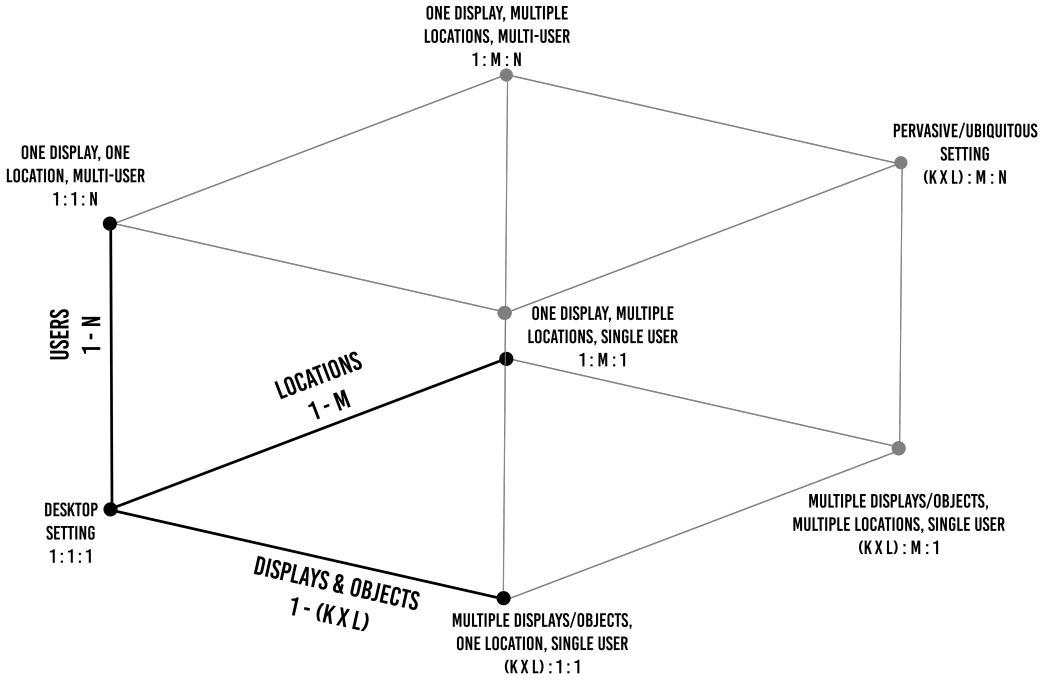


Figure 1.1: *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space highlighting the complexity of the scenario ((Displays x Objects) : Locations : Users), which increases from a simple desktop setting with one display at one location for a single user (1 : 1 : 1) to a pervasive/ubiquitous setting with multiple digital and physical objects distributed at many locations for multiple users (K x L : M : N).

The combination of all three dimensions results in the most complex variant, having multiple persons who interact with physical and digital objects distributed across multiple locations ((K x L) : M : N). With rising complexity through the space of this continuum, the challenges of eye tracking (and thus gaze-based interaction) are also increasing.

Stationary or remote eye trackers are devices that carry out the task from a certain distance. Their purpose is to measure the point of gaze on the object they are mounted on, most often a display. Recently gaze-based interfaces in desktop settings (the easiest setting in the continuum in Figure 1.1) have become a valid option, as remote eye trackers are easily available at a low price point. The latest devices for use with desktop computers are available for under $200 (Tobii 4C [1]). However, making gaze-based interfaces ubiquitous remains an open challenge. In a straight forward approach, the environment (the most complex setting in Figure 1.1) would have to be fully covered by sensors (e.g., cameras) to track people's eyes. Such an approach

[1] https://tobiigaming.com/products/peripherals/

is not feasible, neither from a technical nor an ethical point of view. In theory, head-mounted eye trackers can record a person's eye movements and measure the gaze everywhere and all the time. While a person is mobile and just wearing glasses, a large amount of applications will be enabled (an overview is presented in 1.3). As already mentioned by Weiser, people will forget that they are wearing eyeglasses. Besides ubiquitous computing, virtual reality constitutes an opposite paradigm, in which people dive into a completely computer-generated world. However, both directions can profit from eye tracking technology. In particular, the integration of gaze support in next-generation AR and VR devices seems inevitable, as the latest industrial evolutions suggest[2].

In general, there exist practical challenges that need to be resolved first, to pave the way towards the vision of pervasive gaze-based interfaces [19] by enabling ubiquitous eye tracking (as defined above).

1.2 History of Eye Tracking

Eye tracking has a long history, in which mainly the necessary technology has evolved. First studies go back to 1729, when mere visual observations were used to get insights into human eye movements. Back then, William Charles Wells investigated the perceived visual direction [199]. In 1878, Louis Emile Javal developed a machine to investigate human reading behavior. For his studies, he built an apparatus equipped with mirrors to observe eye movements. He introduced the term saccade, which he used to describe a jerky movement of the eye during reading [70].

By the end of the 19th century, the first complex constructions were developed to record eye movements. Delabarre (1898) mounted a plaster eye-cup to the eye that was connected to a small lever. The eye was stunned with cocaine during the procedure. With that apparatus he was able to record horizontal eye movements, which were drawn by the lever on a kymograph.

Right in the beginning of the 19th century, Dodge and Cline (1901) developed the first version of today's eye tracking technique [33]. They invented the principle of photographing the light reflected on the cornea. This method is much less invasive as it does not require any material directly connected to the human eyeball. They used a falling photographic plate to record only horizontal eye position. The plate

[2] https://pupil-labs.com/vr-ar/

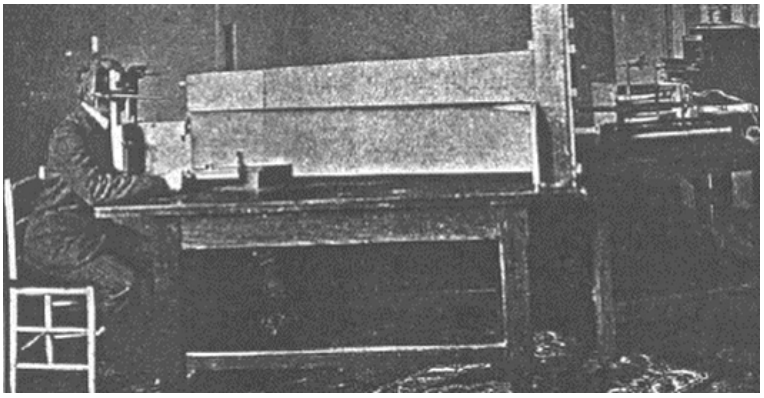


Figure 1.2: Apparatus from Dodge and Cline to record horizontal eye movements [33].

thus showed horizontal eye motion on the X-axis and time on the Y-axis. Figure 1.2 shows the developed machine. Four years later, in 1905, Charles Judd introduced eye movement recording in two dimensions with motion picture photography. Once more, they built a more invasive method: they put a white speckle onto the participant's eye. The first two-dimensional eye movement recording was possible, by combining head position plus the horizontal movement of one and vertical movement of the other eye, in 1920.

Two major improvements of the technology were achieved in 1939 and 1948. First, Jung created the first (theoretical) possibility to process eye tracking data in real time by attaching electrodes to the skin around the eyes. This method became known as electro-oculography (EOG). Later, Hartridge and Tompson invented the first head-mounted eye tracker [54]. This was a big step towards allowing free head movements during eye tracking.

In 1947, Paul Fitts (known for Fitts law) and his colleagues were the first to apply the method of eye tracking to usability testing. They used motion picture cameras to record a pilot's eye movements when landing an airplane. They were interested in how the pilots were using the cockpit's instruments and controls [24].

The 1970s were a productive period in eye tracking activities. A lot of research was conducted in collecting data to solve shortcomings in the technology and analysis. Primarily, the eye tracking devices became less intrusive and achieved a higher accuracy. Cornsweet and Crane developed an approach to separate eye from head movement by using multiple Purkinje images (i.e. reflections of objects from different surfaces of the human eye) [28].

In the 1980s, the computer industry was revolutionized, as the first personal computer was released by IBM. Also, the larger machines improved in terms of more computational power. This was the cornerstone to enable real-time eye tracking by using video-based eye tracking devices. In 1981, Bolt published his vision of using eye tracking for human computer interaction [15].

Since the 1990s, the eye tracking community has experienced a continual development in technology (improved specifications). The use of the devices to track people's eye movement and point of gaze with high accuracy and precision has expanded into several application fields up to today.

1.3 Diversity of Applications

In this section, we will give an overview about the different application areas of eye tracking. Andrew Duchowski provides a deep review of all use cases in his book *Eye Tracking Methodology* [37]. He reviewed two classes of applications, namely *diagnostic* and *interactive* ones (part 4 in [37]). We made a similar differentiation, into two typical application areas which use the information about eye position and movement as well as a person's gaze for *real-time* or *post-hoc* processing. Figure 1.3 depicts a two-dimensional classification scheme, in which we distinguish between the two aforementioned classes, as well as between the commercial and research application sector as a further dimension. Obviously, there is a certain overlap between commercial and research applications.

Marketing and Advertising. Probably the most prominent application use case of eye tracking and gaze estimation is the field of marketing research. For example, large companies, such as Tobii promote this as a primary use case for their head-mounted eye tracking device[3]. As the name suggests, it is about measuring the effectiveness of marketing campaigns to draw or increase people's attention toward a certain product, for example in a retail store. The shopper's visual behavior is analyzed to answer questions like: How is the shopper walking through the store? What products, displays and advertisements has he/she recognized? What objects did he/she miss? Such analysis is done afterwards, i.e. a person makes his/her purchase as usual while wearing an eye tracker. For example, Tobii provides such analyses as a service for retail stores[4]. This information can then be used to reposition products on a shelf, or

[3] https://www.tobiipro.com/fields-of-use/marketing-consumer-research/
[4] https://www.tobiipro.com/services/shopper-retail/

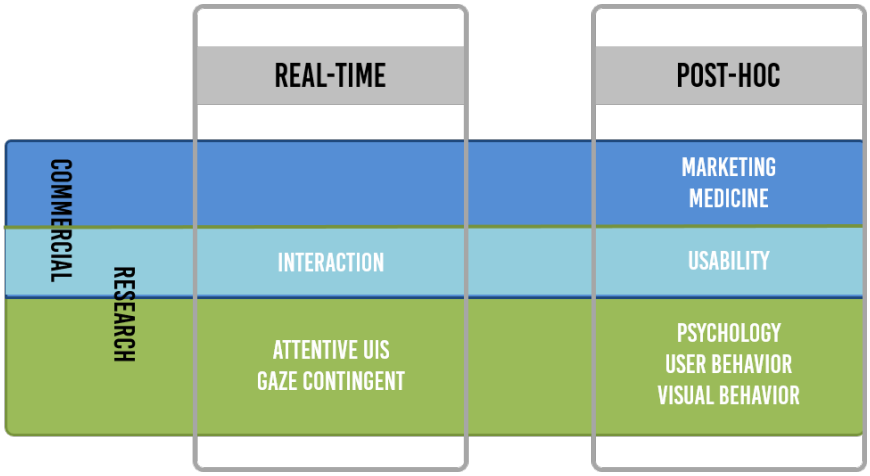


Figure 1.3: Application dimensions of eye tracking, adapted from Duchowski [37]: real-time vs. post-hoc processing and commercial vs. research applications.

advertisements, to improve the visibility. In a similar manner, such analytics can be also done for online stores, usually with remote eye trackers in a desktop scenario.

Usability & User Experience. Another field of use is usability testing and user experience (UX) research. For this application, there is an overlap in objectives of research and commercial services (e.g., SR Research[5]). Using eye tracking data to evaluate the *quality attribute that assesses how easy user interfaces are to use* (as defined by Jakob Nielsen [133]) actually goes back to 1947. As already mentioned in the history section 1.2, experiments by Paul Fitts were done to evaluate the layout of controls in an airplane cockpit. Goldberg and Kotval [47] developed a framework in which they defined various eye tracking metrics (e.g., number and length of fixations, relation between fixations and saccades) as well as the correlations to usability problems. A comprehensive overview is given by Jacob and Karn [72]. Schall and Bergstrom [11] discuss the usage of eye tracking in UX design (e.g., how users interact with mobile devices vs. large screens, or where people expect specific UI elements on a website). With the rise of the World Wide Web, usability testing of web pages based on eye tracking data became a major topic [41]. Usually the specific eye movement data has first recorded, then analyzed post-hoc. The insights gained were used by developers and user interface designers to adapt the UIs to improve user experience. As of now, there are first projects that process the data in real time to detect usability problems and create feedback mechanisms to support the user directly[6].

[5] http://www.eyelinkinfo.com/solutions_use.html

[6] Research project FEUAA funded by the German Federal Ministry of Education and Research under grant number 01IS12050

Psychology & User Behavior. Eye trackers are an indispensable data recording tool for research on visual behavior, perception, cognition and psychology. A prominent example of studying people's perception via eye tracking data is reading experiments [157]. Due to the advances in head-mounted eye tracking, the devices gained considerable attention for analyses of our daily activities [178], people's visual behavior [90] and cognitive processes such as visual memory recall [189] and selected personality traits [60].

Interaction. A field of usage addressed by both the research community and the commercial sector is gaze interaction for end-users that covers a broad area of applications [121]. Nowadays, a well-known use case is in the automotive industry[7], where eye tracking data is used for car control optimization and control handover in autonomous driving. Using gaze information in games has also become a valid option to integrate eye trackers into high-end computers[8]. Especially with the progress made in VR and AR devices, there is an increasing interest in the technology[9,10]. From the research perspective, eye tracking was investigated as an input modality in human computer interaction, including target selection via looking [180], eye typing on a virtual keyboard [111], multi-modal input by combining gaze with touch [148] and controlling mobile devices [40].

1.4 Problem Statement

Today, the development of head-mounted eye trackers is mainly driven from the technological side. With the advanced miniaturization of image sensory technology, the integration of tiny cameras into a head-mounted device has become possible. Such eye trackers consist of a glasses frame that is equipped with at least two cameras: first, an eye camera capturing a close-up image of the user's eye, and second, a world camera partially recording the user's current field of view. The purpose of the eye camera is to detect and track the pupil as well as its movements. A widely used approach to achieve this is through active illumination of the eye with infrared light and the so-called **P**upil **C**entre **C**orneal **R**eflection (PCCR) method. Gaze estimation is the process of mapping the pupil positions from the eye camera's into the world camera's coordinate system. We will present the different approaches for tracking a person's eye and computing the gaze in Chapter 2.

[7] http://smarteye.se/applied-solutions/
[8] https://tobiigaming.com/
[9] https://techcrunch.com/2016/10/24/google-buys-eyefluence-eye-tracking-startup/
[10] https://techcrunch.com/2017/06/26/apple-acquires-smi-eye-tracking-company/

Calibration. A crucial step towards gaze estimation is calibration, which is used to create a function that maps eye to gaze positions. It usually requires the user to fixate a sequence of visual stimuli in the environment. Current state-of-the-art eye trackers, such as Tobii Pro Glasses 2 [11], are built on model-based (geometric) gaze estimation. This method enables *calibration-less* gaze estimation. That means they adapt the model and compute personal parameters like the angle kappa (compare [122]) using at least one calibration point. However, to increase the gaze estimation accuracy and reduce the error, more calibration points are needed [197]. Further, these methods need complex hardware setups with more than two cameras and are extremely expensive, costing more than tens of thousands in USD (during the time of writing). On the other hand, eye trackers that rely on regression-based (interpolation) gaze estimation need a more extensive calibration step, but are usually cheaper (e.g., Pupil Labs[12]). All gaze estimation methods share the same set of parameters, which are acquired through the calibration. According to Hansen and Ji [52], these are:

- **Camera calibration** to determine the intrinsic camera parameters
- **Geometric calibration** to determine relative locations and orientations in the setup, like camera and light sources
- **Personal calibration** to estimate the cornea curvature, the angle kappa (offset between visual and pupillary axes)
- **Gaze mapping calibration** to determine the parameters of the eye to gaze mapping function

Drift. Even if it is possible to set up a fully calibrated eye tracker, i.e. a system in which the camera parameters and the geometry are known, we still face the problem of deterioration of gaze estimation accuracy. This effect is known as calibration drift, which describes the increasing difference between the actual and measured gaze direction [111]. In practice, eye tracking systems are never 100% accurate, and due to calibration drift their accuracy further suffers over time. This effect may be caused by changes in the eye physiology (e.g., wetness of the eye), changes in the environment (e.g., lighting of the room), head movements, or the motion of the cameras. Basically, there are three possibilities to handle such calibration drift:

- Post-hoc drift reduction is done by analyzing already recorded data [176, 61]. The drawback of this approach is obvious, as it cannot be applied for systems that have to react in real time.

[11] `https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/`
[12] `https://pupil-labs.com/store/`

- Dynamic re-centering [177] is an approach to observe the drift during runtime and use the information to immediately counteract it. This method, however, requires the user to actively select targets (usually on a screen) to update the calibration globally.
- The last option to address calibration drift is to periodically invoke the standard calibration procedure to re-establish the initial accuracy. The main drawback of this approach is the interruption of the user during the current task.

So far the different approaches above are mainly applicable in settings where a person's gaze is measured on a display. Practically, in cases like in-the-wild experiments and gaze estimation across multiple objects, a dedicated screen is needed to re-calibrate the device.

Invariance. Given the fact that we want to measure a person's gaze on a specific object (e.g., a display), we are confronted with another problem. Typically the calibration is performed for a fixed position and orientation of the user to a single object. While this is less of an issue for stationary settings, mobile settings and multiple – potentially large – objects evoke two types of motion: (1) user movements in front of an object to inspect other parts of it or get in a more comfortable position; and (2) head movements to reach targets outside the ocular motor range [49]. In addition, there might be multiple objects of interest, causing further movements. Both types of motion considerably reduce gaze estimation accuracy [23]. In order to achieve highly accurate gaze estimation, it is crucial to track a user's (and eye tracker's, respectively) position and orientation relative to the object of interest. So far this has been achieved by augmenting the environment with visual markers [17, 208] as well as using external tracking systems (e.g., OptiTrack[13]). These approaches can achieve high tracking and gaze estimation accuracy, but require a complex deployment (e.g., attaching visual markers to every object a user wants to interact with). Consequently, a truly pervasive and spontaneous gaze-based interaction is severely limited.

Parallax Error. In Figure 1.4 the main principle of the parallax error is depicted. Simply put, the computed gaze point is misaligned with the true gaze, depending on the distance to the object the user is looking at. If the viewed object is at the same distance as the plane used for calibration, the parallax error is zero. This problem is caused by the fact that the scene camera and the user's eye look at the same scene but from different viewing angles. The rays from eye and camera intersect at the distance

[13] https://optitrack.com/

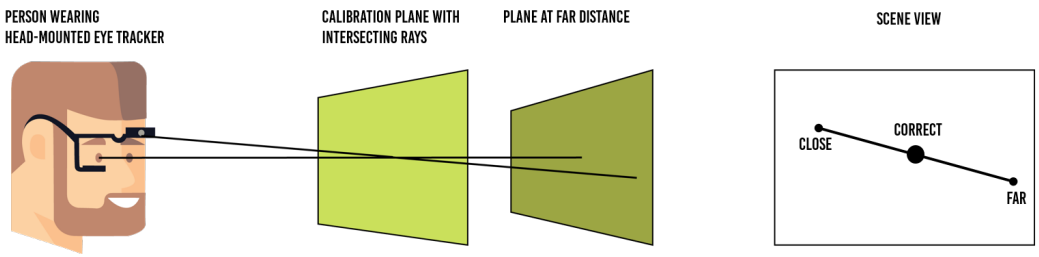


Figure 1.4: A head-mounted eye tracker is calibrated at a fixed position resulting in a fixed calibration plane. Looking at objects at a different distance, the actual and computed gaze point are different.

of the calibration plane and are misaligned from every other distance. To reduce the parallax error, a prominent approach is to use binocular head-mounted eye trackers.

Supervision. To circumvent all aforementioned issues and set up an eye tracker in a correct way, typically a second person is needed. Currently there exist different approaches to setting up the device and sampling the calibration data [142]. With an operator approach, a second person is doing all the work, whereas in a participant-controlled approach, the users themselves monitor the data sampling. A system-controlled approach does an automatic collection of calibration data, as the decision as to whether or not the user is currently fixating the calibration target is an automated process. It is known that a participant-controlled approach yields the best results [142]. Automatic decision by the system still achieves better accuracy and precision than an operator-controlled calibration. It is noteworthy that at least for remote eye tracking devices, the large manufacturers use an automatic calibration (e.g., Tobii 4C[14]). However, modern high-end head-mounted eye trackers still require the support of an additional person acting as operator, such as the Tobii Pro Glasses 2[15]. In fact, a person who is not familiar with the nature of head-mounted eye trackers will not be able to use the device from scratch.

To summarize, the need to calibrate a head-mounted eye tracking system constitutes a serious problem that prevents the application of current head-mounted eye trackers at fully pervasive settings [121], as defined our *Ubiquitous Computing for Eye Tracking* continuum (cf. Figure 1.1). Issues like drift and invariance can be counteracted by using huge, complex hardware setups including more sensors. Existing research addresses the aforementioned issues partially while introducing restrictions elsewhere.

[14]https://help.tobii.com/hc/en-us/articles/213414285-Speci cations-for-4C

[15]https://www.tobiipro.com/siteassets/tobii-pro/user-manuals/tobii-pro-glasses-2-user-manual.pdf/?v=1.20.2

So far, there exists no work that resolves all of them. Desirable attributes of an eye tracker include minimal intrusiveness and obstruction, allowing for free movement while maintaining high accuracy, easy and flexible setup, and low cost. A detailed description of eye tracker preferences is given by Scott and Findlay [168]. Although the latest commercial eye trackers try to include all these attributes, they remain regrettably expensive. Moreover, most of these devices rely on a direct connection to a powerful computer for real-time data processing. As a result, current head-mounted eye trackers are neither applicable for spontaneous nor suitable for long-lasting usage.

1.5 Scenario

In this section we are going to demonstrate the challenges of using head-mounted eye tracking devices that are directly related to efficient long-term, location, orientation & target independent usage and mobile & accurate calibration-free eye tracking, as previously defined in Section 1.4. This is done through a pictorial representation using the Collaborative Newspaper application, published in [100] and demonstrated in [101, 92].

To summarize, the Collaborative Newspaper application enables multiple users to read full news articles presented on a public display. As space is usually limited for several reasons, such as screen dimensions, layout and font size for better readability, the text has to be scrolled to be completely readable. Therefore the application integrates an adaptive scrolling algorithm that automatically moves the text according to the individual reading speed of a person. This is realized by analyzing people's gaze captured through a head-mounted eye tracking device. Finally, the system allows multiple persons to interact and read even the same text simultaneously within their individual reading speed. Figure 1.5 depicts the concept of the system. In our *Ubiquitous Computing for Eye Tracking Continuum*, the system can be categorized as an 1 : 1 : N relation.

Let us assume the system is deployed in an uncontrolled environment (e.g., in an urban setting). In the following we will show the different problem areas regarding the interaction with the Collaborative Newspaper application. In Figure 1.6 the initial state of the system is shown: The public display is presenting the latest news articles and a person is interested in reading a certain text. To estimate which text has to be scrolled, as well as the velocity of moving the text, the system relies on the information about the user's point of gaze on the screen. To accomplish the task of gaze

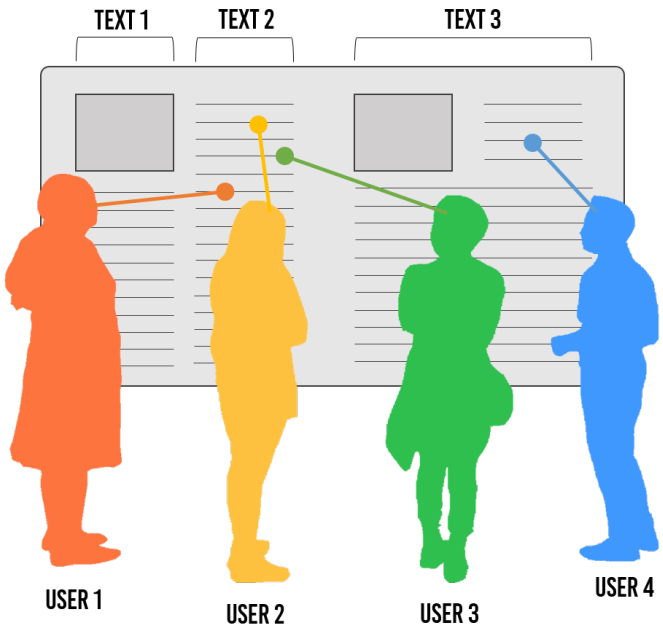


Figure 1.5: Concept of the Collaborative Newspaper: Multiple persons are reading several texts on a public display.

estimation, the user is wearing a head-mounted eye tracker. In contrast to remote systems, these devices can be used across multiple displays and allow for multi-user scenarios.

For smooth and correct operation, the Collaborative Newspaper application requires continuously highly accurate gaze data. A crucial step towards highly accurate gaze estimation is calibration, which is used to create a function that maps eye to gaze positions. The current appropriate calibration procedures usually require the user to fixate a sequence of visual stimuli. In this specific application, a calibration routine flashes the whole screen and moves a visual marker along the screen border (depicted in Figure 1.6). A poor calibration results in a bad pupil-to-gaze mapping. Moreover, an expert usually checks the correct positioning of the cameras, the validity of data sampling and finally the accuracy of the recorded calibration. Although we can take care of all these issues under laboratory conditions, this is not possible in the field.

However, even assuming we have set up a fully calibrated eye tracker, so-called calibration drift will affect the gaze estimation accuracy [111]. This means that, due to changes in the eye physiology (e.g., wetness of the eye), changes in the environment (e.g., sunlight), or the motion of the cameras, the accuracy of the gaze estimation

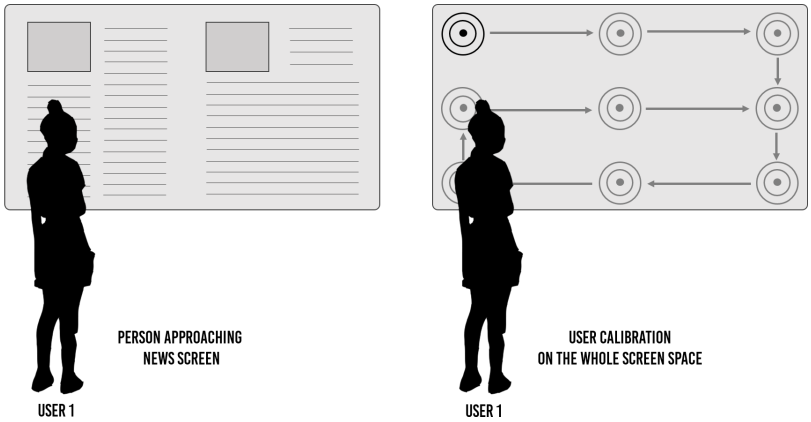


Figure 1.6: A person is interested in reading a text on the public display. After approaching the screen, she first has to calibrate the head-mounted eye tracking device in order to use the Collaborative Newspaper.

will drop, causing an incorrect state of the system. This can result in scrolling at a wrong velocity, moving the wrong text or even no functionality at all. The only option is to re-calibrate the eye tracking device. In a straightforward approach, one would simply trigger the standard calibration procedure again. For this, the screen would be flashed again, showing the calibration routine, which would interrupt the user(s) currently reading. As can be seen, starting to use the application is hard without the help of an experienced person. It might be impossible to use it without any support in an urban setting (e.g., a subway station).

In Figure 1.7 a person is currently using the Collaborative Newspaper application and reading a text shown on the public display. While interacting with the public screen, the person is changing her position and/or location in front of the display. This might happen on a frequent basis, especially as the Collaborative Newspaper is developed for large displays. In particular, two types of motion can occur: (1) user movements in front of the display to change the user's distance/orientation, making the text easier to read; and (2) head movements to view news articles outside the ocular motor range. Usually both movement types will reduce the gaze estimation accuracy that is required for the Collaborative Newspaper to work properly. The reason for this lies in the fact that the calibration is typically recorded for a fixed distance and orientation to the display. There are two valid options to counteract this issue. First, the eye tracking device could be re-calibrated for the new position and orientation of the user to the screen. As previously mentioned, this is not a valid option, as the display will show the calibration procedure (as shown in Figure 1.7),

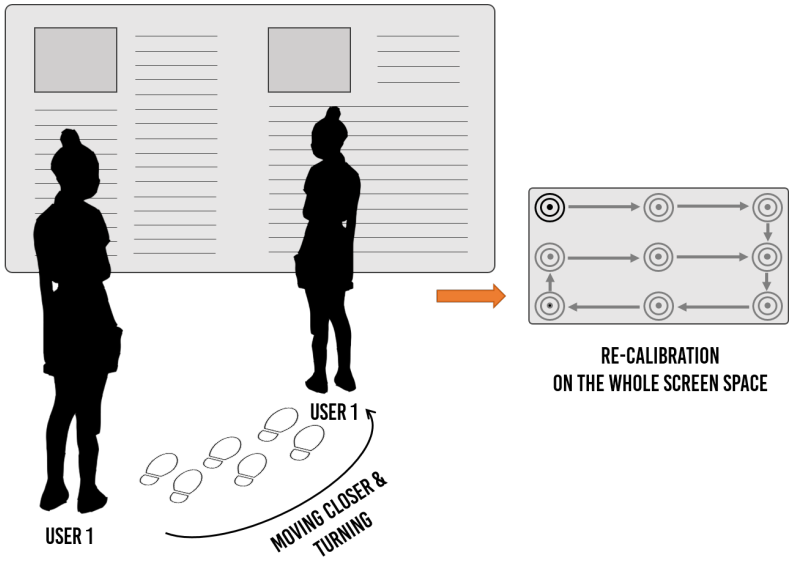


Figure 1.7: The user is moving in front of the display, which requires a re-calibration in general.

which will be distracting during reading.
The second possibility is to track the user's position and orientation in space relative to the display. This is done with the help of visual on-screen markers. This approach is more lightweight than other approaches, such as external tracking systems, but its drawback is that all displays have to be equipped with the markers, and they need to be pre-registered. This can be done by attaching printed ones to a display's frame [209, 17], or directly showing digital ones on the display. Printed markers quickly clutter the environment, in particular in multi-display scenarios. Digital markers could be shown only on demand, but they occupy display space and reduce the content that can be shown.

Taking a look at the example in Figure 1.8, the Collaborative Newspaper is shown on a display with a resolution of 1920 x 1080 pixels. To reliably track and identify the display from arbitrary positions – even if it is not fully visible in the scene camera – we show 14 markers along the screen border. Each one has a resolution of 100 x 100 pixels. In total, there is an available screen space of 2,073,600 pixels. All markers together reserve a space of 14 x 100 x 100 = 140,000 pixels, resulting in 6.75% of the whole display space. To allow better recognition by the software, we are not showing any content between the markers. In fact, we reserve 2 x 1920 x 100 = 384,000 pixels (upper and lower area) plus 2 x 1080 x 100 = 216,000 pixels (left and right area), summing up to 29.93% (one third!) of the available display space. In Chapter 5, we

Figure 1.8: User interface of the Collaborative Newspaper, displaying up to 7 articles. For display tracking, visual AR markers are placed around the content.

will present an alternative technology that does the tracking of the spatial relationship without any markers and is able to achieve even higher gaze estimation accuracy.

Figure 1.9 shows the transition from a single-user to a multi-user setting of the Collaborative Newspaper application. One person (user 1) is already reading a text while standing in front of the screen. The application integrates the functionality that multiple persons are able to read the displayed texts in parallel while being supported by an adaptive scrolling algorithm (explained in [101]). In this case, a second person (user 2) is moving up to the display and wants to read another text or the same text as user 1 (remember that this is possible). Basically the second user is in the same situation as user 1 in Figure 1.6. He or she has to calibrate the head-mounted eye tracking device prior to usage.

Consequently the display will show the calibration routine, which overlays the actual content. Hence, user 1, who is currently reading a news article, is interrupted for about 45-50 seconds. Note that this issue occurs every time another person wants to join the interaction. This is a serious problem that prevents spontaneous usage of the Collaborative Newspaper.

The highlighted problems that arise with a gaze-enabled application were also summarized in [91]. The above example scenario can be expanded to match other complexities of the *Ubiquitous Computing for Eye Tracking Continuum*. Then, again, the challenges to realize eye tracking are becoming increasingly difficult. To lay the foundation for exploiting the full capabilities of head-mounted eye tracking devices and

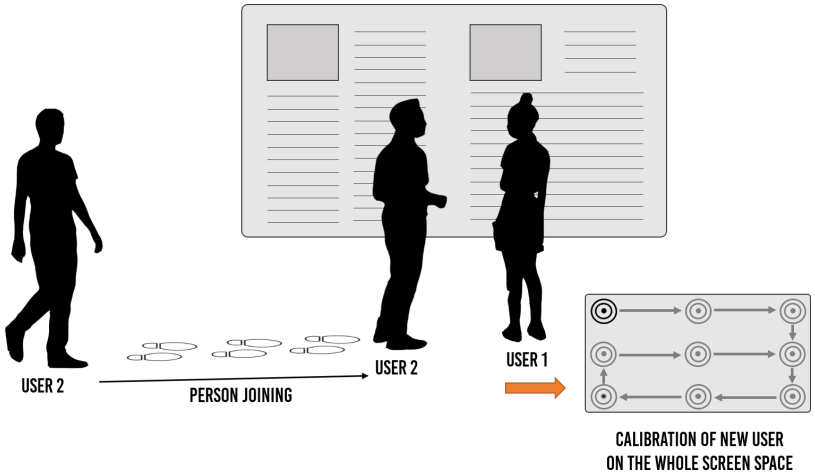


Figure 1.9: A second person is joining the interaction and wants to read a text presented on the public display. For this, a calibration has to be done beforehand.

make them mobile, this thesis addresses the challenges, highlighted above, as follows in the next section.

1.6 Research Questions

In this thesis we address the following fundamental research question:

> *How can we complete the transition of head-mounted eye tracking from a limited, static interaction tool into a ubiquitous computing device?*

We will investigate and finally answer this question by specifically addressing the main challenge of calibration and the arising issues of drift, invariance, parallax error and supervision, from the perspective of human computer interaction. In particular, to find convincing answers to this question, we can break it down into smaller and more straightforward pieces, which this work sets out to answer:

1. What is the long-term accuracy of current head-mounted eye trackers?
2. How can we efficiently re-calibrate head-mounted eye trackers and recover the initial gaze estimation accuracy?
3. How can we overcome the problem that the calibration is orientation, location and target dependent?

4. How can we achieve highly accurate and seamless gaze estimation across multiple surfaces?

5. How can we realize accurate gaze estimation without a user-dependent calibration?

6. What is the benefit of a mobile and wearable eye tracker?

7. How can we design a mobile eye tracking system with a constant high gaze estimation accuracy, usable at pervasive scale?

The goal of this thesis is to contribute methods that help to answer the above questions, and lead to a system which paves the way towards pervasive gaze-based settings and can be used at any level of complexity of the *Ubiquitous Computing for Eye Tracking Continuum* (cf. Figure 1.1).

1.7 Thesis Outline

The work presented in this thesis aims at contributing in two core aspects: Firstly, we will gain insights into the specific issues which are entailed by the need for the calibration process when using a head-mounted eye tracker. Based on that, we will elaborate the *theoretical* principles for solving the different problems step by step. Results of these parts, for instance, are novel concepts and notions/ideas.

Secondly, this thesis contributes with a *practical* part, describing the technical implementation of the corresponding theoretical approach. The results of these parts are, for instance, novel prototypes or applications. Their feasibility, performance and functionality in practice, and their validity as a conceptual solution, are underpinned by an empirical evaluation through laboratory user studies.

To summarize, the two fields of contribution – *theory and practice* – are covered by the whole thesis, and specifically focused on, in three successive and interrelated chapters, to address the following areas and thus the formulated research questions stated above:

- *Efficient Long-Term Usage* of a head-mounted eye tracker (E)
  (Chapter 4)

- *Location, Orientation & Target Independent* head-mounted eye tracking (L)
  (Chapter 5)

- *Mobile & Accurate Calibration-Free* gaze estimation (M)
  (Chapter 6)

In each chapter, we start with an initial section contributing to the theoretical aspects of the discussed problems. The latter part of each chapter concerns the technical development and implementation of the presented idea, transferring the corresponding theories. Figure 1.10 depicts a schematic representation of the structure of this thesis. It shows the relation between the core chapters and their dual contribution. This work follows a step-by-step structure. This reflects the gradual convergence towards a calibration-free and mobile eye tracking system via the above research questions.

Following on the first chapter, the second chapter describes the fundamentals, which are the theoretic and technical basics this thesis is built upon. Chapter 3 frames the work of this document by broadly presenting the related work of this research field. The derived practical problems of Chapter 1 Section 1.4 and will be addressed by all subsequent chapters. The scenario application (cf. Section 1.5) is used to highlight the related progress made in Chapters 4, 5 and 6. Lastly, Chapter 7 closes this thesis by summarizing the contributions of this work, discussing possible ideas for future work and drawing a final conclusion.

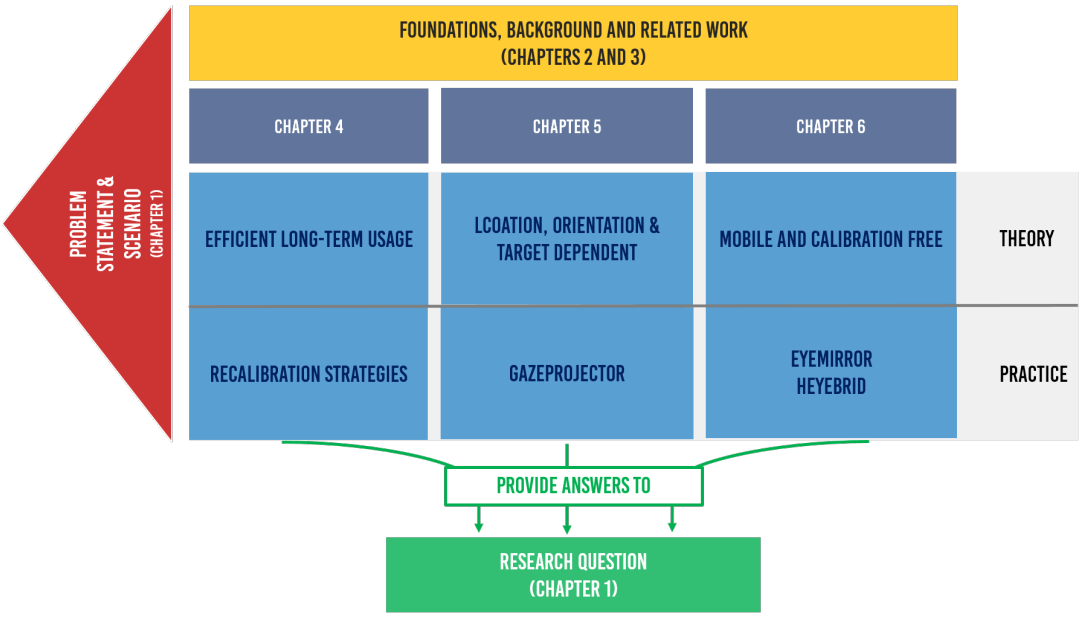


Figure 1.10: Thesis structure: Four main chapters addressing the stated problems and providing answers to the research question.

This thesis was done at the Saarland Informatics Campus of Saarland University and the German Research Center for Artificial Intelligence (DFKI GmbH) and relates to the field of Human Computer Interaction (HCI). Almost all parts of the work that is presented within this thesis was done in conjunction with researchers and students from other institutions.

Chapter 2

Foundations & Background

This chapter provides the foundations and background for this thesis. We start with an introduction into the human visual system by describing the eye, with its anatomy and functionalities. The second part of this chapter presents the existing methods and techniques to acquire people's eye and gaze positions.

2.1 The Human Eye

We start with an overview about the human eye, i.e. the part of the human body we want to equip with technology. To digitize information about someone's eyes and vision, it is essential to understand the structure, interaction and roles of the different parts of the eye.

2.1.1 The Physiological Eye Model

The human eye is the organ of vision. Both eyes are located in the front of the head in such a way, that we have the ability to see three dimensionally and judge distances. Figure 2.1 shows the visible parts of the human eye and illustrates a detailed cross section of the eye ball. Several protective mechanisms are built in to protect the sensitive structure of the eyeball against external influences. To shelter the larger part of each eyeball, the *posterior segment*, the bones of the skull form an orbital cavity. The frontal part of the eyeball, the *anterior segment*, is covered by the *conjunctiva*, which line the upper and lower eyelids. Tears constantly keep the eye wet and clean it up to remove foreign objects. The lids and eyelashes protect the front of the eye against immediate danger.

The eye is complex in its structure. In the following we will explain the main parts

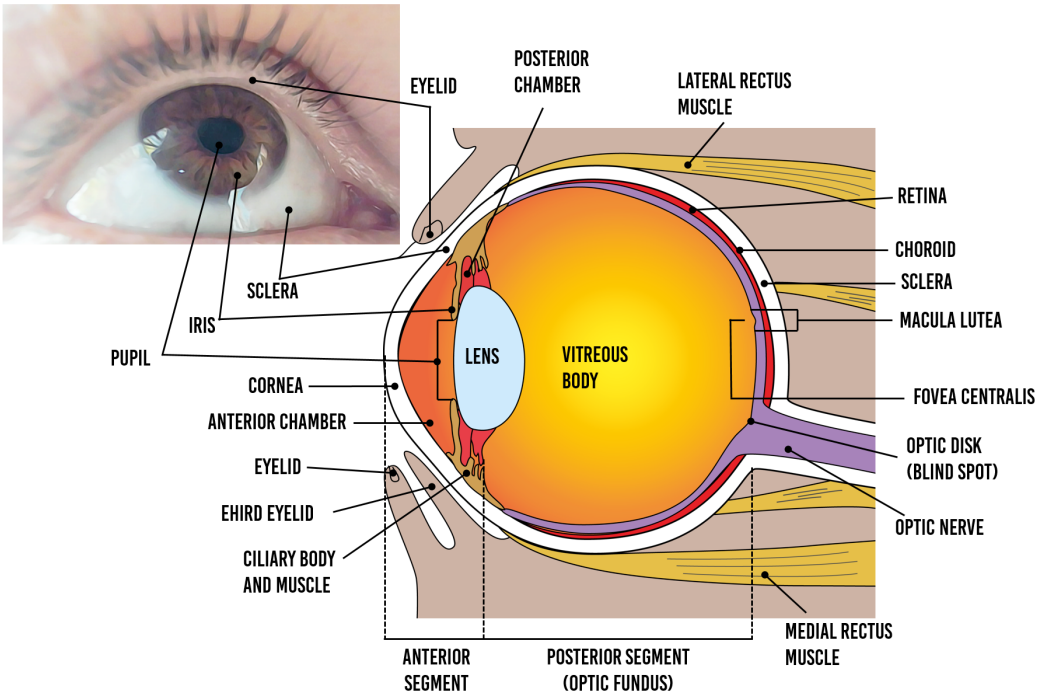


Figure 2.1: Frontal photos of the eye together with a cross section (adapted from [118]). All important parts are labeled.

of it (adapted from [32]), which are all presented in Figure 2.1. Whenever necessary, a more detailed figure will be used for a better explanation of the specific part.

Cornea. The convex, transparent anterior part of the eye allows light to pass through to the lens and provides up to 75% of the eye's focusing power. The cornea is a fibrous structure made up of five layers (corneal epithelium, Bowman's layer, corneal stroma, Descemet's membrane and corneal endothelium) and covers the iris and the pupil. Its primary characteristics are that it is uniform in thickness and nonvascular. The degree of curvature of the structure is different between people and also varies in the same person at different ages. The cornea is continuous with the sclera. The border or edge between the cornea and sclera is called the limbus.

Sclera. The tough, white, opaque and fibrous outer shell of the eyeball covers most of the eye's surface. The front part is visible as the ' white' of the eye. The sclera is made up of three layers (the episclera, the scleral stroma and the suprachoroid) and constantly decreases in thickness from the back to the front of the eye. The external muscles of the eye are directly connected to the sclera.

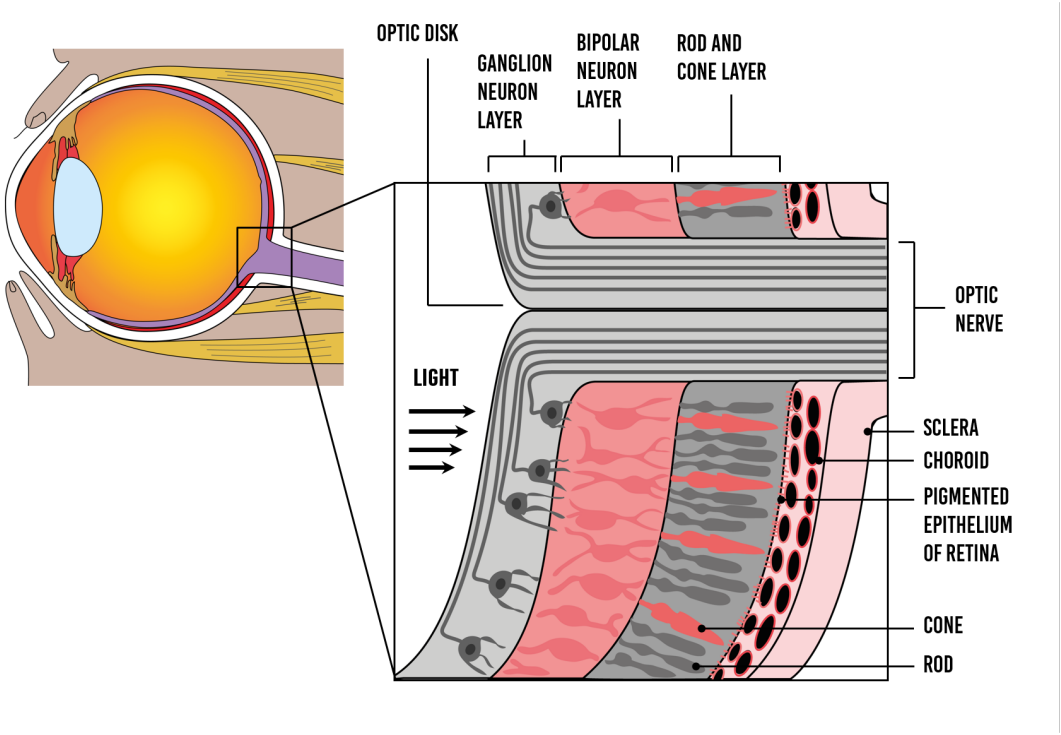


Figure 2.2: Detailed cross section of the eye's retina, showing its structure (adapted from [118]).

Choroid. The vascular layer of the eye is located between the sclera and the retina. The structure consists of four layers (Haller's layer, Sattler's layer, Choriocapillaris, Bruch's membrane). The choroid contains a massive amount of brown pigments that are necessary to reduce the reflection and diffusion of light when it falls on the retina.

Retina. The innermost part of the three coats surrounds the vitreous body and merges into the optic nerve, as shown in Figure 2.2. Since the human eye has to function under several different circumstances, the retina is composed of light sensitive nerve cells, which are arranged in three different layers (the retina is composed of ten layers in total): The two neuron layers (Ganglion and Bipolar) transmit the impulses to the optic nerve. The most prominent nerve cells are the cones and the rods, which are different in shape to cover the full range of adaptation to light. The cones are sensitive to bright light and responsible for color vision, and the rods work in dim light. There are three types of cones, sensitive to the red, green, and violet parts of the visible spectrum. White light stimulates all three types of color cells; any other color arouses one or two. The optic nerve transmits the impulses from the retina to the visual center of the human brain. The part where it leaves the retina is called the optic disk or blind spot, and does not contain any light-sensitive cells. The fovea is

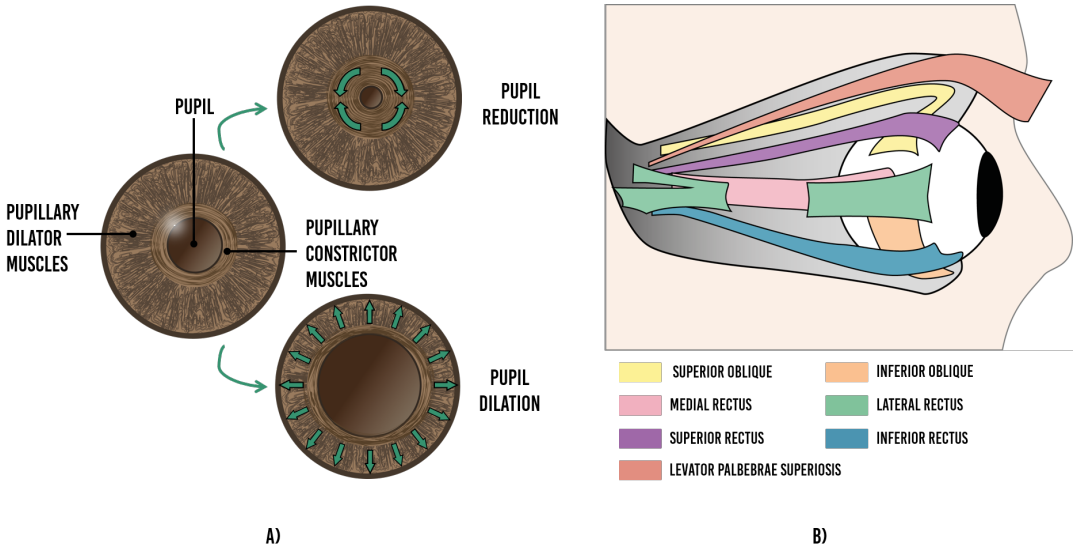


Figure 2.3: a) Frontal illustrations fo the pupil and the states of dilation and reduction issued by the pupillary muscles[1]. b) Structure of the eye muscles necessary for all types of motion[2].

the area of clearest vision in the center of the macula lutea and constitutes about 5° of vision. There the layers of the retina are shifted away; hence the light can directly fall on the cones. The cornea, sclera, the choroid and the retina constitute the three coats of the eyeball.

Pupil. The circular window of the eye to the world is located in the center of the iris. Light enters the eye and falls onto the retina through the pupil. The pupils of both eyes are usually equal. They become smaller (constrict) when exposed to bright light or if the focus is on a near object. The pupil is dilated in the dark or if the focus lies on a distant object. The two types of adjustments are named the pupillary light reflex and the accommodation reflex. Figure 2.3a illustrates the pupil and the muscles needed for dilation and constriction.

Extraocular Muscles. To keep the focus on a specific object, it is necessary that we are able to reposition our eyes. The human eye is equipped with six muscles (as shown in Figure 2.3b), so that it can be moved within six degrees of freedom (three translations and three rotations):

- superior rectus and inferior rectus – up and down movement

[1] https://img.tfd.com/medical/Davis/Tabers/p49.jpg
[2] http://teachmeanatomy.info/head/organs/eye/extraocular-muscles/

- lateral rectus and medial rectus – horizontal movements
- superior oblique and inferior oblique – rotation/twist
- levator palpebrae superioris – an extra muscle to control the eyelid

Iris. The circular, colored membrane is located between the cornea and the lens of the eye and encloses the pupil. It divides the space between the lens and the cornea into anterior and posterior chambers (see Figure 2.1). The iris is made up of muscle fibers that are needed to regulate the amount of light passing through the pupil.

Lens. The transparent, biconvex body separates the anterior and posterior segment (illustrated in Figure 2.1) and refracts the incoming light, so that it can be focused on the retina. The main functionality of the lens (often also called the crystalline lens) is accommodation. That is, it has to adjust its curvature (i.e., the refraction) in order to sharpen near and far objects.

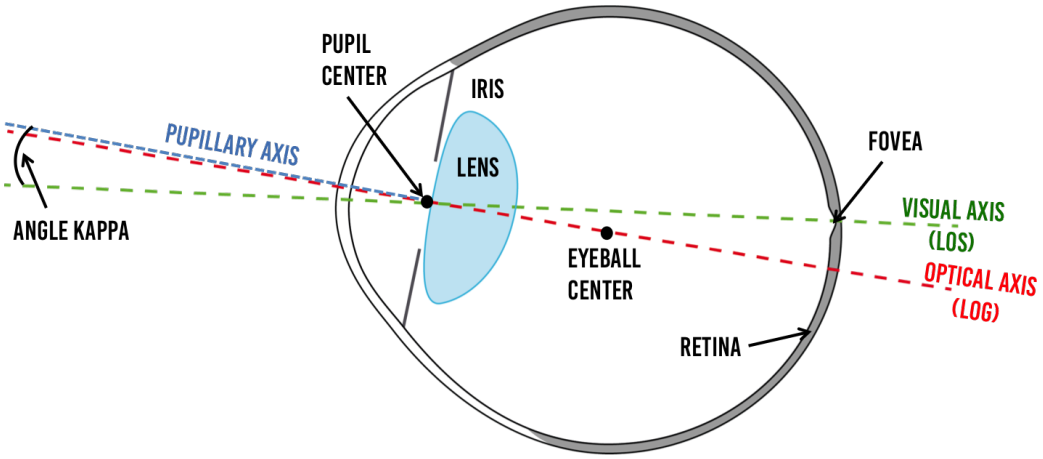


Figure 2.4: Cross section of the human eyeball including the three main axes [98].

Visual Axes. Most important to understand human vision is the definition of the axes of the eye. In a simplified model, there are two main axes, as depicted in Figure 2.4. The optical axis is defined by the line connecting the center of the curvatures of the eye, i.e. the center of the eyeball, the pupil and the lens. It is often referred to as the line of gaze (LoG) and draws sharpest focus when we look at an object. The visual axis is defined as the line passing through the fovea and the center of the pupil. It is often referred to as the line of sight (LoS). The pupillary axis is the normal line to the corneal surface passing through the center of the pupil, which is slightly different

from the optical axis. The angle formed between the line of sight and the pupillary axis is the so-called angle kappa. A definition of all axes and their respective angles can be found in [122].

2.1.2 Eye Movements

We move our eyes to reposition them in order to keep our view focused on a specific object. As already described in Section 2.1.1, each eye is connected to six muscles, of which three pairs are responsible for a specific direction (yaw, pitch and roll) to enable the 3-dimensional rotation of the eye inside the head. The movements that our eyes perform can be categorized into voluntary, involuntary and reflexive ones. In the following we will describe the five basic types [37] (Chapter 4), of which four are depicted in Figure 2.5.

Fixations. The most prominent event is not in fact a movement. Fixation describes the state of resting the gaze at a specific object within the central vision for a certain amount of time, typically around 200-300 milliseconds (ms) (depicted in Figure 2.5a). However, the term is somewhat misleading, as the eye is not standing still but rather doing micro-movements: tremor (frequency around 90 Hz; unclear role), drift (200-1000 ms, moves the eye away from the fixation) and microsaccades (10-30 ms, brings the eye back to the fixation center). Such eye movements are helpful to understand the human neurology [59].

Saccades. Between two consecutive fixations, the eye makes fast, jump-like movements in order to re-focus it. Saccades can be made voluntarily or reflexively and are the fastest movement the body can produce, ranging from 10-100 ms. It is known that we are blind during their execution. They are ballistic movements and do not necessarily follow the shortest path between the start and end point.

Smooth Pursuits. Visually tracking an object that moves with a constant speed (e.g., looking at a passing car as shown in Figure 2.5b) involves pursuit eye movements. Up to a certain threshold, the eyes are able to match the velocity of the followed object, typically 10.30 degrees/second.

Vergence. To bring objects at different depths into focus, the eyes also have to move in relation to each other. Vergence eye movements describe, when the eyes are converging or diverging to prevent double vision (i.e., diplopia), as illustrated in

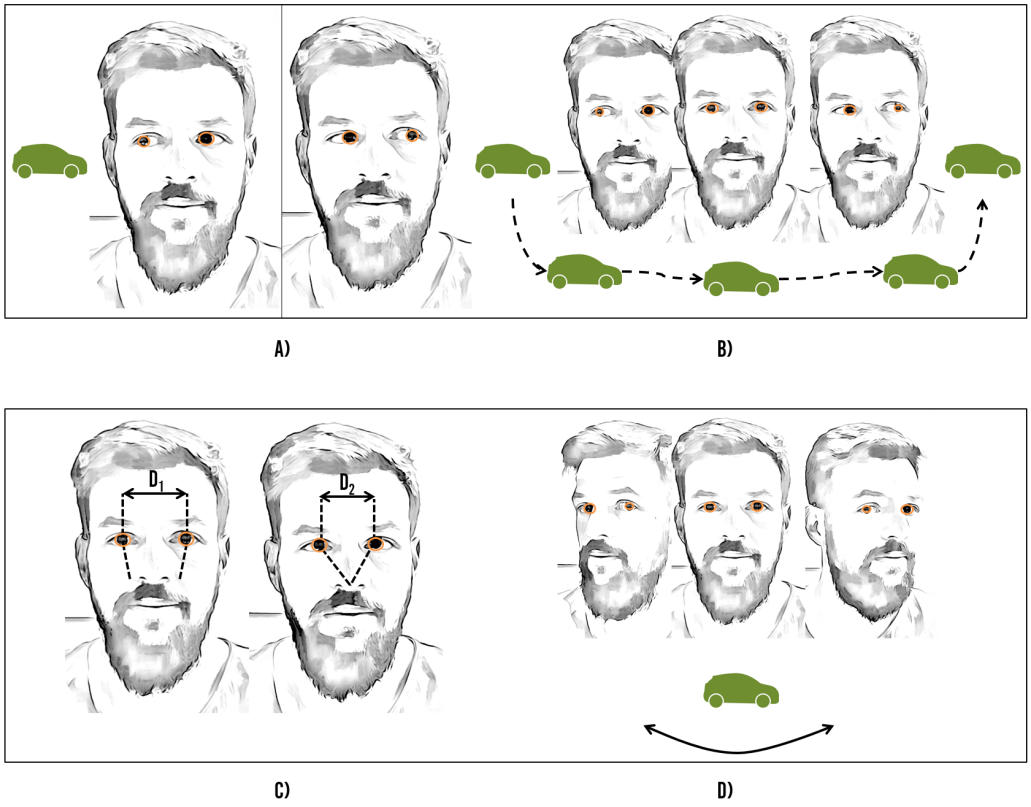


Figure 2.5: Illustration of all major types of eye movements: a) fixation to focus on a object, b) smooth pursuit to constantly follow a moving object, c) vergence to perceive different depths and d) vestibulo-ocular reflex to reposition the eyes when the head is moved.

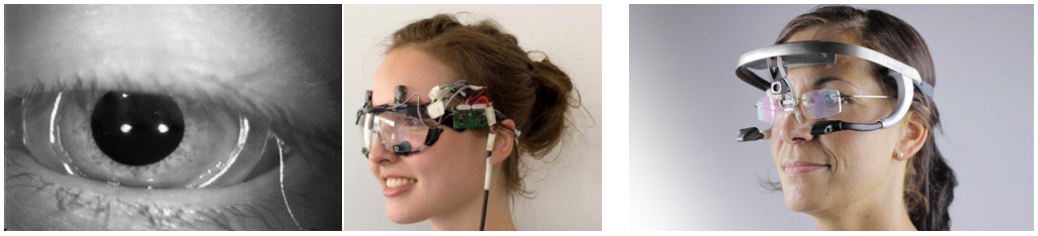
Figure 2.5c. Many people have a dominant and a non-dominant eye. That is, the direction of both eyes may be slightly different.

Vestibulo-ocular reflex (VOR). When we move our head while we are focusing on a specific object, the position of our eyes is compensated for and readjusted (see Figure 2.5d) to keep looking at that object. Hence, a clear image is preserved, which would be noisy while we are moving.

The above-described eye movements are enabled by using the six main eye muscles. But, as described in the previous section, there is another important muscle, required to control the eyelids. The temporary closure of (usually both) eyelids – blinks – can also be counted as a specific case of eye movement. Blinks can be voluntary or involuntary (e.g., corneal reflex). They are necessary to spread the tears over the eye in order to keep it damp.

2.2 Acquiring Eye & Gaze Position

Nowadays, three main techniques exist to track a person's eye and do gaze estimation. The electro-oculography (EOG) based method places electrodes around the eye to measure the skin potentials, as proposed by Kaufman et al. [79]. The scleral search coil method uses a small coil that is embedded into a contact lens. It measures the voltage caused by an external electro-magnetic field [161]. Camera-based (video-based) techniques use the images of the eye to detect its characteristics in combination with images of the field of view to realize gaze estimation. This method is mainly applied nowadays by either using a head-mounted (cameras are mounted on the head) or a remote (cameras are placed in the environment) system. In the following we will restrict ourselves to video based eye/gaze tracking, as it is the main focus of this thesis. Figure 2.6 depicts example devices for each category.



(a) (b) (c)

Figure 2.6: Examples of different types of head-mounted eye tracking approaches: (a) scleral search coil method [66], (b) electro-oculography[3], (c) head-mounted Dikablis eye tracker[4].

We have to distinguish between tracking the eyes' movement and position and estimating where a person is actually looking. In the following we will provide an overview about the common eye tracking and gaze estimation techniques.

2.2.1 Eye Tracking

Detecting and tracking the eye is an essential step towards gaze estimation. Eye detection and tracking is still a challenging task, as there are many unforeseen issues that have to be taken into account, like occlusion of the eye by the eyelids, degree of openness of the eye, variability in size, reflectivity or head pose, etc. The eye can be characterized through different features including the intensity distribution of the pupil, iris, and cornea, as well as by their shapes. Ethnicity, viewing angle, head pose, color, texture, light conditions, the position of the iris within the eye socket, and the

[3] http://www.sciencedaily.com/releases/2008/04/080428083418.htm
[4] http://www.ergoneers.com/en/eye-tracking-head-mounted-1-en/

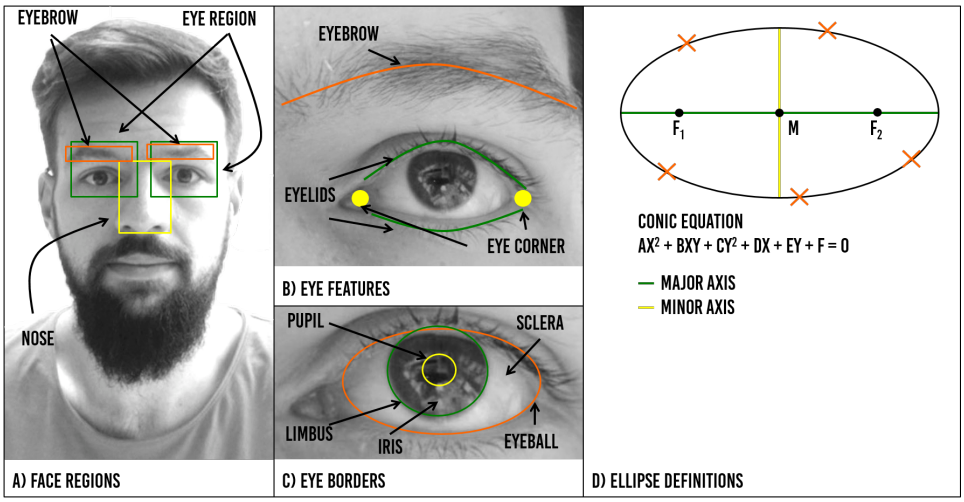


Figure 2.7: a) Face with areas of interest; b) shapes around the the eye; c) prominent features; d) ellipse with its definition

state of the eye (i.e., open/closed) are issues that heavily influence the appearance of the eye. According to Hansen and Ji [52], there exists aa extensive taxonomy of eye detection techniques. We will provide a summary of them in the following.

Shape-Based. The visible parts of the human eye, such as the iris, pupil and eye contours, can be described well by their shape (shown in Figure 2.7b). The shape-based approaches are based on a pre-defined geometric model of the eye's shape that is used to match against. There are many existing methods differing in the properties of the underlying model. That is, each geometric model is defined by a set of parameters that are used to control the level of deformation and transformation. An important aspect of these models is the invariance with regard to scale, rotation and shape. Basically, shape-based approaches can be grouped into two classes. Simple elliptical shape models use the fact that structures like the iris or pupil look like an ellipse depending on the viewing angle. Such models use at least five parameters to describe the ellipse. Existing methods use image thresholding in combination with edge detection [147], the Hough transform [206], image gradients [89] and RANSAC for ellipse fitting of the extracted shape. In Figure 2.7d the parametric form of an ellipse is shown. It is based on the conic equation with which a circle, parabola, hyperbola or ellipse can be represented. An ellipse has five degrees of freedom, i.e. the x and y coordinates of each focus point (F) and the sum of the distance from each focus point to a point on the ellipse. Alternatively, an ellipse can be defined by x and y coordinates of the center (M), the length of each radius and the rotation of the axes

around the center. Complex shape models constitute the second class of shape-based detection techniques. These approaches offer more options for a better approximation of the eye's shape. Usually they take into account the overall eye structure, including eyelids and corners. For example, Yullie et al. [210] propose a model with eleven parameters with which the iris and the eyelids are modeled. In short, existing models differ in the number of parameters and thus are able to handle many or few eye shapes. The number of parameters also has an influence on the computational complexity of the models.

Feature-Based. A human eye can be described by a specific amount of striking features. This set is usually characterized through local features, such as the limbus (border between iris and sclera), pupil and eye corners, as shown in Figure 2.7c. The region around the eye also reveals properties, for example the eyelids and brows, and the nose, used to describe a region of interest, depicted in Figure 2.7. Different computer vision methods [58] or specifically trained neural networks [159] can be used to detect and track boundaries within the eye, such as edges and lines. Together with an appropriate model, the eye is located. To enhance the visibility of local features, particular filters can be applied, such as linear and nonlinear filtering [173] or convolution [35]. When the eye regions are successfully extracted or the eye is captured closely enough, the pupil and/or iris are used to detect and track the eye's movements. Thereby we distinguish between pupil tracking in visible light images and infrared light images. Depending on the image type, different techniques are used. We will provide more details on pupil tracking using IR images in the last part of this subsection. All in all, depending on the type of features, these methods are robust against illumination changes. However, approaches based on edges or lines may not work when different light settings are present.

Appearance-Based. Also called the holistic approach, this is a method that detects the eyes directly based on their photometric appearance, such as color and filter response. Appearance-based methods are comparable with image template matching. For this an image patch model is created and eye detection is done using a similarity measure [213]. Holistic methods are based on the statistical analysis of the intensity distribution across the entire image. Usually a large set of training images is needed to create a classifier or model that is able to detect different eye presentations. Approaches are differentiated, based on whether they are implemented in the spatial or in a transformed domain. When doing eye detection in the transformed domain (e.g.,

frequency domain), one can tolerate slight illumination changes (e.g., removing bands sensitive to illumination).

Hybrid Models. Each of the approaches above has specific advantages and disadvantages. Hybrid methods simply combine the approaches from different categories to exploit their respective benefits.

Eye Detection on IR images

The most prominent eye detection method nowadays is to use active infrared (IR) illumination [52]. This is primarily done to enhance the contrast between pupil and iris. It is a special technique that cannot be mapped exclusively to one of the groups from above. We distinguish between methods relying on visible light, called *passive* (as they use the existing light of the environment), and invisible light, called *active* approaches (as they use additional light sources). Most active methods use near IR light sources (780-880 nm), invisible for the human eye, thus not disturbing the user's field of view. Depending on the location of the light source with respect to the camera, the pupil appears to be bright (close to the optical axis of the camera) [131] or dark (away from the camera's optical axis). Figure 2.8a and b shows the respective examples. The most popular algorithm is Starburst [104], introduced in 2015. It is a video-based eye detection method, using the pupil contours as feature points, which are mapped onto an ellipse shape using RANSAC model fitting. Recently Fuhl et al. [45] compared the latest pupil detection algorithms ElSe [44], ExCuSe [43], Pupil Labs [78], SET [74], Swirski [185] and Starburst using different datasets of active

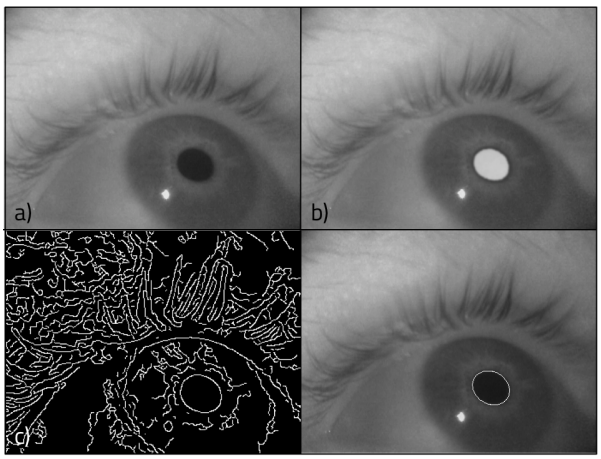


Figure 2.8: a) dark pupil; b) bright pupil; c) Else output based on edge detection and curvature computation.

illuminated dark-pupil images. All these methods combine aspects from different categories, such as eye feature together with ellipse models, while they only work on IR eye images. They found that all algorithms give reasonable results. However, when it comes to real-world images exhibiting problems like changing illumination, occlusions, or reflections [167], the ElSe algorithm (example shown in Figure 2.8c) outperforms existing approaches with respect to detection rate.

In summary, each eye detection/tracking technique is dependent on specific information (e.g., pupil or iris model), requires certain light conditions (e.g., IR or natural) and image properties (e.g, high contrast and/or resolution) and may be robust to special changes (e.g., head pose, occlusion). However, the recent development of more robust and accurate pupil detection algorithms is a major step towards the vision of ubiquitous eye tracking.

2.2.2 Gaze Estimation

Gaze estimation is the primary task of gaze trackers, which are often just referred to as eye trackers. Gaze can be defined as either the gaze direction or the point of regard (PoR). A person's gaze is usually determined by the head pose (position and orientation) together with the eyeball rotation. The gaze changes if at least one of the two values varies. It is common that a person first brings their head into a comfortable position before orientating the eyes. To achieve very accurate gaze estimation, head movements have to be considered as an additional source. This is done by tracking them via extra hardware (e.g., depth sensors, as done with remote trackers, or inertial measurement units on head-mounted devices), or integrating them directly into the method used.

Different methods exist to realize gaze estimation, which is always about finding a suitable mapping from pupil to gaze positions. Depending on the hardware approach, pupil positions are mapped into the scene video of a head-mounted device, or directly on a surface (e.g., screen) if using a remote approach. The gaze estimation method by itself is independent of the hardware approach. Feature-based methods are preferred, as they deliver the most accurate results [52]. These methods are divided into two subclasses.

Geometric-/Model-Based. The human eye can be modeled as a sphere, so that it is possible to calculate the gaze as a 3D direction vector. The center of the cornea

is estimated, and thus the optical and visual axes can be computed. Intersecting one of the lines with the object being viewed gives the gaze point. Usually the line of sight (LoS, the visual axis, see Figure 2.4) is defined as the true gaze direction. Such methods rely on prior knowledge of personal eye parameters, such as size and radii of the eyeball and the cornea, also known as fixed parameters. The 3D eye position, center of the eyeball and pupil as well as the axes, which are specific to different people, are known as extrinsic parameters. Recently, Swirski et al. [186] proposed a new ellipse fitting model that automatically constructs the 3D eye model. They were able to achieve an average gaze error of 1.68°. However, they used perfect simulated noiseless data to provide ground-truth results. In general, model-based approaches may require specific hardware and 3D knowledge about the scene that is not always available, especially in mobile settings.

Regression-/Interpolation-Based. These methods typically compute a mapping from pupil to gaze positions (2D or 3D). Thereby the basic theory is that the mapping can be described in a specific parametric manner. Various approaches exist to assess the mapping by polynomial functions [22], neural networks [75] or homography transformations [209]. Regardless of the mapping, the approaches in this category use infrared eye images, using two pertinent features, the pupil and the corneal reflection. The latter is the reflection of the infrared light source on the cornea of the human eye (depicted in Figure 2.9). Due to the layered structure of the eye, as we described it in Section 2.1, there are four different reflections, also known as the four Purkinje images [80], P1 to P4, as shown in Figure 2.9. There are two main properties thathave made Purkinje images to the indispensable basis for gaze estimation [37]:

1. The Purkinje image, i.e. the position of the corneal reflection of the IR light, is almost constant compared to the pupil position when moving/rotating the eye.

2. The difference between the corneal reflection (also known as glint) and the pupil center is constant when moving the head, but changes when moving/rotating the eye.

According to these characteristics, a mapping from the pupil to first Purkinje image, the so-called glint vector, to a planar surface is estimated. For this a user-dependent calibration process is necessary to evaluate the parameters needed for the mapping, as described in Section 1.4. It requires the user to look at a number of predefined visual stimuli on a surface (e.g., a screen). During this process the relation between the glint vector and gaze position, i.e. the position of the stimulus, is sampled. This data is

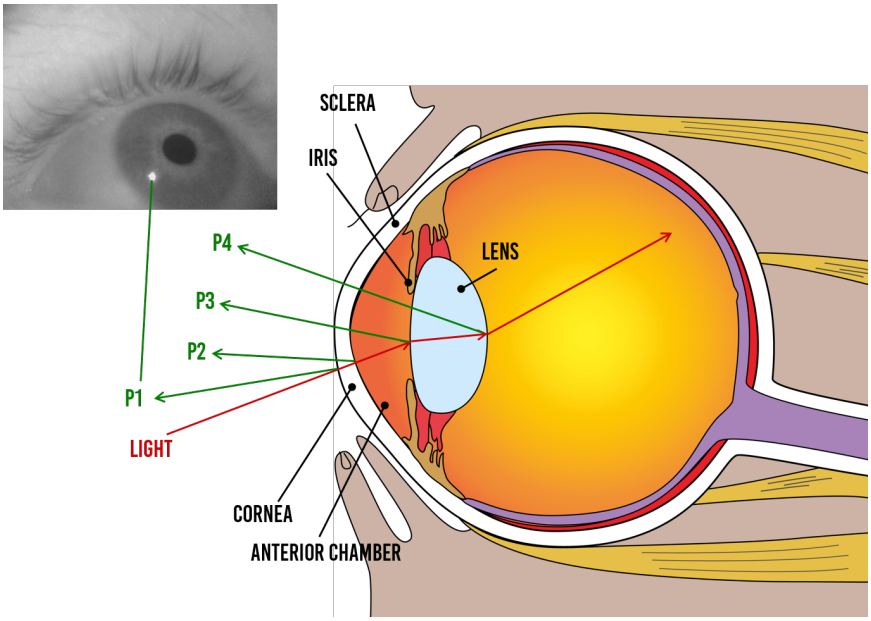


Figure 2.9: Infrared light reflected on the human eye results in Purkinje reflections: P1 on the corner, P2 from the back of the cornea, P3 from the lens, P4 from the back of the lens.

used to compute the parametric mapping. Further, the calibration inherently includes information about the relationship between the cameras as well as personal parameters like the angle kappa. Regression-based gaze estimation is able to achieve highly accurate results (up to 0.5°, as reported in [121]). The described method is also known as *Pupil Center Corneal Reflection (PCCR)*. Both of the described feature-based gaze estimation approaches are the most commonly used ones nowadays in commercially available systems, since they deliver the most accurate and stable results.

Appearance-based systems are an alternative method based on appearance-based eye tracking, as described above. These approaches typically rely on large amounts of user-specific training data. For instance, a neural network can be used to locate the gaze in a normalized space. However, a person's gaze can be only roughly estimated (within around 6.1°, as reported in [213]).

2.3 Summary

In this chapter we presented an overview about the human eye, including its anatomic structure, properties and constraints. We highlighted the different movements that our eyes are able to perform, used by different research areas to assess a person's visual behavior. Using various kinds of technologies, we are able to digitize eye movements

and thus our gaze. Depending on the approach, there are more or fewer requirements and constraints that have to be fulfilled and considered. Eye tracking data can support the creation of different applications in the field of human computer interaction. To be fully applicable in pervasive settings (see *Ubiquitous Computing for Eye Tracking Continuum* in Section 1.1), there are several challenges that are discussed in the next Chapter.

Chapter 3

Related Work

This chapter provides a comprehension of the existing research and state of the art this thesis is built upon. In Section 1.6 we formulated the research question addressed by this work. According to the separation into more streamlined sub-questions, the goal of this thesis is to investigate approaches and technologies to make head-mounted eye tracking usable in a spontaneous manner and in ubiquitous scenarios (anywhere and anytime). Therefore related work in the following categories has to be considered: First we will provide an overview of eye tracking (Section 3.1) covering the existing well-established devices (Section 3.1.1) and applications (Section 3.1.2). Related studies and systems are presented as they apply to the main problems of head-mounted eye tracking, as stated in Section 1.4: calibration and drift, invariance, supervision and the parallax error (Section 3.2). Finally, existing research that opens the way for a different eye tracking and gaze estimation approach is presented in Section 3.3.

3.1 Application of Eye Tracking

In this section, we elucidate eye tracking in general. For this we start with a presentation of current existing devices, including their abilities and limitations (3.1.1). After that, we outline systems from the area of human computer interaction making use of eye tracking data (3.1.2).

3.1.1 State of the Art Devices

When we talk about eye tracking we distinguish between two fundamental classes of approaches that differ in the way they comply with computing paradigms: *static* or *remote eye trackers* do not require any attachments to the users body and are located in the environment (e.g., an eye tracker installed on a computer monitor) to track the

(a) (b)

Figure 3.1: The two categories of eye tracking approaches: (a) remote eye tracker attached onto a computer monitor[1], (b) Head-mounted eye tracking glasses[2].

user's eye(s) from a distant location and estimate the gaze on a pre-defined surface. Remote eye trackers are generally integrated with computer screens or laptops and monitor the user's eyes from a certain distance. *Mobile* or *head-mounted Eye Trackers* present the second category of eye trackers. Their components are fixed on a frame worn on the users head (e.g., mounted on a glasses frame). They are able to always track a person's eye, regardless of the head movement. The recent classification into *remote* and *head-mounted* has become commonly agreed upon in the field of eye tracking, and those terms gradually replaced older categorizations such as *table-mounted* or *head-mounted systems*. In Figure 3.1, examples of devices from these two categories are shown. The hardware is independent from the gaze estimation method, i.e. devices from each class may integrate any of the methods presented in Chapter 2, Section 2.2.1. In the following we will briefly present state-of-the-art devices from both classes.

Three main techniques exist to track a person's eye and do gaze estimation. The electro-oculography (EOG) based method places electrodes around the eye to measure the skin potentials, as proposed by Kaufman et al. [79]. The scleral search coil method uses a small coil that is embedded into a contact lens. It measures the voltage caused by an external electro-magnetic field [161]. Camera-based (video-based) techniques use the images of the eye to detect its characteristics in combination with images of the user's field of view to realize gaze estimation. This method is mainly applied nowadays by using either a head-mounted or a remote system. For a detailed review of eye gaze tracking methods, we refer the reader to Young and Sheena [207]. In the following sections we will focus on video-based eye trackers (introduced in Chapter 2.2).

[1] https://tobiigaming.com/eye-tracker-4c
[2] https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/

Remote Eye Trackers

Just 30 years ago, eye trackers were primarily used in research. Since off-the-shelf devices were not available, people had to build their own prototypes to do gaze estimation [59]. Several previous works used head orientation as an approximation of where people look. For example, Sippl et al. used a remote camera to detect facial features, such as the eyes and nose tip, and estimate head pose on four areas on the display [172]. Nakanishi et al. relied on a stereo face tracking system and the 3D head pose as an approximation of gaze direction [126]. Typically, remote eye trackers are built using one or multiple cameras to extract eye movements. Depending on the applied eye tracking method, additional light sources have to be integrated (cf. Section 2.2.1).

Accurate gaze estimation has remained a significant challenge when remote eye trackers are used. Such trackers only allow a single user to interact with one display at any point in time. The interaction is restricted to the tracking range of typically 50-80 cm in the central area in front of the display, thereby severely limiting the users' mobility [163, 180]. Previous work focused on extending the tracking range of remote trackers [56, 120], or on calibration-free (spontaneous) interaction, but was either limited to interaction along a horizontal axis, i.e., without full 2D gaze estimation [214] or required special interfaces [117]. Stellmach et al. addressed the mobility (interacting from different positions/orientations) of users [181] by using an additional external tracking system.

Remote eye tracking has developed into a relevant interaction modality, as it has become available and affordable for a large user base. In contrast to head-mounted types, remote video-based eye trackers are lower in cost and are available for under $200 (e.g., Tobiigaming)[3]). Characteristic attributes are unobtrusiveness and the ability to be mounted on a display or built into devices[4]. Consequently the interest in using eye tracking data, i.e. information about eye movements and gaze, for purposes other than research is increasing. Some studies were conducted to verify the tracking quality of current state-of-the-art devices. They found a large difference in the numbers reported by manufacturers for tracking accuracy (the offset between computed and true gaze point) and precision (the spatial distribution of the gaze points) across various tracking conditions and persons [13, 59, 142, 143]. In particular for remote

[3] https://tobiigaming.com/products/peripherals/

[4] `https://us.msi.com/Laptop/support/GT72S-G-Tobii-6th-Gen-GTX-980M`

eye trackers, the difference between the estimated and the true gaze point is often found to be larger than 1°, even in controlled environments [13, 142].

In general, remote eye tracking systems still share two common problems: (1) They cannot enable calibration-free and highly accurate gaze estimation and (2) gaze point computation is restricted to a certain set of objects limited by the hardware used. In particular, only a small part of a person's field of view is covered, as the remote camera, requiring the user's eyes to be visible, is stuck on a surface. Ensuring high accuracy is an important aspect in the development of new remote eye tracking systems. There are already investigations and approaches to improve the process of gaze estimation of calibration-less systems to achieve more accurate results [29, 48]. Besides that, Nguyen et al. [130] reported that the creation of systems that tolerate head movements is one of the hardest problems. In recent years, various approaches were published that address these problems [29]. Consequently, current state-of-the-art devices include the possibility to track the user's head movements (e.g., Tobii 4C[5]). But still, remote eye trackers are not usable in fully ubiquitous scenarios. In theory, one would have to equip an unlimited number of objects with such devices to enable spontaneous gaze estimation across various objects, which is impractical.

Head-Mounted Eye Trackers

The latest head-mounted eye trackers are more flexible than remote eye trackers in terms of mobility, as they allow the user to freely move around. In contrast, the first prototypes used desktop-like settings and stationary eye trackers in which a user's head was fixed regarding position and orientation, as depicted in Figure 3.2. The composition of these eye trackers has remained unchanged since then. This class of eye tracking devices consists of a glasses frame that is equipped with at least two cameras: first, one or two eye cameras capture a close-up image of the user's eye(s), and second, a world camera records parts of the user's current field of view. The purpose of the eye camera(s) is to detect and track the pupil as well as its movements. A widely used approach to achieve this is active illumination of the eye with infrared LEDs, as already explained in Section 2.2.1. For this purpose the infrared illuminators are also mounted on the glasses frame, usually next to the eye camera. Although the head-worn device maximizes the mobility of the user, the approach is more invasive (i.e., the user has to wear a device on the head) than a remote system and may be disturbing (e.g., limiting the field of view). In head-mounted eye tracking, gaze

[5] https://tobiigaming.com/product/tobii-eye-tracker-4c/

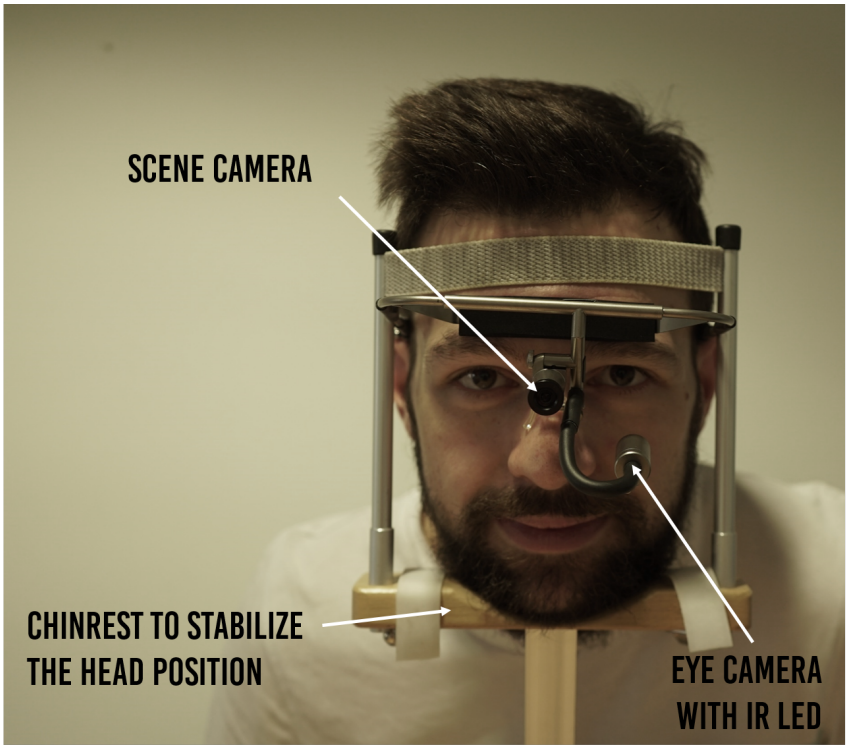


Figure 3.2: Example of an early head-mounted eye tracking device (Ergoneers Dikablis 2[6]) using a chin rest to stabilize the user's head.

estimation is the process of mapping the pupil positions from the eye camera's into the world camera's coordinate system. A crucial step towards gaze estimation is the calibration to a specific user on a dedicated plane (e.g., a display), necessary to create a function that maps eye to gaze positions. This calibration is typically performed for a fixed position and orientation of the user to the plane.

Current state-of-the-art eye trackers, such as Tobii Pro Glasses 2[7], are built on model-based (geometric) gaze estimation and track both eyes of the user (binocular). This setup enables *calibration-less* gaze estimation (cf. Chapter 2, Section 2.2.2). That is, the actual model is predefined based on heuristic data. Using a one-point calibration [122], the model is adapted according to personal parameters like the *angle kappa* (cf. Section 2.1). It was shown that even more calibration points are needed [197] to increase the gaze estimation accuracy. However, these methods need complex hardware setups with more than two cameras and are extremely expensive (costing more than tens of thousands of USD, at the time of writing). On the other hand, eye trackers

[6] http://www.ergoneers.com/en/eye-tracking-head-mounted-1-en/
[7] `https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/`

that rely on regression-based (interpolation) gaze estimation need a more extensive calibration step, but are usually cheaper (e.g., Pupil Labs[8]). We will dive deeper into the topic of calibration in Section 3.2.

Some devices offer a choice between monocular and binocular setup. That means that the device is equipped with two eye cameras, one for each eye, to track the movements of both eyes. However, there is no real consensus about which is better. A large portion of eye tracking research is done monocularly. According to Holmqvist et al. [59], there are two main reasons for that. First, there is the assumption that both eyes are making the same movements more or less simultaneously while looking at approximately the same point in space. Thus, tracking both eyes will produce more data that will need to to be synchronized and analyzed. The second reason is that monocular eye trackers are cheaper. However, due to the low price of cameras nowadays, this should not be a valid argument anymore.

One advantage of a binocular eye tracking setup is the possibility to calculate the 3D gaze point when tracking both eyes. As mentioned, both eyes are looking at the same point in space, when fixating a target. To enable 3D gaze estimation, a common approach is to perform calibrations for a finite set of distances to the calibration plane (e.g., a screen). Hence the computation of the 3D gaze point can be done using stereo geometry calculations [37] (Chapter 7.3). However, in a recent work by Mansouryar et al. [112] an approach was shown to calculate the 3D gaze point based on 2D information using a monocular head-mounted eye tracker.

When we talk about head-mounted eye tracking systems, we have to pay attention to the *parallax error*, as explained in Section 1.4. To eliminate the parallax error, a straightforward approach is to use binocular head-mounted eye trackers. Mardanbegi and Hansen investigated the problem with monocular head-mounted eye trackers. In their studies, they were able to conclude that the parallax error can be compensated for using an error function for different camera setups, i.e. various translations and rotations of the scene and eye cameras [114]. Bartz et al. used a predictive error model, in which the user interface itself is aware of the gaze estimation error [7]. They showed that the parallax error increases about 11.17% (i.e., from 1.68° to 1.9°), after changing the distance to the screen (and thus the calibration plane) by ±50 cm.

[8] `https://pupil-labs.com/store/`

The problem remains a significant challenge when using monocular head-mounted eye tracking devices for real-time applications.

Summary of Head-Mounted vs. Remote Eye Tracking

In the previous sections we familiarized the two different classes of eye tracking devices. To summarize, we provide a direct comparison in Table 3.1 between head-mounted and remote eye trackers concerning the primary characteristics. Remote eye tracking systems still restrict the gaze estimation to a certain surface (usually a screen). With an increasing number of objects that gaze should be estimated on, the complexity of the hardware installation will also increase. Head-mounted eye tracking devices can be used to measure gaze on any kind of object. In principle, the head-mounted approach may allow gaze estimation in pervasive and ubiquitous scenarios (remember the *Ubiquitous Computing for Eye Tracking Continuum*, Section 1.1, Figure 1.1).

The required mobility is limited with remote eye trackers. In a straightforward approach, one would have to instrument the environment by equipping a lot of objects with cameras to enable seamless gaze estimation. It is more convenient to provide the user with a wearable solution. This fact also opens new possibilities for diverse applications of eye tracking. Nevertheless, remote systems also yield benefits over head-mounted eye trackers. Due to the contactless usage, i.e. not being attached to the user in any form, these systems facilitate unobtrusive eye and gaze tracking. However, we believe that with the miniaturization of cameras[9] and the familiarity with spectacles, head-mounted systems will become more convenient. The only issue left is with the camera itself, which concerns both of these aspects. People are usually reluctant about potentially offensive camera shooting.

The second advantage of remote systems is the ease of data analysis. The cameras are fixed to a certain object or surface, which creates the reference frame to measure gaze on. If a head tracking algorithm is integrated, a person's gaze can be directly mapped onto the surface (e.g., a display) as long as the cameras are able to capture the eyes. This allows for distance changes in front of the cameras and thus the calibration plane. The procedure is different with head-mounted systems. There we have two cameras that have to be calibrated to map the eye movements to gaze. For example, if the gaze on a display should be computed, an additional mapping is needed. There are many more issues to be resolved – calibration drift, invariance, parallax error and

[9] http://image-sensors-world.blogspot.de/2013/11/samsung-on-image-sensor-progress.html

supervision – to make these devices ready for ad-hoc everyday usage, as discussed in the problem statement in Chapter 1.4. Investigating respective methods to solve these issues is the task of this thesis.

	Remote Eye Trackers	Head-Mounted Eye Trackers
Setup	Placed at a fixed location in front of the user (e.g., below a monitor or windshield in a car, to estimate gaze on them).	Worn on the user's head (e.g., as a pair of glasses, or frame attached to a hat or a headband).
Application	Used in situations during which the users can sit or stand in one place and gaze at objects presented on a stationary surface (e.g., interact with a screen).	Used in scenarios that require users to move around and interact with physical objects or people (e.g., marketing research, way finding).
Obtrusiveness	Less obtrusive than head-mounted systems – users can easily forget about the eye tracker during usage.	More obtrusive since the system has to be worn on the head and may be partially visible to the user.
Freedom of Movement	User must be positioned in front of the eye tracker and only limited head movements are tolerated. In addition, the space between the eye tracker and the user's eyes must not be occluded by additional objects, i.e. the user cannot hold something in their hands.	The user can move freely around and manipulate objects. However, head-mounted eye trackers usually work best for objects placed at the same distance as the calibration plane (due to parallax error).
Ease of Analysis	As only the eyes of the user are recorded, there is no information about the scene. Typically eye movements are directly mapped to the area surrounding the cameras (e.g., a screen the eye tracker is mounted at). Hence, data analysis is usually easier and faster as aggregation can be automated. But gaze estimation is restricted to a certain area.	Gaze estimation can be realized on nearly the whole field of view of the user, as the recorded scene changes due to head movements by the user. However, the gaze location in the scene camera frame is usually separated from the scene content. Hence, data analysis is more complex, as the association of the user's gaze with an object (e.g., gaze estimation on a screen) may require manual labor.

Table 3.1: Comparison between remote and head-mounted eye tracking concerning five major attributes.

3.1.2 Gaze-based Human Computer Interaction

We already discussed in Chapter 1.3 the variety of application areas making use of eye tracking. In this section we will give a more detailed overview of eye and gaze based interaction. Here, we do not distinguish between systems that are based on remote or head-mounted eye tracking devices.

With gaze, we naturally indicate what we visually attend to and what we are interested in [196]. In addition, it often precedes any action we are planning to do. Consequently, gaze is a powerful modality in human-computer interfaces for both, as an input method and as a data source for analytics. Gaze-based or gaze-enabled user interfaces are systems that react to a user's eye movements, as described by Richard Bolt [15].

Majaranta and Bulling [110] derived a continuum that divides gaze-based systems into four categories. Figure 3.3 shows the different types, starting from applications where *direct* gaze input from the user is required. It ends with systems which monitor the eye and gaze behavior, thus using *indirect* input for analysis. Between these two clearly separate application areas, we have attentive and adaptive systems that react to or learn the user's eye behavior, respectively. In the following we will give an overview of the four categories.

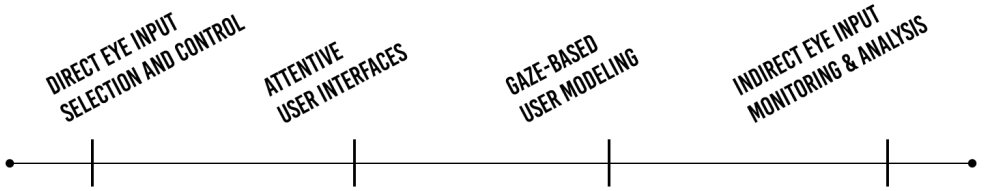


Figure 3.3: Eye tracking application continuum, adapted from Majaranta [110].

Direct Input

Explicit or active eye input is used in gaze-based interfaces that enable a hands-free command or control of a system. People use voluntary eye movements and consciously control their gaze direction to interact with a computer. For example, due to developments in technology, gaze-based input and control is a valid alternative for people with serious physical disabilities to interact with the real world [8, 34]. The most common approach to realize gaze-based input and control is to transfer the ability of the eyes to point at a desired target. The human gaze is faster than other existing

pointing devices, such as a manual mouse [171], although it might not be as accurate. Using gaze as the only input modality in human computer interaction still is subject to two major challenges:

- The **Midas Touch** problem [71] – named after the mythical king who turned everything he touched into gold. In gaze-based interfaces, it stands for the problem that looking at a specific target does not necessarily correspond to the intention to select that target. Simply put, the gaze is misclassified as an interaction input.
- The **region of uncertainty** [107] covers the area around the estimated gaze position in which the true gaze point is located. The size of this region depends on three properties that affect the accuracy of gaze estimation [165]: (1) The precision of the eye tracking device and (2) the goodness of the calibration both might be improved by introducing better hardware and software. In addition (3) the variance of the human fixation itself influences the threshold of the tolerance level.

There are several ways to cope with the region of uncertainty problem. Obviously, one can increase the size of the targets to a certain extent (e.g., larger buttons), so that the user is able to trigger them more easily. Spakov et al. published a technique that dynamically increases the size of menu items, dependent on the calibration [198]. However, increasing the target size will limit the space for other content and user interface elements on the screen. Hence, all control elements would have to be organized in hierarchical structures such as drop-downs and sub-menus to prevent a cluttered screen content. For example, Huckauf et al. proposed a radial menu structure matching the usual circular region of uncertainty [63]. Alternatively, there are approaches that are based on the *Magic Lens* metaphor, invented by Bier et al. [12]. Already in the year 2000, Chris Lankford developed an approach to use the user's gaze as a magnifier to ease interaction [103]. Stellmach et al. use fisheye lenses to explore large data sets like image galleries [182]. For clarification we provide examples of the aforementioned approaches in Figure 3.4.

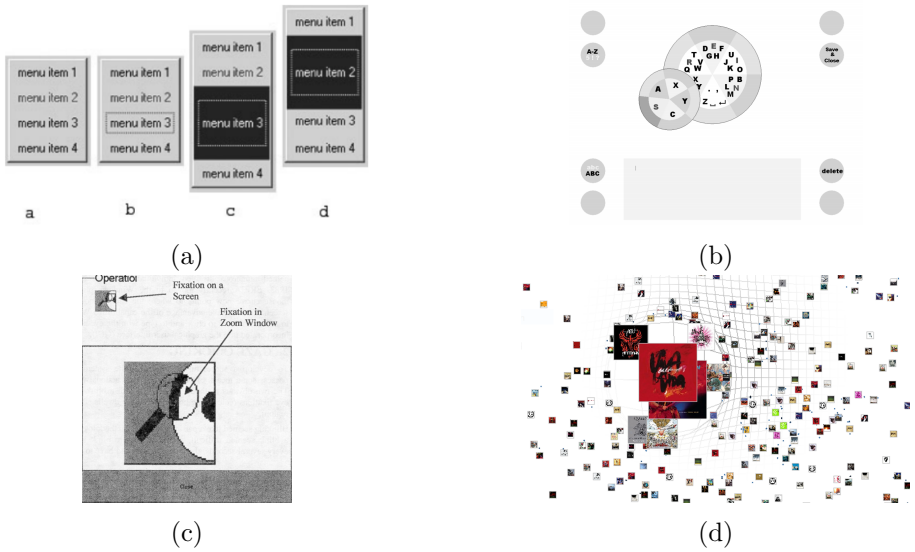


Figure 3.4: Examples to address the region of uncertainty problem: a) dynamic menu [198], b) radial menu layout [63], c) magnifying lens [103] and d) fisheye lens to browse an image gallery [182]

As stated above, it is problematic in gaze-enabled interfaces to distinguish between intentional gaze gestures (e.g., fixations), used as an input modality, vs. the purpose of visual perception. Preventing the Midas Touch problem, where all viewed items or objects are directly selected, is a demanding challenge in gaze interaction. This problem and its possible solutions have been studied for years. Velichkovsky et al. [195] investigated diverse fixation times as an indicator for different levels of cognitive processing. In systems that are based solely on gaze and use static buttons, the most common method to indicate an intended action is dwell time. For this, the user has to gaze for a certain amount of time at the target element (e.g., fixating a button). This concept is easy to learn and ergonomic. Several studies were conducted by researchers to examine the optimal dwell time to reduce false positives. Majaranta et al. [109] proposed an approach that allows adjustment of the dwell time. According to their findings, the optimal value highly depends on the individual users and their familiarity with the input method, resulting in dwell times between 180 ms and 300 ms. Other studies recommend values ranging from 150 ms to 1000 ms [108].

Since there is no general consensus about the right dwell time, another more promising approach is to use gaze gestures to prevent Midas Touch. That means the users perform specific eye movement patterns [69]. For instance, a sequence of eye strokes, as shown in Figure 3.5, is performed by the user to draw a geometric form or even a letter with their eyes. Such gestures are named semaphoric or symbolic, as they

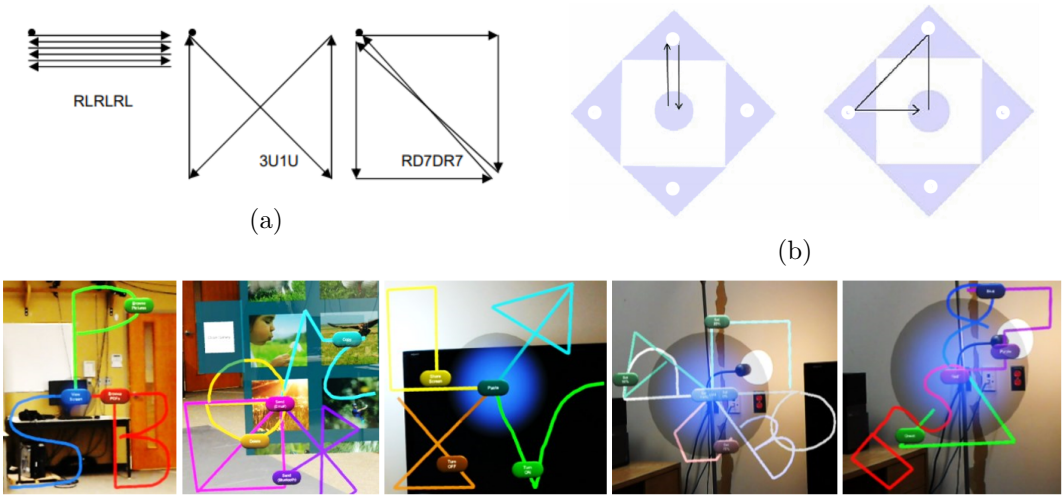
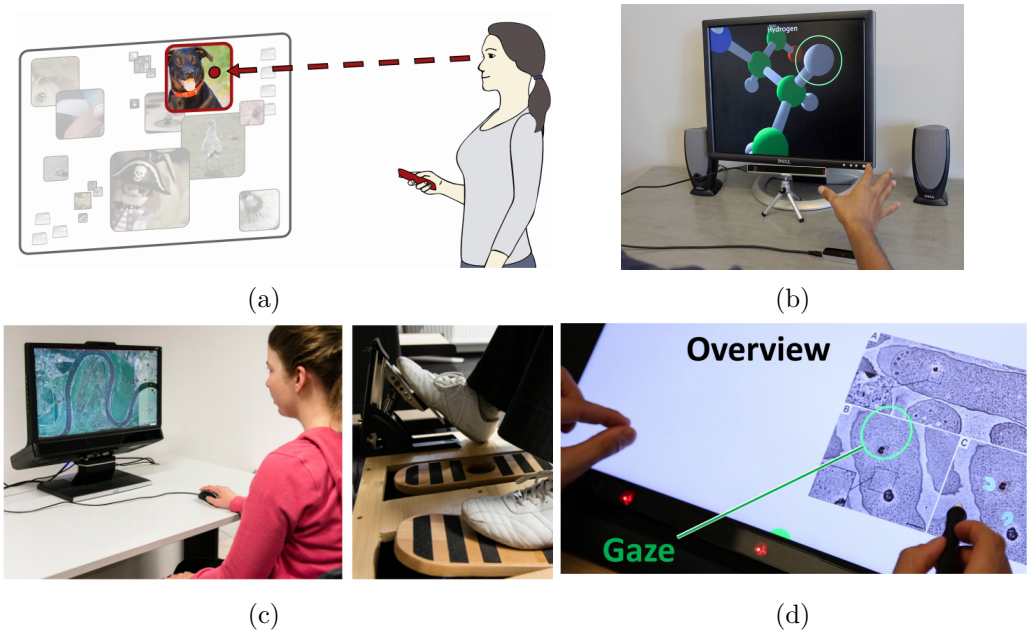


Figure 3.5: Examples of gaze gestures to prevent Midas Touch: a) eye strokes to 'draw' a command with the eyes [36], b) simple eye strokes [64] and c) complex gestures in a VR application [31]

define a certain set of vocabulary that maps gestures to commands. Drewes et al. [36] investigated the ability of users to perform different eye gestures as an interaction technique. They found gestures to be invariant against low gaze estimation accuracy and calibration shift. That is, the concept of detecting a specific eye movement is not dependent on the actual gaze of the user, since only the variations in the position need to be detected. Hyrskykari et al. [64] compared gaze gestures (consisting of 2 or 3 eye strokes, as shown in Figure 3.5b) against dwell time based selection while playing a game. They found gaze gestures to be more robust and accurate. Delamare et al. [31] proposed a gaze gesture guiding system, which is usable with a head-mounted display (e.g., a VR headset), as depicted in Figure 3.5c. With that system they aim to ease the exploration of available gaze gesture command patterns for novice users.

An easy, and in recent years more convenient, way to counteract the issue of unintentional selections is to combine gaze with additional modalities. Qvarfordt provides a deep review of using the human gaze in multimodal interaction [153] (Chapter 9). She proposes a two-dimensional design space to differentiate between active/passive input and mobile/stationary systems. Various possibilities exist to combine our gaze with a second input mode. Already in 1999, Zhai et al. [211] proposed a method combining mouse and gaze input for selection. The so-called MAGIC pointing method includes the advantage of gaze versus hand input concerning spatial movement of a on-screen cursor. The actual selection is realized by a mouse click, to prevent inaccurate gaze



(a) (b)

(c) (d)

Figure 3.6: Examples of multi modal gaze input: a) gaze based pointing and selection via mobile phone [180], c) using gaze together with feed control [85], b) gaze plus hand gestures [25] and d) gaze plus touch and pen input [148]

pointing. The work done by Zhai et al. is fundamental, since many works proposed similar techniques, but replaced the mouse by a different modality. Typically MAGIC pointing serves a baseline in terms of accuracy and performance for selection tasks.

In Figure 3.6 we provide an overview of different gaze plus 'x' input combinations, where 'x' stands for the additional modality. Stellmach and Dachselt [180] developed a system that enables users to select objects on large distant displays. For this, they replaced the mouse by a mobile phone. Utilizing the touch screen of the hand-held device, users are able to manually correct the position of the gaze cursor and finally trigger a selection, as shown in Figure 3.6a. In this way, they counteracted the problem of inaccurate gaze estimation as well as the Midas Touch. Klamka et al. [85] built a custom controller to use a person's feet as an additional input method (see Figure 3.6b). They use it as a modality for precise object selection on the one hand, but also to enable more complex interaction techniques, such as pan and zoom in a map application. In their evaluation, they were able to show that gaze plus foot input can beat traditional mouse-only input and is preferred in specific tasks, such as map navigation. Another different approach was developed by Chatterjee et al. [25]. In their system, Gaze+Gesture, they use a person's gaze for target acquisition, as usual. To perform selection and other additional commands, they implemented hand gestures

using a hand tracker (i.e. a Leap Motion), as shown in Figure 3.6c. They compared the system against other input modalities by conducting the standardized Fitts' study [212]. They were able to achieve results comparable to mouse-only input and outperformed the straightforward dwell time approach. Pfeuffer et al. [148] combined gaze and touch as a novel input modality on the same surface (as depicted in Figure 3.6d). Basically, this approach is taking advantage of the strength of Gaze+Gesture, as the user can focus on a specific element on a touch display, while the content is manipulated through different touch inputs. For example, they showed the potential of their approach in map navigation. In [149] they extended the approach by adding a digital pen as another input device, leading to more novel application designs.

In contrast to these approaches, one can also use a non-physical additional input. Miniotas et al. [119] propose a speech-augmented gaze pointing technique. In their evaluation, they were able to conclude that gaze plus speech is as good as manual pointing (MAGIC) regarding accuracy. Beelders and Blignaut [10] used gaze and speech input for text entry on a virtual versus a physical keyboard. They found the physical keyboard to be faster and more accurate, although this could be explained by users' familiarity with this input modality.

The majority of all the above examples are based on stationary scenarios (i.e. using a remote eye tracker). This may be because of *traditional limitations to eye tracking technology*, highlighted by Qvarfordt [153]. She also notes that with the progress made in mobile computing (e.g., prevalence of smartphones, watches and head-mounted displays), gaze-based interaction may be required to work in mobile and ubiquitous settings (cf. Figure 1.1). Besides this, all the above systems are restricted to a single surface, i.e. a single screen. Both aspects are relevant to the goal of this thesis (cf. Section 1.6). Even if the interaction concepts are transferable to head-mounted settings, there are other issues that have to be addressed first (cf. Section 1.4), to make head-mounted eye tracking usable outside of a controlled environment, in real-world settings.

Regardless of the method used for gaze input, it is important to provide corresponding feedback to the user. According to Majaranta and Bulling [110], it is essential to notify the user of the effects caused by focusing and selection. Feedback is mainly needed for dwell time based, eye gesture and gaze-plus-gesture approaches. In contrast, multimodal systems that combine gaze with a physical device (e.g., mouse, touch enabled

surface, pedal) usually give direct feedback via use of the specific device. Oftentimes, the computed gaze point is visualized on the screen; dwelling is simply animated by a progress bar. Besides this visual feedback, more recent approaches include haptics. For example, Kangas et al. [77] used vibrotactile feedback of a mobile phone while interacting with it using gaze gestures. They found that the use of haptic feedback increased the performance and user experience. Rantala et al. [155] extended that approach by integrating the actuators needed for the feedback mechanism directly into the eye tracking glasses.

Attentive User Interfaces

Attentive user interfaces (AUIs) are an instance of so-called non-command interfaces, as defined by Jakob Nielsen in 1993 [132]. The user does not actively trigger a specific command (e.g., selection) by changing his or her gaze. Instead, the information about a person's eye movements and gaze positions is analyzed and used in the background. Attentive user interfaces may include additional sensor data by monitoring the user's physical proximity and body orientation. An attentive system may be gaze-aware or eye-aware, i.e. for some applications it is sufficient to detect eye contact without information as to where someone is looking. In Figure 3.7, different use case scenarios of the eyeCONTACT sensor [169] are shown. The device simply detects a person's eyes and estimates whether the user is looking at the sensor. It can be attached to an arbitrary physical object, to make it aware of the surrounding persons. For example, placing the device above a TV screen makes it possible to detect if someone is looking at the display. This information can be used to switch off the TV or pause the TV program, if no user is currently gazing at it.

Figure 3.7: Example application of eyeCONTACT sensor to sense a person's attention [169].

Attentive user interfaces have become much more powerful than the above example. According to Bulling [18] the management of on-screen notifications (e.g., messaging) to reduce interruptions is an important research challenge in human computer interaction, which can be addressed via attentive user interfaces. Basically, one can distribute the information shown on a display at different distances from the user's actual gaze point, i.e. the current point of attention on the display. Such displays are called gaze contingent displays (GCDs) [38]. Klauck et al. [86] explored different appearance properties of notifications on GCDs, with a focus on size, opacity, blinking frequency and movement speed. In a user study they examined the effect of these properties on noticeability of notifications and distraction of users. The results showed that changes of static properties is not very distracting, but have a high noticeability. Changing dynamic properties results in maximal noticeability, but also in high levels of distraction.

Some other work investigates how the user's attention, and thus his or her gaze, can be actively changed. Bartram et al. [6] present two subtle gaze direction methods: The first method uses color changes to direct the user's gaze, while the second method is based on blinking. The authors propose that through flickering and changing the color in the near periphery, the user is directed without noticing it. They performed two experiments on how these two gaze direction methods compare in distraction and noticeability. The results showed that motion is most effective in gaining user attention without being distracted and irritating. Color changes are far less effective at gaining the user's attention; blinking is by far the most effective at gaining attention, but it is also the most distracting and irritating gaze direction method. A continuation of this idea, is the system iDict by Hyrskykari et al. [65]. It requires the user to know that their eyes are being tracked. By observing the eye movements during reading, the system supports the user in a proactive fashion. In this case, a text in a foreign language is automatically translated if comprehension problems are detected via eye movement patterns. The reader is also able to actively trigger the translation by staring at the difficult word. There are also approaches that try to predict the user's intentions and next activities based on the previous gaze behavior [76]. Since we do not want to go too deep into detail on this topic, we refer the reader to reviews that provide more information about attentive applications [154, 68].

User Modeling

Analyzing a person's eye movements is a well-known method in experimental psychology to provide insights in visual behavior. Mainly head-mounted eye tracking

systems are used as a tool to conduct studies and record data about a person's gaze in arbitrary environments. This has helped researchers to increase the understanding of visual behavior. The movements that our eyes perform throughout the day are directly linked to our activities. Bulling et al. [20] investigated the detection of five types of office work, solely based on eye movement data. They utilized a head-mounted eye tracking device based on electro-oculography (cf. Section 2.2) to record data from people while they performed different tasks (e.g., reading a text, browsing the web, etc.). By analyzing the data with a SVM classification, they were able to identify the different tasks. Steil and Bulling [179] go one step further by recording eye tracking data, including the main types of eye movements (i.e. blinks, saccades and fixations) plus gaze information, during daily life. They used a video-based head-mounted eye tracker and were able to extract several daily activities, such as 'computer work', 'eating' and 'social interaction'.

Besides human activities, visual behavior is closely linked to various cognitive processes of visual perception. Bulling et al. [21] showed the possibility to automatically recognize the visual memory recall of people based on eye movement data alone. In another study, Tessendorf et al. [190] were able to conclude that a person's cognitive load during focused work can be successfully detected by analyzing her visual behavior. Even selected personality traits that are difficult if not impossible to assess using other sensing modalities are linked to our eye movements. For example, Hoppe et al. [60] propose a method to automatically detect different levels of curiosity.

In human computer interaction, all these findings can be used to gain further insights in how to design and evaluate gaze-based interfaces that are using head-mounted eye tracking devices. However, setting up the experiments and the required hardware still entails extensive work, as head-mounted eye trackers are not ready to use in a spontaneous way.

Indirect Input

Passive eye input is not used to control an interface, in contrast to active eye input. It is a source of data for diagnostic applications. According to Duchowski [37] these cover visual behavior analysis for psychological purposes. Also in marketing and advertising, eye movement data is usually first recorded and analyzed afterwards, since there is no need to react in real time to a person's interaction. For example, Bulling et al. proposed a system called EyeContext with which it is possible to infer cues from eye

movement data. They built a device bundle, which consists of an EOG based head-mounted eye tracker connected to a laptop and a mobile phone, used to annotate the data. They were able to automatically classify the cues into five cases, namely being socially or physically active, being inside our outside a building or doing focused work.

A different application case of passive eye and gaze input is usability and user experience analysis. This is typically done by using a remote eye tracking system, while people are interacting with a desktop computer. Goldberg and Kotval [47] defined different eye tracking metrics and developed a correlation between eye movement data and usability problems. They mainly focused on the detection of visual search, for which they used metrics such as number of fixations, fixation time and the ratio between the two types of motion. Jacob and Karn [72] published a survey about various existing metrics that can be used to identify usability flaws based on eye and gaze data. They also noted the limitations of the technology itself: A good saccade detection algorithm requires a sampling rate higher than 100 Hz. For more information about passive and indirect eye monitoring and their applications, we refer the reader to the survey by Riad Hammoud [51].

3.2 Eye Tracker Calibration

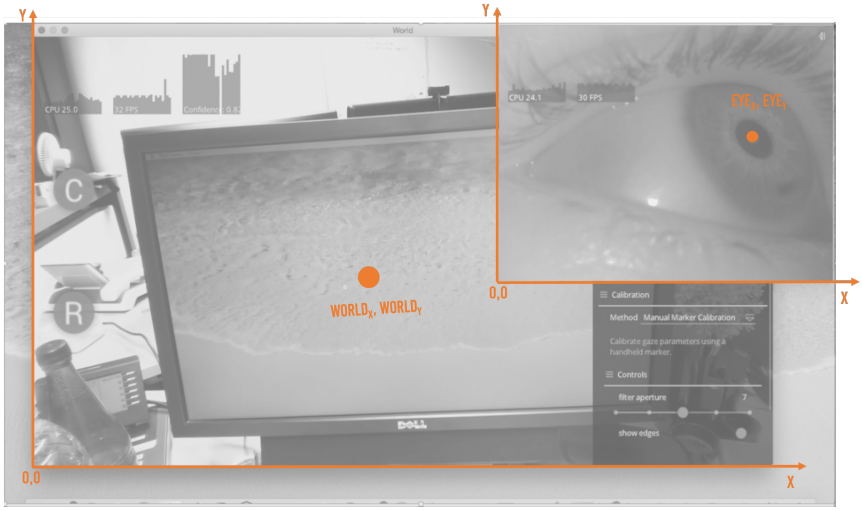
In this section, we will address the problem of calibration, the fundamental issue for which this thesis is going to develop solutions. We already gave an introduction to the problems, caused by calibration in Section 1.4. In the following the standard calibration procedure of head-mounted video-based eye trackers is explained in detail. Subsequently, we present existing approaches that investigate strategies to efficiently re-calibrate an eye tracker in Section 3.2.2. In Section 3.2.3, we present approaches to reduce the need for a calibration procedure, aiming for calibration-less or calibration-free gaze estimation.

3.2.1 Notes on Calibration

According to Holmqvist et al. [59], accuracy is one of the major characteristics of data quality in eye tracking. It is defined as the distance between the estimated gaze position of the device and the actual gaze location in the environment (where the person is really looking). Inaccuracy is directly influenced by the eye tracker itself; it can be a result of bad pupil detection and tracking, a poor mapping from pupil to gaze positions, or both. As we presented in Section 2.2, much research has been done

recently on improving the pupil detection, including robust tracking in realistic conditions [44]. Thus the source of error is more related to a bad pupil-to-gaze mapping, which is a direct result of poor calibration.

(a) Sampling pupil positions and the assumed gaze position on the marker on the screen.



(b) Calibration result showing the estiamted gaze point.

Figure 3.8: Calibration of a head-mounted eye tracking device the Pupil Labs software[10].

Figure 3.8 illustrates the calibration process in the case of an interpolation-based gaze estimation approach using the Pupil Center Corneal Reflection (PCCR) method. This procedure can be used with a monocular or binocular eye tracking setup. Given an infrared illuminated eye image, the pupil center as well as the position of the glint

[10]https://docs.pupil-labs.com/#calibration

(i.e. reflection of IR light source) can be computed. Some approaches (mostly remote systems) use the vector between both points for gaze estimation and others assume a direct correlation between pupil center and a person's gaze direction (cf. Figure 3.8). Kassner et al. [78] argue that the cameras of a head-mounted device are fixed to the user's head, so that the relative position between them and the user's eye will not change when he or she is turning his or her head. Obviously, this assumption is not valid in stationary setups, in which the cameras are mounted at the screen for example, and the user's head is not fixed. However, in both cases a calibration has to be performed to translate eye to gaze positions. We will consider only the calibration of a head-mounted device in the following: In general, a *mapping function* has to be established to successfully translate the pupil-glint vector or the pupil center (denoted as eye_x and eye_y) from the eye camera's into gaze positions (denoted as $world_x$ and $world_y$) in the world camera's coordinate system (see Figure 3.8). The mapping function is of the following form:

$$world_x = a_0 + a_1 eye_x + a_2 eye_y + a_3 eye_x eye_y + a_4 eye_x^2 + a_5 eye_y^2$$
$$world_y = b_0 + b_1 eye_x + b_2 eye_y + b_3 eye_x eye_y + b_4 eye_x^2 + b_5 eye_y^2$$

It comprises two second-order polynomial functions whose coefficients $a_0, ..., a_5$ and $b_0, ..., b_5$ need to be estimated through the calibration procedure. This is performed by looking at a number of pre-defined visual stimuli. While the user is fixating the calibration target, data is sampled that consists of pupil positions in the eye image, the physical orientation of the eye and the location of the target (stimulus) on the calibration plane (i.e. the screen for remote trackers, the scene camera images for head-mounted devices) [51].

In the example in Figure 3.8a, a total of nine calibration points is used; these are distributed along the screen border. Note that the number of calibration points relates to the area gaze is estimated on. This can range from 2 to 16 using monocular setups [59], and can be reduced to one point for binocular systems (cf. Chapter 3.1.1), although this lowers the accuracy. When head-mounted eye tracking systems are used, it is necessary to extract the positions of the presented stimulus in the world camera images. It is assumed, and fundamental, that the user is looking at the location of the stimulus. Each calibration point results in two equations for $world_x$ and $world_y$, which are part of a linear equation system of, in total, 18 equations with 12 unknown coefficients. This can be solved by using linear least squares [202]. The resulting mapping function is used to translate the eye into gaze positions, as depicted in Figure

3.8b. For 3D gaze estimation, the calibration is done by fitting a third-order polynomial, as calibrations at multiple depths are recorded [114].

Coming back to gaze estimation accuracy, the reported values range from 0.3° to around 2° [73, 88]. Remember that the size of the human fovea spans 1.5-2° of the visual field (cf. Section 2.1). Accuracy differs between the gaze estimation methods (cf. Section 2.2). Regression-based gaze estimation, as presented above still achieves the best results (cf. Section 2.2). Its accuracy depends on the quality of the computed mapping function and the correctness of the coefficients. However, the calibration procedure itself leads to many problems.

First of all, the calibration is an individual process that has to be conducted for each user separately prior to usage [37, 59]. The reason is that the calibration includes human-specific parameters (e.g., cornea curvature and angle kappa) and geometric information (e.g., relative location and orientation between cameras) (cf. Section 2.1 and [52]).

Nyström et al. [142] investigated calibration with respect to practical issues. Typically a second person is needed to set up and calibrate the eye tracking device. They compared different approaches for setting up the device and sampling the calibration data. With an operator approach, a second person is doing all the work, whereas in a participant-controlled approach, the users themselves monitor the data sampling. A system-controlled approach does an automatic collection of calibration data, as the decision whether or not the user is currently fixating the calibration target is an automated process. They concluded that a participant-controlled approach yields the best results. Automatic decision by the system still achieves better accuracy and precision than an operator-controlled calibration. It is noteworthy that at least for remote eye tracking devices, the major manufacturers use an automatic calibration (e.g., Tobii 4C [11]). However, modern high-end head-mounted eye trackers such as the Tobii Pro Glasses 2 [12] still require the support of an additional person acting as operator. Also, the developer-friendly Pupil Labs eye tracker applies a combination of the approaches depending on the use case (Pupil Labs Manual[13]). Although this might not be a huge problem per se, the occurrence of the so-called calibration drift requires a regular re-calibration to maintain accurate gaze estimation over time.

[11] https://help.tobii.com/hc/en-us/articles/213414285-Specifications-for-4C
[12] https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/
[13] https://docs.pupil-labs.com/#calibration-methods

This effect describes the deterioration of gaze estimation accuracy due to various reasons. These include changes in eye physiology (e.g., wetness of the eye), the environment (e.g., lighting conditions), the position of the device and/or the camera in relation to the eye, and changes of the user's location and orientation. The latter refers to the problem that the calibration is typically performed for a fixed position and orientation of the user to a single display. While this is less of an issue for stationary settings and TV-sized displays, mobile settings and multiple, potentially large, displays evoke two types of motion: (1) user movements in front of a single display to inspect other parts of the display's content; and (2) head movements to reach targets outside the ocular motor range [49]. In addition, there might be multiple displays present, causing further movements. Both types of motion considerably reduce gaze estimation accuracy [23].

In order to achieve high gaze estimation accuracy, it is crucial to track a user's (and eye tracker's, respectively) position and orientation relative to a display. Solutions that realize this include the augmentation of the environment with visual markers [17, 208] as well as using vision-based motion capturing systems (e.g., OptiTrack). With advances in computer vision, visual markers can be substituted by detecting the display directly in the scene camera's field of view. Mardanbegi et al. detect screens based on quadrilaterals found in the scene [113]. Turner et al. extended this to multiple displays (based on the displays' aspect ratios) by adding a second camera and a method for transparently switching between two calibrations [194]. While such approaches can achieve high tracking and gaze estimation accuracy, the need to deploy them for every display the user might want to interact with currently severely limits uptake and truly pervasive and spontaneous gaze-based interaction.

In the end, a re-calibration is necessary to maintain a constant high accuracy. However, executing the full procedure to replace outdated calibration data is time-consuming and distracts from the actual task. In Section 1.4, we highlighted the problems concerning the calibration procedure, which constitute the foundation of this thesis.

3.2.2 Re-Calibration Strategies

In this section we present some existing work, that counteract the calibration drift by performing an appropriate re-calibration. There are different approaches that aim to reduce the time for re-calibrating an eye tracker while recovering the initial accuracy. The first approach we will present was developed by Stampe [176], already in 1993.

In his approach, *re-centering with one point*, a calibration target out of the set of all calibration points is shown to the user on a recurring basis. Based on the assumption that the user is focusing on the calibration point, the offset between the current eye position and the corresponding one recorded during calibration is computed. The estimated offset value is used as a correction for the next eye positions before translating them into gaze points. Hence, the computed offset serves as a global correction for all recorded eye positions. The point used for the re-centering procedure is freely selectable. It only has to be one of the initial calibration points.

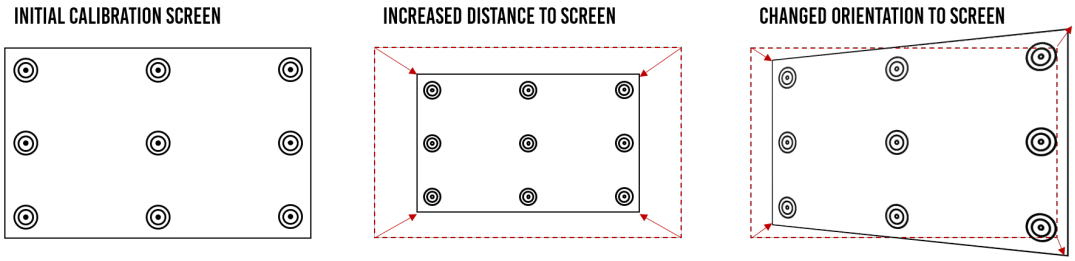


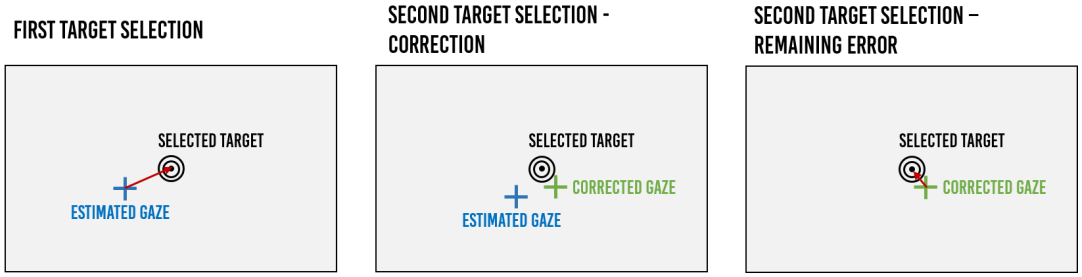
Figure 3.9: Resulting view on the calibration plane (i.e., the screen) after the user has changed their distance or orientation.

An obvious problem with this approach is that it will not resolve a non-uniform drift. It could be the case that the offset between eye positions varies across different calibration points: for example, if the user changes position in front of the screen, i.e. the distance and/or orientation to the screen. In Figure 3.9 we show the different situations, and the resulting changes in the position of the calibration points, from the user's perspective.

When changing the distance to the screen, at least the central calibration point will remain at the initial position. However, if another calibration point is used for the re-centering approach, the computed offset might further increase the already established calibration drift. Even worse, if the user changes orientation to the screen (and thus the calibration plane), the positions of all points will be different. In this case, Stampe's approach will not deliver any usable results.

Stampe and Reingold [177] improved the above approach with their method called *dynamic re-centering via target selection*. In contrast to the initial method, the user is required to actively select targets on the screen that are used for the re-calibration procedure. Target selection is realized by dwelling and triggers the offset computation. In this way, they counteract the aforementioned issues of the previous approach.

However, the correction requires a multi-step procedure, actively involving the user, as depicted in Figure 3.10. Initially the system assumes no error, i.e. the very first target selection serves for data acquisition only. The actual correction is started with the second target selection. Finally, the algorithm adds a fraction of the estimated error to the drift value. Hence, the calibration drift will be constantly reduced in the optimal case.




```
ESTIMATED_DRIFT = (0,0)
FOR EACH TARGET_SELECTION DO:
        CORRECTED_GAZE = ESTIMATED_GAZE - ESTIMATED_DRIFT
        RESIDUAL_ERROR = CORRECTED_GAZE - TARGET_CENTER
        ESTIMATED_DRIFT += ¼ RESIDUAL_ERROR
```

Figure 3.10: Step-wise reduction of the calibration drift.

However, as in the previous approach, the dynamic re-centering applies the offset globally. If the calibration drift is too big (e.g., after re-adjustment of the cameras), a wrong target may be selected, which results in wrong corrections.

The work of Hornof and Halverson [61] is based on so-called *required fixation locations (RFL)*. This approach also measures the gaze estimation errors during target selections performed via mouse input. For this, they introduce implicit (e.g., a button the user clicks on) and explicit (calibration points the user has to look at) RFLs, which have to be fixated by the user at any time. The correction in their approach is not applied globally. Instead, every estimated gaze point is shifted individually.

In Figure 3.11, the idea of their correction algorithms is depicted. In this example the gaze point is corrected according to the following steps: At the estimated gaze point, the screen space is divided into four areas. In each region, the nearest stored error vector is used to compute the average error. Finally, the error offset is weighted by the inverse distance between the selected error vectors and the actual fixation position. In this way, the approach can account for a varying calibration drift across the screen. However, their proposed method does not work in live settings as they use post-hoc

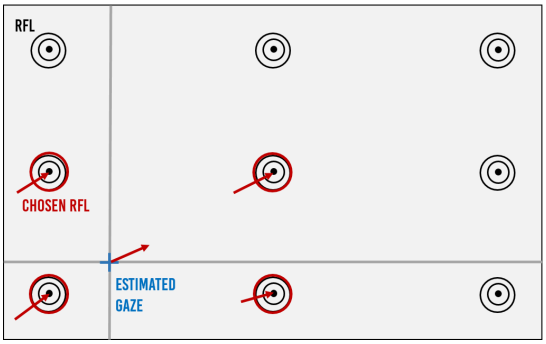


Figure 3.11: Selection of required fixation locations used to correct the calibration drift.

correction. Moreover, it is dependent on active input of the user via a mouse, i.e., it cannot be used for gaze-only systems.

Finally, we summarize the characteristics of these approaches in Table 3.2. Each of the presented techniques has at least one issue that makes it inappropriate for use in real time with head-mounted eye tracking devices. We will propose a different approach in Chapter 4, that accounts for each of the mentioned problems.

	Re-Centering with 1 point	Dynamic Re-Centering	Required Fixation Location
real-time capability	●	●	○
corrects drift error	●	●	●
eye tracking data only	●	●	○
manages abrupt drifts	●	○	●
adaptive to varying error	○	○	●

Table 3.2: Comparing different aspects of the presented re-calibration approaches. None of the existing methods provides a general solution, (● fulfilled, ○ not fulfilled).

3.2.3 Calibration-Less Approaches

In this section we present techniques that aim to ease the usage of eye tracking systems, by introducing alternative calibration approaches. We can group these approaches

into two categories. One group investigates methods that make the calibration more comfortable for the user. In the other, researchers propose techniques without the need for a user-dependent calibration procedure.

A prominent approach is to make use of smooth pursuit movements of the eye (cf. Section 2.1). Pfeuffer et al. [150] calibrate a remote eye tracker while following moving objects on a screen in an unobtrusive way. The presentation of a moving target is used to record calibration data which can be utilized to establish an accurate mapping between the recorded eye position data and the surface where gaze should be tracked. The movements of the eye are correlated with known trajectories of the moving objects (up to 0.6° of gaze estimation accuracy). A big advantage of pursuit calibration is that it makes it possible to determine whether the user is currently focusing on the calibration target or not. Consequently, the system can decide if data should be sampled or not. In this way the whole calibration procedure becomes more flexible because it is robust to interruptions. Moreover, it is more natural for the user to follow a moving target instead of gazing at points for a certain amount of time, making the whole calibration less tedious. TextPursuits is a similar approach developed by Khamis et al. [82]. Instead of abstract objects, they show moving text on a display. In this way they can present content on the screen, while using it to calibrate an eye tracking system, and achieve results similar to those of other approaches.

Flatla et al. [42] investigate a different approach with the goal to make the calibration procedure itself less exhausting and more enjoyable for the user. They developed so-called calibration games, in which they transformed the standard calibration routines into mini games. The basic principle is to use gaze as input, while they can assume that a person is looking at or following a specific object, in order to play the game. Their evaluation shows that the quality of the recorded data does not change. Users found the new way of calibrating an eye tracker more enjoyable and less tedious.

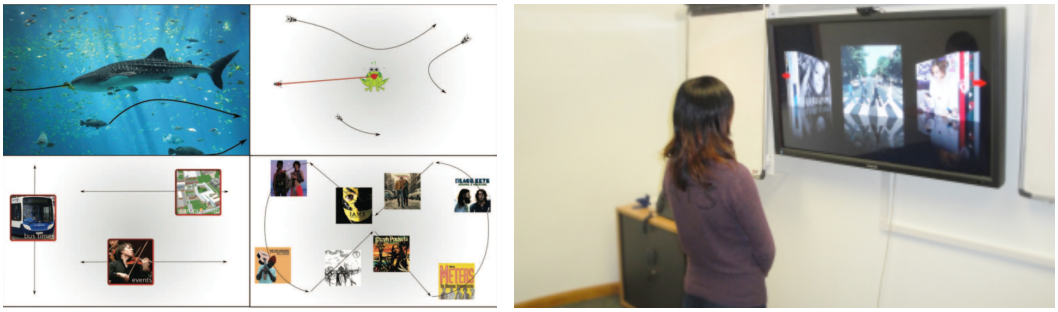
Huang et al. [62] presented PACE, a personalized, auto-calibrating eye tracking system. The idea is to sample calibration data in an unobtrusive way, i.e. invisibly for the user. They use the events occurring when interacting with a desktop computer in combination with an off-the-shelf webcam for face and eye tracking. For instance, they assume the user is looking at a specific user interface element when clicking on it. Similarly dragging items or typing on the keyboard are handled as separate interaction events, which are used as calibration samples. Compared to state-of-the-art

remote eye trackers, they achieved an error of 2.56°. Although existing approaches help to improve the calibration procedure, mainly remote eye tracking systems are used. The principle of pursuit calibration may be transferred to head-mounted eye tracking devices. But the problem that the device is only calibrated onto a specific display still remains unresolved.

A different approach for head-mounted devices is to make use of visual saliency maps to auto-calibrate the eye gaze tracker [26, 184]. In images, the term saliency refers to the occurrence of unique and distinct features in a specific region. Koch and Ullman proposed the initial concept of visual saliency [87]. The basic idea is to predict the gaze point of a person when the saliency map, for example of an image, is known. Combining video and EOG-based devices with saliency maps computed on the scene videos enables one to (re)-calibrate a head-mounted eye tracker [183] (6° of error in the best case). Although these works reduce the calibration drift through an automatic calibration, they create other issues elsewhere, such as cumbersome setups (cameras plus EOG) and inaccuracy. Recently Santini et al. [164] developed an approach for unsupervised calibration of a head-mounted eye tracker. They turn the user's mobile phone into a calibration target that is moved around while fixating it, to recalibrate the system. With this method highly accurate results can be achieved (0.59° of gaze estimation error). But the approach is limited, as the recalibration has to be triggered manually. Hence the user has to know that the accuracy is getting worse and be aware of the calibration drift effect.

Moreover, there are completely calibration-free approaches. Pursuits, developed by Vidal et al. [117], is a framework that allows spontaneous interaction with displays without the need for user-specific calibration. Elements used as a trigger for target selections move on unique trajectories across a graphical user interface. Correlating the eye movements with the known trajectories, input can be detected. It is an alternative way to interact with a gaze-enabled system instead of the previously mentioned dwell-time approach. Another technique was developed by Zhang et al. [214], called SideWays. A webcam mounted on top of a display is used to extract eye movements based on face tracking and eye corner detection. In this way, a calibration-free gaze-based interface can be created that enables users to do high level-input.

However, the existing approaches are either limited to interaction along a horizontal axis (Figure 3.12b), or require dynamic interfaces with moving targets that follow



(a) (b)

Figure 3.12: Two examples for calibration-free gaze interaction: a) Pursuits, utilizing the pursuit movements of the eye [117], and b) SideWays, realizing gaze input via horizontal eye movements [214]

different trajectories (Figure 3.12a). Non-interpolation-based approaches that are also user-calibration-free are only able to achieve coarse gaze approximation [213], as discussed in Section 2.2.

3.3 Corneal Imaging

In this section we will introduce a different approach for eye and gaze tracking, so-called corneal imaging. This method utilizes the fact that the human cornea has mirror-like characteristics. The visible parts of the human eye are the white sclera, the iris, and the black pupil, the latter two of which are surrounded by the cornea (cf. Section 2.1). The corneal surface is covered by the tear fluid, which turns the cornea into a highly reflective surface with mirror-like characteristics (shown in Figure 3.13).

Different research areas took advantage of the specular reflections on the eye. Backes et al. [3] revealed that display reflections could be used to access sensitive data (e.g., passwords) using a telescope. Nishino and Nayar [136, 137] pioneered corneal reflection analysis, giving an overview of what information the reflected image of an eye reveals. They developed the corneal catadioptric imaging system using a geometric model of the cornea. The derived system can be used for several applications like facial reconstruction and relighting [135], face recognition [134] and the calibration of display-camera setups [139]. Besides applications in the field of computer vision [140], several approaches utilize the reflection of the eye to enable gaze estimation [141]. Schnieders et al. [166] used a remote camera to detect a display in the reflected eye image using its geometric properties (e.g., curved edges in the reflection). Nakazawa et al. [127] used infrared light to create patterns in the surroundings, visible in the

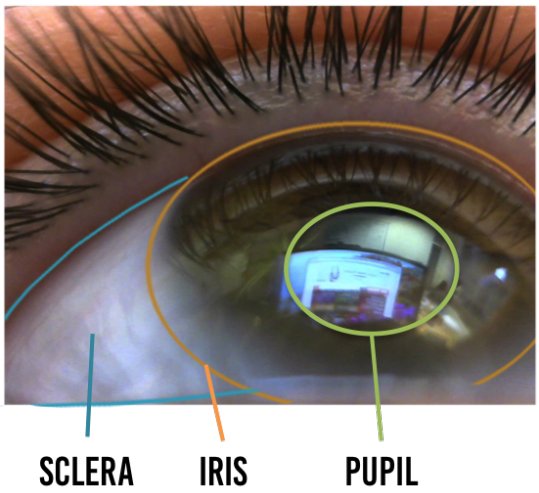


Figure 3.13: Example image of a corneal reflection of a computer monitor.

reflection image, to map the user's gaze into the environment. Nitschke et al. [138] further improved the method by omitting the active illumination. However, these approaches limited the user's mobility by using a remote camera. They achieved highly accurate gaze estimation ($< 1°$ error), but in a highly constrained setting while sitting in front of a large screen. Moreover, additional components, like active illumination or a 3D geometric eye model, were required.

Nakazawa et al. [128] used a head-mounted device to capture corneal reflections and achieved an average gaze estimation accuracy between 2.51° and 4.65° in a highly constrained static desktop setting, not competitive with current commercial remote desktop eye tracking systems. Takemura et al. [187, 188] developed a mobile prototype to estimate the object a user is focusing on. They utilized natural feature tracking to detect objects visible in the corneal images. Both systems are based on 3D eye pose estimation and geometric modeling of the eye, making them computationally expensive. Their current implementations achieve only 7.3 and 1 fps, respectively. In addition, these works lack a detailed evaluation of the gaze estimation accuracy in realistic settings. Their two-camera prototype uses a corneal and a scene camera, and thus relies on a user-specific calibration. Consequently, they cannot be used for gaze based interaction in pervasive settings.

El Hafi et al. [50] developed a mobile corneal imaging system based on high-resolution cameras (4k) capturing both eyes. They show the possibility of reliably detecting objects in the corneal reflections. In their experiment they evaluated the gaze estimation

	remote approaches				head-mounted approaches			
	[136, 137]	[166]	[127]	[138]	[128]	[187]	[188]	[50]
avg accuracy	N/A	15.14 mm to 25.13 mm	<1°		2.51°	N/A	9.5°	2.5°
3D pose estimation	•	•	•	•	•	•	•	•
calibration	•	○	•	○	○	•	○	•
illumination	none	active display	multiple IR LEDs	none	2 LEDs	none	none	4 IR LEDs
cameras	one	one binocular	two scene + eye	one	one	two monocular	one	two binocular

Table 3.3: An overview of existing gaze estimation approaches based on corneal imaging. This shows the main differences between remote and head-mounted techniques, (• fulfilled, ○ not fulfilled).

in a desktop setting. Participants were 500 mm away from a small screen (350 x 300 mm) while looking at targets shown at three different positions around the screen center. With their approach, they only achieved an average error of 2.5°. Moreover, their installed 4K cameras can only deliver images at 11 fps, slowing down the processing speed. Table 3.3 highlights the primary differences of the aforementioned approaches with respect to five important properties: gaze estimation accuracy, required 3D eye model, calibration, illumination and number of cameras. For the sake of completeness, we also included the remote techniques.

Here it becomes clear that only two approaches are able to achieve highly accurate results for gaze estimation (i.e. $< 1°$). Both are implemented for a remote system, for which one needs a user-dependent calibration. The illumination as well as the number of cameras is mainly dependent on the approach itself, i.e. the algorithm used to extract the pupil or the cornea, respectively. It is noteworthy that all proposed techniques rely on a 3D pose estimation of the eye. The reason for this is the following: Basically we can classify all procedures as geometric-based gaze estimation methods. In corneal imaging, the eye images are usually captured via an RGB camera (i.e., natural images), in contrast to the well-known PCCR method that is based on infrared images (cf. Section 2.2). The captured images reveal close-up images of the

eye, making appearance-based methods difficult – if not impossible – to apply. Although the eye does not need to be extracted from the face image, the tracking of the pupil and its movements is challenging. For this, most of the presented approaches rely on limbus tracking (i.e. the iris boundary). The position of the limbus is mapped onto an appropriate 3D model of the eye to compute the pupil position. With that information, it is possible to compute the different axes of the eye used for gaze estimation (cf. Section 2.1).

Up to now, researchers have only investigated how to improve the existing eye models and the limbus extraction to increase the gaze estimation. So far, little attention has been given to alternative methods based on other hardware setups without 3D pose estimation.

3.4 Summary

In this chapter we presented the existing research as well as the current state of eye tracking. We showed how the works serve as a starting point for this thesis, and that there is a need for further investigations provided through our work.

First, we started with an overview about the usage of eye tracking systems in general. We presented state-of-the-art remote and head-mounted technologies. Although remote eye tracking devices are technologically advanced and can also be used by non-experts, these devices are restricted for mostly desktop applications (cf. 1.1, complexity $1:1:1$). We aim for the ubiquitous usage of eye tracking to enable pervasive gaze-based interaction. The group of head-mounted eye tracking glasses comprise the preferred class of devices, as discussed at the end of Section 3.1.1.

In Sections 3.2 and 3.3, we presented works related to the three areas of efficient long-term, location, orientation & target independent usage, and mobile & accurate calibration-free eye tracking. The link between related work and these four areas is as follows:

- *Efficient Long-Term Usage* of a head-mounted eye tracker: In this chapter we presented a detailed discussion about the calibration of head-mounted eye trackers as well as existing approaches to re-calibrate the system in Section 3.2. We learned that all existing strategies can re-establish the initial accuracy, but have several limitations. We investigate two novel methods, extending existing liter-

ature, in Chapter 4 (contribution of paper [95]). There we demonstrate how to efficiently re-calibrate a head-mounted eye tracker, i.e. reducing the calibration time by up to 90%, while recovering 95% of the initial gaze estimation accuracy again. We discuss how this approach can be applied to the reference system to enhance the usability and the user experience.

- *Location, Orientation & Target Independent* head-mounted eye tracking: We also presented related work on gaze estimation on displays using head-mounted eye tracking. Basically in mobile settings with multiple displays (which can be also seen as any kind of surface) the requirement to calibrate to a fixed calibration plane is a serious issue. In Chapter 5 we extend existing literature by presenting a possible solution for this problem. In the evaluation of our system we show that we are able to achieve better results than existing approaches while reducing the need for specific hardware setups (contributed in [93]). In addition, we demonstrate how this concept can be applied it in an in-the-wild study to automatically map a person's gaze into the environment (contribution of [102]).

- *Mobile & Accurate Calibration-Free* eye tracking: A major section of this chapter was dedicated to corneal imaging. We discussed its potential as an alternative method for calibration-free gaze estimation. We extend existing literature by two approaches that we designed and implemented in an iterative process (Chapter 6). With the first technique, we present a new method for extracting the corneal images without the need for 3D pose estimation. We used this information for gaze approximation on displays (contribution of paper [94]), object detection (contribution of [97]) and attention measuring (contribution of [96]). We further improved the method to realize accurate gaze estimation and designed a mobile framework for spontaneous usage in ubiquitous scenarios (contribution of [98]).

In related work Section 3.1.1 on state-of-the-art head-mounted devices we discussed the working method of remote and head-mounted eye tracking devices. We discussed the differences between both approaches, and conclude that the head-mounted approach is most promising ubiquitous settings (cf. Figure 3.14, complexity (K x L) x M x N):

In Section 3.1.2, we presented the whole spectrum of using eye and gaze trackers for various human computer interaction tasks. The developed concepts can be applied for remote and head-mounted devices as well. We learned that gaze is often used in com-

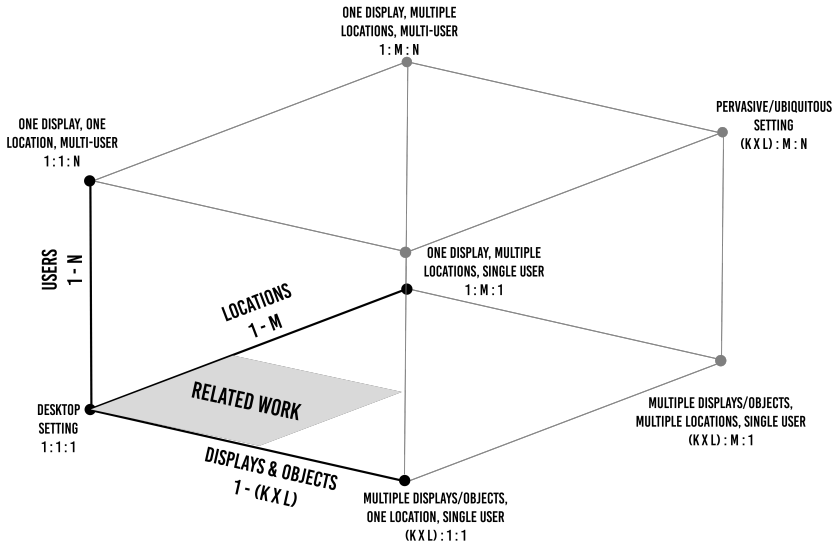


Figure 3.14: *Ubiquitous Computing for Eye Tracking Continuum*: Classification of related work into a 2-dimensional area with varying locations and displays/objects.

bination with other input modalities to counteract precision and accuracy problems. Whereas all the presented works use the information about a person's eye movements to create gaze-enabled interfaces, they miss one important aspect: a discussion on the feasibility of the actual method in a real-world scenario on the basis of the above three areas. All presented methods can be classified used our *Ubiquitous Computing for Eye Tracking Continuum*. Figure 3.14 depicts the classification to an 2-dimensional area with varying number of locations and display/objects. Basically, the existing systems can implement eye tracking on multiple displays/objects at different locations in single user scenarios to a certain extent. However, there exist no approaches that matches the more complex cases of the continuum.

What is required to establish gaze-enabled interfaces at pervasive scale and outside of a controlled environment is an eye tracking device that reduces the omnipresent issues of calibration, with all its negative side-effects, as presented in Chapter 1.4. Khamis et al. [81] discussed three challenges – calibration, user positioning and multi-user interaction – in the case of gaze-enabled public displays. They concluded that there is no approach that addresses all existing issues. With the rest of this thesis, we will contribute by providing approaches to make head-mounted eye tracking usable anywhere and anytime. The developed principles of the gaze-enabled interfaces, presented in Section 3.1.1, are then applicable to create pervasive gaze-based interfaces according to the vision by Bulling et al. [19].

In the following four chapters we present the work, based on the foundations of Chapter 2, that extends existing work presented in this chapter. With our work we contribute to the goal to make head-mounted eye tracking ubiquitous and go beyond the common knowledge from the literature, as we have outlined.

Chapter 4

Investigating the Re-Calibration of Head-Mounted Eye Trackers

In this chapter we will elaborate upon the issues of the so-called calibration drift that occurs with head-mounted eye trackers. We introduced the problem in Chapter 1 (in Section 1.4). We will present a comprehensive evaluation, conducted to investigate the long-term gaze estimation accuracy of a head-mounted eye tracking device while simulating different sources of calibration drift. According to this, we propose four re-calibration strategies to counteract these problems while reducing the required time to re-establish the initial gaze estimation accuracy. Finally we will show how these findings can be applied to the scenario presented in Section 1.5 and enable for eye and gaze tracking settings with multiple users (see *Ubiquitous Computing for Eye Tracking Continuum*, Figure 4.1).

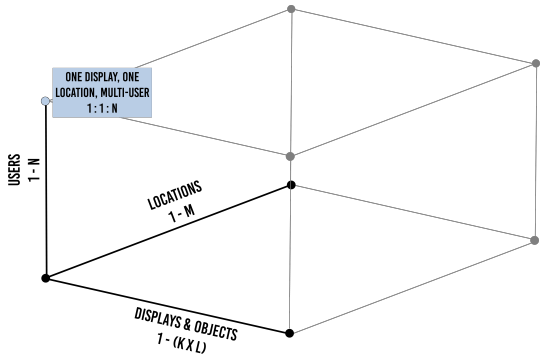


Figure 4.1: *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space highlighting the complexity (1 : 1 : N) that is addressed by this chapter.

The results of this chapter led to one publication [95] and contributed to [91]. Related work that belongs to this chapter can be found in Sections 3.2 on calibration of head-mounted eye tracking systems and in particular in 3.2.2 on re-calibration strategies.

4.1 Introduction

Gaze-based interfaces have been studied for the last two decades. Head-mounted as well as remote eye tracking systems are used to determine people's gaze on a surface, for example a display (outlined in Section 3.1.1). However, most state-of-the-art systems need a calibration procedure before they can be used (discussed in Sections 3.2 and 1.5). When using a head-mounted eye tracking device, many factors can influence the stability of the calibration. These include changes in the eye physiology, in the environment (e.g., light conditions) and in the positions of the cameras attached to the glasses frame. Some of these changes are not controllable, but all lead to less accuracy over time and are denoted correctly as calibration drift. The straightforward approach to counteract these effects is to re-calibrate the eye tracking system at certain time intervals, which is cumbersome and time consuming. Many works exist that investigate different methods for how to re-calibrate the eye tracker (discussed in Section 3.2.2). Each of these approaches has at least one issue that makes it inappropriate to be used in settings, in which real-time eye tracking is required (e.g., interactive scenarios). We will provide methods that resolve the existing drawbacks of these approaches, in particular for head-mounted eye tracking devices.

The chapter is structured as follows. First, in Section 4.2.2, we present our novel time-efficient re-calibration approach. In particular, we address the calibration drift for video- and interpolation-based, head-mounted eye tracking systems. In these systems, the calibration procedure requires the user to look at several visual stimuli, presented, most likely, on a display. During this phase, a mapping function is estimated, that is needed to translate eye into gaze positions. Basically, we are using a subset of the initial calibration points to re-calibrate the system after the occurrence of a calibration drift to estimate an updated mapping function. To achieve this we will present four different adaptation methods. Each one can be used to compute the changes of the stored calibration needed to adapt the gaze estimation accordingly. Second, in Section 4.3.1, we will present the experiment, we conducted to investigate the effectiveness of the different re-calibration strategies. In a controlled laboratory experiment with 16 participants we compared the different adaptation methods with respect to gaze estimation accuracy on a desktop screen. In order to assess the effec-

tiveness, we simulated two kinds of calibration drift: changing the distance between the user and the display, and taking off and putting on the eye tracking device. We were able to show, that we can save up to 90% of the time compared to the required time for a full calibration and are able to reduce the gaze estimation error to 0.5° compared to the initial accuracy. After that, we discuss the findings of the evaluation in Section 4.4. We will derive rules that allow a proper selection of re-calibration-points and give an indication regarding which adaptation method to use. Finally, in Section 4.5, we will discuss how the approach contributes to the the overall research question of this thesis.

The contributions of this chapter are the following: We discuss a new approach for a fast re-calibration of a head-mounted eye tracking system, which improves the gaze estimation accuracy after the occurrence of a calibration drift. Based on this novel approach, we give an overview on the optimal re-calibration point combinations and adaptation methods leading to the best possible improvement of gaze estimation in a display scenario. According to the results of an in-depth evaluation of the developed approach, we define two guidelines to derive the optimal arrangement of calibration-points. Related work can be found in Sections 3.1.1 and 3.2.

4.2 Studying the Long-Term Usage of Head-Mounted Eye Trackers

The most prominent head-mounted eye tracking devices are based on the Pupil Centre Corneal Reflection (PCCR) technique and are usually equipped with two types of cameras (e.g., Pupil Labs[1], Tobii Glasses2[2]). One or more cameras are typically used to capture a close-up view of a person's eye to track its pupil and movements (eye cameras). An additional camera records a person's view (scene camera). Already in Section 3.2 we learned that a user-dependent calibration is needed to create a mapping between the pupil positions in the eye and the gaze positions in the scene camera's coordinate systems. This procedure requires a person to fixate on visual stimuli, for instance AR markers that are shown on a display. In Section 3.2.3 we discussed various investigations to enhance the calibration procedure by using an alternative way of data sampling and making it more convenient (especially [150] and [26]). Due to the occurrence of calibration drift, head-mounted eye trackers have to be repeatedly re-calibrated over time. Reasons for calibration drift include changes

[1] https://pupil-labs.com/pupil/
[2] https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/

in the eye physiology (e.g., wetness of the eye), in the environment (e.g., lighting of the room or sunlight) and change of orientation and/or distance to the object on which gaze should be estimated (parallax error). In particular the latter might occur frequently for gaze-enabled interactive multi-user applications (cf. *Ubiquitous Computing for Eye Tracking Continuum*, Figure 1.1, complexity 1 : 1 : N), as illustrated by the Collaborative Newspaper system [100] in Chapter 1.5. There we also discussed that if no further instrumentation or augmentation of the display is done, re-calibration is the only way to correct for lowered gaze estimation accuracy. As a consequence, using state-of-the-art head-mounted eye tracking systems is cumbersome in such interaction scenarios, in particular in long-term setups. Necessary re-calibrations in the sense of executing full calibrations to replace outdated calibration data is thereby time-consuming and distracts from the actual task (e.g., reading).

We present a new approach to address calibration drift. With our method, we periodically invoke a partial re-calibration procedure to counteract the calibration drift that has been established since the last full calibration routine. The re-calibration algorithm of our approach takes far less time than the standard 9-point calibration routine while achieving competitive gaze estimation results. We use a subset of the full calibration points for re-calibrating the system. The information gathered during this re-calibration is then used to establish an updated mapping function with new coefficients that will improve the gaze estimation accuracy after a calibration drift. We used the head-mounted eye tracking framework developed by Pupil Labs [78] as it can easily be extended through its open-source approach.

Existing methods use only one point on a display for re-calibration [176] or require additional input methods and assumptions [61]. Our approach does the error correction in real-time and does not need any further input devices. Moreover, the method is able to handle sudden increases of calibration drift and a varying error across the screen.

In the following, we will describe the calibration procedure (in Section 4.2.1) we are working with and our new time-efficient approach (in Section 4.2.2). Everything is based on the open-source head-mounted eye tracking framework Pupil Labs and the corresponding device.

4.2.1 Eye Tracker Calibration

Head-mounted eye trackers are calibrated, prior to usage, to map 2D pupil positions from the eye to 2D gaze positions in the scene camera's coordinate system (cf. Chapter 3.2.2). To recap, performing a 9-point calibration, data is sampled as follows: Facing the target display covered by the scene camera's field of view, the user fixates a visual marker shown at nine different subsequent on-screen locations $c \in C$, each for a defined time interval. During this time span, the system records calibration point tuples cp, containing the 2D positions of the pupil $(x_{1_{cal}}, y_{1_{cal}})$ and the corresponding visual marker $(x_{2_{cal}}, y_{2_{cal}})$ in their respective camera coordinate systems (depicted in Figure 6.7).

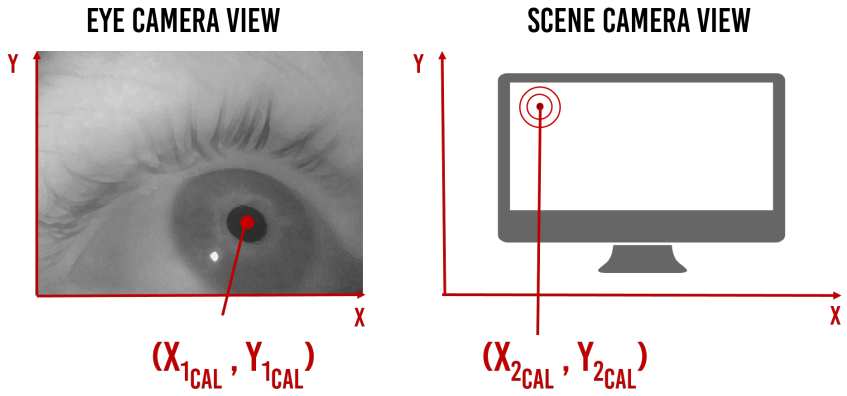


Figure 4.2: Illustration of eye and world view with sampled data of the eye position and the on-screen marker position.

The set of calibration points Cal is used to calculate the coefficients for the mapping function M_{Cal} between the two camera coordinate systems. The coefficients are obtained by solving a linear equation system – one equation per calibration point – aggregating the points in Cal accordingly. Once calculated, M_{Cal} can be changed by modifying the coefficients, leading to different gaze estimations. For more information concerning the calibration procedure, we refer the reader to the first paragraph in Section 3.2.

4.2.2 Time-Efficient Recalibration

Our approach uses a subset of the initial calibration points (C) to reduce the time spent for re-calibration. This minimizes interruption and distraction for the user while preserving the system's accuracy, even for longer-lasting eye tracking sessions. The

set of re-calibration points $R, R \subset C$ delivers new re-calibration tuples rp consisting of pupil and on-screen marker positions for every location $r \in R$:

$$rp_{Recal} = (x_{1_{recal}}, y_{1_{recal}}, x_{2_{recal}}, y_{2_{recal}}).$$

With the set of re-calibration points *Recal*, we compute the offset between the values Cal_{cp}, $cp \in Cal$ recorded during the full calibration and current re-calibration values $Recal_{rp}$ for every re-calibration point $rp \in Recal$:

$$\begin{aligned} offset_{rp} &= Cal_{cp} - Recal_{rp} \\ &= (x_{1_{cal}}, y_{1_{cal}}, x_{2_{cal}}, y_{2_{cal}}) - (x_{1_{recal}}, y_{1_{recal}}, x_{2_{recal}}, y_{2_{recal}}). \end{aligned}$$

The computed offset values $offset_{rp}$ are used to estimate the changes for the values Cal_{cp} of those calibration locations c that were not part of the re-calibration, $c \in (C \setminus R)$. The estimated changes lead to updated values $Cal'_{cp}, \forall cp \in (Cal \setminus Recal)$ that can be used to create an updated mapping function M_{Recal} in combination with the values $Recal_{rp}, \forall rp \in Recal$. In the case that $p \in R$, the values of all calibration point recordings are replaced by the ones obtained during re-calibration; otherwise they are updated based on the shift that was recognized. Hence, the updated mapping function M_{Recal} yields more accurate results than the original mapping function M_{Cal}. We used different methods to estimate the changes in pupil and calibration-marker positions of those calibration points that were not part of the re-calibration:

- Overall Offset (OO) computes a tuple o_{cp} out of the re-calibration points' offset values $offset_{rp}$:

  $$o_{cp} = \frac{\sum_{r \in R} offset_{rp_k}}{|R|}, with\ k \in x_1, y_1, x_2, y_2.$$

  This tuple o_{cp} is added to all tuples Cal_{cp} where $c \in (C \setminus R)$, to obtain the updated calibration values $Cal'_{cp} = Cal_{cp} + o_{cp}$.

- Averaged Offset of Nearest Neighbors (NN) computes a distinct offset for every point $c \in (C \setminus R)$ based on the averaged offsets of the point's nearest neighbors that were part of the re-calibration. The sets of nearest neighbors,

  $$NN_{cp} = \{p \in R \mid d(cp, p)\ is\ minimal\},$$

  are calculated using the Euclidean distance w.r.t. the ratio of the calibration area (NNR) or normalized values (NNN) distance function. Finally, the averaged offset o_{cp},

$$o_{cp} = \frac{\sum_{p \in NN_{cp}} offset_{rp_k}}{|NN_{cp}|}, with\ k \in \{x_1, y_1, x_2, y_2\},$$

is computed for every point $c \in (C \setminus R)$, to update the calibration values $Cal'_{cp} = Cal_{cp} + o_{cp}$.

- Inverse Distance Weighting (IDW) computes a distinct offset for every calibration location $c \in (C \setminus R)$ based on the inverse distance weighting [170]. The overall offset tuple o_{cp} is computed by multiplying a weighting *w(rp, cp, exp)* ($\forall r \in R\ and\ c \in (C \setminus R), exp = 2$) with individual offset values $offset_{rp}$:

$$o_{cp} = \sum_{r \in R} w(rp, cp, exp) \times offset_{rp_k}, with\ k \in \{x_1, y_1, x_2, y_2\}.$$

  The computation for the updated values Cal'_{cp} remains the same as in the previous approaches.

Implementation

Our system consists of two components: (1) a monocular head-mounted Pupil eye tracker connected to a laptop; and (2) a 23-inch display. The eye tracking re-calibration algorithms are written in Python and integrated as a plugin in Pupil's open-source framework running in real time. Thus, the re-calibration methods can be accessed via a graphical user interface. To update and compute the mapping function, the scientific computing package NumPy[3] is used. For gaze estimation on the display, Pupil's integrated marker tracking plugin[4] is used to define and track the display. Therefore, a set of four visual markers on the display corners for tracking the orientation between the eye tracker and the display is used.

4.3 User Evaluation

We conducted a controlled laboratory experiment in order to investigate the effectiveness of our approach as well as to determine the optimal adaption method and selection of re-calibration points in relation to the required time. We thereby focused on two common situations that are known to rapidly increase calibration drift: on the one hand, changes in the distance between user and display and on the other hand, taking off and putting back on the eye tracker. Both circumstances are likely to occur when an eye tracking device is used for a long time.

[3] www.numpy.org
[4] www.pupil-labs.com/blog/2013/12/036-release.html

Participants

We recruited 16 participants (4 female), aged 19 to 58 years (M = 28.5 years, SD = 10.44 years) with a body height between 160 cm and 186 cm (M =174.44 cm, SD = 8.13 cm). Half of the participants reported that they have had prior experience with eye tracking, mainly through participation in other user studies. All of the participants had normal or corrected-to-normal eyesight. To ensure a good tracking result for the pupil, they were asked to participate without using makeup in the area of the tracked eye.

Apparatus

We decided to use the Pupil Labs Pro device[5], a monocular, head-worn eye tracker, as its open source software made it possible to integrate our approach for the evaluation. It furthermore integrates a marker-tracking plugin to define and identify arbitrary surfaces in the environment on which AR markers are attached. We used it to map the participant's estimated gaze onto the display by using visual on-screen markers. The 23 inch monitor used was operated with its native resolution of 1920 x 1080 pixels and equipped with four markers at its corners for the aforementioned tracking plugin. A schematic of our study setup is depicted in Figure 4.3.

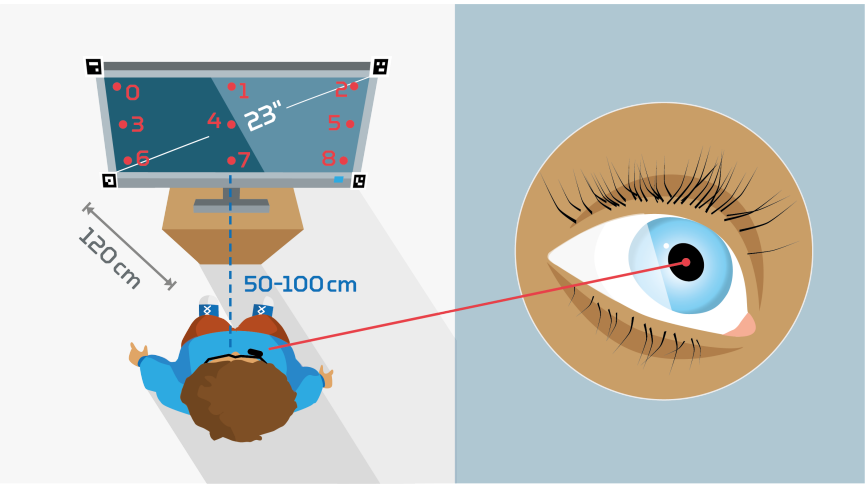


Figure 4.3: Experiment Setup: A 23-inch computer monitor placed on a table at a height of 120 cm. The participant was standing at a distance of 50 or 100 cm in front of the display, wearing a head-mounted eye tracking device.

[5]https://pupil-labs.com/blog/2014-01/new-pupil-pro-headset-capture-software-0-3-7/

Procedure

As outlined before, we focused on two conditions, namely a distance change (50 cm vs. 100 cm, measured from the display) and putting down the eye tracker between measurements (take-off vs. keep-on). In order to test all combinations of the two independent variables, we conducted several experiment blocks, each consisting of two test runs (see Table 4.1). For example, the first test run of the first experiment block was conducted with the short distance, the second one with the long distance, while the eye tracker was taken off between the two runs.

Each test run was divided into two parts: First, the standard 9-point calibration provided by the Pupil software was executed. This enabled us later to evaluate all our proposed re-calibration methods with all possible point combinations in an offline manner. Afterwards, a data recording session for the analysis was conducted. For this, the screen was divided into 24 parts (a grid consisting of 6 columns and 4 rows). In random order, a red dot was displayed in the middle of each of these 24 areas for three seconds and the participants were requested to fixate the points one after another. To increase user comfort during this part of the evaluation, the screen brightness was reduced by using a mid-gray background color.

Experiment block	Distance		Device taken-off
	Test run 1	Test run 2	
1	50 cm	100 cm	●
2	50 cm	100 cm	○
3	100 cm	50 cm	●
4	100 cm	50 cm	○
5	50 cm	50 cm	●
6	50 cm	50 cm	○
7	100 cm	100 cm	●
8	100 cm	100 cm	○

Table 4.1: Design of the experiment blocks and test runs (● device was took off, ○ device was not took off).

In total, 16 test runs (8 blocks with two runs each) were conducted per participant. To reduce fatigue effects, we introduced a break of about 90 seconds between two subsequent test runs. Furthermore, we split the procedure in two sessions with an average duration of 45 minutes each. In order to reduce carry-over effects between different experiment blocks, we counterbalanced the execution order by using a balanced Latin square. The eye tracking data was recorded at 30 Hz, resulting in 90 recorded samples for every presented target in our data recording session. The first 45 samples as well as the last 10 were not considered, in order to account for the time the participants needed to find the target and their tendency to look in advance for the next one towards the end. Based on the remaining 35 frames, the average fixation position was computed.

The collected data consisted of the following: For each block, we got an initial mapping M_{First} for the first full calibration and another one M_{Second} for the subsequent full calibration. Corresponding to each of the mappings, we also obtained a set of 24 test recordings evenly distributed across the screen. Based on a subset R of the calibration points used in M_{Second} and the initial mapping M_{First}, we were able to compute corresponding functions M_{Recal_R} following the approaches presented above. These newly created mapping functions based on a subset of calibration points are those to be evaluated against the ground-truth function, namely M_{Second}. For the evaluation itself, we examined the test data that was recorded after the second full calibration and evaluated it with respect to the average gaze estimation error in degrees of visual angle – first with the mapping function M_{Second} as baseline and then with the newly created mapping functions M_{Recal_R}.

4.3.1 Results

To be able to properly interpret the achieved accuracy values, we first examined the overall gaze estimation accuracy of the test data set based on the corresponding recorded full calibrations as they provide the baseline we want to compete against with our new approach. The gaze estimation accuracy across all participants and study blocks for the initial full calibrations M_{First} was $1.18°(SD = 0.69°)$. For the subsequent full calibrations M_{Second}, an average accuracy of $1.41°$ was achieved. A paired-samples t-test showed a significant difference between the initial calibration $(M = 1.18°, SD = 0.69°)$ and the subsequent calibration $(M = 1.41°, SD = 1.07°)$, $t(127) = 3.08, p < 0.05$. Cohen's effect size value $(d = 0.26)$ suggested a low to moderate practical significance. When evaluating the second test data set using

the initial mapping function M_{First}, the accuracy drops to 4.91° as expected. A paired-samples t-test showed a significant difference between the initial calibration ($M = 4.91°, SD = 4.4°$) and the subsequent calibration ($M = 1.41°, SD = 1.07°$), $t(127) = 9.28; p < 0.05$. Cohen's effect size value ($d = 1.09$) suggested a high practical significance.

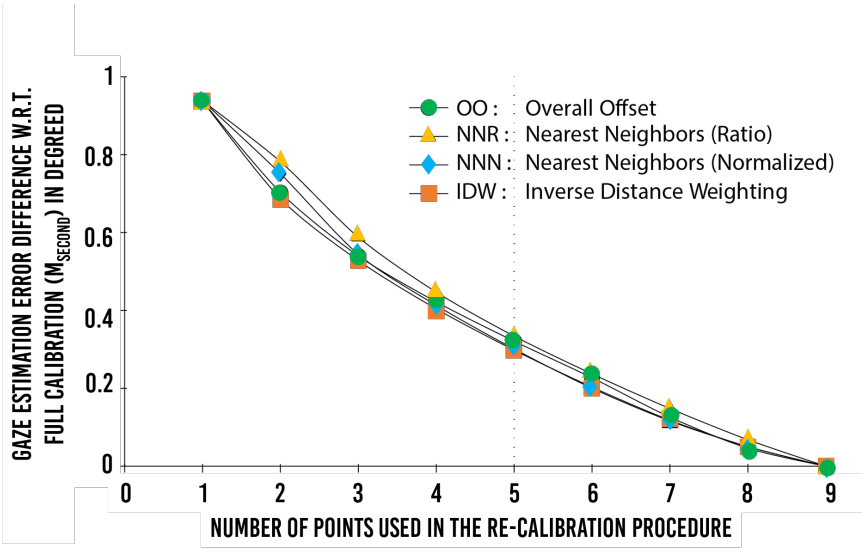


Figure 4.4: Difference in gaze estimation error w.r.t. the full calibration mapping M_{Second}. The results are based on the data recorded during the second test run.

As the presented time-efficient re-calibration approaches can be used with any number of re-calibration points (1 to 9, where 9 equals a full calibration), it is important to check which number of points deliver a good trade-off between time consumption for the re-calibration and resulting accuracy. We therefore examined the relation between number of points used and the difference in gaze estimation error between the resulting re-calibration and the corresponding full calibration as the baseline (shown in Figure 4.4).

We decided to examine only re-calibrations with five points or less. Using more points is not efficient with respect to the trade-off between achieved accuracy and time needed for the re-calibration procedure, as the average improvement in gaze estimation error drops below 0.1° in visual angle. Further analysis of the four different re-calibration methods revealed slight differences (below 0.1° on average) among them. However, for two calibration points, a one-way ANOVA with Greenhouse-Geisser correction applied showed a significant difference between the four methods

Exp. block	1		2		3		4		5		6		7		8	
Mapping	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd
Mean	7.58	1.35	2.73	1	7.92	2.15	3.01	1.25	6.66	1.69	2.41	1.39	7.14	1.3	1.86	1.15
SD	4.98	1.08	2.32	0.78	6.05	2.83	2.2	0.84	3.92	1.56	2.23	1.24	5.75	1.04	1.49	0.8

Table 4.2: Accuracy of mapping functions M_{First} and M_{Second} per experiment block in degrees. Evaluated for the second test data set.

($F(1.234, 156.708) = 7.989, p < 0.05$). A post-hoc analysis based on pair-wise comparisons with Bonferroni correction in effect revealed significant differences ($p < 0.5°$) between all combinations of re-calibration methods, but NNN and OO as well as IDW and OO.

As we previously examined aggregated results across all eight conditions, we now further analyzed the individual experiment blocks. Table 4.2 lists the results for the accuracy of the mapping functions M_{First} and M_{Second} when applied to evaluate the test data of the second test run. The results reveal that the highest error occurs when both the distance was changed and the eye tracker was put down between the test runs. In these cases (blocks 1 and 3), the average error increases to more than 7°. The second highest error occurred when taking down the eye tracker during the break but not changing the distance (blocks 5 and 7, increased by 5.4° on average). Only changing the distance (blocks 2 and 4) introduced an additional error of 1.8°whereas

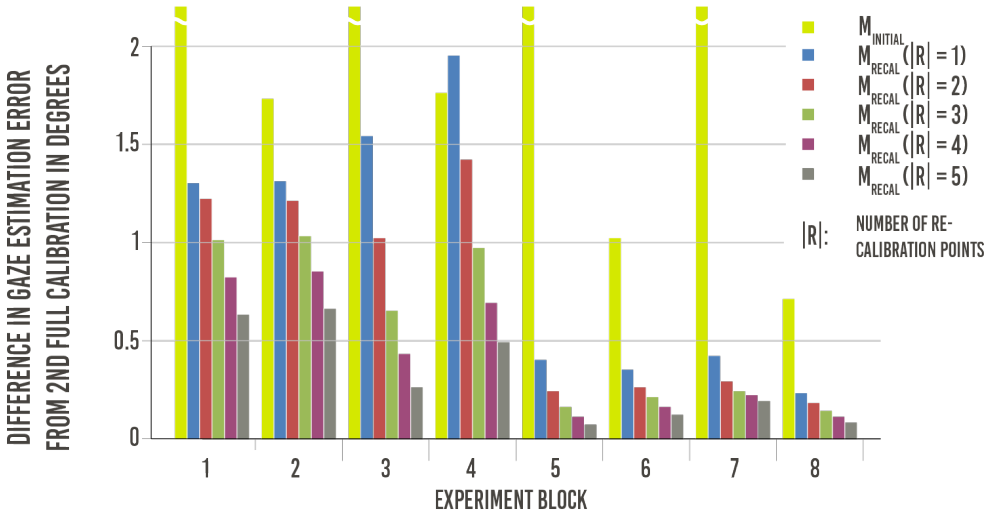


Figure 4.5: Difference in the gaze estimation error w.r.t. M_{Second}. The results are presented per experiment block and number of re-calibration points based on the data recorded during the second test run.

the normal drift over time, without any changes in distance or putting down the device in between test runs results in an additional error of 0.9° (blocks 6 and 8).

Consequently, we investigated how our time-efficient re-calibration algorithm behaves in these situations. Figure 4.5 shows the differences in the gaze estimation errors between re-calibration combinations with the respective amount of re-calibration points $\mid R \mid$, resulting in mapping functions M_{Recal_R}, and the mapping function M_{Second} of the full calibration. To illustrate the improvement compared to the outdated mapping M_{First}, we also illustrated the difference in the gaze estimation error between M_{First} and M_{Second}.

We further analyzed the distribution of points that should be chosen for a re-calibration, i.e. if using $\mid R \mid = 4$ points, which four points should be selected out of the available nine that are used for a standard full calibration (see Figure 4.6). We started by examining the selection for $\mid R \mid = 1$. In this case, our four proposed adaption methods NNN, NNR, IDW and OO do not establish different mapping functions, resulting in only nine different re-calibration settings. A one-way ANOVA with Greenhouse-Geisser correction applied showed a significant difference in the accuracy with respect to the nine single point re-calibrations and the standard full calibration M_{Second} we consider as the baseline ($F(2.664, 39.958) = 42.173, p < 0.05$). A post-hoc analysis based on pair-wise comparisons between M_{Second} and the nine single-point re-calibrations revealed significant differences in eight cases ($p < 0.05$); only the re-calibration based on point 6 did not show a significant difference.

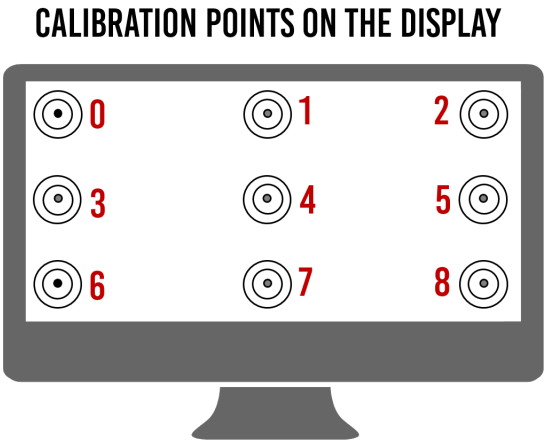


Figure 4.6: The nine points used for recalibration, labeled with their identifiers (0 to 8).

We continued the analysis with the time-efficient re-calibrations with two to five points. We separately looked into the four proposed adaption methods NNN, NNR, IDW and OO. For each of them, four different analyses w.r.t. the number of re-calibration points used were performed. Based on these analyses, we created two sets S_p and NS_p that contain those point selections with p points that result in significantly (S_p) or not significantly (NS_p) different accuracy w.r.t. the baseline M_{Second}. For the point selections in the sets NS_p, the null hypothesis that there is no difference between the means of gaze estimation error cannot be rejected. This means that we were able to create a successful mapping function with the respective re-calibration combination in this special case. However, we cannot make general assumptions about the goodness of this created mapping function.

We continued our examination by checking whether the selection of points for the re-calibration is important. For this, we defined two rating functions to express the calibration area coverage achieved by the selected points. For the case of $\mid R \mid = 2$, i.e. re-calibrations with only two calibration points, we consider the covered axes intercepts based on the following formula:

$$axisIntercept((x_1, y_1), (x_2, y_2)) = \mid x_1 - x_2 \mid + \mid y_1 - y_2 \mid$$

with (x_1, y_1) and (x_2, y_2) indicating the coordinates of two re-calibration points in a normalized coordinate system with an edge length of 1, respectively. Based on this formula, the highest value of 2 can be achieved by selecting two corner points on the diagonal, whereas two adjacent points parallel to one of the axes yield the lowest value of 0.5. A Welch's unequal variances t-test showed a significant difference in the axis intercept values between the groups $NS_2(M = 1.33, SD = 0.49) and S_2(M = 0.95, SD = 0.42)$; $t(20.74) = 3.17, p < 0.05$.

For more than two points ($\mid R \mid > 2$), we consider the area that is covered by the polygon drawn by the selected calibration points. The area of an arbitrary polygon that is not self-crossing can be computed with the following formula:

$$\begin{aligned} area((x_1, y_1), ..., (x_n, y_n)) = 0.5 * (x_1 y_2 - x_2 y_1 + x_2 y_3 \\ -x_3 y_2 + ... + x_{n-1} y_n - x_n y_{n-1} + x_n y_1 - x_1 y_n), \end{aligned}$$

with normalized re-calibration coordinates $(x_1, y_1), ..., (x_n, y_n)$. For all amounts of re-calibration points used (3–5), a Welch's unequal variances t-test showed a significant difference in the covered area (shown in Table 4.3).

$\|R\|$	M(NS)	SD(NS)	M(S)	SD(S)	t-test ($p < 0.05$)
3	0.28	0.15	0.19	0.12	$t(97.06) = 4.49$
4	0.48	0.18	0.37	0.13	$t(273.61) = 7.31$
5	0.62	0.18	0.50	0.13	$t(436.62) = 8.60$

Table 4.3: Results of Welch's unequal variances t-test of the covered polygon area between $NS_{|R|}$ and $S_{|R|}$.

To get a better understanding of point distributions that work well vs. poorly, we illustrated the gaze estimation errors w.r.t. to the 24 screen areas used in the data recording sessions. For space reasons, we show only one example case here. For each target, 128 estimated gaze positions (16 participants × 8 experiment blocks) are depicted by a red circle and connected to the actual target position by a line. Figure 4.7(1) shows the gaze estimations based on the corresponding standard full calibration M_{Second}, 4.7(2) illustrates the results when using the outdated calibration M_{First}, and

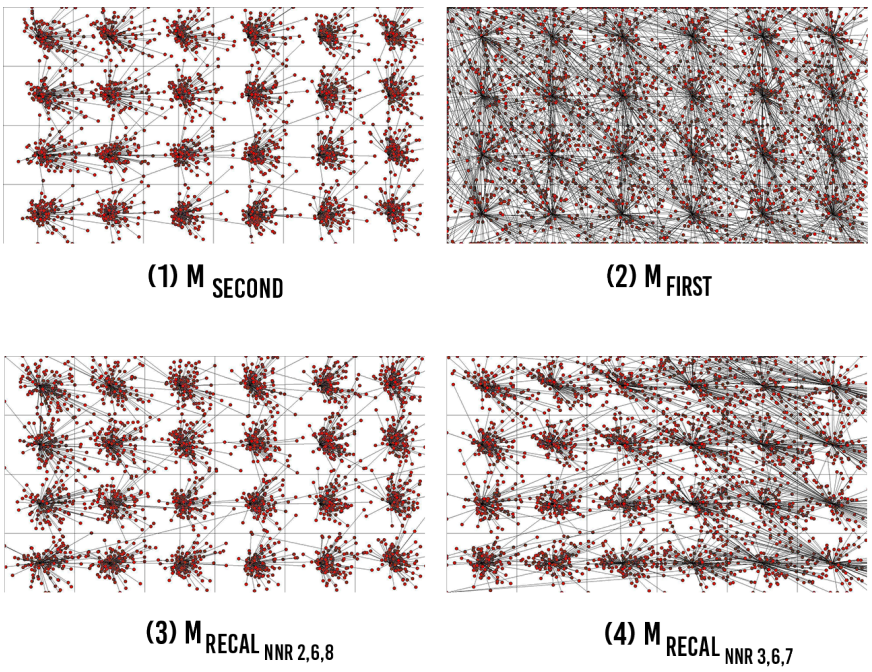


Figure 4.7: Gaze estimations based on different mapping functions of the re-calibration combinations for the data of the second test run. Numbers in subscript indicate the calibration points that were used in this re-calibration.

4.7(3) and 4.7(4) present the results when using the best- and worst-performing time-efficient re-calibration with three points, respectively. The results show that even the worst-performing time-efficient three-point re-calibration yields better results than the out dated first calibration with nine points. However, it can also be seen that carefully choosing the points used in the re-calibration can substantially improve the results. Furthermore, the gaze estimation error in the worst case varies heavily across the screen. As the used calibration points 3, 6 and 7 are all located in the lower left corner, the accuracy in this area is acceptable, whereas it is relatively poor across the rest of the screen.

4.4 Discussion

Considering the accuracy of the full calibrations M_{First} and M_{Second}, we have seen a significant difference between them. As the whole study was conducted within the same environmental conditions and we counterbalanced the experiment conditions, it is highly probable that the major part of the differences originated from participants' fatigue effects. Although we split the experiment into two sessions and introduced breaks after each test run, still 8 x 24 = 192 targets and 8 x 9 = 72 calibration points had to be fixated per experiment session. Consequently, the presence of fatigue despite our countermeasures is not surprising and further emphasizes the necessity to reduce the amount of fixations needed for (re-)calibrations, which is achieved through our proposed time-efficient re-calibration approaches.

The key questions for our approach are (1) how many and (2) which points should be used for a time-efficient re-calibration that again yields an eye-tracking system with high accuracy. As presented above, adding more points to the re-calibration obviously improves the result; however, there is no linear or nearly linear relation, but rather a log-like one (cf. Figure 4.5). Still, the time needed for re-calibration follows a linear trend. As the improvement in the accuracy drops below 0.1° in visual angle per added point when using more than five points, we focused on re-calibrations with five or fewer points, resulting in a possible time savings between 40% and 90% compared to a full calibration.

In our experiment, we focused on two conditions that are known to rapidly increase calibration drift, namely taking the device off and putting it back on, as well as changing the distance between user and tracked screen. We could show that our re-calibration approach is highly effective in counteracting calibration drift that orig-

inated mainly from putting down the eye tracker (experiment blocks 5 and 7). On average, even re-calibrations with only one or two points yield valuable results. They decrease the additional gaze estimation error below 0.5°. However, in cases in which the distance between the user and the tracked display is changed (experiment blocks 1–4), more calibration points may be required to achieve high accuracy values. The outdated full calibration M_{First} adds an additional gaze estimation error of 1° to 1.5°, which might be too much, depending on the use case. Our results suggest to use three or more points for the re-calibration if larger distance changes between the user and the tracked screen are expected.

The previously discussed results already indicate that the time-efficient re-calibration approach with fewer calibration points works in general. However, we have not yet considered the choice of the calibration points. A proper selection is important for the resulting accuracy and might also reduce the necessary number of points further, i.e. a re-calibration with three well-chosen points might yield better results than one with four poorly chosen calibration points. We therefore further examine the point selection process.

Stampe [176] proposed to use the center point, i.e. point 4 according to our naming convention, for re-calibrations with only one point. In contrast to this finding, the re-calibration based on point 6 yielded the best results in our evaluation. However, this could be related to our specific experiment setup, i.e. tracking only the user's right eye, resulting in point 6 having the largest distance to the user's tracked eye. Further investigation is needed to examine whether a correlation between this distance and the resulting accuracy exists. For example, it could be checked whether a similar effect occurs when tracking the left eye, in this case w.r.t. point 8. Also, an investigation of a binocular setup might be of interest for re-calibrations based on one point.

Regarding re-calibrations with two to five points, 372 different calibration point combinations have to be considered. As illustrated above, we can calculate a value that indicates the coverage of the chosen points w.r.t. the screen area. In the case of two points, we consider the axis intercept values; for three or more points, the area of the spanned polygon is considered. Furthermore, we can divide the point selections into two groups: one that leads to mapping functions with significantly different accuracy compared to our baseline M_{Second}, and the other without a significantly different accuracy. The comparison of these two groups w.r.t. axis intercept value or spanned

polygon area revealed that it is better to choose re-calibration combinations with point distributions that maximize the respective value. When considering all re-calibration combinations individually, we see that those resulting in the lowest added gaze estimation error at the same time maximize the axis intercept value or spanned polygon area, which further supports this finding. In contrast, the re-calibration combinations with the highest gaze estimation errors only achieved low values concerning coverage. Furthermore, properly chosen point distributions yield accurate results across the whole screen area, whereas otherwise, parts of the screen show large added errors (cf. Figure 4.7d).

Based on the results from above, we recommend two rules for selecting the appropriate point distribution:

1. Axis intercept values/polygon area should be maximized.
2. Calibration points in the corners should be preferred.

According to these rules, the following sets of points should be chosen for re-calibration in our setting using a nine-point calibration:

#	Combinations	Mean	SD	Gain	Time
2	[0,8], [2,6]	1.62°	1.57°	94.0%	9
3	[0,2,6], [0,2,8], [0,6,8], [2,6,8]	1.52°	1.57°	96.8%	12
4	[0,2,6,8]	1.51°	1.50°	97.1%	15
5	[0,1,2,6,8], [0,2,3,6,8], [0,2,4,6,8], [0,2,5,6,8], [0,2,6,7,8]	1.49°	1.44°	97.7%	18

Table 4.4: Point combinations yielding the best results (mean and standard deviation of the gaze estimation error in degree), chosen based on the two rules defined above. In addition, the gain in accuracy and the time in seconds needed for re-calibration are shown.

The resulting gaze estimation error, averaged across all experiment blocks and participants, was always at most 0.4° worse than the mapping function M_{Second} ($M = 1.41°$, $SD = 1.46°$) which is based on a 9-point full calibration. Table 4.4 states the mean gaze estimation error for the optimal re-calibration point structures. With that, we can take a deeper look into the gain in accuracy when performing a certain re-calibration. Remember that the average gaze estimation accuracy using M_{First} is $4.91°(SD = 4.68°)$. Hence a 100% gain in accuracy is achieved by using M_{Second}, which increases the gaze estimation accuracy by 3.50°on average. For all point combinations, we were able to achieve an increase of more than 90% in gaze estimation

accuracy. In addition the required time for the re-calibration is shown, which results in a savings of 40% to 70% (30 seconds are needed for a full re-calibration).

Apart from the chosen point distribution, the selected adaption method might also be of relevance. For our evaluation, we considered four adaption methods, namely the overall offset approach (OO), the nearest neighbors approach (based on normalized values (NNN) or w.r.t. the calibration area ratio (NNR)) and the one based on inverse distance weighting (IDW). Averaged across all re-calibration point combinations with the same number of calibration points, the comparison of these methods showed that they only deliver significantly different results in the case of two calibration points. In general, our results suggest that OO and IDW are more robust w.r.t. the re-calibration point distribution. This is not surprising when considering the strategy behind the approaches. In the case of OO, all available re-calibration points are taken into consideration to compute the values for the missing points, thereby providing the most global approach. IDW already considers a slightly more local method, as near points are considered to have more impact. Still, all available, new and thereby possibly more accurate information is taken into account. In contrast to that, the nearest neighbor approaches only consider a subset of the available points for computing a missing value. Hence, using an approach that works more globally such as IDW or OO, yields better results, as more information is considered for the computation of each missing point.

Adaptation Method	NNN	NNR	IDW	OO
Gaze Estimation Error (°)	1.59	1.61	1.57	1.60

Table 4.5: Mean gaze estimation error in degrees for each adaptation method when performed with the twelve superior point distributions.

The results of Table 4.5 show that the difference in the gaze estimation error between the four adaption methods drops below 0.05°if properly distributed re-calibration points are used. The inverse distance approach (IDW) performs slightly better than the remaining methods. This result clearly shows that choosing a meaningful point distribution is much more important than the selection of the adaption method.

4.5 Application to research questions

In this section we will elaborate upon the contribution of the time-efficient re-calibration approach to the problems defined in Chapter 1.4 and the research questions defined in Section 1.6. In particular, we will highlight the hypothetical integration of the method into the Collaborative Newspaper application, presented in Chapter 1, to address its practical problems (outlined in Section 1.5).

In our problem statement in the beginning of this thesis, we derived five interrelated issues that prevent the transition of head-mounted eye tracking devices into a ubiquitous computing device. We found the calibration procedure, required prior to the usage of head-mounted eye tracking glasses, to be the main problem, which in fact leads to many other issues. In this chapter we pursued the particular sub-problem of calibration drift. Recall that this effect causes head-mounted eye trackers to need to be repeatedly re-calibrated. In Section 4.2.2 of this chapter, we proposed a new approach to tackle this problem. The developed method reduces the time needed for updating the existing calibration. Note that we do not actively counteract the different types of calibration drift. Changes in the eye physiology, the environment and moving users will still cause negative impacts on gaze estimation accuracy over time. Applying the time-efficient re-calibration approach is a faster way to re-establish the initial gaze estimation accuracy.

However, the work presented in this chapter is an important component of the answer to our formulated research question (see in Section 1.6). After the detailed presentation of our four approaches in Section 4.2.2, as well as the extensive investigation in Sections 4.3.1 and 4.4, we can formulate the answers to the following sub-questions.

2. What is the long-term accuracy of current head-mounted eye trackers?
3. How can we efficiently re-calibrate head-mounted eye trackers and recover the initial gaze estimation accuracy?

In order to effectively study the developed approaches for re-calibration, we let several participants perform a simple gaze pointing task in a single display scenario. We focused on two conditions that increase calibration drift, i.e. removing and replacing the head-mounted device as well as changing the distance between the participant and the target display. On average, the complete experiment took about 90 minutes, split up into two equal sessions. We discovered that the gaze estimation accuracy differed significantly between different test runs (cf. Section 4.3.1 for details about the

study). As we conducted the experiment in the same laboratory and counterbalanced the study conditions we can conclude that the different gaze estimation accuracy originated mainly from the users' fatigue effects. If we consider that the participants had to fixate 240 targets and 90 calibration points per session, the occurrence of fatigue cannot completely be ruled out.

We can now argue definitely more about the long-term accuracy of head-mounted eye trackers. Besides the usual occurrence of different sources of calibration drift, fatigue effects of the individual user have a major impact on gaze estimation accuracy. This may be dependent on the actual task of the user. In our case, people had to perform many fixations on a display. However, by incorporating pauses within the experiment as well, this effect was significant. The fact that the complete study lasted for about 90 minutes per participant indicates that the gaze estimation accuracy will get even worse during longer use of head-mounted eye tracking systems. Finally, we cannot argue about the actual long-term accuracy as a discrete value, but rather we showed that it deteriorates due to the aforementioned reasons. The logical consequence is to reduce the number of fixations required for (re-)calibrations, which leads us to the next question.

We proposed four different methods for re-calibrating a head-mounted eye tracker that is based on the PCCR method for eye tracking and implements interpolation-based gaze estimation. Regardless of the approach – Overall Offset, Averaged Offset of Nearest Neighbors (two versions) and Inverse Distance Weighting – it is possible to achieve a time savings of 40-90% compared to a full calibration with nine points. There is obviously a trade-off between saved time and the updated gaze estimation accuracy, i.e. the better the re-calibration should be in terms of accuracy, the more points have to be used. In fact, a maximum of five points (for a nine-point calibration) recovers the eye tracker's accuracy while still being efficient concerning the required time. However, we learned that the number as well as the distribution of re-calibration points is more important than the actual method.

After answering the research questions, we also want to illustrate the advantages of our developed approaches in an interactive scenario. For this, we want to draw the reader's attention back to the Collaborative Newspaper application that we presented in Section 1.5. A main requirement for the system to work is a steadily high-accuracy gaze estimation. This application use case constitutes a prime example for a long-

term setting that would suffer from disruptive re-calibrations. If a user is reading multiple texts, many situations may occur that cause a calibration drift. For example, for ergonomic reasons, the user may change the position of the head-mounted device. Besides that, the lighting conditions may change, or the task of reading text on a display may cause fatigue effects. Although all these possibilities are hypothetical, the proposed time-efficient re-calibration strategies will definitely counteract the interruption caused by re-calibration. Consequently, the people currently using the system can continue with reading more quickly.

4.6 Summary

Motivated by the increasing use of video- and interpolation-based eye tracking systems and their problems with calibration drift, we developed a new time-efficient re-calibration algorithm to counteract the occurrence of calibration drift. Instead of using the time-consuming standard calibration procedures, which are tedious and cumbersome for the user, we use a re-calibration method that relies on a subset of the calibration points. In contrast to existing approaches, our time-efficient re-calibration method works in live settings, improves accuracy compared to the original calibration, and does not need any additional input devices. Furthermore, it can handle sudden increases of calibration drift, and considers the varying error across the screen.

We conducted an experiment in which we compared the achieved gaze estimation accuracy when re-calibrating with our method against a full calibration and the original calibration after the occurrence of two different drift events. Our results show that our approach compares well to the other possibilities. We were able to achieve accurate results concerning gaze estimation accuracy while considerably reducing the time needed for re-calibration. With this approach, the usage of head-worn eye tracking devices can be improved, especially in long-term settings that do not permit time-consuming and thereby disruptive re-calibrations.

With this chapter we answered two sub-questions of the formulated research question in Section 1.6, investigating the problem of calibration drift and the necessary re-calibration (cf. Sections 1.4 and 1.5). We are now able to address the calibration drift in terms of reducing the time required for recovering the initial gaze estimation accuracy of a head-mounted eye tracking system. The presented approach for re-calibration may thus improve eye tracking in multi-user scenarios, as defined in our continuum (cf. Figure 4.1, complexity $1 : 1 : N$). However, we still have to calibrate

on the surface we want to estimate gaze on and have no ability to counteract the calibration drift itself. In the next Chapter, we will propose a system that further investigates the aforementioned problems and present answers to more questions that are the subject of this thesis.

Chapter 5

Spontaneous Gaze Estimation

In this chapter we will elaborate on the problem, that the calibration should remain invariant despite location and orientation changes, as defined in Section 1.4. Head-mounted eye trackers are usually calibrated from a fixed position and orientation between the user and the object that gaze should be estimated on (i.e. the target object). Oftentimes a display is used to show the calibration procedure. If changing the distance and/or orientation to the target object, or if gaze should be estimated on a different surface, a re-calibration is needed. We will present an approach for accurate and seamless gaze estimation across multiple displays. Note that displays can be seen as a placeholder for various other objects. We will outline, how the approach contributes in making head-mounted eye tracking devices more ubiquitous. Finally, we will present a way of accurate gaze estimation on arbitrary objects in the environment enabled by the proposed technology. Thus it enables complex eye tracking settings with multiple displays and multiple locations (cf. *Ubiquitous Computing for Eye Tracking Continuum*, Figure 5.1).

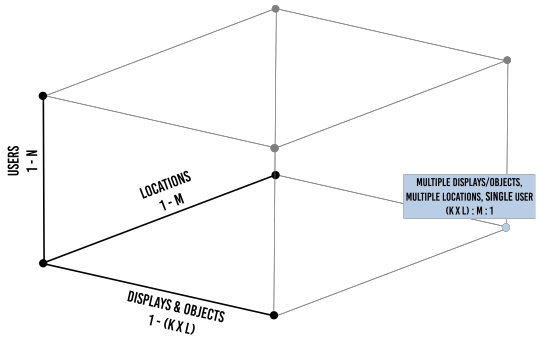


Figure 5.1: *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space highlighting the complexity ((K x L) : M : 1) that is addressed by this chapter.

The results of this Chapter have been presented in one main publication [93] and contributed to two further publications [91, 102]. Work related to this chapter can be found in Section 3.1.1 on gaze-based human computer interaction and Section 4.2.1 on eye tracker calibration.

5.1 Introduction

Gaze is a powerful modality for interacting with the rapidly increasing number of public displays around us. Consequently, gaze-based interaction has received considerable attention with a large range of applications (e.g., desktop control [71, 211], eye typing [111], selection [180]). First prototypes used desktop-like settings and stationary eye trackers in which a user's head was fixed (regarding position and orientation). To enable pervasive gaze-based interaction with situated displays in everyday settings [19], eye tracking still faces the following key challenge: how can spontaneous and transparent gaze-based interaction be facilitated on arbitrary objects and displays?

This chapter is structured as follows. First, in Section 5.2, we describe a novel way of realizing accurate gaze estimation across multiple displays. Specifically, we propose the concept of *GazeProjector*, a system that allows for accurate gaze estimation on arbitrary displays independently of the user's position and orientation. It only requires a one-time calibration of the head-mounted eye tracker per user. The procedure can be performed on any display and not necessarily on the one the user wants to interact with. Afterwards, the system automatically transforms this calibration (i.e., the calibration plane) to other displays in real time. The basic idea is that the eye tracker continuously tracks itself relative to different displays using natural feature tracking.

Second, in Section 5.3, we will present the evaluation in which we compared our approach to a state-of-the-art OptiTrack[1] motion capturing system as well as an AR marker-based approach. In a controlled laboratory experiment with 12 participants, we investigated the gaze estimation accuracy in two different scenarios. In the first task, participants looked at on-screen targets from various positions and orientations in front of a large projected screen. In a second task, we compared *GazeProjector* to the AR marker-based approach on multiple displays. In both tasks, we found that our approach compensated well for head movements (i.e., change of orientation) and user relocation (i.e., change of location). After that, we will discuss the main findings, benefits and limitations of the approach in Section 5.4.

[1] http://optitrack.com/

We also outline some application use cases. In Section 5.6 we are going to elaborate on the contribution of the system to the overall research question of this thesis. Finally, in Section 5.6, we implement a system to allow for gaze estimation in real environments enabled by *GazeProjector*. With this, researchers are able to conduct in-the-wild studies and gain novel insights on pervasive gaze tracking that can be used to analyze people's visual behavior. The contribution of this chapter is a novel method that enables:

- **Location- and orientation-independent** gaze estimation on a display (of varying size/resolution) without requiring any instrumentation in the environment.
- **Accurate gaze estimation on arbitrary displays** in the environment without requiring recalibrating the system for each display independently.
- **Maintaining high gaze estimation accuracy** without requiring recalibrating the system for varying positions, orientations and displays.

We implemented the concepts of this method to create a new system, called *GazeProjector*, to enable both application developers (who wish to employ gaze as an additional input modality) as well as researchers (who wish to study a user's gaze) to apply eye tracking in fully pervasive settings, as defined in Section 1.1 by the *Ubiquitous Computing for Eye Tracking Continuum* (cf. Figure 1.1), without the need for augmenting the environment. Summing up, we provide more answers to the overarching question of how to use head-mounted eye trackers by addressing the problem of calibration and its invariance.

5.2 GazeProjector – Accurate Gaze Estimation and Seamless Gaze Interaction Across Multiple Displays

Mobile gaze-based interaction with multiple displays may occur from arbitrary positions and orientations. However, maintaining high gaze estimation accuracy still represents a significant challenge. We already learned in Chapter 3 (Section 3.1.1), that the latest head-mounted eye tracking devices are commonly equipped with two cameras: (1) a scene camera partly capturing a user's field of view, and (2) an eye camera recording a close-up video of the user's pupil position. Such eye trackers have to be calibrated to a *specific* user for a *specific* display before first use to establish a mapping between the pupil's 2D positions in each of the cameras' coordinate systems.

The calibration is typically performed for a *fixed* position and orientation of the user to a *single* display the user intends to interact with. While this is less of an issue for stationary settings and TV-sized displays, mobile settings and multiple – potentially large – displays evoke two types of motion: (1) user movements in front of a single display to inspect other parts of the display's content, and (2) head movements to reach targets outside the ocular motor range [49]. In addition, there might be multiple displays present, causing further movements. Both types of motion considerably reduce gaze estimation accuracy [23].

The straightforward solution is a re-calibration of the eye tracker. But this is not feasible in scenarios with frequent movements and/or interactive scenarios, as discussed in Chapter 4. In order to achieve high gaze estimation accuracy a more practical option is to track the user's (and eye tracker's, respectively) position and orientation relative to a display. Solutions that realize this include the augmentation of the environment with visual markers as well as using vision-based motion capturing systems. Although these approaches provide high tracking accuracy, they are impractical for spontaneous gaze interaction, particularly when interacting with multiple displays in mobile everyday-life scenarios. Similarly, trackers not relying on visual markers, such as a depth camera system, work well for tracking the rough user position, but are not accurate enough to track head orientation, and fail when people occlude each other. In the Collaborative Newspaper system [101] (cf. Section 1.5) visual on-screen mark-

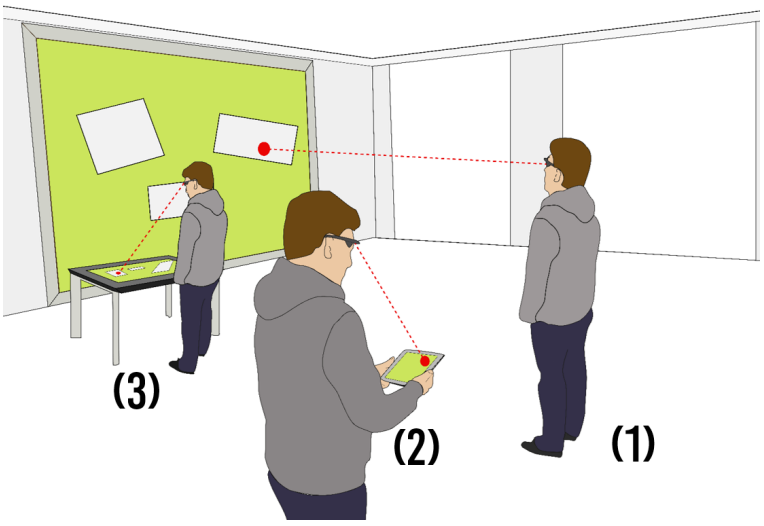


Figure 5.2: GazeProjector enables seamless gaze-based interaction with multiple displays from arbitrary locations and orientations, such as wall-sized displays (1), horizontal screens (2), and handheld devices (3) without active recalibration.

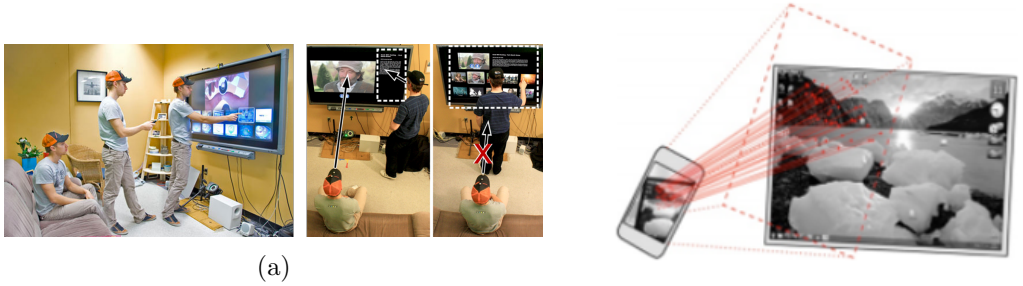
ers were used and required up to one third of the available screen space to enable a reliable tracking and identification of the display.

In order to allow accurate gaze estimation on arbitrary displays placed across different locations, we present a new method to make the initial eye tracker calibration invariant with respect to the user's position and orientation to the displays in the environment. The concepts were implemented in a system called *GazeProjector* (see Figure 5.2). The head-mounted eye tracker continuously tracks its position and orientation relative to different displays using natural feature tracking on the scene camera's video stream. To do so, displays continuously stream their (potentially dynamic) content to a server, which performs the feature matching. Thus, *GazeProjector* neither requires a motion capturing system nor visual markers pre-installed in the environment. After a one-time calibration with an arbitrary display, *GazeProjector* is able to transform pupil positions onto any connected display in the environment, as long as a part of that display is visible to the eye tracker's scene camera. As the calibration is independent of a potential target display, our system allows for accurate gaze estimation and seamless interaction across multiple displays, and thus empowers users to freely move around within the environment.

The developed system builds on methods for gaze approximation and estimation on displays (presented in Section 3.1.2, in particular [172, 174, 181]), (2) gaze interaction using head-mounted eye trackers (discussed in Section 3.2, specifically [209, 113, 194]), as well as (3) tracking the spatial relationship between users and displays. The latter of the three will be briefly discussed in the following extra related work section, 5.2.1. Right after that paragraph, we will elaborate upon the concept of *GazeProjector* (Section 5.2.2), outline its technical realization and conclude with briefly explained application cases (Section 5.2.3).

5.2.1 Tracking Spatial Relationships of Users and Displays

The basic idea of *GazeProjector* is to use the initial calibration on any display for gaze estimation. In order to apply the calibration on different target displays, it is necessary to track the spatial relationship between all displays in the environment and the users (and the users' devices, respectively). There exist two possibilities to do this: First, external tracking equipment can be used to determine a device's exact position in 3D space (and thus its spatial relationship to a display in the environment). The Proximity Toolkit (shown in Figure 5.3) makes use of such high-precision tracking



(a) (b)

Figure 5.3: Examples of tracking a person's spatial relationship to a display: (a) Proximity Toolkit [115] based on a instrumentation of the environment, (b) TouchProjector [4] using a mobile phone's camera to accomplish this task.

equipment and provides an interface to acquire spatial relationships [115]. While such a setup results in extremely high accuracy, it is often impractical for outdoor use.

Alternatively, the device's camera can be used to identify its spatial relationship to a display. Many approaches exist, such as temporarily showing on-screen visual markers [4] or using dynamic markers following a camera's position [146]. More recently, natural feature tracking was used to determine spatial relationships. Herbert et al. used Scale-Invariant Feature Transform (SIFT) to determine the camera's spatial relationship to a display [57]. Their system tried to identify a screenshot of the display in the device's camera stream. Virtual Projection extended this approach to dynamically updated displays [9]. Touch Projector, shown in Figure 5.3, further allowed for tracking multiple displays provided that the display contents differ sufficiently [16]. *GazeProjector* uses these underlying concepts, but advances them with respect to tracking efficiency: we use FAST/FREAK (with their significantly improved matching accuracy [144]). The combination of these two algorithms further increases the processing, allowing for more interactive frame rates at higher precision than previous systems of that kind.

5.2.2 Enabling Gaze Interaction On Large Display

As mentioned before, estimating a user's gaze on a large display and in multi-display environments using a head-mounted eye tracker faces two key challenges: the eye tracker has to be calibrated, and used, from fixed positions and orientations for all displays. During calibration, the entire display has to be visible in the eye tracker's scene camera. Ideally, the eye tracker only has to be calibrated once. To achieve this, we propose the following concept: (1) Calibrate pupil positions to the scene camera

coordinate system; (2) Track the spatial relationship between the eye tracker and a specific display; and (3) Map 2D gaze positions in scene camera coordinate space to the correct display in the environment. In this way, we enable the user to move around while eye tracking is performed on multiple displays in the environment. This empowers the creation of complex gaze-based interactive settings. The concept is depicted in Figure 5.4 and described in the following.

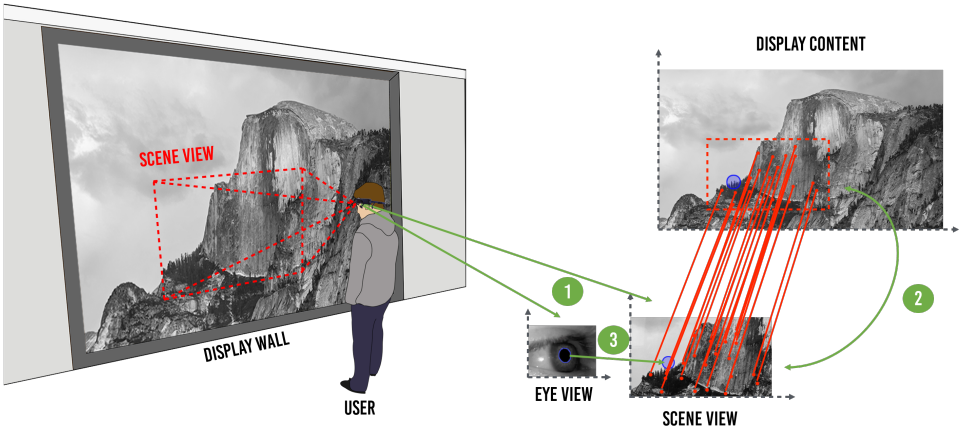


Figure 5.4: Concept of *GazeProjector*: (1) Eye tracker calibration and tracking the user's current field of view and (2) the spatial relationship between eye tracker and display in order to (3) map the user's gaze accordingly

Eye Tracker Calibration

GazeProjector uses a 9-point calibration to map pupil positions to the scene camera's coordinate system (as explained in Section 3.2). There is no need to perform the calibration on each display one intends to interact with. Instead, the system can be calibrated once on any display in the environment (e.g., a laptop). This independence of the calibration to the target display has two advantages: The usage of the eye tracker is not restricted to the same distance and/or orientation to a display while calibrating as this is handled by the self-localization directly; and the calibration does not depend on a single display, thus allowing for seamless gaze estimation across several displays in multi-display environments.

Tracking the Spatial Relationship to Displays

To determine the spatial relationship between the eye tracker and a specific display, we use the approach described by Baur et al. [9]. Basically, the idea is to stream the scene camera's video feed to a server that is aware of all screens (and their displayed content) in the environment. All displays repeatedly stream screen-shots to the server to reflect their current content (i.e., in the case of quickly updated content, such as videos). The server is thus only aware of the physical dimensions of each display (i.e., size and resolution) as well as their current content, but not their physical location in space. This is especially important for mobile devices, which frequently change their position and orientation over time.

In simple words, the server uses the current screenshots of the displays' content as template images, and tries to find them in the observed images (i.e., the eye tracker's scene camera video). If a template matches an observed image, a transformation matrix is calculated.

Mathematically, we compute a projective transformation of planar points from image plane **P** (i.e., the scene camera coordinate system) to **P'** (i.e., the display coordinate system). A special kind of such a projective transformation is a so-called *homography*. In general, a homography is a non-singular, line preserving, projective mapping:

$$h : P^n \rightarrow P^n \tag{5.1}$$

where 5.1 is represented by a square (n + 1)-dimensional matrix with $(n + 1)^2$ - 1 degrees of freedom (DOF).

Back to our scenario, we consider the 2D case, which means a mapping between two planes. Figure 5.5 depicts the concept of 2D *homographies*. The 2D homography is an invertible mapping $h : P^2 \rightarrow P^2$ described by an 3 x 3 matrix **H** with 8 degrees of freedom (i.e., 2 scale, 2 rotation, 2 translation, 2 line at infinity), so that equation 5.2 is true:

$$(5.2) \qquad \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,1} & h_{3,2} & h_{3,3} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad or \quad x' = \mathbf{H}x$$

To compute this projective transformation, a set of coordinate pairs that match from image plane **P** to **P'** have to be known. For this, we use the concept of image feature to extract visual features (e.g., corners, edges, textures) in both image planes (i.e., scene camera and display screenshot), as highlighted in Figure 5.6. The two sets of image features are compared to find similar ones, which result in a set of image feature pairs (cf. Figure5.6c). Based on these pairs, the **homography** matrix can be computed. With this, we are able to estimate and keep track of the spatial relationship between the head-mounted eye tracker (and thus the user) and the display visible in the eye tracker's scene camera video. If multiple displays are present, a transformation matrix is calculated for each display to transform the gaze accordingly.

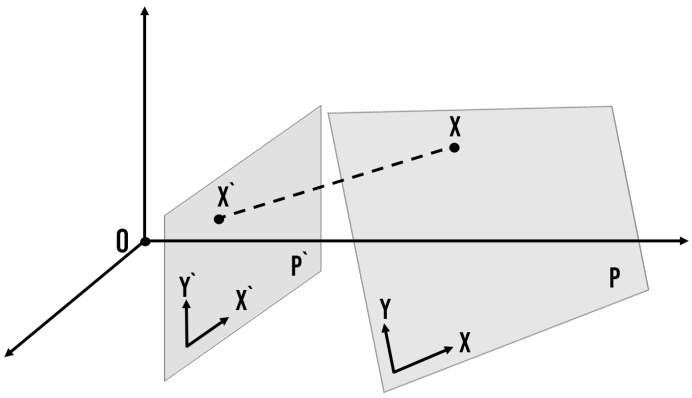


Figure 5.5: 2D homography maps a point x from image plane P to point x' in plane P' (adapted from [27]).

Gaze Estimation

The complete process of gaze estimation is highlighted in Figure 5.6. The initial 9-point calibration of the head-mounted eye tracker is used to map (Figure 5.6c) the user's pupil position from the eye camera's image plane (Figure 5.6a) to the corresponding gaze position in the scene camera's image plane (Figure 5.6b). As described above, to estimate the spatial relationship between eye tracker and display, we extract image feature pairs (Figure 5.6e) between the display's content (Figure

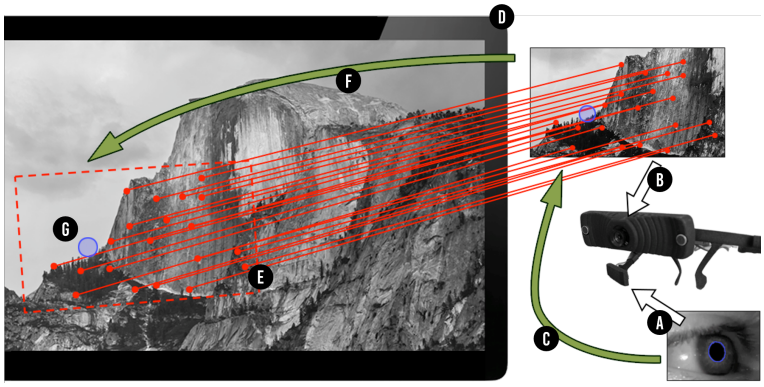


Figure 5.6: Gaze estimation on a display using the eye tracker's (a) eye and (b) scene camera and the display's content (d): the approach combines the eye tracker calibration (c) with the determined projective transformation between the scene camera's image plane and the display (f), based on image feature tracking and matching (e), to obtain the location on the display (g).

5.6d) and the scene camera image (Figure 5.6b). The resulting *homography* matrix **H** allows for bidirectional mapping (5.6e) between points in the scene camera's and target display's coordinate system, as explained above. Combining the initial eye tracker calibration with the homography matrix, the final gaze point of the display is obtained (cf. Figure 5.6f).

5.2.3 Implementation and Applications

In the following we will highlight the implementation of the presented concept from above, which resulted in the final system called *GazeProjector*. Figure 5.7 shows an overview of the system's architecture. It consists of three main components, implemented in a client server structure:

Tracker – one or more monocular head-mounted Pupil Labs eye trackers [78], of which each is connected via USB to a laptop computer. Each laptop is running an instance of *Pupil Capture* (version 3.7[2]) provided by the open source mobile eye tracking platform Pupil Labs. The platform is developed using the programming languages Python and C++. *Pupil Capture* provides the full set of basic features needed for gaze estimation. This includes a graphical user interface to control eye and scene cameras (e.g., parameters such as focus and brightness), adjust the parameters of the eye detection algorithm, perform a 9-point calibration and visualize the gaze point in the scene camera's video feed. In addition, the software already provides a tool for

[2]https://pupil-labs.com/blog/2014-01/new-pupil-pro-headset-capture-software-0-3-7/

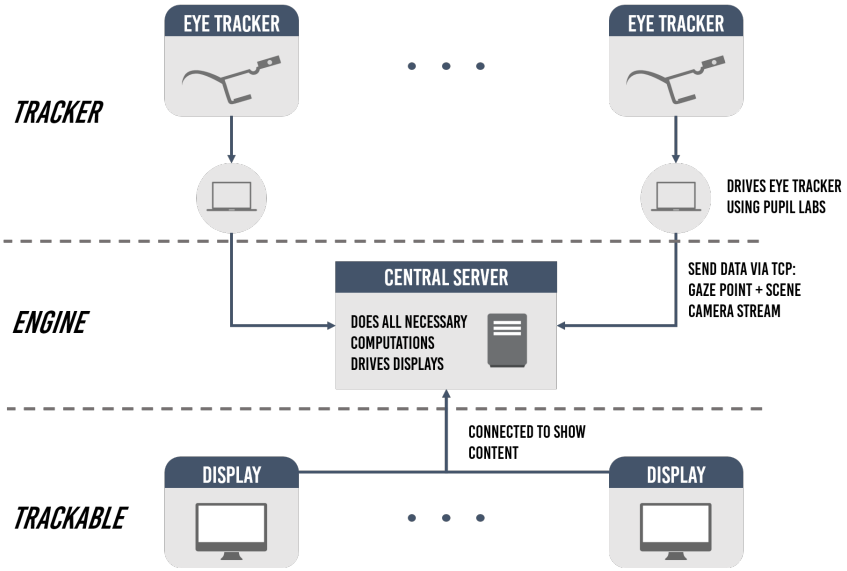


Figure 5.7: Architecture of *GazeProjector* with three components: Tracker, Engine and Trackable

augmented reality (AR) marker tracking, based on the ArUco library[3]. This mechanism can be used to track and map gaze on surfaces (e.g., a display) by equipping it with pre-registered AR markers. We used this functionality later in our evaluation of our approach. Further, the software integrates a network interface to stream all data related to gaze and pupil information. We extended this interface by the ability to integrate the eye tracker's scene camera's images into the data stream, as shown in Listing 5.1. All information is sent to the central server using the ZMQ library[4].

Code Listing 5.1: Camera Server

```
1  class Camera_Server(Plugin):
2    def __init__(self, g_pool, atb_pos=(500,400)):
3      Plugin.__init__(self)
4
5      self.context = zmq.Context()
6      self.pusher = self.context.socket(zmq.PUSH)
7      self.address = create_string_buffer("tcp://*:5562",512)
8      self.set_server(self.address)
9
10     self.messageTracker = None
11
12   def update(self,frame,recent_pupil_positions,events):
13     if frame.compressed_img != None and
```

[3]https://www.uco.es/investiga/grupos/ava/node/26
[4]http://zeromq.org/

```
14      (self.messageTracker == None or (self.messageTracker != None and
                                         self.messageTracker.done)):
15        self.messageTracker = self.pusher.send(base64.b64encode(
16          frame.compressed_img.tostring()), 0, False, True)
```

This framework provides many communication patterns usable in most programming languages. We chose the pipeline pattern that uses push and pull sockets. Since the eye tracker has to send information, it acts as a pusher using the TCP protocol (cf. Listing 5.1 lines 5–8).

The data is broadcasted so as to be independent of the server's IP address. Consequently, the server can consume the data by implementing a pull socket. The update method of the class is called each time a camera frame is processed in the internal program loop of the Pupil Capture software. It sends the scene camera frame as a compressed image (50% JPEG compression) as a base64 encoded string (cf. Listing 5.1 lines 12–16). To enable reliable data transfer and prevent blocking behavior of the socket, we use the option of the method `send(params)` to track the message's state, stored in *messageTracker* (cf. Listing 5.1 lines 15–16). Laptops and the central server are connected via WiFi.

Trackable – one or more planar displays of arbitrary size on which the users' gaze should be estimated. All displays are directly connected to the central server. The physical dimensions (in mm), the resolution (in pixels) and the content are prerequisites for the system to reliably compute a user's gaze on the display.

Engine – a central server that drives all displays in the environment and performs all computations necessary for the gaze estimation. As shown in Figure 5.7, the server is connected to all eye trackers via a WIFI network. The receiving part of the communication is implemented as a pull socket provided by the ZMQ library. All software is written in C#, since Windows 10 is the server's operating system. In the following we will provide listings to show the most important parts of the implementation. Note that we show only simplified versions of different classes in a pseudo-code style. All three components of the architecture are represented as objects within the system.

The object *Tracker* is used to store all necessary information, such as the received image data as well as its size (see Listing 5.2, lines 1–10). Because we might use devices other than an eye tracker (e.g., a mobile phone [9]) as an input device, a designated class *EyeTracker*, inheriting from *Tracker*, is implemented (cf. Listing 5.2,

lines 13–31). It is responsible to stream all eye tracking-related data (i.e., the gaze information) as well as the image data and store these in a list.

Code Listing 5.2: Tracker

```
1  class Tracker
2
3    List dataQueue;
4    Rectangle imageBounds;
5    float scale;
6
7    Tracker(Rectangle frameBounds, double scale)
8      this.imageBounds = frameBounds;
9      this.scale = scale;
10     this.dataQueue = new List();
11
12
13 class EyeTracker: Tracker
14
15   int frame_width;
16   int frame_height;
17   List gazeQueue;
18
19   public EyeTracker(int w, int h)
20     super(new Rectangle(0, 0, w, h), 1.0);
21     this.frame_width = w;
22     this.frame_height = h;
23     this.gazeQueue = new List();
24
25   public void OnCameraFrameReceived(object sender, Event e)
26     double timestamp = System.CurrentTimeInMillis;
27     dataQueue.Enqueue(e.decodeImage(frame_width, frame_height),
                                    timestamp);
28
29   public void OnGazePointReceived(object sender, Event e)
30     double timestamp = System.CurrentTimeInMillis;
31     gazeQueue.Enqueue(e.gazePoint, timestamp)
```

Each display in the environment is represented by a *Trackable* object (Listing 5.3, lines 1–13) and implemented via the class *Display* (Listing 5.3, lines 16–30). We chose this hierarchical implementation to represent any kind of display (e.g., computer monitor, mobile phone). A *Trackable* stores general information, such as resolution and physical dimensions. These properties are necessary to correctly transform the points from the eye tracker's scene image plane to the display's image plane. The actual *Display* class

represents screens that are directly connected to the server. Hence, their content can be streamed by continuously storing screen-shots (see Listing 5.3, lines 23–30). In theory other types of displays can also be supported, whose content may be recorded in a different manner (e.g., network).

Code Listing 5.3: Trackable

```
1  class Trackable
2
3    List dataQueue;
4    Rectangle imageBounds;
5    float scale;
6    Size physicalSize;
7
8    Trackable(Size physicalSize,
9    Rectangle surfaceBounds, double scale = 1.0)
10     this.imageBounds = surfaceBounds;
11     this.physicalSize = physicalSize;
12     this.scale = scale;
13     this.dataQueue = new List();
14
15
16 class Display: Trackable
17
18   public Display(Size physicalSize,
19   Rectangle surfaceBounds, double scale = 1.0)
20     super(physicalSize, surfaceBounds, 1.0);
21
22
23   private void ScreenCapture()
24     Bitmap screenshot = GetRawScreenImage();
25     double timestamp = System.CurrentTimeInMillis;
26
27     Bitmap scaledScreenshot = new Bitmap(screenshot,
28     screenshot.Width * scale, screenshot.Height * scale));
29
30     dataQueue.Enqueue(screenshot, timestamp);
```

The main system logic is implemented in the class *Engine*. It has knowledge about all connected trackers and trackables, stored in a list. The function *trackingAlgorithm* (Listing 5.4, lines 22–50) performs the main loop as follows: Each *Trackable* object is continuously recording its screen content together with the respective timestamps for synchronization. For each such image, the key points and features are extracted using using FAST feature detectors [105] and FREAK feature descriptors [144]. The

correct construction of the necessary objects is shown in Listing 5.4 (lines 1–10). As previously mentioned, the implementation is done in C# (.NET Framework 4.5). For feature detection, description and matching, we use EmguCV[5] (version 2.4.2), a C# wrapper for the computer vision library OpenCV[6]. For detailed information on the chosen parameters, we refer the reader to the documentation of OpenCV.

All *Trackers* are constantly storing the eye tracker's data. Exactly as for the display screenshots, the key points and features of the scene images are extracted. If more than four features were found in the scene camera's image (Listing 5.4, *observedFeatures*), the homography matrix can be computed. This is implemented by using the *Features2DTracker*, provided by EmguCV. The match function of this class uses a brute force matcher that takes a *targetFeature*, matches it to all *observedFeatures* and returns the match with the closest distance. Before computing the actual homography matrix, the computed matches are filtered (Listing 5.4, lines 42–43) to extract unique matches, whose scale and rotation are in line with the majority's scale and rotation.

Code Listing 5.4: Engine

```
1  # class implemented using EmguCV, http://www.emgu.com/wiki/
2  class FeatureDetector
3
4    KeyPointDetector keyPointDetector;
5    DescriptorExtractor descriptorExtractor;
6
7    #used Fast and Freak as implemented in EmguCV based on OpenCV
8    public FeatureDetector(threshold = 35, nonmaxSuppression = true,
                            orientationNormalized = true,
                            scaleNormalized = true,
                            patternScale = 22.0, nOctaves = 4)
9      this.keyPointDetector = new FastDetector(threshold,
                            nonmaxSuppression);
10     this._descriptorExtractor = new Freak(orientationNormalized,
                            scaleNormalized, patternScale,
                            nOctaves);
11
12 class Engine
13
14   List trackers;
15   List trackables;
16
17   public Engine(List trackers, List trackables)
```

[5] http://www.emgu.com/wiki/files/2.4.0/
[6] http://opencv.org/

```
18    this.trackers = trackers;
19    this.trackables = trackables
20
21
22  public void trackingAlgorithm ()
23    for all trackables do_in_parallel:
24      ScreenCapture();
25      for all screenshots in dataQueue do_in_parallel:
26        featureDetector = new FeatureDetector();
27        targetKeyPoints = featureDetector.detectKeyPoints(screenshot);
28        targetFeatures = descriptorExtractor.computeDescriptors(
                                    targetKeyPoints);
29
30    for all trackers do_in_parallel:
31      #Receive all data via ZMQ and store it
32      OnCameraFrameReceived(ZMQ sender, DataReceivedEvent e);
33      for all scene_images in dataQueue do_in_parallel:
34        featureDetector = new FeatureDetector();
35        observedKeyPoints = featureDetector.detectKeyPoints(scene_image
                                    );
36        observedFeatures = descriptorExtractor.computeDescriptors(
                                    observedKeyPoints);
37
38        if (observedFeatures != null && observedFeatures.Length >= 4)
39
40          Features2DTracker tracker = new Features2DTracker<byte>(
                                    target_feature);
41          Features2DTracker.MatchedImageFeature[] matchedFeatures =
                                    tracker.MatchFeature(
                                    observedFeatures, 4);
42          matchedFeatures = Features2DTracker.VoteForUniqueness(
                                    matchedFeatures,
                                    uniquenessThreshold = 1.15);
43          matchedFeatures = Features2DTracker.VoteForSizeAndOrientation
                                    (matchedFeatures, scaleIncrement =
                                    2.0, rotationBins = 8);
44
45          HomographyMatrix matrix = Features2DTracker.
                                    GetHomographyMatrixFromMatched
46          Features(matchedFeatures);
47
48          if (matrix != null)
49            #project correct gaze point, synchronized via timestamp
50            gazeOnDisplay = matrix.Transform(gazeQueue.poll());
```

Finally, the homography matrix can be computed, and with that the gaze point can be projected onto the display. For faster processing, we downscale display screenshots to 384x240 pixels and camera frames to 320x240 pixels. Both loops of the *trackingAlgorithm* function are parallelized and implemented in different threads.

It is important to note that the display does not have to be fully visible in the eye tracker's scene images. Instead, a unique subregion (a region that does not occur anywhere else on that display) is sufficient, given there are enough features within it to allow for robust tracking. Likewise, unique sub-regions and their features on multiple displays present in the scene camera allow for detecting each of the displays within one frame. With our implementation, we achieve up to 30 fps on one display (20 fps with three displays). The number of eye trackers is limited only by the local network bandwidth, as each one needs to stream the camera images to the server. However, a possible improvement to address this issue is to outsource the feature extraction routine into the eye tracking software. Hence the information that has to be sent is minimized.

In its current implementation, the system allows for distances ranging from 0.5 up to six times the display's diagonal. When bthe distance is greater, the accuracy decreases as the display observed in the camera's field of view decreases in size (thus removing several features). We believe that a multi-scale approach of screenshots will increase the operational range, yet we decided not to include it in this proof-of-concept implementation. In addition, the tracking compensates for an angular offset of $\pm 60°$. While this is sufficient for most interactions, fast eye/head movements will have a slight impact on accuracy. However, we believe that the increasing processing capabilities of future devices will allow for both faster image processing on larger images (i.e., less or no scaling required) and for higher accuracy.

Example Application

We built two example applications to demonstrate the use of *GazeProjector* in a multi-display setting. It showcases how people can seamlessly interact with multiple displays while freely moving around in the environment.

We envision an office building where public information screens are distributed showing a calendar application (see Figure 5.8a). We use *GazeProjector* to interact with these displays in a 'walk-up-and-use' fashion. The only requirement is to calibrate

Figure 5.8: First example application: after a user selects an event from a public calendar (a), that information is shown on the personal mobile device, where gaze estimation also works (b).

the eye tracker (e.g., at the beginning of a work day using a tablet). The calibration procedure uses a round marker displayed at nine different positions around the screen, at which the user has to gaze on. In our interaction scenario users are able to transfer information between public and private displays (e.g., a handheld). Looking at a specific event (e.g., a scheduled talk) for a certain dwell time (say: 2 seconds) selects that event, which is then copied to the user's handheld device. *GazeProjector* also works on that device based on the same calibration (cf. Figure 5.8b).

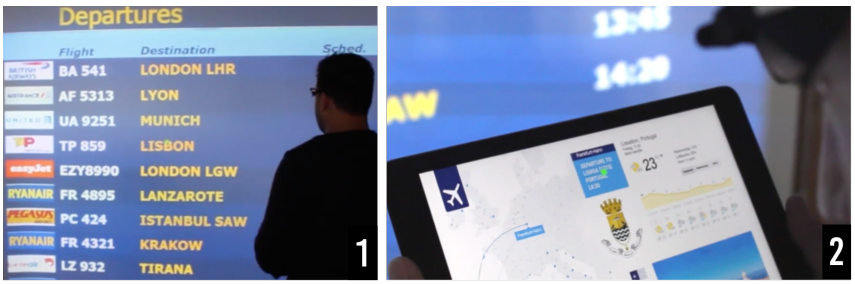


Figure 5.9: Second example application: when users select from a time table of departing flights (1), the corresponding information is shown on their mobile device (2).

In the second example, we consider a timetable on one of the many large screens at an airport, showing flights departing in the next hours (see Figure 5.9(1)). Users can look at a specific flight and its flight number or destination respectively. Once the system recognizes the point users are looking at, additional information, such as a map of the destination or detailed flight data, is transferred to the user's mobile device (e.g., a tablet PC). *GazeProjector* further allows for tracking gaze on the tablet as well, allowing for content adaptation. If the user now gazes at the map, the tablet will show more specific information, such as the weather at the destination (cf. Figure 5.9(2)).

5.3 Evaluation

In this section we describe the two conducted experiments to assess *GazeProjector's* gaze estimation accuracy in various settings. In each of the experiment sections, we also include the corresponding results. We will discuss the findings as a whole in Section 5.4.

5.3.1 Experiment I: Gaze Estimation Accuracy

We first conducted a controlled laboratory study to assess *GazeProjector's* gaze estimation accuracy in comparison to existing but more heavyweight tracking approaches in a single display scenario.

Independent Variables

We had two independent variables in this experiment: *Mode* (i.e., the gaze estimation method used), and *Location* (i.e., where participants stood in front of the display).

Mode: We chose three different modes for gaze estimation: *GazeProjector (GP)* implemented as described before; *Marker Tracking (MT)*, which uses a set of on-screen markers for tracking the orientation between the eye tracker and the display provided by the Pupil Labs framework; and a simple *Head Orientation (HO)* approach, which tracks the participant's head using an external OptiTrack system. For each of these modes, we calibrated the eye tracker from two different locations to investigate the effect of distance during calibration. Both were placed centrally in front of the display, with one location being close to the display and one being further away. We further calibrated the eye tracker for each participant separately instead of using one calibration (see the limitations section for further details).

Location: We chose six different locations in front of the display to simulate a more realistic setting. Three of these locations were close to the display and three were further away (cf. Figure 5.10). The eye tracker was only calibrated for the near central and far central locations. This is more realistic, as users would not calibrate for every position in a walk-in-and-use scenario. Note that we calibrated the eye tracker for each participant separately instead of using one calibration (see the limitations section for further details). Since no visual feedback was given to them, and to keep the experiment a reasonable length, participants had to perform the set of tasks only once. We then computed the gaze estimation accuracy post-hoc for each of the calibrations.

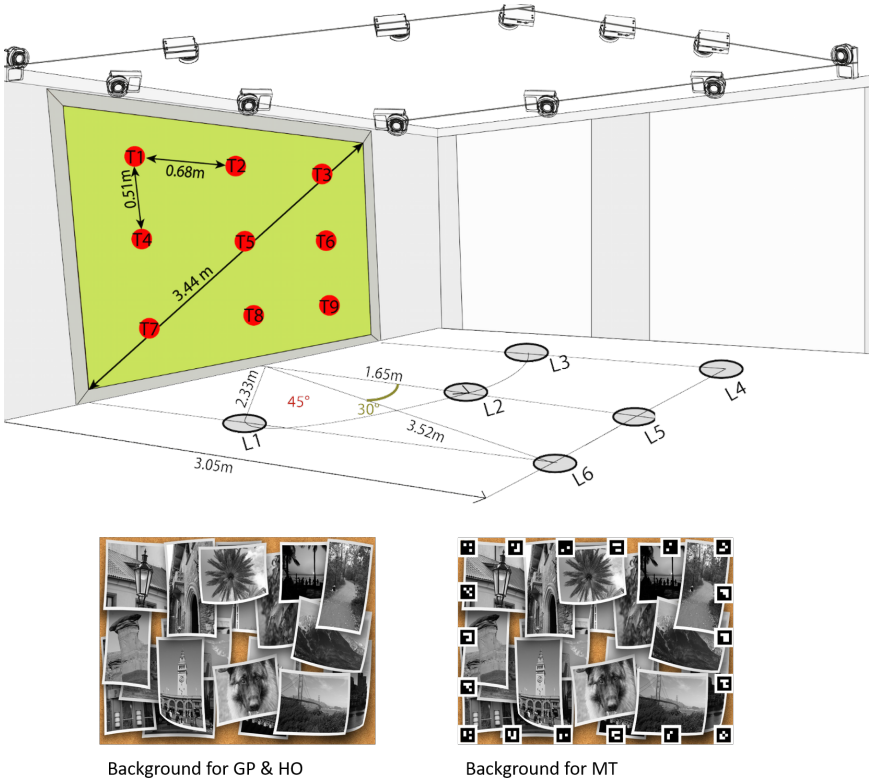


Figure 5.10: Experimental setup showing all locations (L1-L6) and orientations relative to the display, the nine different positions (T1-T9) of on-screen visual targets, and the background image used for feature tracking and marker tracking respectively.

Apparatus

Figure 5.10 shows our experimental setup: we used a large front-projected wall with a size of 2.75x2.07 meters (diagonal: 3.44 meters). The six locations were distributed within a nine-square-meter area in front of the display as follows: three locations at a distance of 1.65 m (*near*), and three locations at a distance of 3.05 m (*far*). The left and right locations for the near distance were exactly 2.33 meters away from the display's center line (i.e., an angular offset of ±45°); those for the far distance were located 3.52 m away from the display's center line (i.e., an angular offset of ±30°). Naturally, the two center locations for near and far had an angular offset of 0°. Locations located far away allow participants to observe the entire screen at once (the display covers 48.52°), while for locations located nearby, the display covers 79.60° thus exceeding the full-scale ocular motor range of ±55° [49]). The maximum visual angles were 3.4° (*near*) and 1.84° (*far*), and the minimal ones were 1.5° (*near*) and 1.3° (*far*). The figure also includes the background image that we used for MT and HO. We chose this image as it compromises many distinct features distributed

across its area. We used the same background with 16 AR markers around the image, each 100x100 px.

Task & Procedure

We implemented a gaze pointing task in which participants had to fixate nine different target locations represented as red circles (50 pixels or 98 mm) on the display with equal distances between them (see Figure 5.10). A pilot study showed that participants were affected by visualizing their gaze point on the display. Especially if the gaze position was incorrect, people tended to move the gaze point to compensate for the error. We therefore opted not to provide any visual feedback to the participants. Participants were instructed to look at each target as quickly and accurately as possible. Each target location was shown for five seconds.

For each *Mode*, participants first calibrated from the *near-center* location and performed the tasks for all other locations. Afterwards, the calibration for the *far-center* location was recorded and gaze positions as well as errors were evaluated post-hoc. Following best practices in gaze estimation experiments, we validated all calibrations by asking participants to fixate once on each point on a 9-point pattern. Finally, we asked for demographic information.

We collected gaze data from the eye tracker and transformation matrices calculated by *GP* as well as *MT*. Furthermore, we recorded data about the head position and orientation with *OptiTrack*. Data was sampled at 30 Hz (i.e., 150 samples per on-screen target) leading to a total of 1,350 samples for each Mode and Location combination. We discarded samples for which the participants' pupil was not detected (7.5%). We dropped the first two seconds of the five seconds per target (60 samples, 40%) for each target, which was the maximum time required to find the target. Altogether we dropped 276,985 out of 583,200 samples (3 modes x 6 locations x 2 calibrations x 12 participants x 1,350 samples), leaving 306,215 samples recorded: 140,532 for *GP*, 165,683 for *MT* (the sample set used for *HO*).

Experimental Design

We used a within-subjects design with the independent variables *Mode* (*GP-near, GP-far, MT-near, MT-far, HO-near, HO-far*) and *Location* (*front-left, front-center, front-right, back-left, back-center, back-right*).

We counterbalanced the order of Location across participants using a Latin square. Although it is possible to record all position information in parallel, we opted to have *GP* and *MT* separate, to prevent positive effects of the AR markers on the feature tracking algorithm of *GazeProjector* (cf. background for MT in Figure 5.10). The *HO* mode was recorded while participants were using the *MT* mode. Half of our participants started with the *MT*, and the other half with *GP*. Thus, each participant performed the task twice per location. For each mode and location, the nine targets (equally distributed in a 3 x 3 grid on-screen) were presented in random order.

Participants

Twelve participants (three female) between 22 and 32 years (mean = 27.45 years, SD = 3.1 years) were recruited from a local university campus. All participants had normal or corrected-to-normal vision; none reported any form of visual impairments (e.g., color blindness).

5.3.2 Experiment I – Results

We corrected all reported gaze estimation accuracies by subtracting the mean calibration error (2.04°with SD = 0.69°). To verify this, we performed a one-way ANOVA with a Bonferroni-corrected post-hoc analysis on calibration accuracies across all Modes, and found no significant differences. In subsequent post hoc analyses, we used Bonferroni-corrected confidence intervals to retain comparisons against $\alpha = 0.05$. Furthermore, we used Greenhouse-Geisser correction in cases where sphericity had been violated.

Gaze Estimation Error

To assess the gaze estimation error, we calculated the average gaze estimation error in degrees of visual angle (cf. Section 2.6.2 in [59]): that is, the difference of the visual angle between the predicted on-screen gaze point and the actual fixation targets for all Modes and Locations. We then performed a 6 x 6 (*Mode x Location*) within-subjects ANOVA on gaze estimation errors and found a main effect for *Mode* ($F(1.989, 21.879) = 8.526, p < .002$), and a main effect for *Location* ($F(5, 55) = 7.363, p < .001$), but we did not find an interaction between the two.

We performed post-hoc tests to further understand the main effect of Mode. Most importantly, we found significant differences within MT and HO for the two calibrations, near and far (all $p < .033$). In both cases, the near calibration led to lower estimation

errors. *GP*, on the other hand, did not show such an effect, suggesting that the point of calibration does not affect its gaze estimation error significantly, and the difference in means was also lower than for the other two ($GP : 0.281°; MT : 0.931°; HO : 0.948°$); yet, also for *GP*, the mean estimation errors were slightly lower for the near calibration than for the far one.

This is further reflected when comparing across Modes: GP-near differed significantly from both MT-far and HO-far (all $p < .01$). However, there was no significant difference between the Modes for the near calibration. Furthermore, GP-far did not differ significantly from any other Mode despite having relatively large differences in error. *GP-near* showed the lowest error ($M = 1.80°, SD = 0.20°$), followed by *GP-far* ($M = 2.08°, SD = 0.27°$), and *HO-near* ($M = 2.09°, SD = 0.23°$). *MT-near* ($M = 2.23°, SD = 0.31°$) also has an estimated gaze error of less than 3 degrees. The other Modes performed slightly worse: *MT-far* ($M = 3.16°, SD = 0.32°$) and *HO-far* ($M = 3.04°, SD = 0.31°$). Figure 5.11 summarizes these results.

Post-hoc tests on Location revealed that the significant main effect stems from participants' distance to the display: front-left differed significantly from back-center

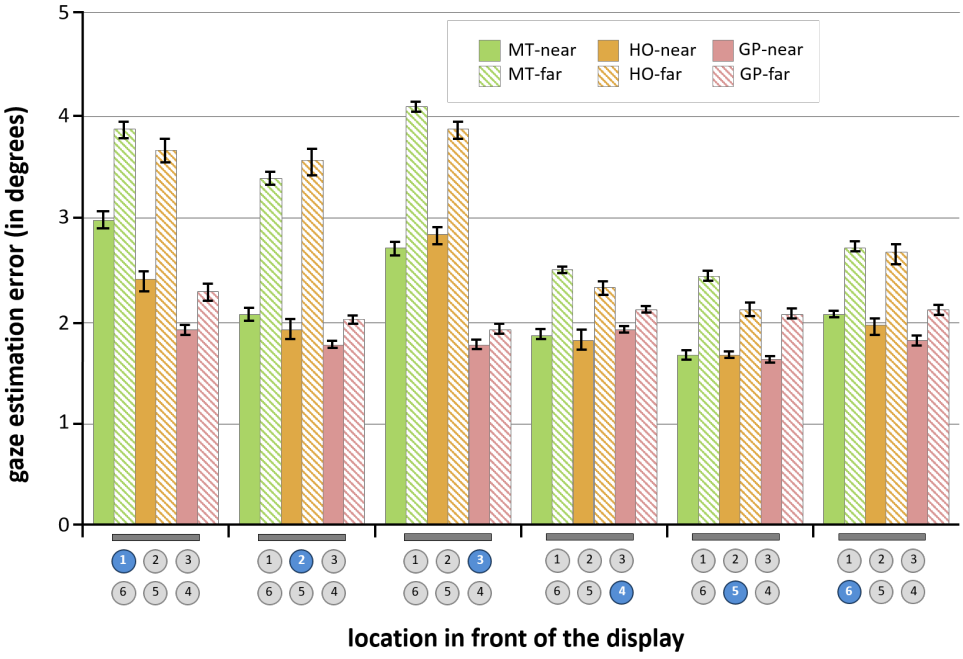


Figure 5.11: Mean gaze estimation error for every location for MT-near, MT-far, HO-near, HO-far, GP-near and GP-far. Error bars indicate ± standard error of the mean.

and back-right (all $p < .019$). Front-right also differed significantly from back-center ($p < .011$). Overall, back-center led to the lowest estimation errors ($M = 1.9°, SD = 0.23°$), followed by back-right ($M = 2.01°, SD = 0.17°$), and back-left ($M = 2.22°, SD = 0.30°$). The front locations performed worse, with front-center having the lowest errors ($M = 2.45°, SD = 0.28°$), followed by front-left ($M = 2.86°, SD = 0.29°$) and front-right ($M = 2.86°, SD = 0.30°$). On average, the back locations had a lower estimation error of $2.08°(SD = 0.23°)$ compared to the front locations with $2.72°(SD = 0.29°)$.

Differences for On-screen Target Positions

We did not expect high gaze estimation errors for each of the Modes. However, we wanted to analyze whether the on-screen targets resulted in different estimation errors and thus analyzed the results separately for each on-screen target. For *MT*, we found no significant main effects on gaze estimation error for Target. We found the same for *HO*. Only for *GP*, we found significant differences for gaze estimation for *Target*. Our analysis revealed that predominantly the *bottom-left* target T7 differed significantly from a few others (T2, T3, T6 and T8) and led to higher estimation errors. We assume that this is due to the scene camera seeing too few features, which in turn increased the error of the transformation matrix. Figure 5.12 shows gaze estimation errors for the different modes averaged over all targets.

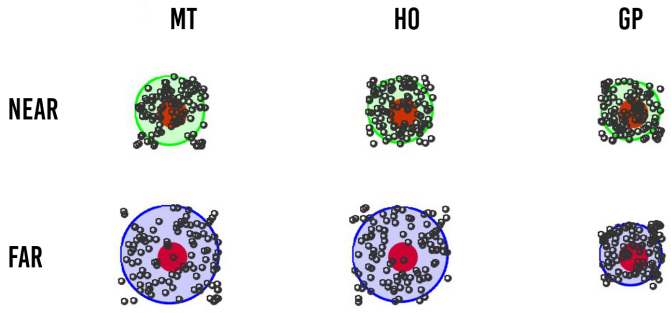


Figure 5.12: Visualization of the mean gaze error (ellipses) for the three modes MT, HO and GP and all calibrations averaged over all targets. Black circles visualize the mean gaze points.

Eye and Head Movements

We were further interested in whether participants mainly moved their head or their eyes to point at an on-screen target location. As expected, we found that the average

Location	Mean(x,y)	SD(x,y)	Var(x,y)
front-left	0.43,0.45	0.19,0.24	0.038,0.060
front-center	0.45,0.47	0.20,0.25	0.040,0.062
front-right	0.46,0.46	0.22,0.27	0.052,0.076
back-left	0.46,0.48	0.23,0.25	0.054,0.065
back-center	0.45,0.48	0.20,0.24	0.044,0.058
back-right	0.43,0.48	0.20,0.23	0.043,0.057

Table 5.1: Mean, standard deviation and variance for x,y-coordinates of normalized gaze positions in the participants' field of view.

normalized gaze position in the field camera's video was x = 0.44 and y = 0.47 ($SD_x = 0.21; SD_y = 0.25$). Thus, gaze positions remained near the center of the participants' field of view. We subsequently analyzed the gaze position for every Location in front of the display and found no significant differences between them. The largest average difference was 0.03. Table 5.1 lists these results for each *Location*. The *OptiTrack* data provided detailed information on participants' head orientation (*HO*). We found that the largest head turns covered the entire width of the display (far: 51.2°, near: 83.66°). On average, head motions covered an angle of 31.61° ($SD = 2.04°$). This further confirms our results in that HO might be a suitable approximation for gaze estimation with an average error of 2.09° ($SD = 0.23°$) for HO-near and 3.04° ($SD = 0.31°$) for *HO-far*.

5.3.3 Experiment II: Multiple Displays

We conducted a second controlled laboratory study to assess *GazeProjector's* gaze estimation accuracy across multiple displays of varying form factors with only a single calibration performed on one of the displays.

Independent Variables

We had two independent variables: *Mode* (i.e., the gaze estimation method used), and *Screen* (i.e., on which display the target was shown). There were no fixed positions,

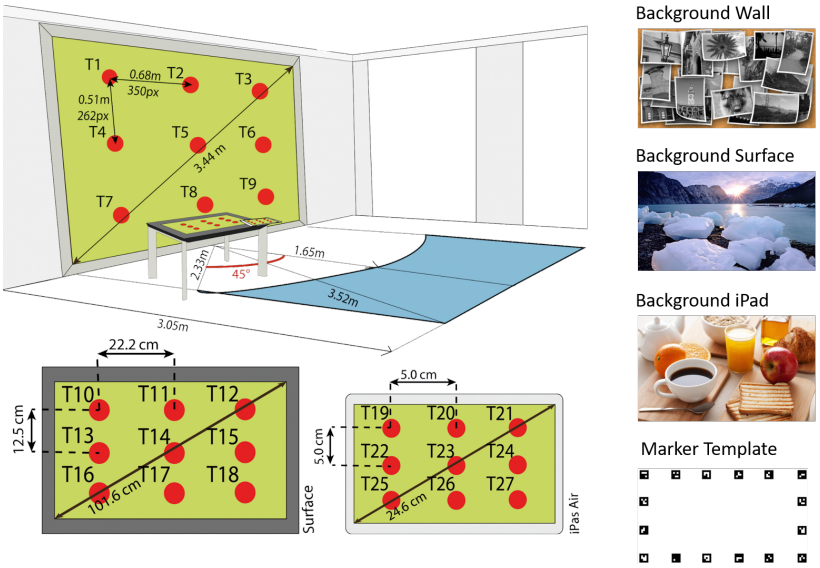


Figure 5.13: Our setup showing the three screens (including their background images for feature tracking) used during the experiment, as well as the placement of the nine targets per screen. Note that all participants were free to choose a location within the blue area throughout the experiment.

to mimic a more realistic scenario where participants were free to move in the environment.

Mode: In this experiment we chose to use only *GazeProjector (GP)* and *Marker Tracking (MT)*, not head orientation, as we believe it will perform similarly across displays. We calibrated for two locations (as in the first experiment), but additionally recorded calibrations on a 40-inch tabletop display (*Surface*), and on a 9.7-inch iPad Air (*iPad*). We chose to do so to investigate the effects on gaze estimation accuracy of calibrating (1) on surfaces not orthogonal to the participant, and (2) on personal devices with a considerably smaller display. The latter resembles a more realistic scenario where users calibrate the eye tracker once on a personal device. Again, calibrations were analyzed post hoc.

Screen: In addition to the large display used in the first experiment (*Wall*), we chose to add the other two displays used for calibration as well (here *Surface* and *iPad*).

Apparatus

We used the same front-projected *Wall* as in the first experiment. In addition, we had a 40-inch Microsoft Surface 2 (*Surface*), and a 9.7-inch iPad Air tablet (*iPad*). Figure

5.13 shows our setup. The tabletop display was placed in front of the projection wall in an area where the participant would occlude the projection. Participants held the tablet in their hands during the experiment. They could freely choose their location within a nine-square-meter area. In addition, the backgrounds used for each display for mode *GP* are shown. An AR marker overlay is displayed on top of each background image for *MT*.

Task & Procedure

The task used in this experiment was the same as in the first one: participants had to fixate on-screen targets. However, since we had three displays, participants now had to acquire nine targets per display (27 in total) as shown in Figure 5.13. As mentioned before, participants could freely choose and change their position between the displays. We again opted not to provide any feedback to participants for the same reasons as before. All participants were instructed to look at each target as quickly and accurately as possible. Each target location was shown for ten seconds to give the participants enough time to find the target on the correct display. There was only *one* target on *one* display shown at a time.

The procedure was nearly the same as for the first experiment, but with an additional calibration for *Surface* and *iPad* after all tasks were completed. On the additional displays we used the same 9-point calibration pattern. At the end of the study we asked for demographic information.

We used the same data collection method as in the first experiment. Data was sampled at 30 Hz (i.e., 300 samples for each target, 8100 samples for each *Mode*), and samples were discarded if the participants' pupil was not detected. As we expected an increase in search time for the target, we dropped the first five seconds (150 samples) for each target, leaving 259,745 samples (*GP*: 124,421; *MT*: 135,324).

Experimental Design

We used a within-subjects 8 *Mode* (*GP-near, GP-far, GP-Surface, GP-iPad, MT-near, MT-far, MT-Surface, MTiPad*) x 3 *Screens* (*Wall, Surface, iPad*) design. Half of our participants started with *GP*, the other half with *MT* (as in Experiment I). The targets were randomized: thus, the next target could appear on any of the three *Screens*. The 27 targets were again placed in 3 x 3 grids (i.e., nine per display, 50

pixels in radius, or 10 mm on *iPad*, 23 mm on *Surface*) on each display. In total, participants acquired 54 targets.

5.3.4 Experiment II – Results

We again corrected gaze estimation accuracy by subtracting the mean calibration error. The mean calibration error was 2.18° ($SD = 0.69°$). We again verified that we could do so by performing an ANOVA with a Bonferroni-corrected post-hoc analysis on calibration accuracies across all *Modes*, and found no significant differences. As in Experiment I, we used Bonferroni-corrected confidence intervals in all post-hoc analyses and Greenhouse-Geisser correction in cases where sphericity had been violated.

Gaze Estimation Error

We calculated the average gaze estimation error as in Experiment I and subsequently performed an 8 x 3 (*Mode* x *Screen*) within-subjects ANOVA on them. We found main effects for Mode ($(F7, 77) = 21.733, p < .001$), and for Screen ($F(2, 22) = 82.705, p < .001$) as well as an interaction effect between the two ($F(14, 154) = 9.100, p < .001$).

Post-hoc pairwise multiple means comparisons revealed that *GP-near* and *GP-far* differed significantly from *MT-far*, *MT-Surface* and *MT-iPad* (all $p < .001$). Furthermore, *GP-Surface* differed significantly from *MT-Surface* and *MT-iPad* (all $p < .007$). And finally, *GP-iPad* also differed significantly from *MT-far*, *MT-Surface*, and *MT-iPad* (all $p < .039$). It is noteworthy, however, that both *GP* and *MT* did not show any significant differences between their different calibrations, suggesting that the device on which they were calibrated did not impact accuracy.

Overall, *GP-near* had the lowest estimation error ($M = 2.77°, SD = 0.20°$), followed by *GP-far* ($M = 3.01°, SD = 0.16°$), *GP-iPad* ($M = 3.24°, SD = 0.17°$) and *GP-Surface* ($M = 3.31°, SD = 0.16°$) across all *Screens*. For all *MT* variations, the estimated gaze errors were larger than 4 degrees. Figure 5.14 summarizes these results.

As for the main effect for Screen, post-hoc multiple means comparisons revealed that *Wall* was significantly different from the other two *Screens* (all $p < .001$). However, there was no significant difference between *Surface* and *iPad*. Overall, targets on the *Wall* had the least estimation error ($M = 2.07°, SD = 0.07°$), followed by *Surface* ($M = 4.52°, SD = 0.22°$) and *iPad* ($M = 5.12°, SD = 0.23°$).

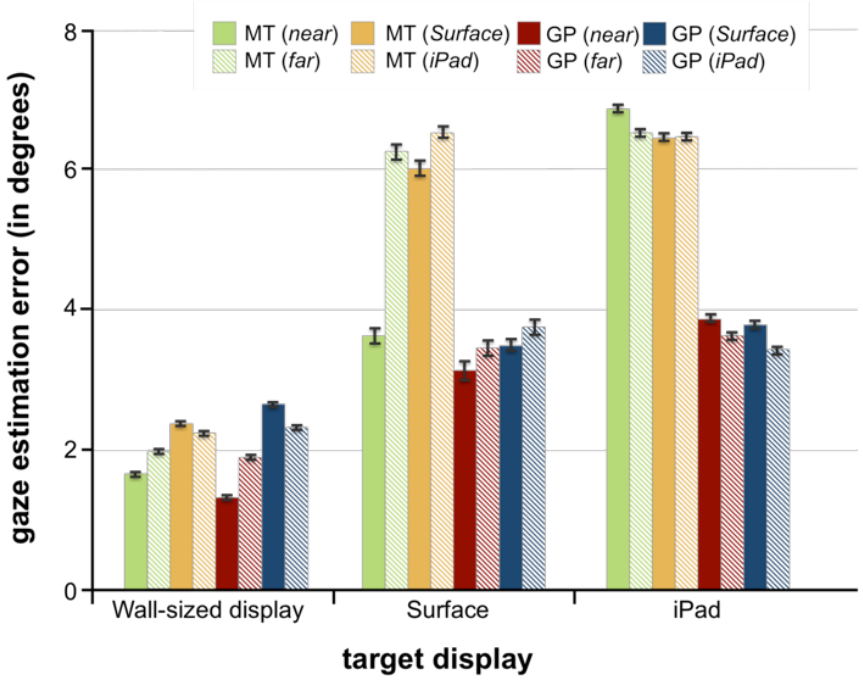


Figure 5.14: Mean gaze estimation error of each mode for each display. Error bars indicate ± standard error of the mean.

As shown in Figure 5.14, the source of the *Mode* x *Screen* interaction is the increased difference between *MT* and *GP* (all calibration modes) between the *Wall* and *Surface/iPad*, with the *Wall* resulting in much lower estimation errors than the other two. It is noteworthy that *MT-near* performs similarly to all *GP* modes on the *Surface*, but its estimation error increases drastically on the *iPad*, although all *GP* modes remain at their level. We subsequently ran separate ANOVAs on *Modes* for each *Screen*, and found several significant effects. On the *Wall*, only *MT-iPad* and *GP-near* differed significantly ($p < .008$), indicating that nearly all modes performed similarly.

On the *Surface*, the differences become more prominent, with *GP-near* and *GP-far* outperforming all *MT* modes except *MT-near* (all $p < .016$). Furthermore, *GP-Surface* and *GP-iPad* differed significantly from *MT-iPad* (all $p < .012$). We found the most differences on the *iPad*, where all *GP* modes are significantly less error-prone than all *MT* modes (all $p < 0.03$). Here we again did not find any differences within *GP* and *MT* for different calibration modes. Figure 5.15 visualizes the mean gaze estimation errors for each of the screens and targets for both *MT* and *GP*.

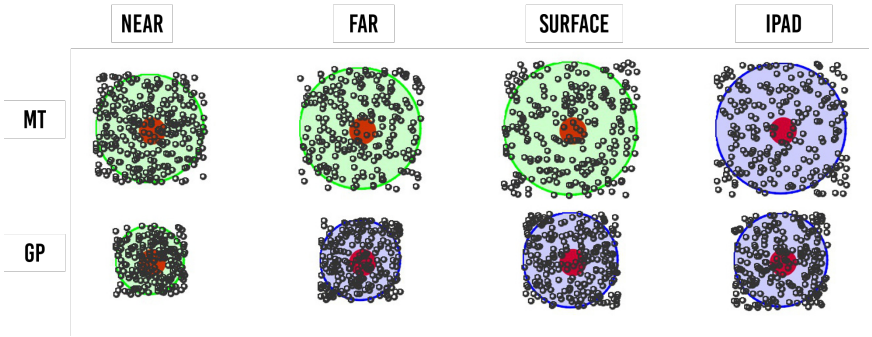


Figure 5.15: Visualization of the mean gaze error (ellipses) for all different modes, MT and GP, and all calibrations averaged over all targets over all screens. Additionally, black circles visualize the mean gaze estimations.

5.4 Discussion

Our results show that – on a single display – *GazeProjector* achieves an average gaze estimation accuracy of 1.78° compared to 2.64° for *MT*, and 2.65° for *HO*. We used the same calibration grid recorded for both the near and far location, resulting in different visual angles in view space: the size of the calibrated visual field decreases when distance increases. Thus, the near calibration achieves better results than the far one. When used on multiple displays (and only being calibrated on a single screen), *GazeProjector* achieves an average gaze estimation accuracy of 2.47°, compared to 3.60° for *MT* over all modes and target screens. Although this accuracy is slightly lower than the 0.5°–1° reported for the Pupil Labs eye tracking glasses under ideal conditions (i.e., in a stationary desktop setting with a 27-inch screen and optimal lighting conditions ([78]), we achieve this accuracy in a fully unconstrained, pervasive interaction setting.

The poor results of *MT* can be explained as follows. When using the AR marker tracking approach, the complete display is always tracked, which can lead to inaccuracies while gaze mapping. In contrast to that, *GazeProjector* computes the projective transformation based on a partial area of the display (with a high density of image features). Thus, the computed mapping and the gaze estimation is much more accurate. With Head Tracking we also have the same issue, as we are just transforming the coordinates in the scene camera's frame. In the multi-display setup, the large error of *MT* comes from the fact that the software does not know which screen the user is currently focusing on, if one is located behind another (i.e., they overlap in the camera image). For example, the participant might be holding the iPad in front of their face, but the software also recognized the AR markers of the wall or the surface behind

it. *GazeProjector* prevents this issue, as the feature tracking and matching algorithm will always compute the projection for the screen the user is currently focusing on (i.e., the most prominent screen in the eye tracker's scene camera image).

Pervasive Settings

The first advantage of *GazeProjector* is its suitability for eye tracking in complex environments with multiple screens and locations of the user, as defined in the *Ubiquitous Computing for Eye Tracking Continuum* (Section 1.1, Figure 1.1, (K x L) : M : 1 complexity). Current approaches that allow for gaze interaction on multiple displays using monocular mobile eye trackers require heavyweight external motion capturing systems or visual markers. While motion capture systems allow for high-precision tracking, they (1) are costly and (2) cannot easily be installed in public environments. Markers mitigate this, but have another drawback: all displays have to be augmented with them either with printed ones attached to a display's frame [209, 17], or digital ones shown on the display. However, printed markers quickly clutter the environment, in particular in settings with a large number of displays. While digital markers could be shown only on demand, they still take away display space and "compete" with the main content.

While binocular systems can automatically compensate for vergence error (cf. Section 3.1.1), estimating gaze in display coordinates still requires tracking changes in the user's position and orientation relative to these displays. This severely limits the use of these devices to instrumented environments. *GazeProjector*, however, allows users to interact from arbitrary locations and orientations relative to multiple displays without requirement need and, as our experimental results show, *GazeProjector* does so without lowering the accuracy. Thus, our approach allows for unconstrained and seamless gaze interaction with multiple displays while users are on the move. Since our approach theoretically allows for multi-user settings (see architecture in Section 5.7), it can be used to create pervasive gaze-based interfaces (see Figure 1.1, (K x L) : M : N complexity). Practically, the current implementation might hinder such scenarios, as it requires many computers to communicate with each other.

Display Visibility & Multi-Display Interactions

Display tracking using visual markers requires the whole target display to be visible in the eye tracker scene camera's field of view during calibration and interaction. In contrast, *GazeProjector* relies on natural feature tracking and provides competitive gaze

estimation accuracy even if only a fraction of the target display is visible. Naturally, the larger the visible portion of the display, the lower the tracking error. However, we found that a quarter of the display is usually sufficient, provided that enough features (e.g., high frequencies) are found in that portion. This allows *GazeProjector* to work on much larger displays as well as with more extreme head movements than current eye trackers.

Our results show that *GazeProjector* provides robust gaze estimation accuracy for different displays of different sizes without a need for recalibrating the eye tracker to each of the displays. Instead, the eye tracker only needs to be calibrated once (on *any* display) and gaze estimates are then automatically mapped to the other displays during runtime. Applying our calibration method in the presented experiments, we were still able to achieve an accuracy of 3.24° when the eye tracker had been calibrated on a 9.7-inch iPad Air screen. This is a significant advancement over state-of-the-art gaze estimation approaches.

Head Movement and Orientation

We further found that head movements are more prevalent in gaze interaction with large displays compared to smaller displays (e.g., mobile devices). This finding is in line with controlled laboratory studies on human vision: humans employ head movements for gaze shifts with ocular orbital eccentricity exceeding 20° [175]. While head movements pose a significant challenge to current head-mounted eye trackers, *GazeProjector* proved to be robust to head movements, which is an essential feature for using head-mounted eye tracking systems for large screens.

Limitations

Despite its numerous advantages over state-of-the-art eye-tracking systems, *GazeProjector* also comes with some limitations: first, our current implementation requires continuous snapshots of the target displays to be transferred to a central server. Consequently, all displays need to be registered with such a server a priori. Furthermore, increasing the number of displays also increases the network load for transferring real-time updates of a display's content. However, we believe that future network technologies may overcome this limitation.

Second, *GazeProjector's* gaze estimation accuracy depends on the quality of image data from the scene camera: these cameras usually come with wide-angle lenses to

cover a larger field of view, resulting in smaller representations of a target display. This may increase errors in the transformation matrix due to insufficient image features. This technical limitation can be overcome by using different lenses for scene cameras. Furthermore, as with all optical tracking systems, environmental conditions such as changes in lighting will affect our system, as this influences the video quality of the scene camera. And finally, *GazeProjector's* accuracy is dependent on the number of features of a display's content [57], thus requiring feature-rich content on displays. For multiple displays, we found that wallpapers in Windows are sufficiently different. Here, the server can detect potential similarities across displays through feature matching of their respective content.

Nevertheless, we believe that *GazeProjector* is a promising system that realizes continuous gaze-based interaction in pervasive settings, in particular for multi-display environments.

5.5 Application to research question

In this section we will work out the details of how the concept of *GazeProjector* contributes to the problems defined in Section 1.4 and the derived research questions (cf. Section 1.6). In particular, we will first formulate answers to further sub-questions of this thesis. After that, we will highlight a possible integration into the Collaborative Newspaper application (cf. Section 1.5).

Back of the beginning of this thesis, we formulated five coherent problems within the current area of head-mounted eye tracking that prevent the application of a head-mounted eye tracker as a ubiquitous computing device. As discussed in Chapter 1 (Section 1.4), the source of error in using head-mounted eye tracking devices that makes the creation of gaze-based interfaces difficult is mainly a result of a poor calibration. Calibration is performed by looking at a number of pre-defined visual stimuli. While the user is fixating the calibration target, data is sampled that consists of pupil positions in the eye image, the physical orientation of the eye and the location of the target on the calibration plane (i.e. the scene camera images). The process itself leads to many issues. The problem of calibration drift was already discussed in Section 4.

In the current chapter, we consider a possible solution for the problem of invariance. The calibration is typically performed for a fixed position and orientation of the user to a specific object that gaze should be estimated on. Moreover, in a multi-user

scenario, the process of calibration has to be done for each user separately (discussed in Chapter 3.2).

In Section 5.2 of this chapter, we presented the concept of *GazeProjector*, a system to enable accurate gaze estimation on various surfaces (e.g., displays) independently of the user's position and orientation. With *GazeProjector* we require the user to calibrate the head-mounted eye tracker only once per user. The procedure of looking at the visual stimuli to obtain the pupil-to-gaze mapping can be performed on any display, independent of the one the user's gaze is estimated on. As we highlighted in Section 5.2, the underlying idea of our approach is to continuously estimate the position (including orientation) between the eye tracker's scene camera and the surfaces (in our example, displays) in the environment using natural feature tracking. Although the calibration is now independent of a potential target display, it is noteworthy that each person still has to accomplish the task of calibration prior to usage.

However, the work presented in this chapter is a key achievement to formulate an answer to the research question raised in Section 1.6. Remember that we reduced the overarching question into eight smaller sub-questions that are more specific. After the detailed elaboration of the concept and implementation of *GazeProjector* in Section 5.2 as well as the thorough evaluation in two separate experiments described in Section 5.3, we can formulate the answers to the following questions:

4. How can we overcome the problem that the calibration is orientation, location and target dependent?

5. How can we achieve highly accurate and seamless gaze estimation across multiple surfaces?

In order to resolve the issue of the calibration being invariant when changing the user's location and orientation in front of the target object (e.g., the display on which gaze should be estimated), we have to follow these three steps:

1. Initially we have to calibrate the eye tracker, i.e. map the pupil positions from the eye to the scene camera's coordinate system. Remember that we are able to calibrate on an arbitrary display.

2. We track the spatial relationship between the eye tracker and a specific display from the device's perspective. Hence, we do not require any external tracking system.

3. Finally, we map the 2D gaze positions from the scene camera's to the display's coordinate space using this information.

With *GazeProjector*, we proposed a method (cf. Section 5.2.2) consisting of the above three steps. According to our proof-of-concept implementation (described in Section 5.2.3), we can definitely answer the questions from above. With *GazeProjector* we are able to finally overcome the aforementioned calibration issues of the and achieve highly accurate gaze estimation across multiple displays. The results of the two experiments presented in Section 5.3 additionally underline the advantages of our approach over existing methods in terms of gaze estimation accuracy and scalability.

Having successfully answered sub-questions numbers 4 and 5, we want to illustrate the benefits of *GazeProjector* in an interactive scenario. For this purpose, we want to return to the Collaborative Newspaper application [101], presented in Section 1.5, and the resultant problems for multi-user eye tracking scenarios. We discussed the use of visual AR markers that were shown on the screen to realize the tracking of the display in the scene camera images. This feature is required to correctly map the users' gaze onto the screen and the right text position. It is a key requirement for the system to properly scroll the text being read. In the example, we estimated the portion of the screen space that is wasted to display the AR markers. It turned out that up to one third of the screen must be used for the marker tracking to work.

GazeProjector provides an opportunity to replace the well-established marker tracking technique. The only requirement for the system to work is that the screen's content contains many visual features. The default layout of the Collaborative Newspaper application consists of several news articles, showing a static image and text accordingly (cf. Figure 1.8). There are at least three news articles displayed, sufficient for the feature tracking algorithm to work properly. Consequently, integrating *GazeProjector* into the application will definitely solve the problem of wasting display space, and instead use it to show more content. Moreover, the gaze estimation is proven to be more accurate than with marker tracking, while people are still able to move freely in front of the public screen. Finally, *GazeProjector* can calibrate the head-mounted eye tracker on any screen in the environment while achieving high gaze estimation accuracy, also on other displays. Thus, the interaction, i.e. the reading, will not be disturbed by any calibration procedure, enabling spontaneous use. One could simply use a second display to calibrate the head-mounted device.

5.6 Application of GazeProjector in a Real-World Setting

In this section we will show the applicability of *GazeProjector* in a real-world scenario for gaze estimation on any object in the environment. In the following we will present our work to realize a novel way of using head-mounted eye tracking devices for in-the-wild data analysis utilizing the concept described in Section 5.2. In particular, we discuss how to use head-mounted eye tracking together with Google Street View data for pedestrian navigation. We start with an introduction on the topic of navigation and landmark generation and briefly review the existing work in this research area. After that, we will present our approach utilizing *GazeProjector* to efficiently analyze head-mounted eye tracking data. Finally, we will highlight the results of a small pilot study to demonstrate that the concept works well.

5.6.1 Eye Tracking & Navigation

Navigating in partially familiar or completely unfamiliar environments is a complex task that requires spatial reasoning, memorization, and close examination of the surroundings. Incorporating landmarks – geographic objects that help structuring human mental representations of space [160] – in the route description has been proven to enhance understanding and performance of way-finding [192, 39]. May et al. [116] found that landmarks, namely bars, specific shops, restaurants, supermarkets, gas stations, traffic lights and parks should be the primary means of providing directions to pedestrians.

Even though these advantages are well known, very few commercial systems exist that include landmarks in the description process. The primary reason for that is the lack of available landmark data, as stated by Giannopoulos et al. [46]. Many existing approaches require active user input [204, 55, 14]. But selection of landmarks might be biased when using a specific source (e.g., social network), or restricted by the available data for certain characteristics of objects. A possible way to overcome these issues is to use eye tracking. In the area of geographic exploration, eye-tracking has been employed successfully before. For example, Kiefer et al. [83] identified factors that influence the duration of the visual exploration of a city panorama. Eye tracking has also been used to understand the process of self-localization on a physical paper-based map [84]. Giannopoulos et al. [46] developed a navigation system that incorporates the user's gaze at decision points to communicate the route. This prior work makes

eye tracking a promising approach to automatically identify landmarks as well. Since landmarks are normally characterized by a high visual saliency, they should attract the visual attention of the user [203].

The overall problem of the existing approaches is the mapping of people's gaze on the correct object (e.g., building) in the environment to automatically extract landmarks. We present our approach towards a system to automatically infer suitable landmarks for pedestrian navigation instructions from head-mounted eye-tracking data. Utilizing *GazeProjectior*, we match the video feed of the eye tracker's scene camera to Google Street View imagery. Thus, our system is able to cluster the visual attention of the users on specific elements of the environment. From this aggregation, we can infer the saliency of the environmental elements and the potential for use as landmarks for navigational instructions. In the following, we focus on extracting visual landmarks using eye-tracking data to show the applicability of *GazeProjector* in a large complex environment.

5.6.2 Automatic Gaze Estimation In The wild

The computation of people's viewing and gazing behavior in an in-the-wild setting (i.e., on buildings and other surroundings), where the environment is dynamically

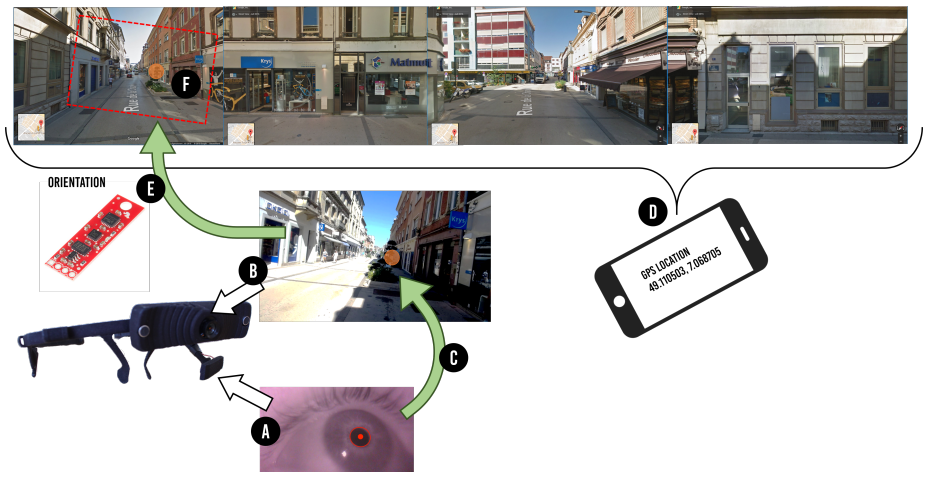


Figure 5.16: Concept of the system using *GazeProjector* for gaze estimation on objects in the environment: The head-mounted eye tracker (a,b) enables gaze estiamtion in the user's field of view obtained through the scene camera (c). With the GPS location, possible candidates for the environment the user is facing are estimated (d). The orientation sensor (compass) reduces the environment images to the correct one according to the user's head orientation (e). *GazeProjector* matches the user's gazepoint from the scene camera image to the selected environment (f).

changing, is a key requirement for an automatic extraction of visual landmarks. Figure 5.16 depicts the concept of applying *GazeProjector* in such a setting. The goal to estimate a user's gaze on an object (i.e., a building) in the environment is realized as follows: As before, we are using a head-mounted eye tracker to obtain a person's gaze information based on a one-time 9-point calibration (e.g., done on a laptop display). By applying the method of *GazeProjector*, the gaze points can be mapped from the eye tracker's scene camera to corresponding gaze points on a display in the environment the user is currently looking at (cf. Section 5.2). coming back to our scenario, the environment serves as a display, in simple terms. However, as this setting is quite a bit more complex, we need more data for an accurate gaze estimation.

To get information about the user's current surroundings, we record the location through a mobile phone. With this, potential candidates from the environment are obtained via Google Street View imagery (details on the implementation are presented in the next section). These 'screenshots' of the environment (comparable with multiple displays placed in the environment) represent a 360° view around the user's location. To reduce the number of candidates, we additionally include the orientation of the user.

A 9-degree-of-freedom (accelerometer, magnetometer, gyroscope) sensor, attached to the head-mounted eye tracker, records the exact orientation of the user's head in 3D. Together with the location information, one single candidate can be extracted. Applying *GazeProjector* is comparable to the gaze estimation in a single display scenario. The eye tracker's scene camera image is used as a template that is searched for in the corresponding Google Street View image. If the template matches, we can calculate the projective transformation (i.e., the homography matrix H) from the recorded scene camera image into the Street View image, as described in Section 5.2.2. The transformed gaze point on the Street View image/screenshot represents the user's true gaze in the environment. After clustering the gaze points to estimate the user's attention, the set of visually most attractive landmarks could be extracted.

In sum, we continuously record a person's location, head orientation and gaze data. All data streams have to be synchronized for an accurate estimation of the user's gaze in the environment, and thus of their visual attention. Mapping the user's gaze onto the environment in an automated fashion, used to be problematic in the past due to

the lack of holistic imagery of the environment. In the following we will present the implementation of the above concept.

5.6.3 Implementation

Figure 5.17 shows the adapted architecture of the system, according to the one presented in Section 5.2.3. It consists of the same three main components – *Tracker, Engine* and *Trackable*. For reasons of mobility, we moved all software parts to one computer. In the following we will provide more details on the implementation of each component.

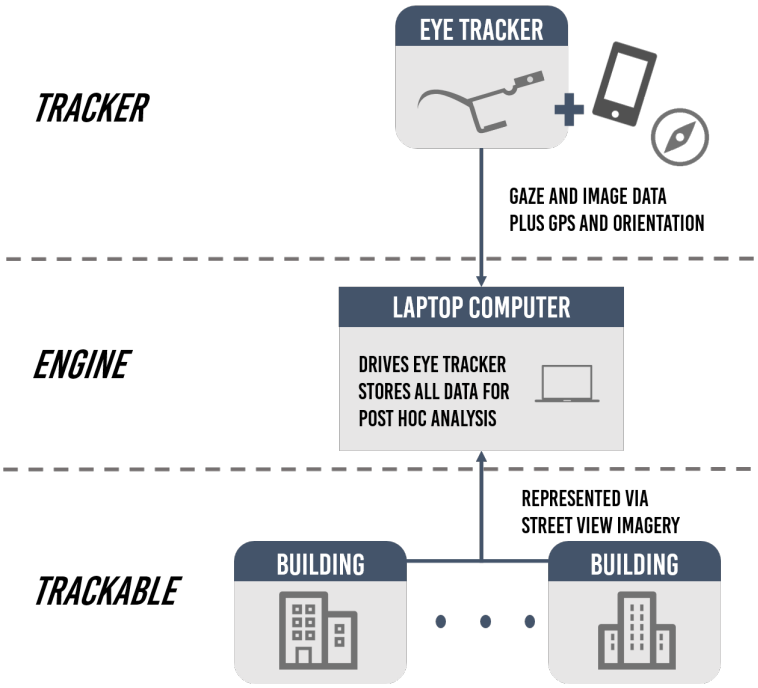


Figure 5.17: Architecture of the system using *GazeProjector* for gaze estimation on objects in the environment.

Tracker – a hardware prototype that bundles different devices into a wearable solution to record all required data at once. Figure 5.18 depicts the hardware implementation of the device, which consists of the following components: (1) a monocular head-mounted Pupil Labs eye tracker [78] that is connected via USB to a laptop computer; (2) a 9-DOF inertial measurement unit (IMU)[7], attached on the eye tracker's frame, that is powered by a battery; and (3) an iPhone SE to record GPS locations.

[7]Sparkfun 9-DOF sensor stick with ADXL345 Accelerometer, HMC5883L 3-axis magnetometer, and ITG-3200 gyroscope

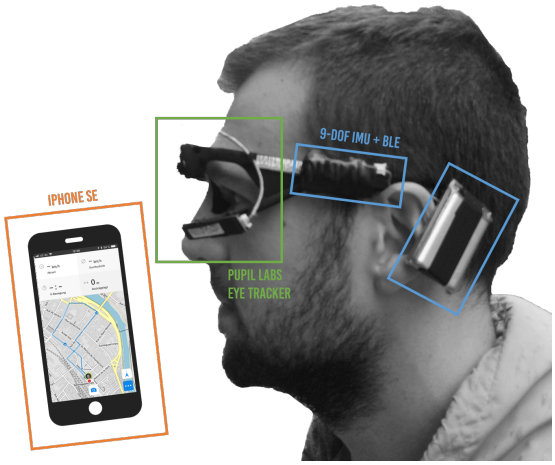


Figure 5.18: Head-mounted device consisting of Pupil Labs eye tracker and a wireless inertial measurement unit to detect the user's head orientation; iPhone SE used for GPS location.

The mounted IMU is able to measure the user's absolute head orientation as yaw, pitch, and roll independently of the eye-tracking device. We used an RFDuino4 to connect the IMU via Bluetooth LE to the Laptop. The way we mounted the sensor breakout shield ensured that the x-axis was pointing forward, the y-axis to the right and z-axis down with respect to the viewing direction. The three sensors were mounted on the left earpiece of the headset, where clearance to other cables and the cameras was sufficient in terms of magnetometer deviation. We took special care with the compass calibration to compensate for hard and soft iron errors. We experienced deviations from magnetic north due to electromagnetic induction when the eye-tracking system was running.

The GPS locations were stored as a GPX track in the cloud, using komoot[8], a web-based routing framework. We used the *Pupil Capture* software, as it provides a recording functionality to record the user's field of view, by storing the scene view images, and the gaze information in a dedicated folder[9]. The data of the IMU (magnetometer, accelerometer and gyroscope with timestamp) was streamed via a UDP socket and stored as CSV data. All software is developed in Python.

Trackable – the environment on which the users' gaze should be estimated. It is represented by multiple screenshots provided through Google Street View imagery, queried by the laptop computer. Listing 5.5 shows the implementation of the track-

[8]https://www.komoot.de/
[9]https://docs.pupil-labs.com/#recording

able to get screenshots of the environment. We make use of the Google RESTful APIs for Maps Static[10] and Street View[11]. To obtain a map showing the route walked by a user, a one-time query is necessary (Listing 5.5, lines 9–17). The screenshots of the environment are based on the GPS location (as latitude and longitude) and potential information about the head orientation (Listing 5.5, lines 20–48). In the case that no orientation information is available, a 360° image (i.e. a panorama view) is created by concatenating four Street View images with a field of view of about 90° (Listing 5.5, lines 34–44). All necessary data can be queried by passing the location, heading, pitch and size of the target image for the Street View image, and the zoom level for the Static Map image.

Code Listing 5.5: Using Google Street View to get a screenshot of the environment

```
1  class GoogleMaps():
2  #doc: https://developers.google.com/maps/documentation/streetview/
                                    intro#url_parameter
3    def __init__(self):
4
5    self.street_base = "https://maps.googleapis.com/maps/api/streetview
                                    ?size=640x360&location="
6    self.maps_base = "https://maps.googleapis.com/maps/api/staticmap?
                                    zoom=17&size=380x640&maptype=
                                    roadmap&markers=color:blue%7Clabel:
                                    P%7C"
7
8  # returns an image of the map displaying the path
9  def get_static_map(self, lat, lng, path):
10   parameters = str(lat) + "," + str(lng) + "&path=" + path + "&key="
                                    + GOOGLE_MAPS_API_KEY
11   url = self.maps_base + parameters
12
13   resp = urllib.urlopen(url)
14   image = np.asarray(bytearray(resp.read()), dtype="uint8")
15   image = cv2.imdecode(image, cv2.IMREAD_COLOR)
16
17   return image
18
19 # get street view image
20 def get_street_view(self, lat, lng, heading=None, pitch=None):
21   parameters = str(lat) + "," + str(lng) + "&key=" +
                                    GOOGLE_MAPS_API_KEY
22   image = None
```

[10] https://developers.google.com/maps/documentation/maps-static/intro
[11] https://developers.google.com/maps/documentation/streetview

```
23      url = ""
24      if heading:
25        parameters = str(lat) + "," + str(lng) + "&heading=" + str(
                                       heading) + "&pitch=" + str(pitch) +
                                        "&fov=90" + "&key=" +
                                       GOOGLE_MAPS_API_KEY
26
27        url = self.street_base + parameters
28
29        resp = urllib.urlopen(url)
30        image = np.asarray(bytearray(resp.read()), dtype="uint8")
31        image = cv2.imdecode(image, cv2.IMREAD_COLOR)
32      else:
33        #stitch them together
34        pano = np.zeros((360, 2560,3), np.uint8)
35        idx = 0
36        for h in [0,90,180,270]:
37          parameters = str(lat) + "," + str(lng) + "&heading=" + str(h) +
                                        "&pitch=" + str(p) + "&fov=90" + "
                                       &key=" + GOOGLE_MAPS_API_KEY
38
39          url = self.street_base + parameters
40          resp = urllib.urlopen(url)
41          image = np.asarray(bytearray(resp.read()), dtype="uint8")
42          image = cv2.imdecode(image, cv2.IMREAD_COLOR)
43          pano[:360, idx * 640:640+idx*640] = image
44          idx += 1
45
46        images = pano
47
48      return image, url
```

Engine – the software running on the laptop that uses all provided data streams to performs all computations, necessary for the gaze estimation in the environment. As we use the *Pupil Capture* software to record all eye tracking data, we also use the *Pupil Player*[12] software to play back and analyze the recorded data. This complete Pupil Labs framework is developed in Python and easily extensible through plugins. We implemented the following features: (1) automatic correlation and playback of all recorded data, i.e. a folder containing files recorded via the Pupil Capture folder, a GPX track provided by komoot and the user's head orientations stored in a CSV file. The synchronization is done based on timestamps. (2) The recorded gaze information is processed using *GazeProjector*. Listing 5.6 shows parts of the implementation of

[12] https://docs.pupil-labs.com/#pupil-player

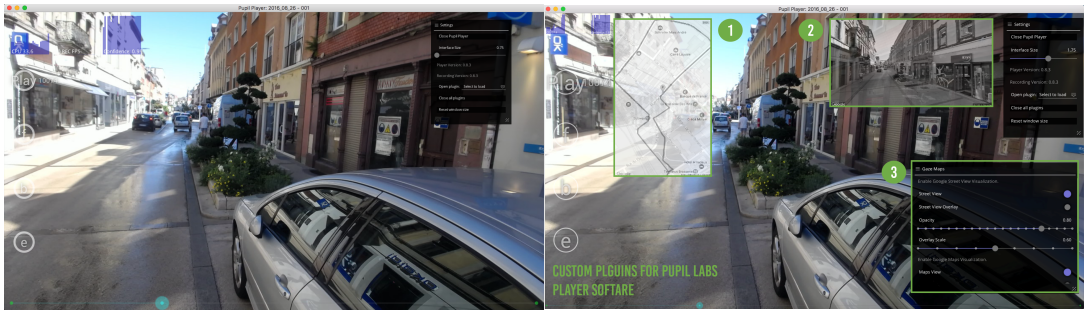
the class Engine. The actual transformation of the gaze point is implemented in the *update* method (Listing 5.6, lines 10–30). We re-implemented the feature tracking and matching algorithm of *GazeProjector* using the OpenCV 3.1 bindings for Python. We use the same parameters and algorithms as presented in Section 5.2.2. For faster processing, we downscale the Street View images to 640 x 480 and the scene camera images to 360 x 180 pixels. We achieve up to 30 fps, i.e. the processing can be done in real time, and thus also during the data recording. (3) The user interface of the Pupil Player was extended to highlight the current location of the user on Google Maps and the extracted user's view and gaze point.

Code Listing 5.6: Using Google Street View to get a screenshot of the environment

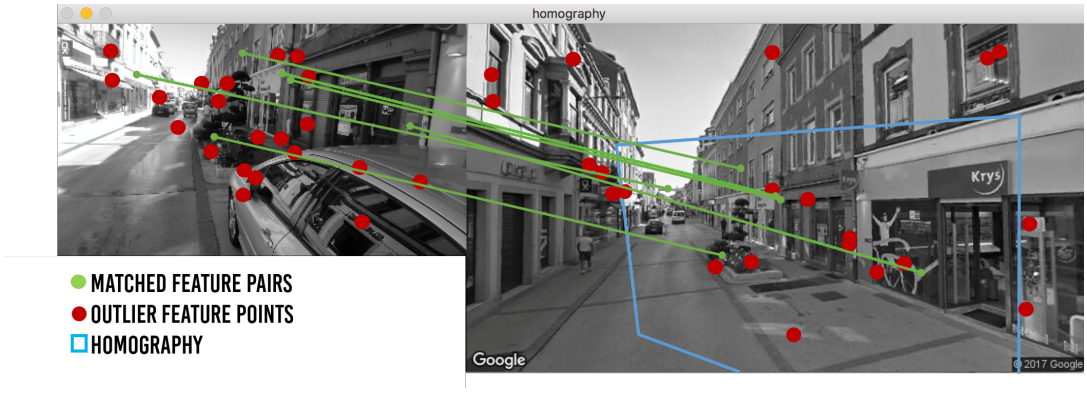
```
1  class Engine(plugin):
2    def __init__(self,g_pool,street_view=True,street_view_window=False,
                                    street_view_alpha=0.8,street_scale=
                                    0.8,maps_view=True,maps_view_window
                                    =False,maps_view_alpha=0.8,
                                    maps_scale=0.8,move_around=1,pos=[(
                                    150,10),(800,10)],gps_data=True,
                                    orientation_data=False,gaze_data=
                                    True, broadcast=False):
3
4      self.algorithm = GazeProjector("freak")
5      self.gmaps = GoogleMaps()
6
7      #more initialization concerning GUI
8
9    # method called every frame
10   def update(self,frame,events):
11     positions = events.get('gps_positions', [])
12     imu = events.get('imu_positions', [])
13
14     if imu:
15       self.heading = imu[0]['heading']#+180
16       self.pitch = imu[0]['pitch']
17
18     if positions:
19       self.positions = positions
20
21     self.street_view_image, self.url = self.gmaps.get_street_view(
                                    self.positions[0]['lat'], self.
                                    positions[0]['lng'], self.heading,
                                    self.pitch)
22
```

```
23        # computes homography H and matches
24        H, matches, good = self.gp.find_homography(small_frame_g, cv2.
                                      cvtColor(img, cv2.COLOR_BGR2GRAY),
                                      idx)
25
26        if H != None and len(matches) > num_matches:
27          homography = H
28
29          for pt in events.get('gaze_positions',[]):
30            mapped_gaze = H.dot(pt)
```

Figure 5.19 illustrates the extended *Pupil Player*. In the leftmost image, the software is displaying the current field of view, captured by the scene camera. The center image depicts the plugin used to visualize the aforementioned data. Specifically, it overlays the scene camera image with the current Google Street View image, and a



(a) (b)



(c)

Figure 5.19: Playback and analysis of the recorded data: (a) The standard Pupil Player is able to play back the data recorded by the eye tracker. (b) The developed custom plugin is able to correlate GPS location, head orientation and eye tracking data. For manual analysis, the path on a Google Map is shown (1), as well as the Street View image (2). The user is able to set some parameters for the visualization (3). (c) Using feature tracking, the plugin determines the transformation between the scene camera's image and the Google Street View image.

map indicating the current location of the user. The rightmost image highlights the result of the feature tracking and matching algorithm, used to map the scene camera image to a Google Street View image.

5.6.4 Evaluation

We conducted a user experiment to verify our proof-of-concept implementation. In particular, we want to investigate the performance of *GazeProjector's* concept in real-world environments by evaluating its algorithm when used with Google Street View images. To do this, we equipped six participants between 22 and 58 years old ($M = 39.6, SD = 14.8$ years), 4 male and 2 female, with our device bundle. All participants were recruited from a local university campus and had corrected or normal vision; none reported any visual impairments (e.g., color blindness).

The task of the participants was to walk a pre-defined route through the nearby French town of Saareguemines. All participants rated their familiarity with this city as "rather unfamiliar" or "unfamiliar". The generated route went through the inner part of the city and was constructed as a 1.5 km circular course, depicted in Figure 5.20. The participants were guided using the off-the-shelf audio navigation part of the Komoot Mobile App. As Wenczel et al. showed that the amount of visual attention on more salient landmarks is not affected by whether the user learned the route beforehand, or is incidentally learning it, we decided not to explain the route beforehand [203].

Each participant was first asked to calibrate the head-mounted eye tracker while standing. We used the built-in nine-point 2D calibration procedure of the Pupil Labs framework on a 15-inch laptop screen. Remember that *GazeProjector* allows calibration on any screen while maintaining a highly accurate gaze estimation accuracy on other displays. In our case the displays are the environment. After the calibration, we synchronized the software manually with the mobile phone used for location tracking. This was done by capturing a start-button press on the iPhones screen with the scene camera of the eye tracker. The participants were instructed to follow the audio instructions to finish the route.

During the task, all data was sampled in the following way: Right after the eye tracker calibration, each participant was asked to look straight ahead. We sampled data from the eye tracker for six seconds. This was necessary to have a set of samples

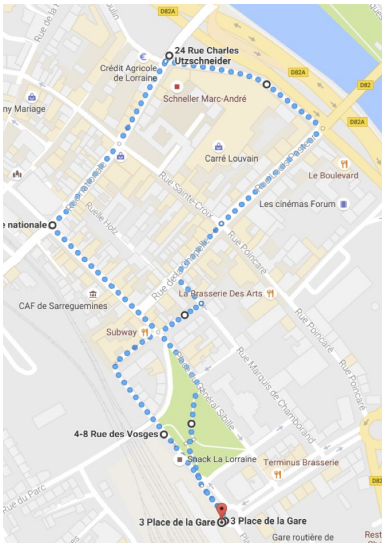


Figure 5.20: The route that was walked by the participants. Each person started through the park, beginning and ending in front of the railway station at the red marker.

to create a mapping between the gaze direction and head orientation relative to each other. On the way through the city, the scene camera video stream and gaze data were recorded through the Pupil eye tracking device at 30 Hz; IMU data and location data were recorded at 40Hz and 2Hz respectively.

Results

To do a first evaluation of our proof-of-concept prototype, we aggregated all recorded data of each participant, using the Pupil framework together with the developed plugins. We then computed the number of scene camera images which we were able to match to Google Street View images across all participants and recordings. In a frame-by-frame evaluation, we found that it is possible to successfully match 31.99% of the scene camera images to Street View image data ($SD = 5.6\%$) averaged over all participants. That means we are also able to compute the homography matrix (i.e., *GazeProjector's* basic algorithm needed for gaze estimation) along a third of the walked route.

We further wanted to investigate the variability in eye and head movements: Figure 5.21 plots the mean eye movements we recorded during the experiment. We noticed a horizontal eye movement of 7.22° on average ($SD = 3.26°$), compared to 5.42° in the vertical direction ($SD = 1.41°$). It also shows the same plot for the observed head movements. Here we saw 144.23° for horizontal head movements (yaw angle) on

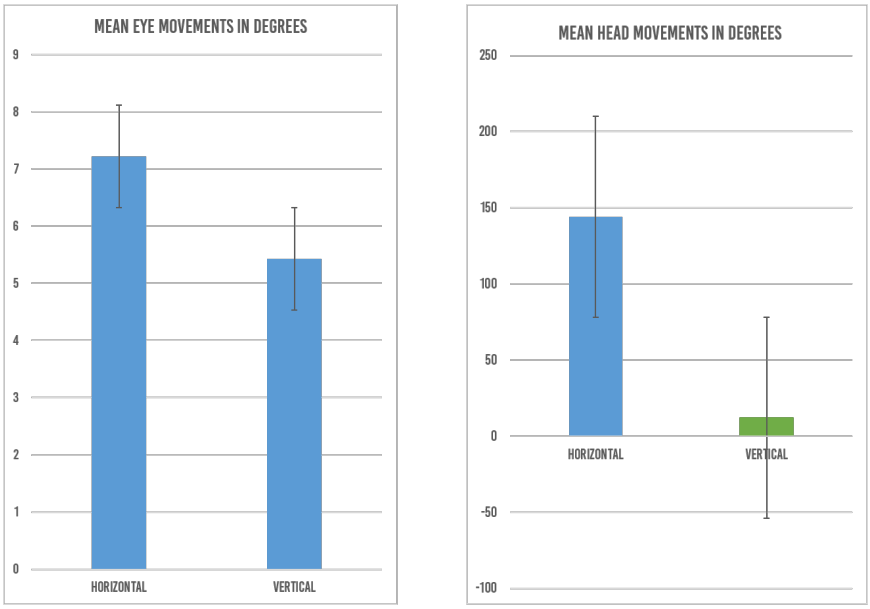


Figure 5.21: Left: Mean eye movement in degrees, horizontal vs. vertical direction. Right: Mean head movements in degrees, horizontal vs. vertical direction.

average ($SD = 86.51°$). In the vertical direction (pitch angle), we observed 12.06° on average ($SD = 12.07°$).

5.6.5 Discussion

The conducted experiment gives first insights into the applicability of *GazeProjector* in an in-the-wild scenario. We found that the feature tracking and matching algorithm is able to compute a homography matrix for almost a third of all image data, and thereby project the user's gaze into the environment. For this subset, the attention of the user can be extracted by computing the fixations based on the gaze behavior. The object at the location of the user's fixation in the environment (i.e., the Google Street View image) is defined as a potential landmark. To obtain further information about the attended object, further investigation is required. For example, Google Places[13] can be integrated to retrieve information about a building (e.g., whether it is a restaurant).

GazeProjector's poor performance can be explained as follows. We processed the raw data that was sampled. That means we did not take into account the fact that Google Street View images are usually taken from the center of the road. To increase the accuracy of the results, one would have to incorporate the offset between the user's

[13]https://developers.google.com/places/web-service/intro?hl=de

position and the position of the Google Street View imagery and adapt the orientation accordingly. Further, it is very likely that Google Street View images differ from the current scene images, as they could contain other objects like cars, or might have been recorded in a different season (e.g., winter versus spring).

We noticed a very small variability in the eye movement data, compared to the head movement. The observed 7.22° for horizontal movements is within the macular region of the peripheral system (see Chapter 2). The large values for head movements indicate that people tend to move their head instead of their eyes. On the one hand this could be caused by the fact that people were walking through an unfamiliar city and tried to see as much as possible. On the other hand it might be sufficient to use only the head orientation and location information, which would be a subject for further research.

5.7 Summary

In this chapter, we presented *GazeProjector*, an approach for accurate gaze estimation and seamless interaction with multiple large displays using head-mounted eye trackers, in Section 5.2. In contrast to existing systems, *GazeProjector* only requires a single calibration performed with an arbitrary display, and is robust to the user's location and orientation to the displays as well as head movements. Furthermore, *GazeProjector* works without external tracking equipment, such as motion capturing systems or markers attached to displays.

We conducted two experiments in which we compared *GazeProjector* to existing, well-established techniques (which require additional equipment), and found that our approach compares well to these techniques (see Section 5.3). When used on multiple displays, the results are even more promising. Overall, they underline the significant potential to finally bring gaze-based interaction into settings that involve gaze estimation on multiple displays and even with multiple locations (see Figure 5.1, complexity (K x L) : M : 1).

So far, we have tested *GazeProjector* in a laboratory environment, to gain insights into its performance compared to existing techniques, and in a real-world scenario (cf. Section 5.6). There, the automatic combination of head-mounted eye tracking data with Google Street View imagery was demonstrated. Accordingly, we enabled the mapping of gaze data into the environment, which can be used for automatic

extraction of visual landmarks for navigational tasks. We saw that especially in this real-world scenario, the gaze estimation on arbitrary objects in the environment is a complex task that cannot be fully solved using *GazeProjector's* concept, as it involves much more data than just the pure eye tracking data.

However, we believe that the method can be improved by including novel machine learning algorithms, in particular for object and scene detection, to take our approach one step further. In addition, we want to evaluate *GazeProjector's* performance with multiple users simultaneously. This will further contribute to the eye tracking community, as it has been virtually impossible to test eye-tracking systems on such large scales with multiple persons.

With the developed system presented in this chapter, we answered two more sub-questions of the research question this thesis is based upon (see Section 1.6). In particular, *GazeProjector* makes the calibration of a head-mounted eye tracking system invariant against changes in distance and orientation to the surface gaze should be estimated on. That is, the calibration plane, which is usually the screen used for the calibration procedure, is projected onto any surface in the environment that a user wants to interact with. Moreover, the gaze estimation accuracy remains constant across multiple displays (discussed in Sections 5.4 and 5.5). Moreover the application presented in Section 5.6 highlights the contribution of *GazeProjector* in a different domain of research.

However, it is still necessary to calibrate the head-mounted eye tracker prior to use. In the next chapter we will elaborate on the design process resulting in two systems that enable calibration-free gaze estimation. Finally, we show a system for real mobile eye tracking without the need for carrying a powerful laptop required to drive the eye tracking system.

Chapter 6

Mobile and Calibration Free Gaze Estimation Approach

In this chapter we will elucidate the development process of a mobile and calibration-free approach for gaze estimation that results in two connected systems. Both address the problem of calibration and the remaining issues of supervision and parallax error (cf. Section 1.4), which were not addressed so far. There we discussed that each gaze estimation method relies on a specific set of input parameters which have to be acquired through a calibration procedure, which in turn might differ depending on the method. To make a head-mounted eye tracker ready for usage, different approaches exist to carry out the calibration. Although there is the possibility to automatically make the device ready to use (as done with remote devices through an automatic calibration; see Section 3.2), the state-of-the-art head mounted eye trackers still implement a supervised calibration procedure (e.g., Tobii Pro Glasses 2[1] and Pupil Labs[2]).

In this chapter, we will present alternative gaze estimation approaches, which are based on corneal imaging (see Section 3.3) and which overcome the user dependent calibration procedure, including also the problems of invariance and calibration drift. With the developed approaches we are able to answer the remaining open sub-questions of the formulated research question of Chapter 1.6. Thus, they enable ubiquitous and pervasive eye tracking scenarios, i.e. gaze estimation for multiple persons on multiple objects and locations (cf. *Ubiquitous Computing for Eye Tracking Continuum*, Figure 6.1).

[1] `https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/`
[2] `https://docs.pupil-labs.com/\#calibration`

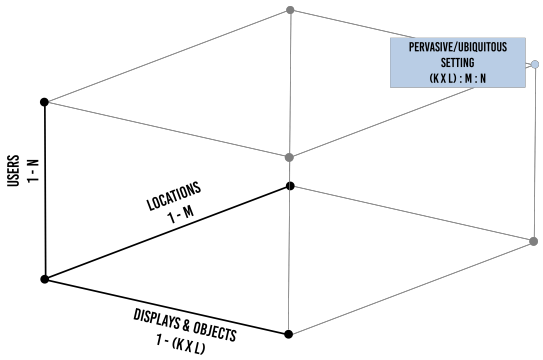


Figure 6.1: *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space highlighting the complexity ((K x L) : M : N) that is addressed by this chapter.

The results of this Chapter have been presented in two main publications [94, 98] and applied in three more works [97, 96, 99]. Work related to this Chapter can be found in Section 3.2 on eye tracker calibration and calibration-less approaches as well as in Section 3.3 on corneal imaging.

6.1 Introduction

The calibration procedure, required to make head-mounted eye tracking devices ready for use, involves various problems, as discussed in Section 1.4. Although much research exists to enhance the calibration process and counteract its issues (cf. Section 3.2), it remains the main problem that hinders gaze-enabled pervasive interfaces, as highlighted in Section 3.4. Desirable attributes of an eye tracker include minimal intrusiveness and obstruction, allowing for free movements while maintaining high accuracy, easy and flexible setup, and low cost [168]. Although the latest commercial eye trackers try to support all these attributes, they rely on a user-dependent calibration. Moreover, most of these devices need a direct connection to a powerful computer for real-time data processing. As a result, current head-mounted eye trackers are neither applicable for spontaneous interaction nor suitable to conduct long-lasting in-the-wild experiments.

So far we have presented several approaches to tackle most of the issues along with the calibration procedure of head-mounted eye tracking systems. But to complete the answer of the research question formulated in this thesis, we have to investigate the two remaining sub questions. This chapter presents two calibration-free gaze estimation approaches, which are based on corneal imaging (cf. Section 3.3). Figure 6.2

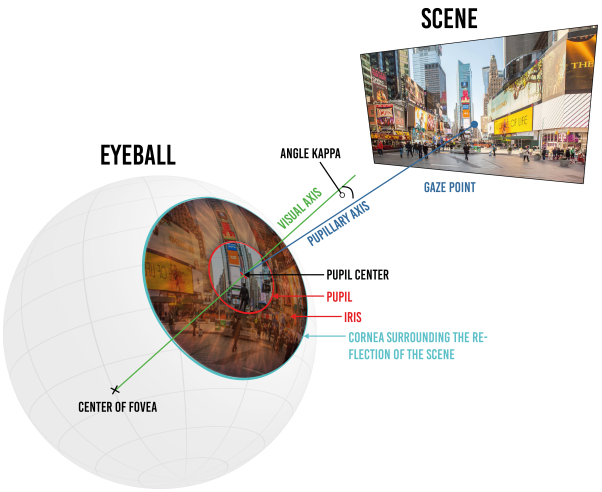


Figure 6.2: Basic idea of using the corneal images (i.e. the environment reflected on the human eye) for gaze estimation in the current scene.

depicts the underlying idea of using the environment, reflected on the human eye, for gaze estimation. Simply put, the usual scene images of a head-mounted eye tracker are replaced by corneal images, making the scene camera obsolete. Following this, a calibration procedure is no longer required. While exploring this approach, we aim to conclude the work that paves the way for head-mounted eye tracking devices that are applicable in ubiquitous scenarios (cf. *Ubiquitous Computing for Eye Tracking Continuum* Section 1.1).

The chapter is structured as follows: First, in Section 6.2, we will present the concept of *EyeMirror*, a head-mounted device that is able to approximate a person's gaze on surfaces (e.g., displays). In particular, we developed a device, consisting of a head-mounted camera, connected to a wearable minicomputer, capturing the corneal images. We propose two simple ways of gaze approximation, both based on natural feature tracking on the corneal images and the surfaces in the environment. Necessary eye features, such as the cornea (also known as the limbus) and the eye center are approximated. In two lab studies we compared variations of *EyeMirror* against established methods for gaze estimation in a display scenario, and investigated the effect of display content (i.e. number of features). We finally achieved an average accuracy of 4.03°.

Second, in Section 6.3, we will demonstrate the approach, called *hEYEbrid*, achieving an average gaze estimation accuracy of 2.19°. It is based on a hybrid concept that combines infrared eye images with corneal imaging in order to realize highly accurate

pupil tracking. For this, two eye cameras are mounted side-by-side on a glasses frame. In this way, the pupil can be tracked with high precision. This information is translated into the corneal image, which is used to create a connection to the environment, acting like a scene camera. In a user study with 20 participants, we evaluated our approach against an extended version of the system, and a state-of-the-art head-mounted Pupil Labs eye tracker. Finally we present a prototype, connecting the head-mounted device to a mobile phone, enabling gaze estimation in real time by using the dedicated *hEYEbrid* mobile application.

After presenting both developed systems, we will discuss their contribution to the fundamental research question of this thesis. To make the new generation of head-mounted eye tracking systems, presented in this Chapter, usable in eye tracking experiments, the application of classical methods to extract and analyze eye movement data has to be explored. Therefore, we will finally highlight a working approach to adapt standard eye tracking metrics such as fixation extraction on corneal imaging based gaze data in Section 6.5. With this method we can cluster and compress the eye tracking data and make it ready for further analysis (e.g. attention measurement and object detection).

Summing up, we provide the final answer to our research question: how to turn head-mounted eye trackers into ubiquitous computing devices? In addition, we underline the benefits of our systems with applications to demonstrate their applicability.

6.2 EyeMirror – Mobile Calibration-Free Gaze Approximation

In this Section we present *EyeMirror*, a mobile and wearable system for corneal imaging. It allows for calibration-free, moderately accurate gaze approximation on surfaces (e.g., displays) in the environment, while tolerating changes in distance and orientation to them. Our eyes literally serve as a mirror of our everyday doings and whereabouts. The parts that we can see externally are the white sclera, the iris, and the black pupil, the latter two of which are covered by the cornea. The corneal surface is covered by the tear fluid, which turns it into a highly reflective surface (see Section 2.1). The basic idea is to place a lightweight camera in front of the eye to capture close-up images of the human eye. They contain a distorted partial reflection of the user's current field of view (see Figure 6.3). These corneal images are compared with the

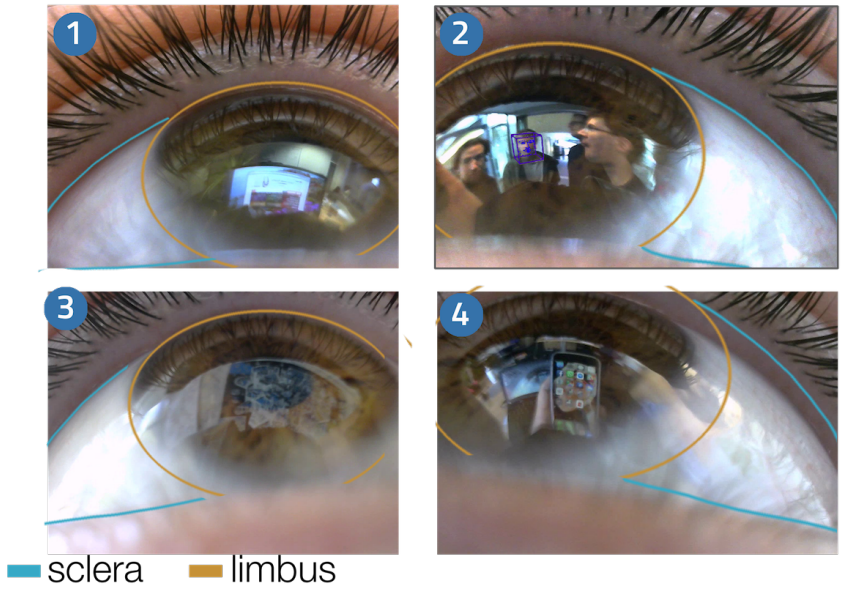


Figure 6.3: Corneal images, showing (1) a computer monitor, (2) faces, (3) a poster and (4) an iPhone6 display.

content of surfaces in the environment (e.g., displays, posters or books) using natural feature tracking.

We developed two approaches to approximate a person's gaze: The first version approximates the user's gaze point on the surface by clustering all key feature pairs, extracted between the corneal image and surface content (e.g., display). The second version approximates the pupil center and maps it to the surface, applying a transformation matrix based on the extracted key feature pairs (similar to the concept of *GazeProjector*; cf. Section 5.2). Both versions require knowledge about the surface's content. In the case of displays, the screen content can be streamed; for books and posters, the content has to be available. The underlying concept is solely based on natural feature tracking and a single head-mounted camera, which makes a calibration procedure unnecessary. As the device is connected to a wearable computer, users are able to freely walk around, which enables pervasive scenarios.

We conducted two consecutive laboratory experiments to evaluate our approach. In the first experiment with 10 participants we compared four versions of *EyeMirror* – the two already described approaches, and each approach using distortion-corrected corneal images – against a state-of-the-art Pupil Labs eye tracker [78], and using head orientation as gaze direction based on a Microsoft Kinect v2 sensor.In the first ex-

periment we were primarily interested in gaze estimation accuracy in a single-display scenario. The task was to look at different on-screen targets from multiple distances and orientations in front of a projected display. *EyeMirror* achieves moderate gaze estimation accuracy of about 5° in each version. The second experiment explored the effect of the number of features on gaze estimation accuracy. This is of great importance, as the system is based on natural feature tracking. In a single-desktop setting, we repeated the same task as in the first experiment, but changed between six different content types. We found no significant change in gaze estimation accuracy among five of them. One content type did not work, since it contained too few features. Hence, our work provides the following contributions:

- **Fully implemented wearable corneal reflection system** enabling **calibration-free** gaze mapping on ambient surfaces in real time.
- **Investigation of two algorithms** based on established concepts (natural feature tracking) executable on-board for use in the wild.
- **Guidelines for the quality of surfaces** (i.e., content); the device can be used in accordance with the results of the evaluations.

In the following parts of this section we will present *EyeMirror*'s concept and its implementation in detail. After that, we will present the two experiments to assess *EyeMirror*'s gaze estimation accuracy and the effect of the display content in a single-display scenario. Finally we will discuss our results, pointing out the current limitations and giving an outlook for future work.

Difference from Related Work

We already presented the necessary related work on which our approach is based in Chapter 3, in Sections 3.2 and 3.3. With *EyeMirror* we built a wearable mobile corneal imaging system using a single off-the-shelf webcam without any additional components, such as active infrared light [174] or optical parts [187, 188]. Iris contour detection, as well as tracking the eye center point, is based only on processing the close-up eye images using computer vision methods, and works without any highly complex computations based on a 3D model of the eye [50]. In our approach we solely investigate the method of natural feature tracking for calibration-free gaze approximation in real time. We use lightweight algorithms of a low computational complexity, executable on a single-board computer. Thus *EyeMirror* is a system made for the

exploration of gaze estimation in the wild. We evaluated our approach against established methods for gaze estimation, as well as the effect of the number of features used to align *EyeMirror*.

6.2.1 The EyeMirror System

The *EyeMirror* system is designed to detect known aspects visible in a person's field of view to estimate gaze on ambient surfaces (e.g., it is usable to measure attention on displays). The only hardware required for a working system is a single off-the-shelf RGB webcam. It is positioned underneath the eye to capture a close-up video, revealing objects in the near environment, reflected on the eye's pupil and iris. The camera is slightly rotatable and movable to center the eye in the image, as needed for an optimal reflection image. The camera frames are analyzed with image processing and computer vision methods.

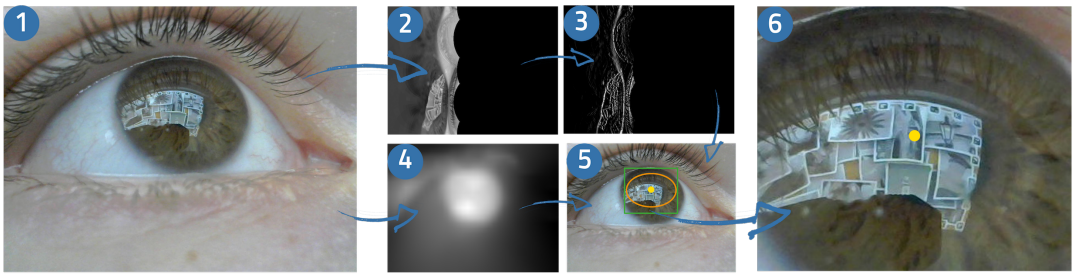


Figure 6.4: Limbus detection is done via ellipse fitting (5, marked by the green rectangle and the orange ellipse) after a polar transformation (2) and radial derivation (3). The eye center is extracted using image gradients (4). The final region of interest (6) contains the limbus and the eye center location (yellow circle).

Figure 6.4 illustrates the processing pipeline for extracting the limbus (iris contour) as well as the eye center, used to approximate the gaze location. The output is a cropped version of the raw input image, containing the region within the iris contour, used for further operations. In the following we explain the limbus extraction, the eye center localization, and gaze approximation on displays.

Limbus Detection

Figure 6.4(1–3) visualizes the pipeline used for limbus extraction. The algorithm continuously receives close-up images, shown in Figure 6.4(1). These raw video frames contain much information that is not needed for later processing: The sclera can occupy up to one-third of the image, depending on the eye pose and camera position, and does not reveal any relevant information. The eyelashes may corrupt the result

of later processing steps (e.g., feature tracking). Hence, these areas are removed from the raw image through a pre-defined region of interest.

The limbus of the eye has two main image characteristics: (1) it has an elliptical shape, and (2) it can be distinguished well from the surrounding structures (sclera and eyelids). A well-known approach is to look for the radial edges and classify them as the limbus boundary using ellipse fitting. *EyeMirror*'s algorithm is based on the approach by Wood et al. [205]. As suggested by them, the derivative of the polar transformation of the raw image is used, as shown in Figure 6.4(2–3). The maximum of each row is marked as a potential limbus point and fitted using a least-squares method for ellipse fitting. This approach is highly robust across different users and under varying lighting conditions, as it is not based on pre-defined thresholds for edge detection.

Eye Center Localization

To realize gaze estimation, it is necessary to have a reference point in the environment of what the user is currently looking at. The second version of *EyeMirror* declares the reflection at the eye center as the gaze reference point. We assume that this point correlates with the actual eye center. For eye center localization, the method of image gradients, proposed by Timm et al. [191], is used. Figure 6.4(3) shows the input image processed with the function they developed, which extracts the location where the most gradient vectors intersect. Like limbus detection, this method is robust under changing lighting conditions and various eye movements.

Gaze Estimation on Displays

The corneal images contain the actual part of the environment a person is currently looking at. In the case of interacting with a surface, e.g. a display, its content is partially reflected on the eye's iris and pupil, as shown in Figure 6.4(1). In *EyeMirror*, we developed two techniques for realizing gaze estimation on surfaces, such as public displays, using the corneal images (shown in Figure 6.5): The first approach is based on the concept of *GazeProjector*, developed in Section 5.2. The main idea is to compute the spatial relationship between the user's eye (and respectively the camera) and the surface the user is currently looking at. But, instead of utilizing the world camera of a mobile eye tracker (as in *GazeProjector*), we use the area within the limbus, reflecting the user's current field of view. The content of the surface (e.g., a display screenshot) a person is currently looking at is used as a template image that is searched for in the

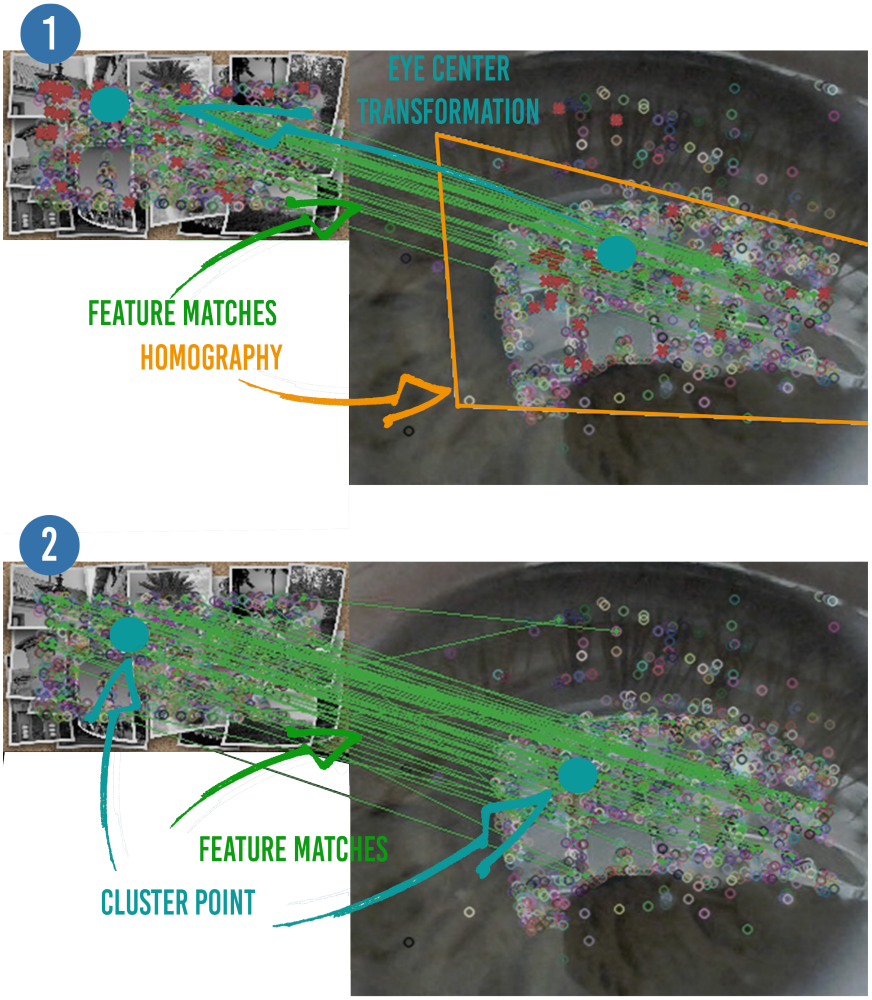


Figure 6.5: *EyeMirror*'s two approaches for gaze estimation on surfaces, here displays: (1) Gaze is approximated by transforming the extracted eye center onto the display using a homography matrix. (2) Gaze is approximated by clustering the image feature pairs of the display's content and the corneal image.

pre-processed corneal image. Both images are compared using natural image features. Whenever we find a match, the homography matrix, describing a transformation of points from the surface's image plane to the corneal image plane, is computed. As the homography is a bi-directional mapping, we use the inverse to transform the eye center point to the corresponding location on the surface. Figure 6.5(1) illustrates the procedure for gaze approximation on a display applying the described approach.

The second approach works in a similar manner in terms of calculating the relation between the eye reflection and the displays. In contrast to the previous version,

we are computing a k-means cluster for the found key feature pairs (with k = 1) to extract the gaze point instead of detecting the eye center. The result is immediately chosen as the current gaze point, as we assume it is usually located near to the pupil's center. The result is shown by the cyan point in Figure 6.5(2).

Note that most of the features are found and matched within the area of the pupil. The reasons are that (a) the distortion of the reflection in that part is relatively small for most eye poses, and (b) the corneal reflection is less noisy at the pupil than on the iris. This is caused by the fact that the reflection on the iris is mixed with its color, making it blurrier. Also, other distraction factors (e.g., contact lenses) can corrupt the reflection there. Both described methods are applicable to any surface providing feature-rich content (e.g., posters or books).

Implementation

The *EyeMirror* system consists of three components: (1) the head-mounted prototype built from a 3D-printed glasses frame [78] and a 3D-printed camera mount to place the webcam underneath a person's eye; (2) a single-board computer, running the *EyeMirror* software component, and (3) the surfaces in the environment. The software is designed to be easily extendable through plugins (e.g., further image processing algorithms). In the following, we will outline the implementation to use *EyeMirror* with displays.

Each mobile device and the displays are connected over WiFi. To keep the prototype device as lightweight as possible, we use a Logitech c270 camera (Figure 6.6(1)), capturing frames with a maximal resolution of 1280 x 960 px at 30 fps. The camera covers a 60° field of view and has a fixed focus of 4 mm. To enable the camera to capture with macro resolution, the camera has been stripped of its original housing and the glue around the lens was removed to adjust the focus manually (Figure 6.6(2)). The camera board is mounted on a 3D-printed glasses frame using a custom enclosure (Figure 6.6(3-4)). In this way, the camera is movable and rotatable; thus we do not need any further optical parts like prisms [187]. As the joint between the frame and a custom camera hold is lockable with a screw, the camera will not move without further effort. Consequently, the relationship between eye, camera and scene is fixed and re-adjustments are not done more often than for other head-mounted eye tracking devices. Figure 6.6 illustrates the building steps and the final hardware prototype.

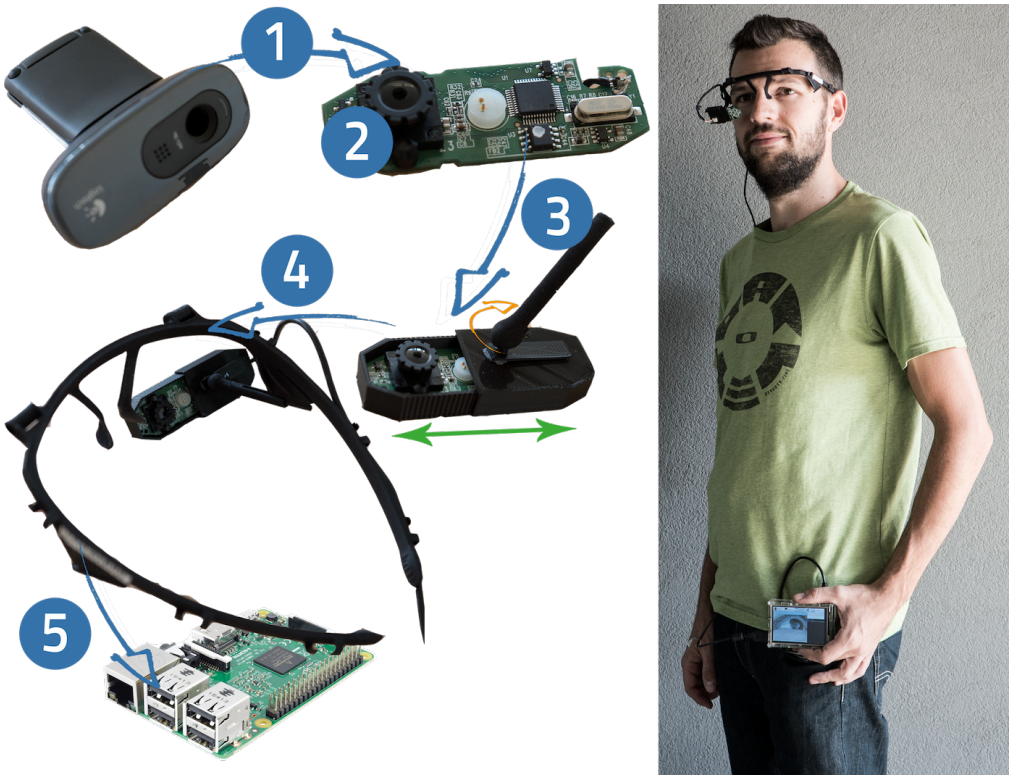


Figure 6.6: Building steps: (1) Extraction of camera board; (2) removing glue around the lens to adjust focus; (3) camera is built into a custom rotatable and movable enclosure and (4) mounted on a glasses frame, connected to a (5) RaspberryPi. On the right, the final wearable and fully functional prototype is shown.

All software components are developed in Python. For image processing methods, the OpenCV 3.0[3] library is used. It provides methods for FAST and FLANN to extract and match key features. For simple image operations (e.g., rotation and maxima search) we are using numpy[4], as it provides a fast way to process arrays. Eye images are captured with a resolution of 1280 x 960 px, resulting in an image with varying resolution after limbus extraction. Depending on the eye pose (i.e., the location of the limbus) the resolution ranges from 410 x 400 pixels to 400 x 360 pixels. Hence, these images are not further down-scaled, to preserve a high number of features.

The described pre-processing steps, i.e., limbus detection and eye center localization, are implemented as a separate sub-process. Listing 6.1 depicts the method to processes the raw camera frame. To speed up the computation, the input image is down-scaled using a Gaussian pyramid. As mentioned above, we extract the limbus

[3]http://opencv.org
[4]http://numpy.org

boundaries by looking for the maximum derivative (using the Sobel operator) of the image in the polar space (see Listing 6.1, lines 10–29). The method to find the eye center was taken from an open-source project[5]. Finally, both eye characteristics are scaled to match the initial size of the input image.

Code Listing 6.1: Limbus Detection & Eye Center Localization

```
1  def extract_eye_information_raw(frame):
2    # image pyramids to speed up computation
3    frame_pyr = make_gauss_pyr(frame)
4
5    # do polar transformation and the same procedure
6    # as proposed by Wood et al.
7    # find limbus with polar transformation and 2nd order derivative
8    new_frame = frame_pyr[2].copy()
9    gray = cv2.cvtColor(new_frame, cv2.COLOR_BGR2GRAY)
10   polar = cv2.logPolar(gray, (gray.shape[1]/2, gray.shape[0]/2), 45,
                                    cv2.WARP_FILL_OUTLIERS)
11
12   # first order dervative twice and add them
13   gxx = cv2.Sobel(polar, cv2.CV_8U, 1, 0, ksize=3)
14   derivative = np.add(gxx, gxx)
15
16   # find maxima and select them in the polar image
17   maxima = np.argmax(derivative, axis=1)
18   polar = cv2.cvtColor(polar, cv2.COLOR_GRAY2BGR)
19   for idx in range(0,len(maxima)):
20     polar[idx][maxima[idx]] = (0,0,255)
21
22   gray = cv2.logPolar(polar, (polar.shape[1]/2, polar.shape[0]/2), 45
                                    , cv2.WARP_INVERSE_MAP + cv2.
                                    WARP_FILL_OUTLIERS)
23
24   # store the maxima values
25   a,b = np.where((gray == [0,0,255]).all(axis = 2))
26   data_limbus =  np.transpose((b,a))
27
28   # ellipse fitting of the extracted limbus border points
29   limbus_bounding_rectangle = cv2.boundingRect(data_limbus)
30
31   #find eye center with gradients with method
32   # from https://github.com/errollw/EyeTab
33   pupil_x1, pupil_y1 = eye_center_locator_combined.find_pupil(
                                    new_frame,
```

[5] https://github.com/errollw/EyeTab

```
34        fast_width_grads=25.0,
35        fast_width_iso=80.0,
36        weight_grads=0.9,
37        weight_iso=0.1,
38        debug_index=0)
39
40    # remap the information to the initial image size
41    eye_center = (pupil_x1 * 2.0, pupil_y1 * 2.0)
42    limbus = (int(limbus_bounding_rectangle[0]*2), int(
                                    limbus_bounding_rectangle[1]*2),
                                    int(2*limbus_bounding_rectangle[2])
                                    , int(2*limbus_bounding_rectangle[3
                                    ]))
43
44    return (eye_center, limbus)
```

The implementation of both versions for gaze estimation is shown in Listing 6.2. Both methods rely on natural image feature tracking, as explained above. For image key feature detection and extraction we use FREAK [144]. To compute the corresponding matches, a FLANN based matcher [124] is used (see lines 3–9 and lines 13–21). The first version of gaze estimation computes the cluster of all key feature matches, applying k-means with k=1 (see Listing 6.2, lines 23–31). In the second version, the homography matrix is computed if more than four matches are found. Since we use the display content as a template that is searched for in the corneal image, we have to use the inverse of the homography to project the eye center to the screen (see Listing 6.2, lines 33–38).

Code Listing 6.2: Gaze Estimation

```
1   class FeatureTracker:
2   def __init__(self):
3   self.freak = cv2.xfeatures2d.FREAK_create()
4   self.detector = cv2.FastFeatureDetector_create()
5   self.detector.setThreshold(35)
6   norm = cv2.NORM_HAMMING
7
8   flann_params= dict(algorithm = 6, table_number = 6, key_size = 12,
                                    multi_probe_level = 1)
9   self.matcher = cv2.FlannBasedMatcher(flann_params)
10
11  def gazeEstimation(self, template, corneal_image, pupil):
12  # find the keypoints
13  kp1 = self.fast.detect(template,None)
14  kp2 = self.fast.detect(corneal_image,None)
```

```
15
16    # find the keypoints and descriptors with FREAK
17    kp1, desc1 = self.freak.compute(template,kp1)
18    kp2, desc2 = self.freak.compute(corneal_image,kp2)
19
20    raw_matches = self.matcher.knnMatch(desc1, trainDescriptors = desc2,
                                          k = 2)
21    p1, p2, kp_pairs = filter_matches(kp1, kp2, raw_matches)
22
23    # clustering approach
24    # Define criteria = ( type, max_iter = 10 , epsilon = 1.0 )
25    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1
                                          .0)
26    # Set flags (Just to avoid line break in the code)
27    flags = cv2.KMEANS_RANDOM_CENTERS
28    # Apply KMeans
29    compactness,labels,center = cv2.kmeans(p1,1,None,criteria,10,flags)
30
31    gaze_cluster = (int(center[0][0] * scale1), int(center[0][1] * scale1
                                          ))
32
33    # gaze estimation with homography
34    if len(p1) >= 4:
35    H, status = cv2.findHomography(p1, p2, cv2.RANSAC, 5.0)
36    # get the inverse
37    homography = np.linalg.inv(H_freak)
38    projected_gaze = cv2.perspectiveTransform(pupil, homography)
39
40    return (gaze_cluster, projected_gaze)
```

The software component runs on a Raspberry 3 single-board computer, based on Raspbian OS, with up to 25fps. It is mountable to a belt, enabling the whole system to be portable, while it is powered by a power bank. Camera frames are captured by using pyuvc[6], a python wrapper for libuvc[7], offered by Pupil Labs. It is a cross-platform library for USB video devices to access UVC devices. We use this library, as it is more robust and faster than the built-in OpenCV camera plugin. When using *EyeMirror* with displays, their content has to be streamed to the software component. We developed a python script that broadcasts screenshots with a resolution of 240 x 180 px to the RaspberryPi 3. In the case of other ambient surfaces (e.g., books), the images have to be known beforehand.

[6]https://github.com/pupil-labs/pyuvc
[7]https://github.com/ktossell/libuvc

6.2.2 Experiment I – Gaze Estimation on a Display

We conducted a controlled laboratory study to evaluate *EyeMirror*'s accuracy in estimating gaze on a projected display. We compared our two approaches with corresponding versions using distortion-corrected corneal images, a head-mounted Pupil Labs eye tracker that uses marker tracking to estimate its relative position to the display[8], and an approach with a Kinect v2 sensor that uses only the head position and orientation.

Independent Variables

We had two independent variables in the experiment, *Mode* (i.e., method used for gaze estimation) and *Location* (i.e., where participants were standing in front of the display), as follows:

Mode: We had 6 different modes for gaze estimation on the projected display: *EyeMirror-Pupil (EM-P)*, transforming the pupil center using a homography, and *EyeMirror-Cluster (EM-C)* taking the cluster of the key feature pairs as the gaze point, both described above; *EyeMirror-Pupil-Undistorted (EM-P-U)* and *EyeMirror-Cluster-Undistorted (EM-C-U)*, both using a corrected corneal image to investigate the effect of the spherical distortion of the corneal reflection (as in [151]); *Marker Tracking (MT)*, using a set of visual markers shown on the screen to track the orientation between the display and the eye tracker provided by the Pupil framework[9]; and a simple *Head Orientation (HO)* approach, tracking the participant's head with a Kinect v2 sensor, placed underneath the projected display.

For *MT* we calibrated the eye tracker for each participant separately from the centered location in front of the display. For all *EM* modes we adjusted the camera (position and focus) to capture images with a centered eye location. For *EM-P-U* and *EM-C-U* we sampled data while the people were looking at a chessboard pattern for 5 seconds to dewarp the images. The pattern filled out the whole projected display. To compute the distortion map, we used OpenCV's camera calibration[10], using 20 samples for each calibration. Figure 6.7 highlights the calibration procedure and feature matching as well as homography computation of the undistorted corneal image.

[8] https://pupil-labs.com/blog/2013-12/pupil-v0-3-6-marker-tracking/
[9] http://www.pupil-labs.com/blog/2013/12/036-release.html
[10] `https://docs.opencv.org/3.0.0/d4/d94/tutorial_camera_calibration.html`

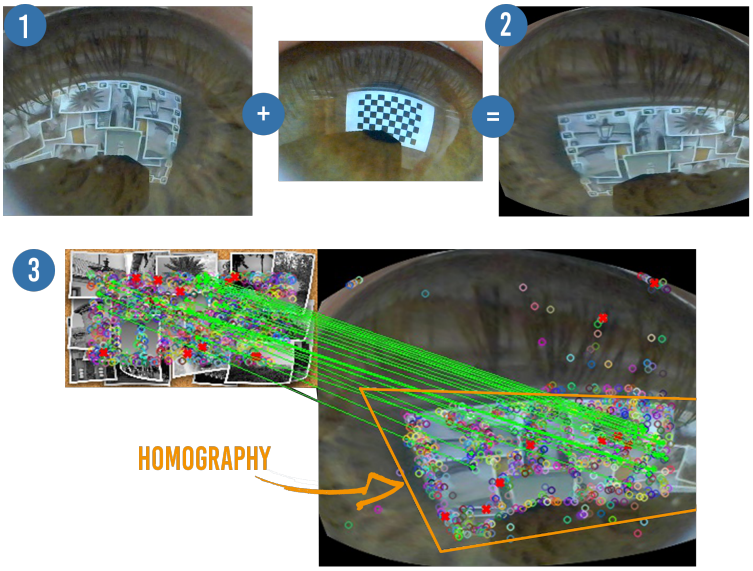
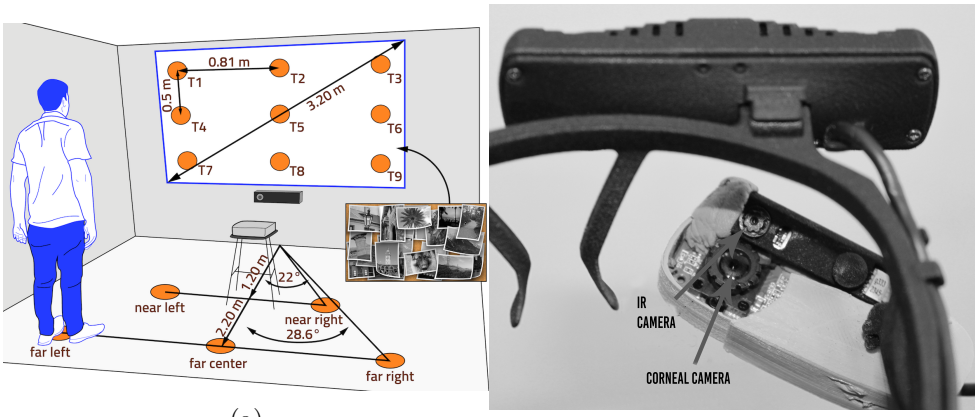


Figure 6.7: Calibration with chessboard pattern (1) to undistort the corneal image (2). Computation of a homography may be more exact (3).

Locations: We chose five different locations (2 near, 3 far) to investigate the effect of varying positions and orientations in front of the display. In doing so, we obtained images containing many different reflections of the display (i.e., different size and distortion), while simulating a more realistic setting. The eye tracker was calibrated only for the central location, as it is not likely that users re-calibrate for every position in a dynamic setting. We did not give any visual feedback to the participants, to prevent false positives. In addition, this allowed us to keep the length of the experiment reasonable, as all the data was sampled in two runs. We had to record the HO data separately, as the head tracking was error-prone while the mobile devices (eye tracker and prototype) were being worn. We computed the gaze estimation accuracy as well as the correction post-hoc for every mode.

Apparatus

Figure 6.8 illustrates the experimental setup: we used a large front-projected display with a size of 2.80 x 1.56 meters (diagonal: 3.20 meters) using a short-throw projector. The five locations (L1-L5) were distributed as follows: the *near* locations at a distance of 1.2 m, and the *far* locations at a distance of 2.20 m from the display center. The *near-left* and *near-right* locations were 1.35 meters away, and the *far-left* and *far-right* locations were 2.50 meters away from the display's center with an angular offset of $\pm 26.5°$ and $\pm 28.6°$, respectively. The *far-center* location had an angular offset of $0°$.



(a) (b)

Figure 6.8: Study setup: (a) A large projected screen and the Kinect sensor v2. In addition, all locations (*near-left, near-right, far-left, far-center and far-right*) and targets (T1-T9) are shown. (b) The prototype combining the Pupil Labs eye tracker with EyeMirror.

Standing at the *far* locations, the display covers 64.8°, while at the *near* locations the display covers 98.8°.

Choosing the locations this way forces participants to move their heads, as the region covered by the display exceeds the ocular motor range of ±55° [49]. To record the *EM* and *MT* in parallel, we mounted *EyeMirror*'s camera underneath the eye camera of the monocular Pupil Labs eye tracker, capturing the right eye ((see Figure 6.8)). The Pupil Labs system was running on a Thinkpad X201, transmitting the data via WiFi to a MacBook Pro driving the display and capturing the *EyeMirror* camera frames. For feature tracking we used the background image (containing 9262 features, shown in Figure 6.8) without markers as a template to prevent any effect on *EM's* performance.

Task & Procedure

We implemented a simple gaze-pointing task in which participants had to focus on targets shown at nine different positions (T1–T9). These were represented as red circles (40 pixels, or approx. 58 mm) on the projection with equal distances between them (see Figure 6.8). A pilot study showed that artificial lighting conditions could affect the quality of the reflected eye image. We therefore created a realistic scenario by illuminating the room with natural light. Every participant was first asked to calibrate the head-mounted eye tracker while standing at the center location in front of the projected screen. Each *Mode*, except *HO*, was recorded in parallel, as it is possible

to wear both mobile devices at once. The participants were instructed to look at each target as quickly and accurately as possible, while being free to move their heads. The targets were shown for six seconds each. At the end, participants were asked for demographic information including the color of their eyes.

The task was done while standing in front of the projected display at five locations (see Figure 6.8). Looking straight ahead, they looked approximately at the vertical center of the projection. We collected gaze data from the eye tracker for *MT* as well as the location of the on-screen targets and their time-stamps needed for post-hoc analysis. Furthermore, we recorded raw video material for all EM modes including calibration frames, as described above. Data was sampled at 30 Hz for all six modes (i.e., 180 samples per on-screen target = 6sec x 30 Hz), leading to 1620 samples for each Mode and Location combination. In total we recorded 30 (Hz) x 6 (sec) x 9 (targets) x 6 (Modes) x 5 (Locations) x 10 (participants) = 486000 samples. We dropped the first 2 out of 6 seconds per target, leading to 30 x 2 x 9 x 6 x 5 x 10 = 162000 (30% of all samples), resulting in 324000 samples in total (i.e. 54000 for each Mode). We discarded all data points of the first two seconds for each target, as this was the maximal required timespan to find the current target.

Experimental Design

We used a within-subject design for our experiment with the two defined independent variables from above, *Mode* and *Location*. We counterbalanced the order of *Location* between all participants using a Latin square. All modes except *HO* were recorded in parallel. Thus each participant performed the task twice. For each location, the nine targets were displayed in a random sequential order, different between participants, but identical within them (i.e., for both runs).

Participants

Ten participants (3 female) between 23 and 37 years old ($M = 27.5$ years, $SD = 4.55$ years) and having three different iris colors (5 brown, 4 blue, 1 green) were recruited from a local university campus. All participants had corrected or normal vision; none reported any visual impairments (e.g., color blindness).

Results

To evaluate the *EyeMirror* concept, we calculated the average gaze estimation error in degrees of visual angle (as defined in [59], Section 2.6.2). This value states the

		MT			EM-C			EM-P			EM-C-U			EM-P-U			HO		
		x	y	2D	x	y	2D	x	y	2D	x	y	2D	x	y	2D	x	y	
near	M	2.27°	1.39°	2.92°	3.89°	2.35°	4.90°	4.45°	3.50°	6.17°	4.23°	2.50°	5.28°	4.36°	3.12°	5.91°	3.93°	2.71°	5.32°
left	SD	1.46°	0.94°	1.27°	3.16°	1.75°	3.12°	3.35°	2.15°	3.14°	3.43°	1.81°	3.38°	3.79°	2.13°	3.57°	2.69°	2.95°	3.23°
near	M	2.44°	1.41°	3.12°	3.97°	2.38°	4.96°	4.08°	3.33°	5.79°	4.23°	2.57°	5.31°	4.53°	3.21°	6.08°	3.92°	3.01°	5.55°
right	SD	1.74°	1.02°	1.51°	2.92°	1.78°	2.93°	3.34°	2.22°	3.23°	3.19°	1.89°	3.17°	3.68°	2.21°	3.51°	2.58°	3.14°	3.20°
far	M	1.67°	0.96°	2.10°	2.80°	1.62°	3.48°	3.61°	2.27°	4.60°	2.92°	1.66°	3.61°	3.01°	2.29°	4.16°	2.43°	2.47°	3.97°
left	SD	1.07°	0.68°	0.96°	1.93°	1.12°	1.83°	2.63°	1.52°	2.50°	2.11°	1.18°	2.03°	2.49°	1.54°	2.35°	1.94°	4.59°	4.60°
far	M	1.50°	0.97°	1.99°	2.70°	1.71°	3.44°	3.27°	2.37°	4.39°	2.84°	1.80°	3.62°	2.79°	2.28°	3.99°	2.71°	2.37°	4.03°
center	SD	0.98°	0.68°	0.80°	1.96°	1.11°	1.86°	2.40°	1.59°	2.31°	2.12°	1.18°	2.03°	2.37°	1.53°	2.25°	1.70°	3.86°	3.82°
far	M	1.70°	.89°	2.10°	2.79°	1.55°	3.41°	3.24°	2.32°	4.33°	2.84°	1.61°	3.50°	3.08°	2.29°	4.23°	2.65°	2.74°	4.35°
right	SD	1.20°	1.01°	1.32°	1.76°	1.05°	1.66°	2.37°	1.53°	2.26°	1.90°	1.09°	1.80°	2.52°	1.57°	2.40°	1.74°	4.92°	4.78°

Table 6.1: Means and standard deviations of the overall, horizontal and vertical gaze estimation for all modes and locations.

difference between the position of the estimated gaze point and the actual on-screen target of the six *Modes (EM-P, EM-C, EM-P-U, EM-C-U, MT and HO)* and the five *Locations* (*near-left, near-right, far-center, far-left, far-right*). We performed a 6×5 (*Mode*×*Location*) within-subjects ANOVA on gaze estimation errors and found a main effect for *Mode* ($F_{5,25} = 34.52, p < .001$), and for *Location* ($F_{4,20} = 30.73, p < .001$), but not for an interaction between them.

In a subsequent post-hoc analysis in gaze estimation accuracy across all *Modes*, we found that *MT* differed significantly from all *EM* modes (all $p < .001$) as well as *HO* ($p < 0.05$). We found no significant difference in gaze estimation accuracy between *HO* and all *EM* modes. Comparing the gaze estimation accuracy concerning different targets between *EM* modes and *HO* also revealed no significant difference. All *EM* modes performed better than *HO*, as shown in Table 6.1. *EM-C* differed significantly from *EM-P* ($p < .01$), *EM-C-U* and *EM-P-U* (both $p < .05$). Finally, we found a significant difference between *EM-P* and *EM-C-U* ($p < .01$).

Overall, *EM-C* achieved the highest accuracy for all *EM* modes ($M = 4.03°$, $SD = .04°$), followed by *EM-C-U* ($M = 4.22°$, $SD = .03°$). *EM-P* ($M = 4.99°$, $SD = .07°$) and *EM-P-U* ($M = 4.87°$, $SD = .13°$) showed the worst results overall. *HO* ($M = 4.66°$, $SD = .36°$) yields better results than *EM-P* and *EM-P-U*, but worse than

EM-C and *EM-C-U*. Finally, *MT* achieved the best results overall, i.e., the lowest gaze estimation error ($M = 2.41°$, $SD = .06°$). All values were averaged over all locations and are summarized in Table 6.1. Transformed to absolute rounded pixels, these values correspond to 64px ($SD = 31$px) for *MT*, 106px ($SD = 59$px) for *EM-C*, 133px($SD = 72$px) for *EM-P*, 111px ($SD = 64$px) for *EM-C-U*, 128px ($SD = 73$px) for *EM-P-U* and 122px ($SD = 116$px) for *HO*.

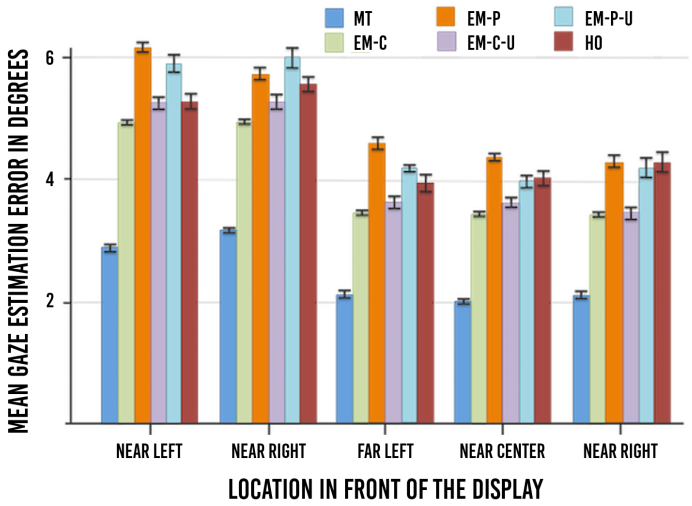


Figure 6.9: Mean gaze estimation error for every location and mode. Error bars indicate ± standard error of the mean.

Figure 6.9 depicts the average gaze estimation error for every *Mode* and *Location*. *MT-far-center* performed best ($M = 1.99°$, $SD = 0.22°$), followed by *MT-far-right* ($M = 2.08°$, $SD = 0.35°$), *MT-far-left* ($M = 2.09°$, $SD = 0.25°$), *MT-near-left* ($M = 2.95°$, $SD = 0.47°$) and *MT-near-right* ($M = 3.10°$, $SD = 0.44°$). All *EM* modes performed worse than *MT*, but within *EM-C* showed the lowest error for *EM-far-right* ($M = 3.42°$, $SD = 0.18°$), followed by *EM-far-center* ($M = 3.44°$, $SD = 0.40°$), *EM-far-left* ($M = 3.48°$, $SD = 0.19°$), *EM-near-left* ($M = 4.86°$, $SD = 0.43°$) and *EM-near-right* ($M = 4.95°$, $SD = 0.36°$). The other *Modes* performed slightly worse.

Post-hoc tests on *Location* revealed that the significant main effect stems from the participants' distance from the display: *near-left* differed significantly from *far-left, far-center* and *far-right* (all$p < .01$). *Near-right* also differed significantly from *far-left* ($p < .01$), *far-center* and *far-right* (all $p < .05$). Overall, *far-center* showed the highest gaze estimation accuracy, i.e., the lowest error ($M = 3.62°$, $SD = 0.1°$), followed by *far-left* $M = 3.64°$, $SD = 0.11°$), and *far-right* ($M = 3.75°$, $SD = 0.17°$). The

near locations led to worse results: *near-right* had the highest error in gaze estimation accuracy ($M = 4.97°$, $SD = 0.15°$), followed by *near-left* ($M = 4.96°$, $SD = 0.09°$). To further investigate the gaze estimation of all *EM* modes, we split the error measured in degrees of visual angle into two values, showing horizontal (x-direction) and vertical (y-direction) errors separately. We found that for all *Locations*, the vertical gaze estimation error is lower than the horizontal across all *Modes*. Table 6.1 summarizes the results. On average *MT* showed the lowest difference between horizontal and vertical gaze estimation error, followed by *EM-C*.

To complete the analysis we further wanted to find out whether the screen targets and thus screen regions resulted in different levels of gaze estimation accuracy. In doing so we analyzed the results separately for each on-screen target. We found significant differences between most of the targets across all *Modes*. For *MT* we found significant differences between all targets except (T1, T6), (T3,T4), (T3,T7), (T3,T9), (T4,T7), (T4,T9) and (T7,T9). *EM-C* showed no significant difference for gaze estimation accuracy between targets (T1,T3) and (T7,T9). For *HO* the target pairs that showed no significant difference are (T2,T4), (T2,T9), (T2,T6), (T4,T6) and (T5,T7). Every *Mode* performed best for T5, whereas *EM-C* achieved the lowest gaze estimation error overall (M = 1.43°, SD = 1.71°), followed by *EM-C-U* (M = 1.54°, SD = 1.95°), *MT* (M = 1.67°, SD = 0.92°), *HO* (M = 3.85°, SD = 5.36°), *EM-P-U* (M = 4.08°, SD = 2.07°) and *EM-P* (M = 4.70°, SD = 2.08°).

Finally, we found no significant difference in gaze estimation error based on iris color for all *EM* modes. *EM-C* performed best, with 3.84° (SD= 2.31°); brown eyes achieved the lowest gaze estimation error on average, followed by blue eyes with 4.12° (SD = 2.56°). Green eyes had the highest gaze estimation error with 4.20° (SD = 2.41°) on average for *EM-C*. We found the same order for all other *EM* modes.

6.2.3 Experiment II – Influence of display content

In a second laboratory experiment we wanted to investigate the connection between *EyeMirror*'s gaze estimation approach and the content of the display. For this, we compared the gaze estimation accuracy of EM-C across various display content, where the content used differed in the number of natural features.

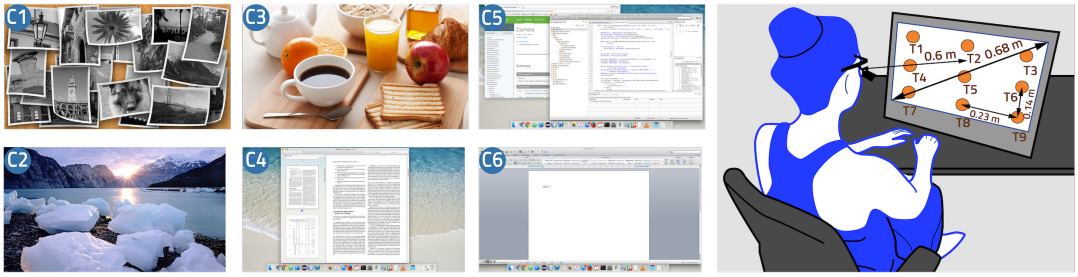


Figure 6.10: The different display contents used in the experiment: C1 with 9262, C2 with 6988, C3 with 3314, C4 with 9476, C5 with 4239 and C6 with 934 features. The right picture shows the setup for the second experiment.

Independent Variables

We used a within-subjects design in this experiment with one independent variable, *Content*: we used six different display content images (C1–C6), while computing the gaze estimation accuracy for the display. The different images are depicted in Figure 6.10 together with their number of features. We used three different wallpapers (C1–C3) as well as realistic desktop scenes where different kinds of applications were opened (C4–C6).

Apparatus

Figure 6.10 illustrates the experimental setup: we used a multi-touch enabled monitor with a size of 0.59 x 0.33 meters (diagonal: 0.68 meters). The participant was sitting centered in front of the display at a distance of 0.6 meters, as she would be sitting when interacting with the touch-enabled display. At this position the display covers 31.6° of the field of view. The *EyeMirror* camera frames were recorded by plugging it into a MacBook Pro that also was driving the display.

Task & Procedure

We reused the same gaze-pointing task as in the first experiment to be able to compare the findings to our initial results. Looking at red circles, shown consecutively at nine different positions on a desktop monitor, was done while sitting in front of the display (shown in Figure 6.10). Raw video material for *EyeMirror*'s clustering approach (EM-C) was recorded at 30hz. Again, we discarded the first 2 seconds for each target. We counterbalanced the order of *Content* between all participants using a Latin square. As we had six different images, each participant did the task six times. For each content image, the targets (T1–T9) were shown in a random sequential order.

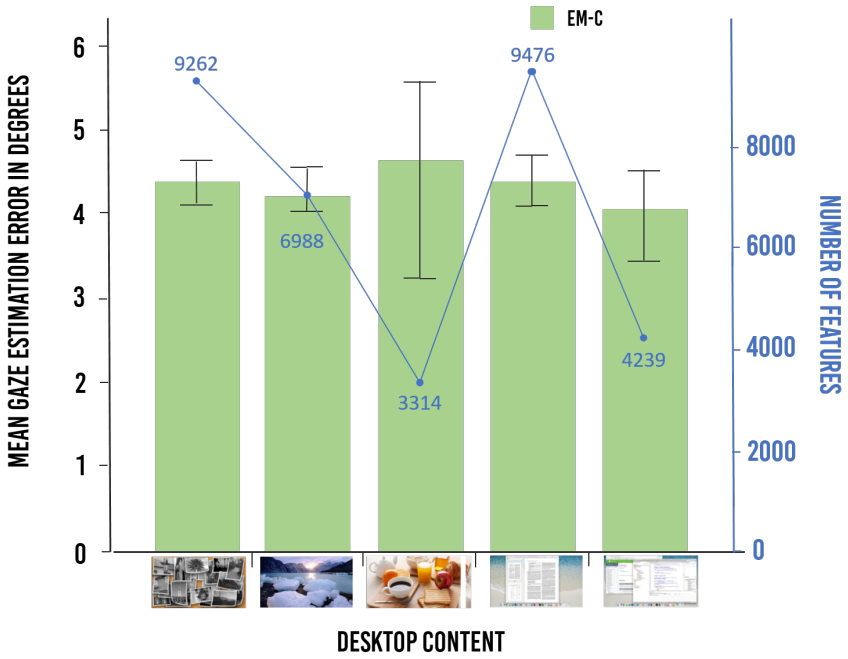


Figure 6.11: Mean gaze estimation error for the desktop content images C1–C5, combined with the number of features.

Participants

A group of six participants (2 female) between 23 and 37 years old ($M = 27.5$ years, $SD = 4.55$ years) were recruited from a local university campus. All participants had normal vision; none reported any visual impairments (e.g., color blindness). Power analysis showed that the number of participants is adequate, as we achieve a power > 0.65 for this experiment with an effect size f = 0.63.

Results

To evaluate *EyeMirror*'s accuracy, we calculated the average gaze estimation error in degrees of visual angle. This value states the difference between the position of the estimated gaze point and the actual on-screen target for all different *Content* images using the cluster approach of *EM-C*. While computing the gaze estimation for all different *Content* images, we discovered that our approach did not work for C6. The blank page in the middle of the screen revealed too few features, so that *EyeMirror* returned arbitrary values.

Hence we performed a 5×1 (Content (C1–C5) x EM-C) within-subjects ANOVA on gaze estimation errors and found no effect for *Content*. In Figure 6.11 the gaze

estimation error across all targets and participants is plotted against the number of features. We achieved the highest accuracy, i.e., the lowest gaze estimation error, for C2 (wallpaper with 6988 features) ($M = 4.30°$, $SD = 0.04°$), followed by C5 (Mac desktop: opened browser and IDE, 4239 features) ($M = 4.27°$, $SD = 0.02°$), C1 (wallpaper used in first experiment with 9262 features), C3 (wallpaper, 3314 features) ($M = 4.34°$, $SD = 0.06°$) and C4 (Mac desktop, opened pdf, 9476 features) ($M = 4.36°$, $SD = 0.06°$).

6.2.4 Discussion

Our results show that – using corneal reflections (i.e., the area covering the iris and pupil) – *EyeMirror* is capable of achieving a gaze estimation accuracy of 4.03° using the clustering approach (*EM-C*) compared to 2.41° for *MT* and 4.66° for *HO*.

Experiment I – Gaze Estimation

We used different approaches within *EyeMirror* to compute the gaze point of the participants. Our results reveal that it is sufficient to extract the cluster of all key feature pairs to achieve still-reasonable results. Taking the eye center into account and transforming it onto the display by using homography matrices performs 0.77° worse (*EM-P*). While it seems counterintuitive, it can be explained when we have a closer look at the difference between the two approaches. Each method uses as an input image the extracted limbus area (i.e., the pupil and the iris). The reflection within the area of the pupil is less noisy and brighter than within the iris. Whereas features are found on both areas, most of the matches are detected within the pupil, leading to key feature pairs. The main drawback of *EM-P* and *EM-P-U* is the use of the eye center as a gaze reference point. The method used for eye center localization finds only an approximation of the actual pupil center. Consequently, it does not always correspond to a person's real gaze, and only transforms rough estimations.

In addition, we investigated the use of a camera calibration to correct the input images for our approaches. As the eyeball has a spherical structure, the corneal images are distorted. As expected, we found a small, but not significant, effect when using the homography matrix for transformation, such that *EM-P-U* is 0.12° more accurate than *EM-P*. Having a look at Figures 6.7(3) and 6.5, the approach is shown based on the raw and the corrected image. The shape of the reflected display is more rectangular if using the undistorted version. This supports the concept of computing a homography matrix, as the template (i.e., the screen's content) is also of a rectan-

gular shape. Otherwise, we found the opposite for the cluster approach. Using the undistorted version of the corneal image (*EM-C-U*) significantly decreases the gaze estimation accuracy by 0.19°. Correcting the images changes the distribution of the key feature pairs. As most of the matches are found at the pupil, they are arranged in an elliptical shape supporting the structure of the pupil. After a correction this is no longer the case, leading to worse results.

As we used different locations in front of the screen, we were able to investigate the effect of different orientations to and distances from the screen. We calibrated once for *MT* from the far-center position for each participant to simulate a realistic scenario. Hence, we achieved the optimal results for a one-calibration scenario, as the calibration plane was orthogonal to the participants. Using different locations results in various distorted reflections on the human eye. *EyeMirror* is able to partly deal with these changes, since its accuracy remains constant among different orientations for the same distance. In the case of *EM-C* and *EM-C-U*, this is primarily caused by the sharp angle between participant and display for both *near* locations. There, the highest gaze estimation error was found for targets at the opposite side of the location (i.e., for *Location near-left* the targets T3, T6 and T9). With increasing distance, the angle between participant and display also increases. For *EM-P* and *EM-P-U* there is also another reason: the larger the distance to the display, the larger the reflected area. This means that the template of the surface – here the display – is detected more accurately, and thus the computation of the homography matrix is more robust. The results for *MT* are in line with existing findings [93]. Calibrating from the *far* location leads to a smaller calibration plane than for *near* locations. Hence the eye tracker extrapolates for gaze positions lying outside that region, causing worse gaze estimation.

Although we found no significant difference between *HO* and all *EM* modes, *EM* achieved a consistently better gaze estimation accuracy than *HO*. The good results of *HO* stem from the experiment design, i.e., the location layout that forced participants to move their heads quite a lot, mostly for *near* locations. This fits with the very high standard deviation (see Table 6.1) in gaze estimation for *far* locations. This is mainly caused by the large variability in head movement propensity [123]. If we also take a look at the gaze estimation between targets with minimal distance (e.g., far-left EM-C T4: 3.39°, T5: 1.14° compared to HO T4: 7.52°, T5: 2.08°), we can see that *HO* performs much worse than *EM*. This fact indicates that the gaze of users

who do a lot of eye movements might be better approximated with *EM* than *HO*. In general, *EM* gives a direct connection to the objects in the environment, as they can be extracted from the corneal images. *HO* only provides the position and orientation of the head in 3D space, which has to be combined with knowledge about the position of surrounding objects.

Thus, *HO* is not preferable for gaze approximation, especially in settings where gaze is computed across a variety of surfaces (e.g., multiple different displays). The first experiment investigated the gaze estimation accuracy of *EyeMirror* in a lab setting. In real-world scenarios, *HO* is still not usable. If the head tracking is done remotely, it requires the trackers to be attached on every surface in the environment. Doing head tracking via an IMU sensor will require a scene camera to create a connection to the environment. However, *EyeMirror* has a major advantage, as we get a realistic representation of the human vision, combined with a value for gaze approximation.

Experiment II – Surface Content

We evaluated *EM-C's* accuracy across six display content images, all different in terms of the kind of information shown and the numbers of provided features. Overall we found no significant difference in gaze estimation error, using content with a number of features between 3314 and 9476. Obviously, our approach does not work with surfaces providing too few features (e.g., C6 with 934 features). Interestingly, we achieved the best results for C5 (4239 features). This may be caused by the distribution of the features being rather uniform. Hence, our results show that *EyeMirror* enables gaze approximation in a normal desktop setting.

Example Applications

To summarize, *EyeMirror* is sufficient for settings in which spontaneous gaze approximation is sufficient. The system can be used to detect if someone is looking at a screen in a multi-display setting, and at which region, useful for gaze-contingent displays. *EyeMirror* is built as a mobile wearable system, so it could be used as a tool for exploring the human gaze (e.g., attention measuring) in more unconstrained and mobile settings [97, 96].

For example, information like detected faces (shown in Figure 6.3(2)) can be used to estimate social interaction. Tracking other objects can be used to measure attention on many different things. *EyeMirror* might be easily integrated into AR and VR

devices. A virtual reality headset provides the best conditions, since nothing other than the screen content (i.e., the VR environment) is reflected on the cornea of the eye. Approximating someone's gaze in VR can be used for foveated rendering [152]. With *EyeMirror* we developed a system that provides a base for further investigations in a pervasive setting. The system will be made open-source. As *EyeMirror* is easily extendable, new algorithms and methods for different purposes can be plugged in and explored.

Limitations

Apart from its advantages and applicability to pervasive application scenarios, *EyeMirror* also comes with some limitations: First, our concept of gaze estimation on surfaces is based on natural feature tracking that requires an information stream about the content (e.g., display content). However, with the rise of IoT, we believe that information like display content will be accessible. For example, information about the environment outside is available through Google Street View. Moreover, we think that objects can be detected via a trained image classifier and specific neural networks [158].

Second, *EyeMirror*'s performance is influenced by the quality of the corneal images. The camera is placed close below the eye to capture most of the eye movements. To handle large eye movements, the use of more cameras should be explored. In our current prototype, we have to adjust the focus manually by rotating the lens in the thread in order to retrieve sharp corneal images. Whereas limbus and pupil center extraction are tolerant of various lighting conditions, they might have an effect on the overall approach. Obviously our proposed method does not work in dark environments, as no information will be reflected on the cornea.

Besides lighting, the eye-object distance is another source of blurred corneal images. If the user focuses on an object at a far distance, its representation on the corneal image is rather unsharp, caused by the mirror properties. To adapt the system to this new setting, the focus of the camera has to be adjusted. Using cameras with adjustable parameters, like ISO sensitivity, aperture size and auto-focus, might help to counteract the above limitations.

So far we have not explored the long-term usage of *EyeMirror*. We can only argue about the duration of the experiment, which lasted for at least 27 minutes (6

(sec) x 9 (targets) x 6 (Modes) x 5 (Locations)). The presented results are representative for a usage time within half an hour.

Nevertheless, we believe that *EyeMirror* is a promising case toward a new generation of less-invasive head-mounted eye trackers and a good baseline for using corneal reflections in pervasive settings (cf. *Ubiquitous Computing for Eye Tracking Continuum* in Figure 6.1), where a gaze approximation is sufficient.

6.2.5 Application Example

In this section, we present two approaches utilizing corneal imaging. In particular, we demonstrate the potential of *EyeMirror* in a life logging scenario to analyze people's visual behavior in the wild. We conducted a life logging experiment where we collected data in un-instrumented environments. Based on the explorative analysis of the recordings, we demonstrate what kind of information can be extracted out of the corneal image reflections and derive suitable applications use cases.

Data Collection

We carried out an experiment in the wild, applying our system to a daily routine. We are going to explain the opportunities arising when using the information of the environment reflected on a person's eye. In doing so, we will differentiate between (1) scenarios where object detection delivers valuable insights and (2) cases in which gaze estimation is achievable.

We equipped a 28-year-old male person, who is working in a research lab, with the *EyeMirror* system. The test person was wearing the device on two consecutive days, for approximately 11 hours. The device recorded the camera stream at 30 fps. To prevent data loss, the captured camera streams were analyzed post-hoc. For this, we manually examined the recorded video material and selected different types of scenes. As the material contained many settings involving sitting in front of a computer monitor, we reduced the amount of data by removing most of them. The analysis was done on the pre-processed camera frames, containing the region within the pupil. For object detection we used a small set of template images (found with Google image search) as well as self-taken images of items possibly visible in each scene. In the following we will briefly present the main findings by showing the processed recordings and possible future applications of the system:

- **Human to Human Interaction** – It is possible we are able to detect the reflection of a person's face on the human eye. Applying the OpenFace framework [5] for face tracking, we can extract the face and its orientation in the reflection.

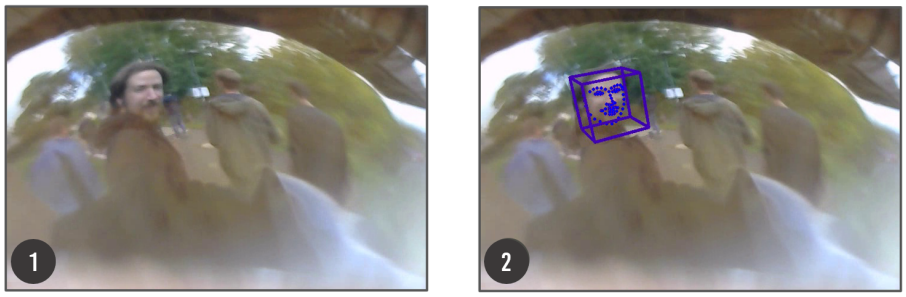


Figure 6.12: Face detection and its 3D orientation (2) using the pre processed camera frame (1).

Figure 6.12 depicts the case that another person is looking towards the test person. That kind of corneal image information can be used to detect human-to-human interaction.

- **Display Interaction** – Displays are pervasive in our environment and exist in various forms and sizes, used to transport information. Display blindness [125] is a common problem of public screens. We found that it is possible to detect and track displays in the environment that are currently visible in a person's field of view. We implemented a modified version of the OpenCV sample square detector to handle the distorted display representations in the reflected images. The algorithm combines Canny edge detection together with a contour detection. In Figure 6.13 the processed images are illustrated. Such information could be used to detect if a person is recognizing a display they are currently passing.

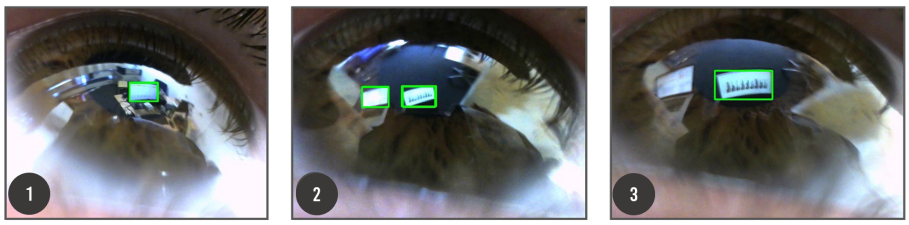


Figure 6.13: Display Detection and Tracking (1)–(3) in images reflected on a person's eye; detected displays are marked with a green rectangle.

- **Object Interaction** – In our everyday environment there are many objects other than displays that we use and interact with. We integrated a function for template matching. In fact, the algorithm utilizes the correlation coefficient method using a 2D convolution. In Figure 6.14 recordings from several situations

are depicted. We were able to estimate where the participant was looking within a poster (see Figure 6.14), a book-shelf, and even when he looked at smaller objects, such as tools.

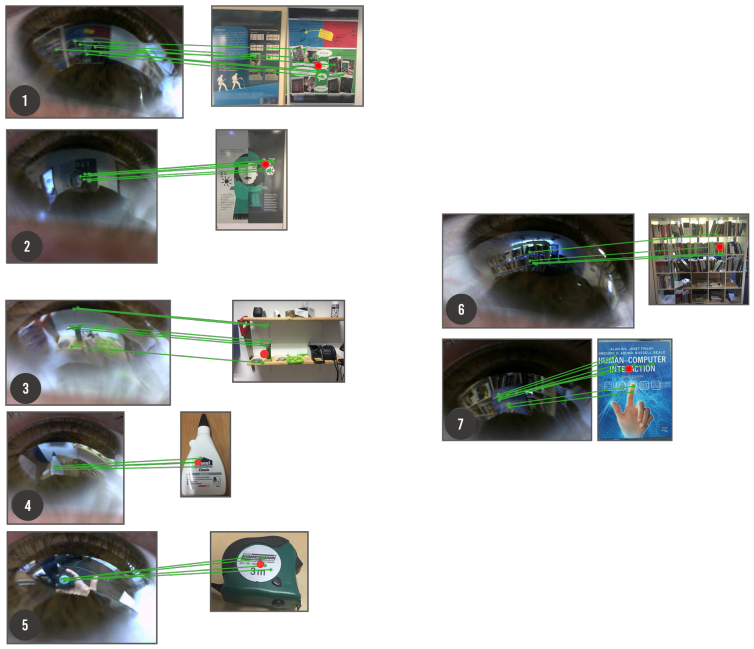


Figure 6.14: Gaze estimation on posters (1,2) using the feature tracking algorithm. Gaze estimation on a bookshelf (6) and a book (7); on other real-world objects (3,4,5).

- **Driving** – In situations where our attention is not restricted to interaction with a single object, even more information from the environment is important. The corneal images can be used to extract information about the location of the reflected objects, i.e. if they are within the foveal and the peripheral vision.

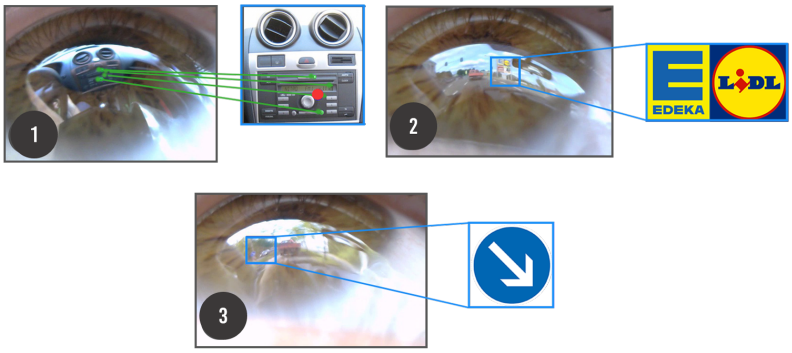


Figure 6.15: Gaze estimation on car hifi system (1), and detection of retailer logos (2) and road sign (3).

Figure 6.15 summarizes several scenarios while driving a car. You can determine whether the driver is looking at the controls, and also which information is out of the current focus (grocery logo and road sign).

With this data set and its explorative analysis, we investigated the potential of corneal imaging. It serves as a pilot data analysis. Further investigation is needed in terms of automatic object detection. Such a system could be used to create visual diaries enriched with information about gaze and attention.

6.2.6 Summary of Findings

In this Section, we presented *EyeMirror*, a mobile system using a novel approach for calibration-free gaze approximation. We built a low-budget prototype, which in contrast to existing head-mounted eye-trackers, only requires a single camera. Capturing the environment through corneal images, it enables gaze approximation on various surfaces (e.g., displays), solely based on natural feature tracking. For this, a representation of the object's content has to be known and provide enough visual features. In a laboratory experiment we compared different modifications of our approach against a Pupil Labs eye tracker and using the head orientation as gaze. We found better results for *EyeMirror* compared to *HO* in gaze estimation accuracy, although they were not significant. The head-mounted eye tracker gives the best results, but needs to be calibrated.

Since we are following the iterative development model, we aim to further improve the system in the next cycle. In particular, the major part of the progress has to be made in the method itself to realize better gaze estimation.

6.3 hEYEbrid – A Hybrid Approach for Mobile Calibration-Free Gaze Estimation

In this Section we present *hEYEbrid*, a concept that enables calibration-free and accurate gaze estimation using a head-mounted device. It is the result of the iterative development cycle performed to resolve the main issue of *EyeMirror's* inaccuracy. We developed a hybrid approach to estimate a person's gaze, utilizing two eye cameras: (1) an infrared eye camera that is tracking the pupil's positions and its movements, and (2) a natural light eye camera that captures closeup RGB images of the eye. The latter captures the environment reflected on the corneal surface, i.e. the iris and the pupil of the human eye, and is referred to as the *corneal camera* in the rest of the

Figure 6.16: The developed *hEYEbrid* concept used to build a mobile and wearable device for spontaneous competitive gaze estimation at any place and any time. The head-mounted device is connected to a Nexus 6P that processes both camera streams in real time.

paper. The underlying assumption is that the pupil center in the corneal camera's images corresponds to the actual gaze point in the real environment. This assumption is similar to the one that most current eye-trackers make. Pupil position and center are extracted in the infrared image and mapped onto the corneal image. Hence we are able to extract the reflection within the pupil area, which roughly represents the user's current field of view. Both cameras are placed next to each other in a fixed position, focusing on the same eye. With *hEYEbrid*, we do eliminate the need for a cumbersome calibration as well as regular re-calibration procedures. To compute the relationship between the infrared and corneal camera image plane, a mapping needs to be computed at one time during the construction process of the head-mounted device. Thus *hEYEbrid* is an enabling technology for any kind of gaze estimation in pervasive settings, such as spontaneous gaze estimation or user studies in the wild.

In a controlled user study with 20 participants we evaluated the gaze estimation accuracy of three approaches, each differing in the number and location of cameras: (1) *hEYEbrid* using two eye cameras; (2) *3Camera hEYEbrid* (*hEYEbrid*plus), which

combines the two eye cameras of *hEYEbrid* with an additional scene camera; and (3) a state-of-the-art monocular *Pupil Labs* eye tracking system[11]. In three validation sessions, each participant had to focus on 15 different objects. Between sessions, we either re-calibrated the Pupil Labs eye tracker or asked the participant to take off and put on the device to simulate a calibration drift. With *hEYEbrid*, we achieved the highest gaze estimation accuracy compared to the other approaches, namely 2.19°. The simulated calibration drift had no significant impact on *hEYEbrid*'s performance, as it still achieved 2.36° accuracy in a fully unconstrained setting.

Finally, we designed and implemented a mobile and wearable system by connecting a head-mounted device based on *hEYEbrid* to an Android phone, as illustrated in Figure 6.16. It includes the functionalities for ad-hoc gaze estimation on a public screen and focused object detection. Moreover, the mobile application is able to drive a head-mounted Pupil Labs eye tracker as well as the *hEYEbrid-3C* device. In sum, our work provides the following contributions:

- **Novel hybrid approach** connecting IR eye and corneal images.
- **Accurate and constant long-term** gaze estimation without (re-)calibration.
- **Wearable and mobile** system to employ gaze as an input modality in fully pervasive settings.

The remainder of this section is structured as follows: We first introduce the *hEYEbrid* concept as well as its extension, *hEYEbrid-3C*, and outline the implementation. We then describe the conducted study and present its results. In the last part, we illustrate the design and implementation of the mobile prototype.

The required related work was already presented in Chapter 3. *hEYEbrid* advances the existing head-mounted approaches, outlined in Sections 3.2 and 3.3, as it (1) introduces a new concept, without 3D pose estimation and limbus tracking; (2) is evaluated in a mobile setting and (3) achieves better gaze estimation accuracy than existing approaches (especially for on-screen gaze estimation, at an average of 1.64°). Further, the necessary background knowledge was introduced in Chapter 2.

6.3.1 hEYEbrid System

As stated already in Section 1.4, gaze estimation using a head-mounted eye tracker faces several challenges that come along with the main problem of calibration. Ideally,

[11] https://pupil-labs.com/

the user should be able to put on a device that is immediately ready to use and provides robust and accurate gaze estimation. In keeping with this idea, we introduce the concept of *hEYEbrid* in the following.

Concept

The basic idea of the approach is that the pupil center in the corneal image, i.e. in the reflected scene, coincides with the actual gaze in the real environment, as explained in Section 6.1 (see Figure 6.2). Remember that the area within the corneal limbus (often referred to as just the limbus), that is the pupil and the iris, reflects the scene a person is currently facing. As the eye is a spherical object, it acts like a fish-eye view into the world from the person's perspective. Hence the reflection is a distorted representation that is blended with the structure of the iris and is clearest within the area of the pupil. As explained in Chapter 2, there are two main visual axes of the eye when it comes to gaze estimation, the pupillary axis and the visual axis (also depicted in Figure 6.2). Standard head-mounted eye tracking systems model the angle kappa, i.e. the difference between visual and pupillary axes, using the vector of the first Purkinje and the pupil center (cf. Section 2.2). Note that both axes are going through the pupil center; thus, we are using the pupil center on the corneal image as a reference to the gaze in the scene.

The concept is different from the approach of *EyeMirror*, where we used the eye center as an approximation for the gaze point. The eye center does not necessarily coincide with the pupil center. In addition, we do not require limbus tracking in the *hEYEbrid* approach and consider only the reflection within the pupil.

Existing works that use corneal reflections for gaze estimation are based on limbus tracking [138, 128, 187]. To accomplish an exact and robust extraction from the corneal images, these approaches rely on a combination of feature- and model-based methods. Although the limbus can be distinguished well from the surrounding white sclera, other image features, such as eyelashes and eyelids, make the separation harder. In addition, the underlying models require high computational power [188, 187, 141] or are inaccurate, worsening the results of the gaze estimation, as we learned in Section 6.2. That is why we decided to use a different approach in *hEYEbrid*.

Figure 6.17 illustrates the processing pipeline which we propose. Our concept is based on pupil detection and tracking. The advantages are that the pupil naturally reflects

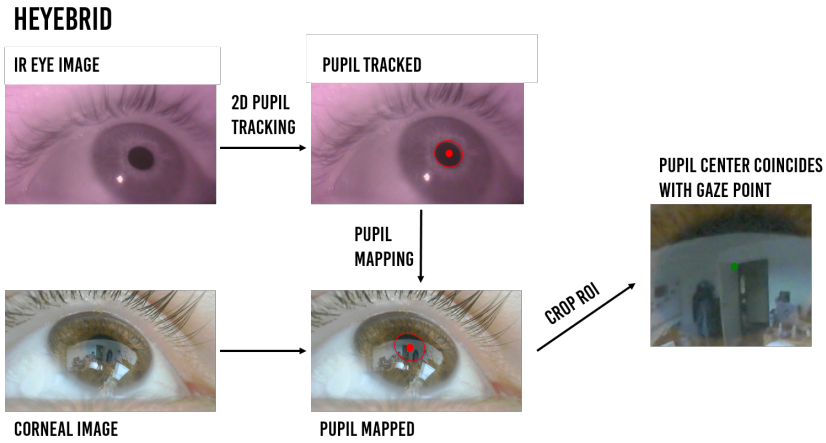


Figure 6.17: Basic processing pipeline of *hEYEbrid*, which combines infrared eye and corneal images. The pupil is tracked in the IR image and mapped onto the corneal image to finally crop the reflection within the pupil. The mapped pupil center coincides with the gaze point.

our current field of view and the reflection within this area is explicit (unlike the iris' reflectance). *hEYEbrid* is based on a two-camera approach, as it is hardly possible to extract the pupil in the corneal image itself, due to the reflective characteristic of the pupil and iris. This is different for infrared illuminated eye images. Therefore we combine two different types of eye images – an infrared eye and a corneal image – and process them simultaneously. In this way we can profit from the mature algorithms enabling a fast and robust pupil detection/tracking based on IR images. In each IR frame we obtain an ellipse describing the pupil's structure and center. This is mapped onto the corneal image, to finally crop the reflection around the pupil, as depicted in Figure 6.17.

To project information from the infrared to the corneal image, it is necessary to obtain a mapping between the two image planes. Both cameras, the infrared as well as the corneal imaging camera, are placed in front of the same eye in a fixed position relative to each other. The necessary mapping has to be computed only once after the construction process of the head-mounted device. This mapping is a 3D transformation problem. Figure 6.18 visualizes the geometric solution of how to find the pupil, which we extracted in the IR plane, in the corneal plane. In our system, we have placed the camera lenses beside each other, so that their fields of view converge on the person's eye. According to the *Epipolar Constraint* [53] , the following equation holds:

$$m_2^T F m_1 = 0 \text{ with a suitable } 3 \times 3 \text{ matrix } \mathbf{F}, \text{ called a } \textit{fundamental matrix}$$

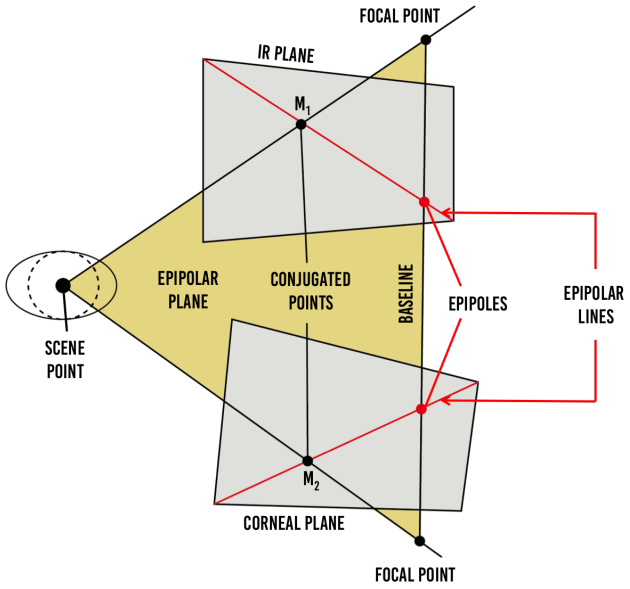


Figure 6.18: Estimating a mapping between the two eye camera image planes based on epipolar geometry. In *hEYEbrid*, the problem is reduced from 3D to 2D and solved via a planar homography mapping matrix.

It says that a pixel m_2 of the corneal image plane, which corresponds to a pixel m_1 in the infrared image plane, cannot lie anywhere. The fundamental matrix has rank 2, thus is not invertible, and has seven degrees of freedom (DOF), i.e. nine minus two for rank and scale. If the fundamental matrix is known, we know for each pixel m_1 in the IR frame the corresponding epipolar line l_2 in the corneal frame and vice versa:

$$m_2^T l_2 = 0 \text{ with } l_2 = Fm_1$$
$$m_1^T l_1 = 0 \text{ with } l_1 = Fm_2$$

where the vector $l_i = (a, b, c)^T$ describes the epipolar line $ax + by + c = 0$. The fundamental matrix F can be computed from correspondences of image points. Given m_1, the search for m_2 is reduced to searching along the epipolar line.

In *hEYEbrid*, we simplify the camera transformation from a 3D to a 2D problem. For this, we assume the eye to be flat, so that the usual rotations of the pupil in 3D space are represented by a moving ellipse on a plane. That is the camera image plane, as it just records a 2D image. The original 3D information is encoded in changes of the elliptical parameters, such as minor and major axes, which result in different shapes of the pupil. In that case, the mapping reduces to searching for a transformation matrix

between two planes. A planar *homography matrix* **H** satisfies the following constraint:

$$m_2 = Hm_1 \text{ for a } 3 \times 3 \text{ matrix } \mathbf{H} \text{ with rank } 3$$

There is a one-to-one point correspondence in this particular case, which can be computed using image features for instance. The homography matrix H is actually a projective transformation and has 8 degrees of freedom: 2 for scale, 2 for rotation, 2 for translation and 2 for line of infinity. It is important to note that it is not robust against changes in distance to the planes one wants to find the correspondence between. Consequently, we have to ensure that the distance from both cameras to the eye remains rather stable.

Finally, the combination of two eye cameras in a hybrid method is such that (1) we exactly compute the position of the pupil and its center without relying on 3D eye pose estimation using pre-defined parameters [138, 128] and (2) we do not require a user calibration [188]. The concept can be seen as just moving the scene camera of a video-based head-mounted eye tracker below the eye to capture the world through the eye of the user. In doing so, we eliminate the need for a calibration procedure.

Extension: 3C-hEYEbrid

We modified *hEYEbrid* by adding scene images to the concept, similar to the scene registration approach by Nakazawa et al. [129]. They use a two-camera approach combining corneal images with scene images. Using an aspherical eye model, they developed an iterative registration algorithm to match both image types in order to compute the user's gaze in the scene camera's image. In Figure 6.19 the additional concept of *hEYEbrid-3C* is illustrated.

It projects the result of *hEYEbrid*'s pipeline, the cropped corneal image with the pupil center, to the current scene image. The mapping is done as in the method from above based on a planar projection. As we cannot guarantee that the spatial relationship between the corneal and scene camera is fixed, the correspondence is computed for each pair of corneal and scene frames. In particular, the cropped corneal image is used as a *template* image that we try to find in the *observed* image of the scene camera stream. We use a natural image feature approach that computes the homography matrix H if enough matches are found. The matrix describes a bidirectional mapping between the corneal and the scene camera's image plane. The pupil position that was already mapped on the corneal image is further transformed to a gaze position

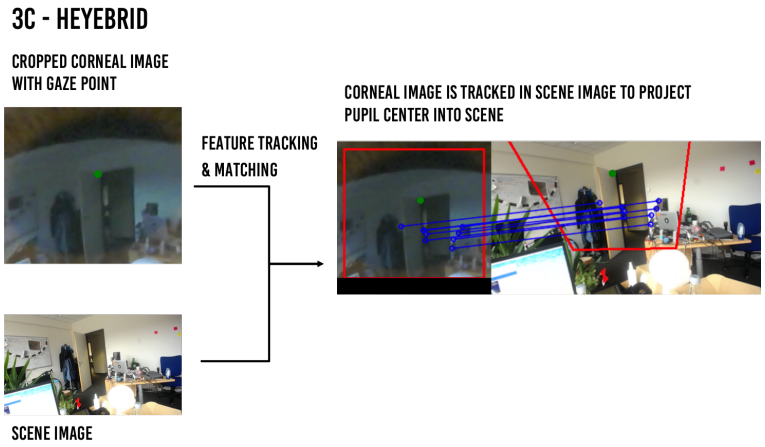


Figure 6.19: The extension *hEYEbrid-3C* connects the result of *hEYEbrid*'s processing pipeline with the scene image to project the pupil center into the scene image plane.

in the scene image by applying the transformation matrix. This concept is similar to the standard approach of head-mounted eye trackers. But in contrast to performing a user calibration, *hEYEbrid-3C* is continuously doing a frame-based calibration to update the relationship between the cameras, without actively requiring a specific user interaction. *hEYEbrid-3C* does not account for any image corrections (e.g., distortion) and does not rely on 3D eye pose estimation, as in [129]. Hence, it is more lightweight and faster, though we accept potential inaccuracies during the matching process between corneal and scene images.

Implementation

Both approaches are implemented using a monocular Pupil Labs Pro mobile eye tracking headset (2014 version[12]) in which the eye camera was extended by a corneal camera. Figure 6.20 illustrates the building steps to integrate *hEYEbrid* into the head-mounted device. We used a Logitech C270 webcam to capture corneal images with a maximal resolution of 1280 x 960 pixels at 30Hz. The original housing and the glue around the lens were removed to manually adjust the focus, so as to enable macro shots. This camera covers a 60° field of view and has a fixed focus of 4 mm. The cropped corneal images have a varying resolution after slicing the pupil region. Depending on the eye pose (i.e., the location of the pupil) the resolution ranges from 200 x 200 pixels to 300 x 300 pixels.

[12]https://pupil-labs.com/blog/2014-01/new-pupil-pro-headset-capture-software-0-3-7/

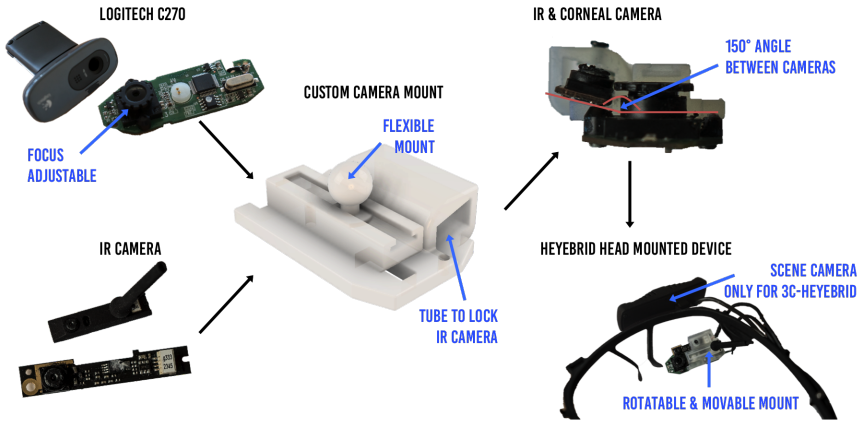


Figure 6.20: Building steps to integrate *hEYEbrid* into the Pupil Labs head-mounted device. The corneal camera is affixed to a custom mount together with the device's IR camera. The mount is movable and rotatable.

We reused the default infrared camera of the Pupil Labs eye tracker, but removed the housing. This camera can record images with a maximal resolution of 640 x 480 pixels at 30 Hz. We designed a new mount to position both cameras in front of the user's eye. It preserves the basic possibility to rotate the eye cameras and adds the option to be moved in front of the eye to position the cameras, so that the pupil is best covered. Both cameras are fixed with screws and placed at an angle of 150° to each other. Hence, it is guaranteed that the cameras do not move and can be placed close to the eye (around 25 mm away), while still capturing the pupil and its movements.

The joint between the frame and the custom mount is lockable with a screw, so the mount and the cameras will not move without further effort. Consequently, the relationship between eye, cameras and scene is rather fixed and re-adjustments are done no more frequently than for other head-mounted eye tracking devices. The custom mount was designed with Autodesk Fusion 360[13] and printed by a Formlabs Form2 3D printer[14] using transparent synthetic resin (GPCL02). All parts together (IR + corneal camera + custom mount) have a weight of 14 grams (compared to the 4 grams of the original Pupil Labs eye camera), and are therefore still comfortable to wear. The total weight is comparable to the default monocular device, as we removed the scene camera for *hEYEbrid*. The total size is $58 \times 28.25 \times 14.40mm$ (width x height x depth) compared to the $46.20 \times 10.60 \times 7mm$ of the original eye camera. Hence it does not disturb the user's field of view any more than the usual device. For *hEYEbrid-3C*,

[13] http://www.autodesk.com/products/fusion-360/overview
[14] http://formlabs.com/3d-printers/form-2/

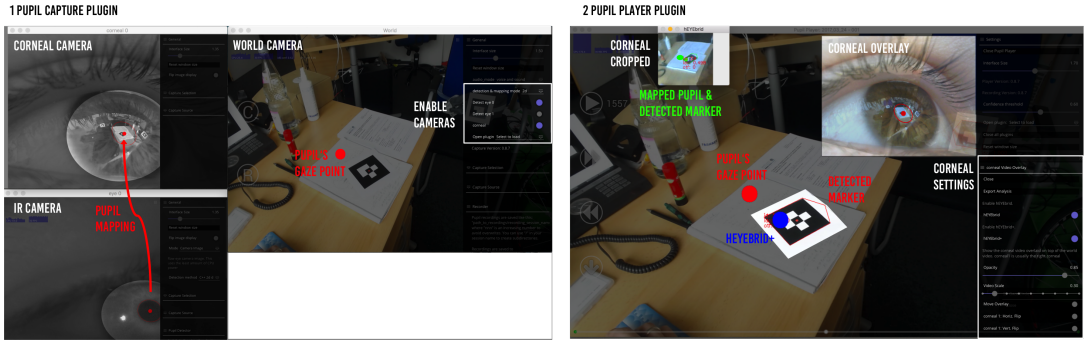


Figure 6.21: Integration of *hEYEbrid* and *hEYEbrid-3C* into the Pupil Labs framework. The Capture software is able to record data, whereas the Player is used for analysis and gaze computation in the different approaches.

we attached the device's default scene camera, a Logitech c920, capturing frames of 1920 x 1080 pixels at 30Hz. It has a 90° field of view with an auto-focus lens (shown in Figure 6.20) and a weight of about 80 grams.

We integrated *hEYEbrid* and *hEYEbrid-3C* into the Pupil Labs open source eye tracking platform, version 0.8.7[15]. The modified devices are thus usable with laptops and desktop computers. We followed the best practices to extend the software[16] and implemented two new plugins for the Pupil Labs Capture and Player software, as shown in Figure 6.21. The first extension we made is needed to integrate the new type of images from the corneal camera. It is similar to the existing plugin for IR images, in that it allows you to choose the camera source and to change its parameters (e.g., resolution, brightness, exposure time). The main purpose is to give direct feedback, i.e. the corneal image is overlaid with the pupil information, that is the ellipse and its center. We integrated the information about the corneal images into the software's interprocess communication. Hence it is possible to record the data of *hEYEbrid*, such as image data and coordinates of the mapped pupil, in conformity with Pupil Labs's logging and message format.

The second extension is made to play back recorded data while using *hEYEbrid* or *hEYEbrid-3C*. We integrated additional functionalities to visualize the corneal images as well as their cropped version. All computed gaze points are highlighted in a different color depending on the approach. The built-in mechanism for AR marker tracking is also enabled for corneal images. For instance, it is possible to label and

[15] http://github.com/pupil-labs/pupil/releases/tag/v0.8.7
[16] https://docs.pupil-labs.com/#plugin-guide

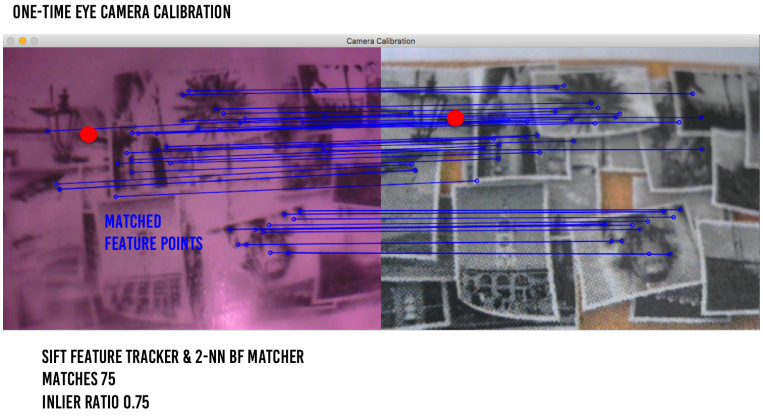


Figure 6.22: Setting up the eye cameras by computing the transformation between the infrared and corneal image planes. Image features are extracted and matched to compute a homography matrix, used to map a point from one image into the other.

track objects. Finally, the user is able to export recorded data as CSV files for further processing and analysis.

We actively made use of all extensions for the analysis of the user study. All described software components are developed in Python v2.7.6. By default, the Pupil Labs software is able to manage camera streaming via pyuvc[17] (based on libuvc) and provides a toolkit for a graphical user interface through pyglui[18] (based on OpenGL). For image processing (e.g., feature tracking and matching, homography computation), methods of the OpenCV 3.2 library[19] are used. For simple image manipulations (e.g., flipping and slicing), we use numpy[20], as it provides fast array processing.

For pupil tracking, we used the software's built-in pupil detector[78] (implemented in detect_2d.hpp). It implements dark pupil tracking and is based on edge detection, contour tracking and ellipse fitting. To map the pupil position onto the corneal images, the relation between both cameras was computed beforehand. This was done by estimating a homography matrix, needed for a bi-directional projective transformation between the IR and corneal image planes, as described before. Figure 6.22 depicts the one-time mapping computation. We captured images of the IR and corneal camera focusing on a printed image (size: $22 \times 15mm$), revealing lots of features. Both images are scaled to a resolution of 640×480 pixels.

[17] https://github.com/pupil-labs/pyuvc
[18] https://github.com/pupil-labs/pyglui
[19] http://opencv.org/
[20] http://www.numpy.org/

The homography matrix **H** is computed by first doing a feature detection using the scale-invariant SIFT algorithm[106]. In a second step, the image and the extracted features are matched with a 2-nn brute-force matcher. This simple matcher computes the distance between two features of two images, using the Hamming distance, and returns the two best matches. The resulting matches are used to find the homography matrix, by computing the perspective transformation, using the functions provided by OpenCV (cv2.findHomography()). Note that this procedure is only done once, during the construction process of the head-mounted device. As long as the cameras are fixed to each other, the procedure does not have to be repeated. Instead of applying a feature-based approach, one could also use fiducial markers for the one-time camera calibration.

For *hEYEbrid-3C*, the Pupil Labs eye tracker's scene camera is used to retrieve the scene images. The homography matrix between the cropped corneal and scene image planes is computed as described above. We also use the methods for SIFT and a 2-nn brute-force matcher provided by OpenCV. All extensions needed to integrate *hEYEbrid* and *hEYEbrid-3C* with the Pupil Labs framework together with the 3D models to build the hardware prototype are available on our GitHub repository[21].

6.3.2 Evaluation

We conducted a controlled user study to assess the validity of *hEYEbrid*'s approach and its gaze estimation accuracy in comparison to an existing state-of-the-art eye tracking approach and *hEYEbrid-3C*. We collected data using the head-mounted device that was described above. With this it is possible to record data from a Pupil Labs eye tracker, *hEYEbrid*, and *hEYEbrid-3C* simultaneously and compare the gaze estimation accuracy.

Experimental Design

We used a *within-subject* 3 x 1 design with the independent variable *Mode*, i.e. the method used for gaze estimation. We chose the following three different modes for gaze computation: *hEYEbrid* and *hEYEbrid-3C* implemented as described above. In addition we used a state-of-the-art monocular *Pupil Labs* head-mounted eye tracking device. We calibrated the Pupil Labs eye tracker for each participant separately. For this a 9-point marker calibration, standing in front of a 15-inch laptop screen, was performed using the built-in procedure of the Pupil Labs software. It is important

[21] `https://github.com/landerc/hEYEbrid`

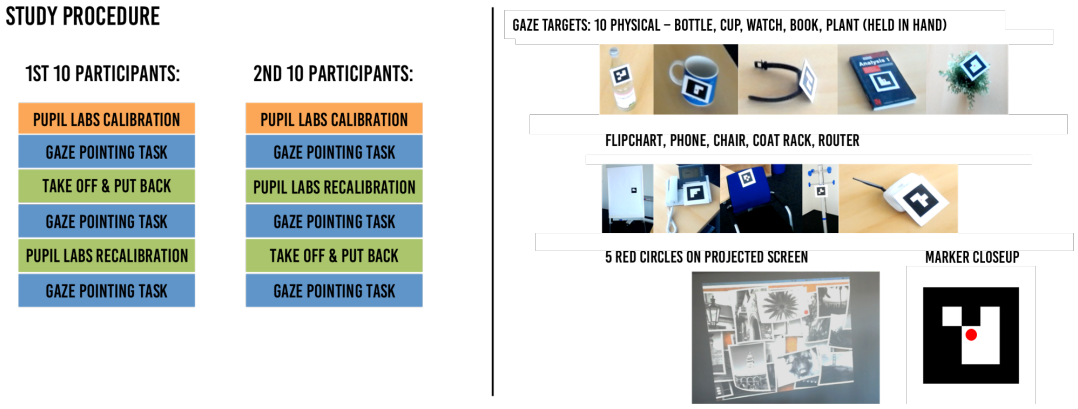


Figure 6.23: The study protocol showing the procedure of the experiment as well as the gaze targets used for the gaze pointing task.

to note that we did not perform any calibration for *hEYEbrid* and *hEYEbrid-3C*, as both methods are calibration-free.

Task & Procedure

We implemented a gaze pointing task, in which each participant had to focus on 15 targets in total (10 physical and 5 on-screen), as shown in Figure 6.23. The instructor named an object (e.g., *book*), that the participant then had to look for. When the object was found, the participant had to focus on the center of a visual marker (indicated by a red dot) that was attached to the object. For on-screen targets, the participant had to look at a red circle every time the instructor named the target *projection*. Looking at the target was verbally acknowledged by the participant. Only then, the sampling was started, which lasted eight seconds for each target. During the task, the participant was able to freely move around. However, while fixating a target, participants tend to keep standing at the same location. Figure 6.23 shows the schedule of the procedure. In the beginning of the experiment, we calibrated the Pupil Labs eye tracker. After this, each participant had to perform the task three times. Between the tasks half of the participants first took off and put back on the head-mounted device (used to record each mode) and finally re-calibrated the Pupil Labs eye tracker. The other half of the participants did this the other way around, to eliminate the order effect.

We collected all the data, i.e. the raw video data from the infrared, corneal and scene camera, as well as information about the pupil tracking, its mapping and the computed gaze values for each approach in parallel at 30Hz. This was possible since

Figure 6.24: The study setup showing the distribution of the tangible objects (attached with AR markers) in the room and a participant gazing at the book. In addition, the projection including all targets is shown.

we already bundled the necessary three cameras (IR eye, corneal and scene camera) for *hEYEbrid-3C* in one device. Note, that all other modes required only a subset of these three cameras. We captured 240 samples per object (8 seconds x 30 Hz), leading to a total of 3,600 samples per task (240 x 15 targets). As we only recorded after the acknowledgment of the user – when she started to fixate at the target center – we did not have to discard samples to account for the time that was needed to search for the object. Thus, we recorded 10,800 samples per participant (3,600 x 3 sessions), leading to a total number of 216,000 samples for each mode (*Pupil*, *hEYEbrid* and *hEYEbrid-3C*).

Apparatus

Figure 6.8 shows the study setup in a 360° picture. We set up a $4m^2$ (200 x 200 cm) square area, in which the participants were able to freely walk around. We did so to simulate a more realistic setting, in which people are moving around. Eight large tables (each 100 x 200 cm), forming four blocks of 200 x 200 cm, were arranged around this area. We used ten tangible objects, each different in size and form, of which seven were placed on the tables. The order of the objects' positions remained the same during the entire experiment. To track the objects and mark the gaze target, each object was equipped with an AR marker with a red dot at its center (see close-up image in Figure 6.24). Therefore, we printed ten AR markers – five of size 5 x 5cm and five 10 x 10 cm – and attached them onto the objects. The small markers (5 x 5cm) indicated that the object had to be picked up by the user before focusing on it. In addition we used a projected screen with a size of 210 x 160 cm (diagonal: 264 cm), which had a resolution of 1280 x 1024 pixels. Five additional targets were

shown on the projected screen represented as red circles (diagonal: 40 px = 6.4 cm). The distribution of the on-screen targets is shown in Figure 6.24. Note that only one on-screen marker was shown at a time.

To validate the approach of *hEYEbrid*, we computed the gaze estimation accuracy and compared it against the values for *hEYEbrid-3C* and the Pupil Labs eye tracker. This eye tracker provides a gaze estimation accuracy of 0.6° with a precision of 0.08° [78]. Other prominent state-of-the-art devices, such as the SMI Eye Tracking Glasses 2, achieve a slightly better accuracy of 0.5° [22]. Note that the values reported by the manufacturers were measured under optimal laboratory conditions. Since Pupil Labs provides an open source development framework, it perfectly fits the purpose of our research. Following best practice in eye tracking, we computed the gaze estimation error in degrees of visual angle to compare the results obtained by each approach (as defined in [59], Section 2.6.2). This error describes the difference between the gaze target (indicated by the red dot of an AR marker or by a red on-screen circle) and the computed gaze point according to the mode. The smaller the error, the more accurate the mode.

In the case of the Pupil Labs eye tracker and *hEYEbrid-3C*, we used the scene camera image as a basis to compute the distance between the gaze point and the center location of the gaze target (AR marker or on-screen target). For *hEYEbrid*, we used the AR marker or on-screen target reflected on the corneal image and computed its distance from the pupil center. For marker detection and tracking, we used Pupil Labs's built-in marker tracking plugin that is based on ArUco[23]. For the analysis, i.e. the marker tracking and the computation of gaze accuracy, we used the player software including the custom plugins, as described above. We were able to detect the markers without problems in the scene and the corneal images. To convert the raw pixel distances to the degree of visual angle, the pixels per degree have to be estimated. For *Pupil* and *hEYEbrid-3C*, we used the field of view of the scene camera (90°). For *hEYEbrid*, we used 30° for the field of view of the pupil, as reported in [128, 129].

Participants

In total 20 participants (5 female) between 23 and 38 years old (M=28.8 years, SD=3.31) participated in the data collection. All participants were recruited from a

[22] https://www.smivision.com/wp-content/uploads/2017/05/smi_prod_ETG_120Hz_asgm.pdf
[23] http://www.pupil-labs.com/blog/2013/12/036-release.html

local university campus and had normal or corrected-to-normal vision; none reported any form of visual impairments (e.g., color blindness).

Results

We first computed the overall gaze estimation error for all three modes across all tasks and all objects. Using *hEYEbrid* gave the lowest error (M=2.19°, SD=0.92°), followed by *Pupil* (M=3.09°, SD=2.26°), and *hEYEbrid-3C* (M=4.05°, SD=1.04°). We performed a one-way ANOVA on the gaze estimation accuracy across all three modes and found a significant difference ($F(2,17) = 7422.22$, p<0.001). To further assess this finding, we computed three paired independent-samples t-tests on gaze estimation error. We found a significant difference in gaze estimation error between *hEYEbrid* and *Pupil* ($t(48)$=-4.11, p<0.001). No significant difference was found between *hEYEbrid-3C* and *Pupil* ($t(48)$=-0.45, p=0.65) and *hEYEbrid* and *hEYEbrid-3C* ($t(48)$=-0.87, p=0.38).

	hEYEbrid		hEYEbrid-3C		Pupil	
	M	*SD*	*M*	*SD*	*M*	*SD*
physical	2.22°	0.91°	4.26°	1.12°	3.27°	2.22°
on-screen	1.64°	0.28°	2.17°	1.21°	2.84°	1.38°

Table 6.2: Means and standard deviations for the computed gaze estimation errors for all modes on the two target groups.

We subsequently analyzed the gaze estimation error for the different target groups – on-screen and tangible objects. The lowest error was achieved for the on-screen targets using the *hEYEbrid* mode. Table 6.2 lists these results for each mode.

Independent-samples t-tests were conducted and we found the same results for the tangible target group as for the overall results. *hEYEbrid* and *Pupil* differed significantly according to gaze estimation error ($t(48)$=-3.06, p<0.05), as well as *hEYEbrid* and *hEYEbrid-3C* ($t(48)$=-1.3, p<0.05). No significant difference was found between *hEYEbrid-3C* and *Pupil*. For the on-screen markers, we were able to measure a significant difference in gaze estimation error between *hEYEbrid* and *Pupil* ($t(48)$=-2.93, p<0.05). We found no significant difference between *hEYEbrid-3C* and *Pupil* or *hEYEbrid* and *hEYEbrid-3C*.

We were further interested in the impact of the calibration of the *Pupil* eye tracker. We compared the gaze estimation accuracy of all three modes for each task sepa-

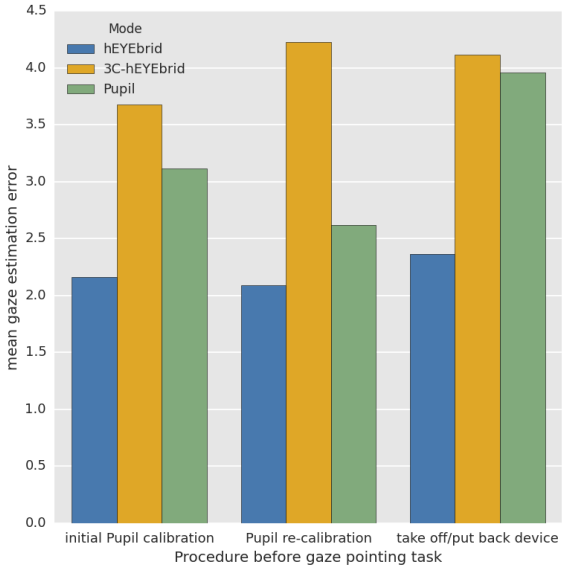


Figure 6.25: Mean gaze estimation error for each mode for each of the three gaze pointing tasks. Each column shows the mean gaze estimation for each mode after the procedure done before the gaze pointing task, i.e. after the initial *Pupil* calibration, the *Pupil* re-calibration and taking off the device and putting it back on.

rately. The tasks differ in the procedure that was done before doing the gaze pointing task (shown in Figure 6.23). Figure 6.25 summarizes the results. It shows the mean gaze estimation error for all three modes after the initial calibration of the *Pupil* eye tracker (shown in the first bar group). In the second group the mean gaze error after a re-calibration of the *Pupil* eye tracker is shown. In the third group the mean gaze estimation error of the different modes after a simulated calibration drift, i.e after taking the device off and putting it on again, is shown. Note that we conducted no re-calibration of the *Pupil* eye tracker in the gaze pointing task after this procedure. We noticed the lowest error for *hEYEbrid* after the initial *Pupil* calibration (M=2.16°, SD=0.72°), the re-calibration of the *Pupil* eye tracker (M=2.09°, SD=0.79°) and the simulated calibration drift (M=2.36°, SD=0.80°). Remember that *hEYEbrid* was not calibrated.

We conducted independent-samples t-tests for all pause combinations within each mode. We noticed a significant difference in gaze estimation error between re-calibration (M=2.61°, SD=2.03°) and taking off/putting back the device (M=3.95°, SD=1.85°) for *Pupil* ($t(48)$=-3.27, p<0.05). We did not find further significant differences for the other modes.

We further conducted independent-samples t-tests for all pause combinations between modes. There we found a significant difference in gaze estimation error between *hEYEbrid* and *Pupil* after the initial calibration of the *Pupil* eye tracker ($p<0.05$) and after taking off/putting back the device ($p<0.001$). In addition we found a significant difference in gaze estimation accuracy between *hEYEbrid* and *hEYEbrid-3C* for all pauses (all $p<0.05$), as well as between *Pupil* and *hEYEbrid-3C* after initial calibration and re-calibration of the Pupil Labs eye tracker.

6.3.3 Discussion

Our results show that combining the corneal reflection within the pupil together with the pupil center in a hybrid approach achieves an average gaze estimation accuracy of 2.19°, compared to 3.09° for *Pupil* and 4.05° for *hEYEbrid-3C*. This supports our initial assumption that the pupil center, mapped from the infrared eye image on the reflected environment (i.e the corneal image), converges with the actual gaze point in the real scene.

With *hEYEbrid*, we are able to achieve a higher gaze estimation accuracy than existing approaches using corneal reflections in a mobile system. Takemura et al. [188] built a wearable device using a mobile phone. Their approach is based on corneal images and a model-based tracking approach for gaze estimation. They evaluated their system in a static single display scenario (24-inch screen), in which the participant's head was fixed at a distance of 70 cm from the screen. There, they were able to achieve 9.5° gaze estimation error on average.

Nitschke et al. [128] also built a head-mounted device using a corneal imaging camera. To realize gaze estimation they rely on 3D eye pose estimation. Similar to [188], they evaluated their approach in a single-display scenario (23-inch screen), in which participants were sitting 60 cm away from the monitor. They were able to achieve a gaze estimation error of 2.51°.

With *hEYEbrid*, we developed a method that enables accurate gaze estimation in a mobile and pervasive interaction setting. In our evaluation, we set up a realistic environment using a screen as well as tangible objects as gaze targets. Thus we simulated a more dynamic yet realistic setting. We intentionally chose this kind of accuracy evaluation, instead of doing a simple 9-point accuracy test on a screen. Nevertheless, we implemented a similar accuracy test by including the screen as one of

the gaze targets in the task. Note that we performed even better than *Pupil* using marker tracking for gaze estimation on a screen, since we achieved an average gaze estimation error of 1.64° compared to 2.84°. We noticed an even larger error of 3.27°for *Pupil* on the tangible gaze targets. This error stems from the 9-point calibration itself, which is a mixture of operator and system-controlled procedure. It was done on the laptop screen only once. We did not validate the calibration to account for correction. We did so to create a realistic baseline for comparison, in contrast to the reported values by the manufacturer. In doing so, we underline the drawback of this type of calibration, as already shown in [142], and highlight the advantage of *hEYEbrid*. How error-prone and uncontrollable the calibration process is can be also seen on the results during the task after initial calibration and after the re-calibration of the *Pupil* Labs eye tracker. We noticed a better gaze estimation accuracy for the latter of the two, although the participants did the same gaze pointing task in both cases.

Furthermore, the developed method can be considered as an advance in comparison to existing approaches that realize high gaze estimation accuracy [93]. Such approaches rely on an active user calibration that has to be renewed at regular time intervals. *hEYEbrid* is not influenced by a calibration drift caused by taking off the head-mounted device.

We did not find a significant effect on gaze estimation for *hEYEbrid* when taking off the device and putting it back on. However, we noticed a slightly worse result in this case. This may be caused by minimal changes in the image quality of the corneal images or less accurate pupil tracking. This also has a negative effect on the results for *hEYEbrid-3C*, as it builds on the output of *hEYEbrid*. As expected, the Pupil Labs eye tracker achieves much worse results caused by the simulated calibration drift, which invalidates the mapping function (as shown in [95]).

hEYEbrid-3C uses the two eye cameras plus an additional scene camera and is able to compute the gaze in the scene by matching the corneal image onto the scene image. Note that the detection of objects via AR markers and the display tracking via natural features is performed in the scene images. It achieves less accurate results than the other methods, except for the on-screen targets. In this case an average gaze estimation error of 2.17° was achieved, compared to 2.83° for *Pupil* and 1.64° for *hEYEbrid*. A possible explanation is that *hEYEbrid-3C* relies on natural feature tracking to update its frame-based calibration. To match the pupil center in

the scene image, it needs to compute the homography matrix between the corneal and scene camera images. Hence the method does two transformations (from IR to corneal and from corneal to scene images) that might introduce errors. Note that we also did not account for any corrections of the images; for instance, we did no undistortion of corneal images. The study was conducted in a minimally furnished room. Therefore the image features were very limited, except when looking at the projected screen, as it showed feature-rich content. Thus our setup shows the limitations of *hEYEbrid-3C* by design. However, when enough features are available, this method also performs well, as indicated by the good results for the on-screen targets (2.84° on average). Nevertheless, this approach relies on three cameras, of which the scene camera especially is an issue, as it can cause discomfort for other people.

In our experiment we showed the applicability of our approach in a mobile indoor lab setting. Although we did not enforce constraints such as remaining seated and keeping the head still, we controlled the correct initial setup of the device (i.e. correct camera-eye distance). Consequently the results show the potential of using *hEYEbrid* in indoor settings, such as retail stores, shopping centers or airports. We also believe that the concept is usable in outdoor settings to a certain extent. The current implementation calls for at least stable lighting conditions (e.g. sunlit or partially cloudy).

6.3.4 Making hEYEbrid Mobile

The results of the conducted user study showed that connecting an infrared eye and a corneal camera in a hybrid approach is suitable for user-calibration-free gaze estimation. Thus, we designed and implemented a mobile and wearable eye gaze tracking system that can be used to conduct eye-tracking experiments in the wild and to build interactive systems. For this, we connected the head-mounted device, equipped with the two eye cameras, to a mobile phone. We developed a *hEYEbrid* mobile application that is able to drive the two eye cameras. Moreover, it takes care of the pupil tracking as well as the mapping of the pupil from IR to corneal images, as described earlier. Basically, one can spontaneously start to use the system in everyday life settings. The application is able to record the camera streams as well as the information about the pupil positions, which can be imported into the Pupil Labs player software for later analysis and processing.

Figure 6.16 illustrates the application's interface, which we designed to be easy to

use. It guides the user through the setup of the system, showing the required hardware parts (the mobile phone and the head-mounted device). The main screen of the application is divided into three expandable views, integrated in a list view. The control view is used to inform the user about the connection and tracking state. Here, several plugins can be activated, like gaze estimation on screens. Two more views – the IR and Corneal views – show the camera streams and give direct feedback about the processing, i.e. the pupil tracking and mapping. This is of great importance, as it provides the user the ability to adjust the eye-camera distance to get sharp corneal images, which at the same time results in the optimal mapping. In this way we overcome one of the aforementioned limitations. The application can be easily extended by several functionalities which make use of the computed gaze point and the corneal reflection.

Besides the *hEYEbrid* mode, we also implemented mobile versions for *hEYEbrid-3C* and Pupil Labs monocular eye trackers. With a fully equipped head-mounted device, i.e. three cameras (infrared, corneal and scene) one can decide which mode to use. However, keep in mind that only *hEYEbrid* and *hEYEbrid-3C* are calibration free in that they only need to establish a camera mapping once, unlike the Pupil Labs eye tracker.

Applications

Mobile calibration-free gaze estimation opens up the space for a variety of application cases. We distinguish between the usage of *hEYEbrid* as an analysis tool and an input device.

Coming back to the introduction, there is increasing interest in the integration of gaze tracking in AR and VR devices. Both the reflection of an AR glasses screen (e.g. HoloLens) as well as that of a VR headset display are visible on the corneal images. The availability of information about where a person is looking in tasks such as industrial maintenance will be of great benefit to guide a worker, for example. A virtual reality headset offers the ideal conditions to integrate *hEYEbrid*, since nothing other than the screen content (i.e. the VR environment) is reflected on the human cornea. Moreover, the distance between the VR headset's display and the eye is fixed; thus, we do not have to deal with blurry corneal images due to defocus. However, the current camera clip would have to be scaled down to fit into a headset. Possible use cases range from foveated rendering [152] to hands-free interaction.

hEYEbrid also enables the creation of interactive applications at pervasive scale. A person's gaze, which is available all times, can be shared with connected devices in the environment. Gaze estimation on public displays can be used to enable the development of interactive screens. A person's gaze can be used as a direct or indirect input modality to control the screen by pointing, or support the user in reading text on a public display, as in [101].

Besides utilizing *hEYEbrid* for creating gaze-based interfaces, corneal images can be analyzed to gain insights in human cognitive processes. That is, *hEYEbrid* enables gaze estimation for real-life in-the-wild scenarios. The mobile system can be used to conduct user studies in a lightweight manner. Multiple persons can be equipped with the system, to collect a large amount of data in parallel. This data can be analyzed post-hoc for different application use cases, such as landmark extraction [102] or activity analysis [178, 67]. Since our daily behavior is reflected on our eyes, corneal images can serve as a basis for camera-based lifelogging [97]. If a person is fixating an object, this behavior is reflected in the corneal image. For example, such information can be used for attention measurements.

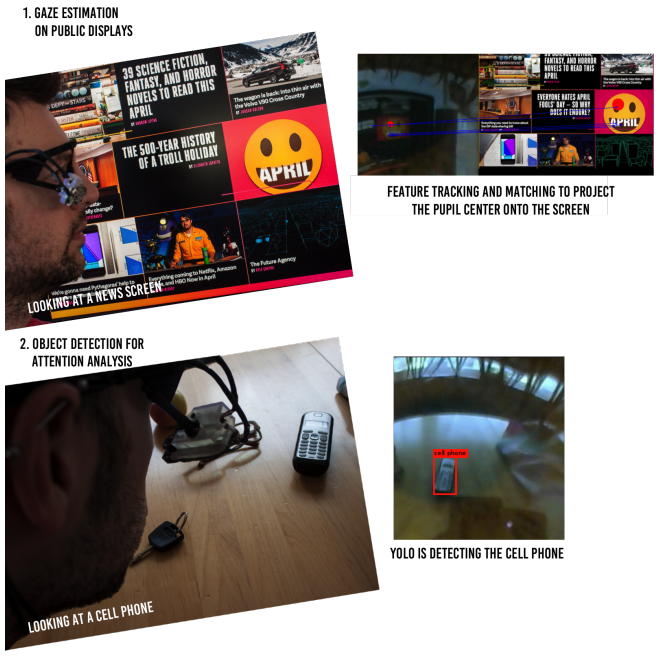


Figure 6.26: Example applications: 1. Gaze estimation on a display, based on natural feature tracking. 2. Object detection in the corneal image using a CNN.

To demonstrate the potential of *hEYEbrid*, we integrated two example plugins: gaze estimation on public displays, and automatic object detection, reflected on the user's eye within the area of the pupil. Both example applications are shown in Figure 6.26. The implementation of the example applications as well as the dedicated mobile *hEYEbrid* application is outlined in the following.

Implementation

The architecture of the final mobile prototype is shown in Figure 6.27. It consists of two main components on the hardware side: (1) the head-mounted device and (2) a Nexus 6P phone, running Android 7.1, to drive the device. All cameras are attached to a USB 2.0 hub that is directly connected to the phone via USB-C. There is no need for an extra power source to power two cameras. However, to run *hEYEbrid-3C* or a Pupil Labs eye tracker, an additional power source has to be used. Running the application on the phone, its battery lasts for approximately five hours. Note that during this measurement, Bluetooth, WIFI and GPS were also enabled. We will explain the software architecture only for *hEYEbrid* in the following. The other modes are developed in the same manner.

The software architecture of the Android application consists of three main parts: (1) the backend to process all images, (2) the frontend for live visualization and to control the app, and (3) the plugins to add advanced functionalities. The *hEYEbrid* application mode is developed as a native Android application, based on the Android SDK 25. The backend parts are developed in C++ and included via Android's native development kit (NDK), version 7. To stream both camera streams at once over USB, we use libuvc, a cross-platform library for USB video devices. In particular we include an existing library, wrapping libuvc[24], for usage with Android. This library allows us to set the available camera settings (e.g., frame rate, resolution, brightness) and to control the bandwidth factor. The IR camera frames are streamed with a resolution of 640 x 480 pixels, corneal images with a resolution of 1280 x 960 px, both at 30Hz. For image processing we use the Java native interface to realize the pupil tracking. We included PUPIL's built-in pupil detector (defined in detect_2d.hpp) and all its dependencies. For this, we had to include specific versions of Eigen[25] and TBB[26] for Android. The method returns an ellipse describing the pupil structure. This is transformed onto the corneal image applying the homography matrix that was computed

[24]https://github.com/saki4510t/UVCCamera
[25]http://eigen.tuxfamily.org
[26]https://www.threadingbuildingblocks.org/tbb-mobile

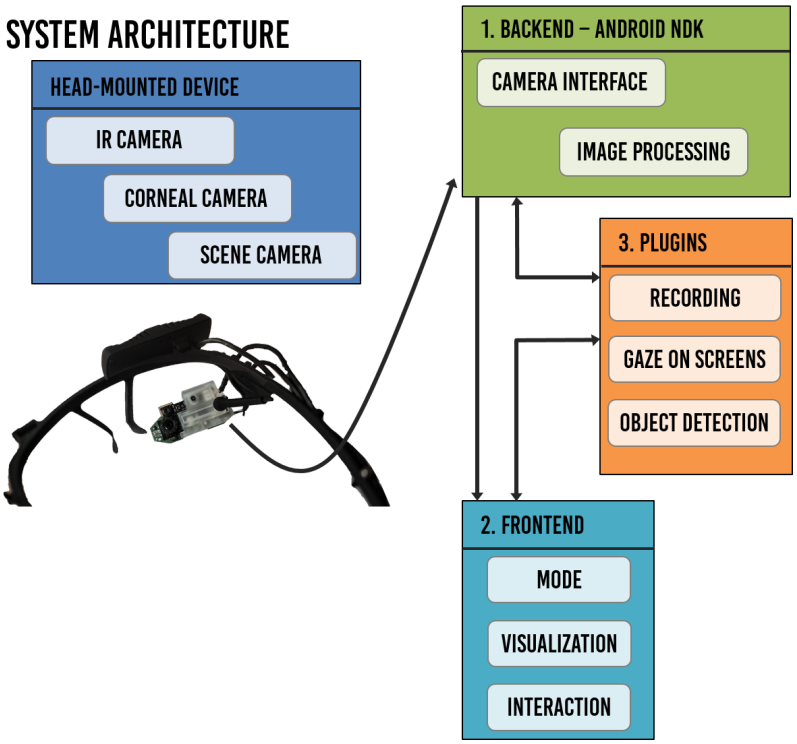


Figure 6.27: Architecture of the mobile system usable with *hEYEbrid*, *hEYEbrid-3C* and a Pupil Labs eye tracker. The hardware is made up of a head-mounted device containing two or three cameras, depending on the mode. The software side contains three main parts.

beforehand during construction. For the projection we make use of the OpenCV 3.2 library for C++. Finally, the backend offers a callback delivering the pupil center and the cut-out corneal image.

The frontend implements three views in different panels. We chose a list-like visualization of the views and included the ExpandableLayout library[27] to smoothly expand and collapse the views. We used Android's Material design for all UI components.

To integrate advanced functionalities, the application provides interfaces to the backend and frontend usable by plugins. We implemented two example plugins: For gaze estimation on displays, we use a similar approach to GazeProjector [93]. The idea is to compute the spatial relationship between the head-mounted device and the surrounding displays. In contrast to existing approaches [193], we use the cropped corneal image as a template to look for on the display's content. In both images key features

[27] https://github.com/cachapa/ExpandableLayout

are extracted using AKAZE [145] and matched using a 2-nn brute-force matcher. To estimate the spatial relationship, the found key feature pairs are used to compute a homography matrix. For the algorithm, the screen's content is streamed via screenshots from the display to the mobile application. For this a script has to be executed on the computer connected to the display. It resizes the screenshots to 480 x 360 pixels and pushes them onto the mobile phone. The mobile application sends back the estimated gaze position to use this for further processing.

For object extraction, we use the YOLO[28] framework for real-time object detection [158] based on neural networks. We use the pre-trained weight file (yolo.weights) that is already available within the framework and the standard configuration (yolo.cfg). To speed up the processing, the cropped corneal images are sent as batches to a server that does the processing. After a successful object detection, the result with the highest detection probability, closest to the pupil center, is sent back to the mobile phone. The main software part (i.e., the backend) runs at 25–30 fps on the Android phone (Nexus 6P). The gaze estimation plugin has to do some more complex computations (e.g., feature matching) and thus runs at 25 fps. The object detection plugin is only limited by the transfer rate of the images to the processing server, as YOLO is able to process the stream at up to 90 fps. Hence the plugin runs at at 25–30 fps like the main program. The source of the mobile application is also publicly available on Github[29].

6.3.5 Overall Limitations

Apart from its numerous advantages over state-of-the-art eye tracking systems, *hEYEbrid* also comes with some limitations. Our concept of gaze estimation requires two cameras, bundled into one enclosure that is mounted in front of the user's eye. The construction may interfere with the field of view. However, using a dedicated hardware prototype with both cameras on one board, and smaller cameras, could reduce its total size. This would make the head-mounted device even smaller, lighter in weight and more comfortable.

Further, we used AR markers for object detection in our experiment, to robustly track the objects. Nevertheless, we believe that with a higher resolution eye camera, objects are able to be detected via feature tracking or a trained classifier.

[28] https://pjreddie.com/darknet/yolo/
[29] `https://github.com/landerc/hEYEbrid`

Currently, computing the mapping between the infrared and corneal camera image plane is the biggest limitation. As described above, we rely on a homography matrix that computes a planar mapping in 2D. If the distance between the camera and the user's eye changes to a great extent, the mapping will become more incorrect. However, we handle this limitation in the following way. To compute the homography matrix we record images from both cameras at a fixed distance with sharp focus. Consequently, we obtain an optimal mapping in this setting. When putting on the device, the camera is placed in such a way that the focus is also sharp. Hence, the distance between cameras and eye is nearly the same as during the mapping step. The good results of our experiment after taking off and putting back on the device support this assumption. In a totally uncontrolled environment the user has to make sure to place the camera at a focused distance from the eye. It would be possible to support that procedure by computing the current focus value of the corneal image (e.g. by using depth sensors). In this way it might be possible to guide the user in placing the camera at the right distance from the eye.

Besides camera-eye distance, eye-object distance is the second source of blurry corneal images. That is, if the user is focusing on an object at a far distance, its representation on the corneal image is rather unsharp, caused by the mirror properties. Consequently our current implementation has a certain working distance. Using cameras with autofocus may help to circumvent this limitation. Another possibility is to use neural networks that are specifically trained on such blurry images to handle these cases.

We have little knowledge about the long-term accuracy of *hEYEbrid*. The experiment took thirty minutes at maximum per participant, caused by longer pauses between the tasks. Consequently the presented results are representative for a usage time within half an hour. It is noteworthy that the homography matrix between both cameras was not updated during the entire experiment, which lasted over several days.

Also, *hEYEbrid* has difficulties working in dark environments, as nearly no information will be reflected on the cornea. Installing cameras with a higher ISO sensitivity could help to make information visible on the corneal images, even in darker settings. Also, changes in lighting might affect the video quality of corneal images and thus the processing. Since the human pupil is sensitive to light, a bright environment will have a miotic effect (i.e. shrinking of the pupil). This directly influences the size and

resolution of the corneal image extracted by *hEYEbrid*. However, the approach might still work, as we rely only on the pupil center that gives us the gaze point in the reflected environment. The extracted field of view will be smaller, but sufficient to map the gaze on smaller objects (e.g. a mobile phone), whereas it will be insufficient for large objects (e.g. cars or buildings).

In summary it makes sense to explore cameras with different properties and configurations, especially to make this approach robust for outdoor usage. Note that when integrating a camera that offers capabilities like auto-focus, auto-iris and high ISO sensitivity, the complete device will get more expensive. Nevertheless, we believe that *hEYEbrid* is a promising concept for user-calibration-free gaze estimation in pervasive settings to enable spontaneous gaze-based interaction as well as eye-tracking experiments in the wild.

6.3.6 Summary of Findings

In this section, we presented *hEYEbrid*, an approach for calibration-free and accurate gaze estimation in pervasive settings (cf. *Ubiquitous Computing for Eye Tracking Continuum* in Figure 6.1, complexity (K x L) : M : N). In contrast to existing systems, *hEYEbrid* combines the images from an infrared eye and a corneal camera in a hybrid approach and works without any prior user calibration. Hence, its accuracy is not influenced by calibration drifts. The concept is integrated into Pupil Labs' open-source eye-tracking framework, by adding a corneal camera to the head-mounted device and developing plugins for the capture and player software.

We conducted a user study in which we used a mobile setup to evaluate *hEYEbrid*'s gaze estimation accuracy against a state-of-the-art Pupil Labs eye tracker and *hEYEbrid-3C*, an extended version using the scene camera as additional image information. We found that our approach performs better than the well-established Pupil Labs eye tracker. In addition, it compares well to existing approaches based on corneal imaging. The results are very promising and underline the potential to realize mobile gaze-based interaction in everyday life settings.

We finally transferred the approach into a mobile and wearable system which is able to react in real time. It consists of a head-mounted device (with the two eye cameras) and an Android-based mobile phone, making it usable out of the box. Finally, we developed a novel approach for real mobile gaze estimation achieving reasonable ac-

curacy. In the following we will discuss how this approach contributes to the research question of this thesis.

6.4 Application to research question

In this section we will demonstrate the contribution of the presented system to the problems defined in Section 1.4 and the remaining parts of the research question formulated in Section 1.6. We will discuss the involvement of *EyeMirror* and *hEYEbrid*, as both constitute major achievements investigating calibration-free gaze estimation. In the following we will derive the answers to the two remaining sub-questions while highlighting the comparison between the above presented systems.

At the outset of this thesis, we emphasized the key challenge of current state-of-the-art head-mounted eye tracking devices and the problems that come along with it. In Section 1.4, we discussed that these systems are not ready for use outside a controlled environment (like a laboratory) in a mobile setting (e.g. for spontaneous interaction or long-lasting in-the-wild experiments). The main reason for that is the need to perform a calibration procedure in order to estimate different parameters to translate pupil into gaze positions. The issues interrelated with the calibration are user-dependency, drift, lack of invariance under location and orientation change, the need for supervision and the parallax error (cf. Section 1.4). So far, the presented approaches in the previous chapters dealt with most of these sub-problems. For the first time, we addressed the calibration problem itself in this chapter. To create solutions for the main problem, we propose two consecutive concepts enabling calibration-free gaze estimation, both using the concept of corneal imaging (cf. Section 3.3). We continuously refined the developed prototypes concerning performance in terms of gaze estimation accuracy and mobility. Figure 6.28 highlights the progress from a low- to a high-fidelity prototype, i.e. *EyeMirror* connected to a laptop or a mini-computer, to *hEYEbrid*, usable with a mobile phone.

In Section 6.2 of this chapter, we presented *EyeMirror*, a wearable system that requires only one camera placed in front of the user's eye. By capturing the environment reflected on the human cornea (i.e., the area covering the eye's iris and pupil), we realize gaze approximation on surfaces (e.g., a display). Its underlying concept is to match the corneal image into the environment using natural feature tracking. Therefore it is necessary to have visual information about the surroundings present. Our proof-of-concept realization demonstrates the gaze approximation of *EyeMirror* on displays,

Figure 6.28: Progress made within the development of a device for mobile calibration-free gaze estimation.

similar to the concept of *GazeProjector* (cf. Chapter 5). Obviously one can substitute displays by any other surface, as long as the system is aware about the appearance, as we did it with Section 5.6. Although the developed method gets along without any user calibration, it is only possible to approximate the user's gaze with an average angular error of 4.03°. This is about four times worse than with *GazeProjector* (avg. gaze estimation error 1.78°).

However, the *EyeMirror* system is a first step towards calibration-free gaze estimation while also addressing the parallax error. Hence we can start to think about the answer to the fundamental research question, formulated in Section 1.6. For this, we will take a look at the sub-questions that have so far remained unanswered:

6. How can we realize accurate gaze estimation without a user-dependent calibration?

7. What is the benefit of a mobile and wearable eye tracker?

8. How can we design a mobile eye tracking framework without creating new restrictions?

With the *EyeMirror* approach, we can at least deliver a partial answer to question number six. For this purpose we reiterate the idea behind *EyeMirror*: The corneal

image (on the left) contains information about what a person is currently looking at, in this case a screen's content. As we already showed with *GazeProjector*, it is possible to compute the spatial relation between the display content and its representation in the scene camera image. The corneal images are comparable to the scene camera images of a head-mounted eye tracker, as they reveal a distorted version of the user's field of view.

In the evaluation of *EyeMirror*, we showed that it is sufficient to cluster the extracted key feature pairs and define its center as gaze point. This approach can moderately compute the current gaze of a person on a display. This might be sufficient to indicate the region or area of interest useful for attention measurements. However, *EyeMirror* is not the right answer to the questions from above, as it is too imprecise and not as easy to use as one might have imagined. Note that it requires a specific mini-computer plus an external battery pack. We consider it as a promising step towards a new generation of less-invasive head-mounted eye trackers and a good baseline for further investigations on corneal imaging in human computer interaction. We identified the inaccurate limbus extraction together with using the eye center as gaze reference as the key problems that cause the inaccurate gaze estimation. The limbus tracking results in low-quality corneal images and gives no exact reference to the actual gaze. Consequently we pursued another cycle of the iterative development model to resolve these issues, which led to the *hEYEbrid* system.

In Section 6.3 of this chapter, we proposed the concept of a revised prototype based on corneal imaging for gaze estimation. With *hEYEbrid* we implemented a system that enables head-mounted eye and gaze tracking at any time and any place in a suitable fashion. In order to accurately extract the pupil position and its movements, we employ the well-established method of active IR illumination eye tracking (cf. Chapter 2). We combined two cameras in front of the eye to acquire both IR images for pupil tracking and corneal images simultaneously. Simply put, we moved the scene camera of a head-mounted eye tracker below the eye and turned it into a corneal imaging camera. As explained in Section 6.3, we transform the pupil positions into the corneal images to accurately extract the reflection of the environment within the area of the pupil. In the same way, we obtain a reference point for the user's gaze given by the pupil center. Figure 6.28 depicts the final-high fidelity hardware prototype and the software running on a mobile phone, enabling gaze estimation in a plug & play manner.

With *hEYEbrid* we gave the answers to the remaining two of the three open questions from above. With *hEYEbrid*, we finally created a method that is competitive in terms of accuracy with our previous approaches (such as *GazeProjector* in Chapter 5). The fact that this approach is calibration-free underlines its benefit towards state-of-the-art head-mounted and remote eye tracking systems.

6.5 Attention Maps in Corneal Imaging using hEYEbrid

In Sections 6.2 and 6.3 we presented an alternative generation of head-mounted eye tracking approaches. We showed the potentials of these methods, as they resolve the problem of a user-dependent calibration. In order to further enhance the benefit, we want to focus on methods for analyzing corneal image based eye tracking data. As when conducting eye tracking experiments, researchers face fundamental questions like:

How many fixations took place? What is the mean duration of all fixations? and What are the areas of interest?

Currently, such analytics can be done by either using the eye tracking vendor's software tools (e.g., Tobii Pro Lab[30] or Pupil Labs Player[31]), or independent software [30]. However, these tools are not applicable to corneal imaging-based eye tracking and gaze estimation. This is because of the difference in the approach itself, i.e. world plus eye camera versus eye camera only.

In this section, we propose a first approach to include analytics in corneal imaging-based eye tracking data. In particular, we developed an extension for *hEYEbrid* (see Section 6.3) to realize fixation extraction to answer the aforementioned questions.

Fixation Extraction Approach

To make corneal imaging based eye tracking applicable in research studies and usable as an input modality, we have to investigate how to transfer the existing analysis metrics. The human eye is able to perform five different kinds of movements, namely fixation, saccades, smooth pursuits, vergence and the vestibulo-ocular reflex, as outlined in Chapter 2. With our approach, we focus on the extraction of fixations, as

[30] https://www.tobiipro.com/product-listing/tobii-pro-lab/
[31] https://docs.pupil-labs.com/#analysis-plugins

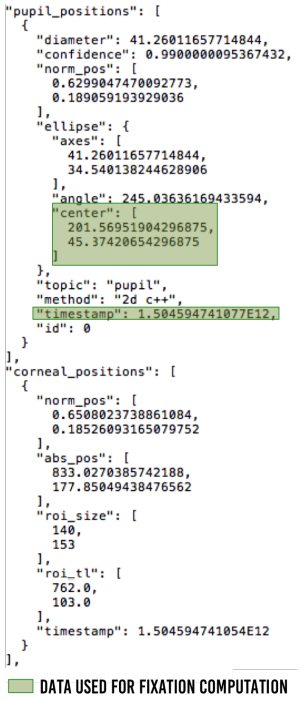
they are the most prominent events in eye tracking data. Fixations describe the state of resting the gaze on a specific object within the central vision for a certain amount of time, typically around 200–400 msec [162]. The term is somewhat misleading, as the eye is not staying still but rather doing micro-movements. Usually, fixations are extracted to measure the attention of the user on specific objects in the field of view. In head-mounted eye tracking, fixations are computed using the estimated gaze point in the world camera. This is done by estimating the spatial variation of the gaze point within a certain time interval. Simply put, if the gaze point is rather constant, a fixation is detected.

It is important to note that head-mounted corneal imaging systems usually do not have a world camera. Thus, we adapted the general computation of fixations as follows. We use the corneal images together with the absolute pupil positions as input. Basically, the spatial and temporal variances of the pupil movements are used to define a fixation event. We set a time window according to [162] in which pupil movements within a certain threshold are allowed. If a fixation is found, the corneal images connected with the pupil positions are stitched together. Finally, a fixation is associated with one corneal image.

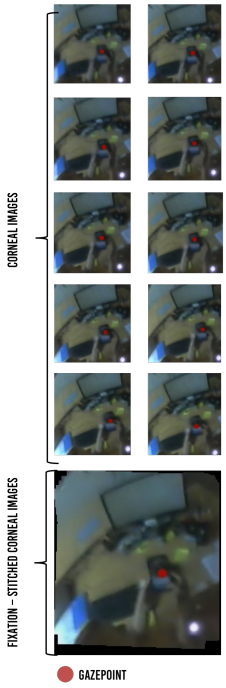
To implement the described method for fixation extraction, we extended the mobile *hEYEbrid* application to enable data recording and implemented a Pupil Player plugin[32] for the actual fixation extraction afterwards. A complete record of one data frame is depicted in Figure 6.29. It contains information about the pupil positions (timestamp and ellipse describing the pupil) and the matched corneal positions (timestamp and transformed pupil position). The scheme is similar to the standard Pupil Labs[33] data format to ease compatibility.

Figure 6.29 shows the workflow of the algorithm on data recorded with the head-mounted device. The pupil positions are analyzed using an alternative version of the fixation detector plugin of the Pupil Labs player. The dispersion of the pupil positions is thereby computed sequentially. If this value is within one degree, and data for at least 200 msec was analyzed, a fixation event is created. If the dispersion threshold is exceeded, the algorithm restarts. If a fixation event was found, the corresponding corneal images are clustered to one image to match the fixation. For implementation,

[32] https://docs.pupil-labs.com/#pupil-player
[33] https://docs.pupil-labs.com/#data-format



(a) An example of a recorded data sample containing the information on pupil and corneal positions. The absolute position of the tracked pupil center, together with the timestamp, is used for fixation extraction.



(b) Stitching ten corneal images that correspond to a fixation with a duration of 300 msec. The resulting image contains the gaze point and all scene information after the stitching process.

Figure 6.29: (a) The data format used to extract fixation data; (b) the stitching approach used to cluster image data

we are using OpenCV's image stitcher[34] with mode *scans*. The resulting image also includes the averaged gaze point (equal to the pupil center as shown in Section 6.3) of all gaze points of each data frame of the current fixation. In this way, no information is lost and the overall image data is compressed. In the example presented in Figure 6.29b, ten images (126 KB) are reduced to one image (11 KB), containing all necessary information. While this is not a huge improvement in this case, data recorded over several hours and days can be dramatically reduced to a minimum.

After extracting the fixations, we can go one step further and cluster several fixations within a certain time interval. Simply put, we do another round of image stitching of the clustered corneal images and their associated fixations. Figure 6.30 shows the idea behind this additional step. It results in a corneal panorama image showing the spatial distribution of the chosen fixations. This can be used to indicate the points of interest and create areas of interest for a large set of data.

[34] https://docs.opencv.org/trunk/d8/d19/tutorial_stitcher.html

Figure 6.30: Panorama stitching to cluster multiple fixations. The resulting image contains all three fixations mapped onto the stitched image.

The resulting information could be used to create attention maps which can be integrated over time and space: that is, a cluster of fixations on a panorama of corneal images. A side-effect of the whole process is the compression of a sequence of corneal images and eye tracking data.

The work presented in this section lays the foundation for further improvement in the future. First of all, the extensions we developed enable only post-hoc fixation computation. The next step is to realize a real-time version of the presented approach. Hence, it will be possible to directly record compressed data, i.e., only the extracted fixations. In this way an automatic information extraction can be realized, as sketched in Section 6.2, because the size of the data set is smaller while containing the same amount of information.

6.6 Summary

In this chapter, we began to address the matter of the calibration issue itself. For the first time, we tackled the actual problem at its source. As discussed, gaze estimation in general requires a specific set of input parameters which have to be acquired through a calibration procedure, which in turn causes various side-effects that have a bad influence on gaze estimation accuracy (see Chapter 1).

With *EyeMirror* we first demonstrated the possibility to build a head-mounted eye tracking device utilizing corneal imaging. Capturing the environment reflected on the human eye, it enables gaze approximation on various surfaces (e.g., displays), solely based on natural feature tracking. In the corresponding laboratory experiment, we found *EyeMirror* (M = 4.03°) to be less accurate than a state-of-the-art head-mounted Pupil Labs eye tracker (M = 2.41°), but more accurate than using head

orientation only (M = 4.66°). With *hEYEbrid* we presented the second approach for a calibration-free gaze estimation method. It was a significant improvement over *EyeMirror* in terms of accuracy (M = 2.22°), as shown by the user study. In summary, we gave answers to the remaining open questions, resulting in the following findings:

- Both presented approaches do not require a user-dependent calibration procedure. Based on corneal imaging, they utilize an alternative method for eye and gaze tracking. *EyeMirror* is able to do rough gaze estimation and does not rely on any calibration at all; however, *hEYEbrid* enables a more accurate gaze estimation (competitive with a monocular Pupil Labs device), while requiring a one-time camera calibration.

- Utilizing corneal imaging, we implemented wearable solutions to accomplish the task of gaze estimation in a spontaneous and unobtrusive fashion. Both systems demonstrate the benefit of head-mounted devices, as they can be used on any object at any time and any place, in contrast to remote technologies. We remind the reader that this is only possible due to the fact that our systems are user-calibration-free.

- Finally, we transferred *hEYEbrid* into a mobile and wearable system which is able to react in real time. People can simply use the system with their mobile phone. The additional integration into an existing eye tracking platform makes the approach ready to be used by others for research and to create gaze-based interfaces.

Furthermore, we have discussed possible applications of the two developed systems. We demonstrated the usage of corneal imaging-based head-mounted eye tracking in a live logging scenario. We concluded that information about the environment can be extracted from corneal images applying established computer vision methods for face tracking and object detection. Besides that, we finally demonstrated the integration of well-known existing metrics for eye tracking data analysis. In order to address the lack of options, we proposed a way to compute fixations on data recorded with our specific corneal image-based head-mounted device. Moreover, our fixation extraction algorithm efficiently compresses the recorded eye tracking data, to ease its analysis. Consequently, *hEYEbrid* is ready for use in research studies and interactive scenarios in ubiquitous settings (cf. *Ubiquitous Computing for Eye Tracking Continuum* in Figure 6.1 complexity (K x L) : M : N).

Chapter 7

Conclusion

In this last chapter we are going to summarize the key contributions of this thesis. We will also discuss possible directions for future work that result from our work but go beyond the scope of this thesis. In the end, we will close this thesis with final remarks.

7.1 Major Contributions

The work presented in this thesis has focused on the consideration of eye tracking from the perspective of human-computer interaction, in particular the usage of eye trackers in ubiquitous scenarios. Making eye tracking devices wearable and mobile as well as ready for usage in a spontaneous and simplified fashion is challenging due to the practical issues caused by the necessary calibration.

In the first chapter (cf. Section 1.1), we defined and introduced our *Ubiquitous Computing for Eye Tracking Continuum* (see Figure 1.1), inspired by the vision of Mark Weiser [201] and Robert J. K. Jacob [71]. Along this three-dimensional continuum, we highlighted the challenges of eye and gaze tracking along the rising complexity of the number of devices, locations and users from a *1 : 1 : 1* to a *(K x L) : M : N* relation (K = number of digital objects, L = number of physical objects, M = number of locations, N = number of users): The simplest scenario (1 : 1 : 1) is constituted by a desktop setting, in which a single user interacts with one digital device (e.g., a single display) at one location. Usually in such settings, remote eye trackers can be used to enable gaze-based interaction and interfaces (cf. Section 3.1.1). Remember that the complexity can be increased in three different directions: (1) A single user can interact with several digital devices (e.g., display and mobile phone) and physical

objects at the same location ((K x L) : 1 : 1). (2) One person can interact with one display at different locations (e.g., information screens at different urban places, 1 : M : 1). (3) Multiple users can interact with the same display at the same location (1 : 1 : N). The combination of all three dimensions results in the most complex variant, having multiple persons who interact with physical and digital objects distributed across multiple locations ((K x L) : M : N).

Throughout this thesis, we developed and implemented various approaches that can be applied to realize gaze estimation for the scenarios, as highlighted in Figure 7.1.

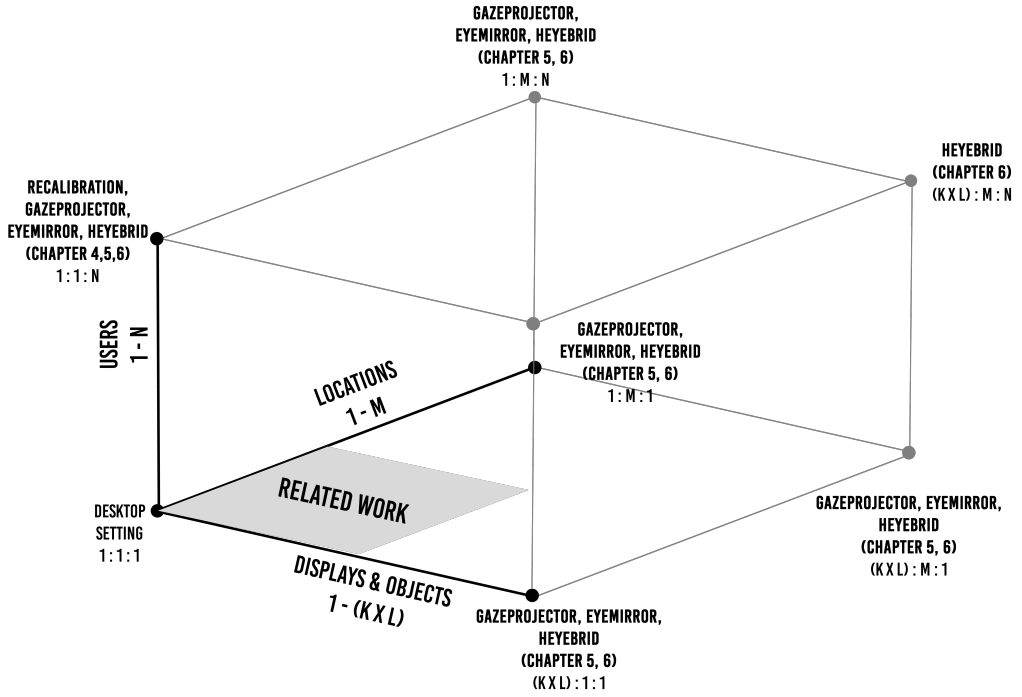


Figure 7.1: *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space ((Displays x Objects) : Locations : Users) highlighting the complexity of scenarios from a simple desktop setting with one display at one location for a single user (1 : 1 : 1) to a pervasive/ubiquitous setting with multiple digital and physical objects distributed at many locations for multiple users (K x L : M : N). The developed approaches are classified within the continuum which increases along the three axes.

We demonstrated the practical problems of head-mounted eye tracking systems via an interactive multi-user application in Section 1.5. The Collaborative Newspaper supports people in reading text on a public display by an adaptive scrolling that is based on their individual read speed and belongs to the complexity level (1 : M : N), i.e. one screen, with multiple locations and users. We use the Collaborative Newspaper system to draw attention to the practical issues of (supervised) user-dependent

calibration, the resultant calibration drift, position and orientation invariance and the parallax error. All these problems (together or separately) make head-mounted eye trackers impractical for ubiquitous and pervasive scenarios.

Starting in Chapter 4, we presented the first approach to tackle the derived problems. The comprehensive study investigated the long-term gaze estimation accuracy of using head-mounted eye trackers in a single display scenario, while simulating different sources of calibration drift. Base on these findings, we developed and tested different strategies to counteract the problem of calibration drift while reducing the time spent for the re-calibration procedure to recover the initial accuracy. Using the developed method for efficient recalibration, eye tracking scenarios of the complexity level $1:1:N$ (multi-user scenarios in front of one display) are possible (see Figure 7.1).

In Chapter 5, we investigated the problem of invariance, i.e. the need to calibrate a head-mounted eye tracker when the user changed their orientation and / or location with respect to the calibration plane (e.g., the display the device was calibrated on). With *GazeProjector*, we developed a system for accurate gaze estimation across multiple displays, independent of which display the eye tracker was calibrated on. In addition, we showed the general applicability of the method in an in-the-wild setting. The system can be applied at several points within the *Ubiquitous Computing for Eye Tracking Continuum*, as shown in Figure 7.1.

In Chapter 6, we presented *EyeMirror* and *hEYEbrid*, two alternative approaches that enable calibration-free gaze estimation based on corneal imaging. In particular, *hEYEbrid* enables eye tracking and gaze estimation in ubiquitous scenarios (see Figure 7.1). With *hEYEbrid*, we presented a ready-to-use wearable and mobile system that is usable with any newer Android-based mobile phone and integrated into the existing open-source eye tracking platform Pupil Labs[1].

The contributions of this work were made by the development and implementation of novel approaches. In particular, we investigated the theoretical methods, and implemented a prototype accordingly, to solve the shortcomings of current state-of-the-art head-mounted eye tracking systems. Further, each prototype was evaluated by conducting a user study, in which it was compared to existing well-established technologies. The work presented in this thesis has made contributions to the areas

[1] https://pupil-labs.com/

of *Efficient Long-Term Usage* of head-mounted eye trackers (E), as well as *Location, Orientation & Target Independent* (L) and *Mobile & Accurate Calibration-Free* eye tracking (M):

(E) Three time-efficient re-calibration strategies counteracting the calibration drift

(E) A large study showing that the order of the calibration points is more important than the used strategy

(E) A guide for the optimal re-calibration strategy

(L) A system for accurate and seamless gaze estimation across multiple displays

(L) A study to demonstrate the advantages against state-of-the-art approaches on a single display

(L) A second study in a multi-display environment to further support the benefits of the approach

(L) A system to effectively match gaze to the environment in an in-the-wild setting

(M) A system for calibration-free gaze approximation

(M) A study on the performance of the approach

(M) A system for calibration-free competitive gaze estimation

(M) A study in a semi-mobile scenario to support the potential of the approach

(M) A fully wearable and mobile system integrating the well-established eye tracking metrics

In addition to these contributions, which individually can be used to tackle aspects of the calibration issues and improve the design of future gaze-based interactive systems, we also contribute the latest versions of the calibration-free eye tracking system as open source[2]. Resources of other systems can be accessed upon request. This contribution will allow others to use our methods for further investigations and continue the development in the open-source community. By answering all our research questions and thus the fundamental research question, we have shown the possibility of and the approach for making eye trackers a ubiquitous computing device.

[2]https://github.com/landerc

7.2 Future Work

The work presented in this thesis, provides new opportunities and leave open challenges that can be addressed in future work. We identified the extension of the conducted studies, improve the developed and implemented systems and build an adaptive solution (in the terms of one-fits-all) as possible directions that can be followed.

Conducting further Evaluations

In the future, we want to take our approach one step further and do an evaluation of our re-calibration approach in a real-world scenario (e.g., in final industrial assembly). We tested *hEYEbrid* in a laboratory environment, in which we simulated a mobile setting where users could freely move around, to gain insights into its performance compared to existing techniques. However, we want to take our approach one step further. One obvious step is to take it to the real world, and evaluate its performance with multiple users simultaneously in scenarios where real-time gaze estimation is required.

The next step is to evaluate the system in an in-the-wild study. We plan to equip multiple persons with the device and ask them to use it during the day while doing several activities indoors and outdoors. We thereby want to investigate the long-term usage as well as the influence of environmental factors (e.g., lighting). In addition, we would like to test prototypes with cameras having different properties to resolve the issue of defocused and unsharp images. Finally, having a system that can be used in fully unconstrained settings for long-term data recording will bring further insights into people's gaze behavior, and create novel interactions that could benefit from such a device.

To further investigate *hEYEbrid* it may be beneficial to include a 3D eye model for 3D pose estimation of the eye. Then we could exactly compare our approach against the existing methods of corneal imaging-based gaze estimation. As we already highlighted, the existing approaches have only been evaluated in a constrained desktop scenario. To make our results more comparable, the latest approaches should also be evaluated in a mobile scenario against *hEYEbrid*.

Enhancement of Systems

The methods for re-calibration presented in Chapter 4 merit further investigation. We think of subsequently optimize the presented methods. The re-calibration algorithm could be tested using other study setups, such as a multi-display setup or a more mobile setting. Further, the integration of the findings in the calibration-free approaches to further increase their accuracy could be explored. We imagine that the mapping from the infrared to the corneal image plane could benefit from this approach.

The work presented in Chapter 6, Section 6.5 lays the foundation for further improvements for future use of the *hEYEbrid* system. First of all, the extensions we developed enable only post-hoc fixation computation. The next step is to realize a real-time version of the presented approach. This would make it possible to directly record compressed data, i.e., only the extracted fixations. In addition, our proposed method will have to be evaluated with a large data set. For this, we envision a comparison of the proposed fixation computation against a state-of-the-art eye tracker. The number and duration of fixations can be verified in this way.

To further improve the gaze estimation of *hEYEbrid*, another possible direction is to look into smooth pursuits. Remember that this concept was used as an alternative possibility to calibrate remote systems (cf. Section 3.2). Investigating the occurrence of smooth pursuit movements in real environments (e.g., following a driving car or a flying bird) is needed to assess the method's applicability. In theory, these events could be used to update the relation between both eye cameras of the device while actually hiding the process of calibration to the user.

Another option is to reduce the system's complexity by using a one-camera approach. The idea is to utilize the already-recorded data of the evaluations done with *hEYEbrid*. This would include labeled information about the position of the pupil in the corneal images, which could be used to train a neural network in order to extract the pupil and its center without requiring the infrared images.

Finally, the developed approaches may also be used to investigate novel systems based on remote setups. In particular, remote approaches are not applicable for ubiquitous scenarios, as discussed in Chapters 1 and 3, they could profit from our developed approaches. For example, using multiple steerable cameras in combination with corneal

imaging would enable calibration-free gaze estimation in (1 : M : N) scenarios (one display, multiple locations users).

Ubiquitous Eye Tracking

As a first step towards this vision, we built a mobile and wearable system which is able to react in real time. This system, which consists of a head-mounted device (with the two eye cameras) and an Android phone, is usable in different application settings. Researchers can profit from this device, as they can use it for eye-tracking experiments in unconstrained environments. Besides, the device can be used to build gaze-based interfaces in pervasive settings. We developed two example applications to showcase the potential of *hEYEbrid*. But, as discussed in the section as limitations above, there is lot of room for future work to resolve current issues and improve the approach.

To use our final prototype *hEYEbrid* for concrete eye tracking experiments, it is necessary to include further metrics for data analysis. Besides fixations, saccades are one of the most important events in eye tracking data (cf. Chapter 2). Consequently, the integration of other eye tracking data events into corneal imaging data analysis has to be further investigated and implemented. If we can reach a state in which this kind of eye tracking data can be analyzed with the same metrics as state-of-the-art head-mounted eye trackers, corneal imaging will be applicable in research studies.

Through the integration of *hEYEbrid* into the Pupil Labs universe and the open-source release, we expect our approach to be used in the future to create an open-source solution that fits the needs of any use-case.

7.3 Closing Remarks

This thesis has investigated the problems of the current generation of eye trackers in the context of ubiquitous computing. In the case of personal computers, the shift to this paradigm has already been reached (from a single computer to multiple devices), whereas the development of ubiquitous eye glasses (as envisioned by Mark Weiser in [201]) is yet unfinished (cf. *Ubiquitous Computing for Eye Tracking Continuum*, Figure 1.1). In the course of this thesis we concentrated on head-mounted state-of-the-art devices, their problems, and the question of how to solve these problems (see Chapter 1). We cannot foresee the development of these computing devices and the application of eye tracking for interaction in the future. Latest industrial developments (e.g.,

Apple bought SMI, Google bought Eyefluence) indicate an increase in the importance of eye tracking (especially with the rise of VR devices and applications). With the novel methods we developed and implemented, and the lessons learned to counteract existing issues, we are doing our part in order to pursue the development of a ubiquitous eye tracking device. To further substantiate our work, we published the relevant part of this work as open source. With this we want to inspire other researchers and technically affine persons to take up our results and enhance the use of eye and gaze tracking for everyone.

List of Figures

1.1 *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space highlighting the complexity of the scenario ((Displays x Objects) : Locations : Users), which increases from a simple desktop setting with one display at one location for a single user (1 : 1 : 1) to a pervasive/ubiquitous setting with multiple digital and physical objects distributed at many locations for multiple users (K x L : M : N). . . . 3
1.2 Apparatus from Dodge and Cline to record horizontal eye movements [33]. . . . 5
1.3 Application dimensions of eye tracking, adapted from Duchowski [37]: real-time vs. post-hoc processing and commercial vs. research applications. . . . 7
1.4 A head-mounted eye tracker is calibrated at a fixed position resulting in a fixed calibration plane. Looking at objects at a different distance, the actual and computed gaze point are different. . . . 11
1.5 Concept of the Collaborative Newspaper: Multiple persons are reading several texts on a public display. . . . 13
1.6 A person is interested in reading a text on the public display. After approaching the screen, she first has to calibrate the head-mounted eye tracking device in order to use the Collaborative Newspaper. . . . 14
1.7 The user is moving in front of the display, which requires a re-calibration in general. . . . 15
1.8 User interface of the Collaborative Newspaper, displaying up to 7 articles. For display tracking, visual AR markers are placed around the content. . . . 16
1.9 A second person is joining the interaction and wants to read a text presented on the public display. For this, a calibration has to be done beforehand. . . . 17

1.10 Thesis structure: Four main chapters addressing the stated problems and providing answers to the research question. 19

2.1 Frontal photos of the eye together with a cross section (adapted from [118]). All important parts are labeled. 22
2.2 Detailed cross section of the eye's retina, showing its structure (adapted from [118]). 23
2.3 a) Frontal illustrations fo the pupil and the states of dilation and reduction issued by the pupillary muscles[3]. b) Structure of the eye muscles necessary for all types of motion[4]. 24
2.4 Cross section of the human eyeball including the three main axes [98]. 25
2.5 Illustration of all major types of eye movements: a) fixation to focus on a object, b) smooth pursuit to constantly follow a moving object, c) vergence to perceive different depths and d) vestibulo-ocular reflex to reposition the eyes when the head is moved. 27
2.6 Examples of different types of head-mounted eye tracking approaches: (a) scleral search coil method [66], (b) electro-oculography[5], (c) head-mounted Dikablis eye tracker[6]. 28
2.7 a) Face with areas of interest; b) shapes around the the eye; c) prominent features; d) ellipse with its definition 29
2.8 a) dark pupil; b) bright pupil; c) Else output based on edge detection and curvature computation. 31
2.9 Infrared light reflected on the human eye results in Purkinje reflections: P1 on the corner, P2 from the back of the cornea, P3 from the lens, P4 from the back of the lens. 34

3.1 The two categories of eye tracking approaches: (a) remote eye tracker attached onto a computer monitor[7], (b) Head-mounted eye tracking glasses[8]. 38
3.2 Example of an early head-mounted eye tracking device (Ergoneers Dikablis 2[9]) using a chin rest to stabilize the user's head. 41
3.3 Eye tracking application continuum, adapted from Majaranta [110]. . 45
3.4 Examples to address the region of uncertainty problem: a) dynamic menu [198], b) radial menu layout [63], c) magnifying lens [103] and d) fisheye lens to browse an image gallery [182] 47

3.5 Examples of gaze gestures to prevent Midas Touch: a) eye strokes to 'draw' a command with the eyes [36], b) simple eye strokes [64] and c) complex gestures in a VR application [31] 48
3.6 Examples of multi modal gaze input: a) gaze based pointing and selection via mobile phone [180], c) using gaze together with feed control [85], b) gaze plus hand gestures [25] and d) gaze plus touch and pen input [148] 49
3.7 Example application of eyeCONTACT sensor to sense a person's attention [169]. 51
3.8 Calibration of a head-mounted eye tracking device the Pupil Labs software[10]. 55
3.9 Resulting view on the calibration plane (i.e., the screen) after the user has changed their distance or orientation. 59
3.10 Step-wise reduction of the calibration drift. 60
3.11 Selection of required fixation locations used to correct the calibration drift. 61
3.12 Two examples for calibration-free gaze interaction: a) Pursuits, utilizing the pursuit movements of the eye [117], and b) SideWays, realizing gaze input via horizontal eye movements [214] 64
3.13 Example image of a corneal reflection of a computer monitor. 65
3.14 *Ubiquitous Computing for Eye Tracking Continuum*: Classification of related work into a 2-dimensional area with varying locations and displays/objects. 69

4.1 *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space highlighting the complexity (1 : 1 : N) that is addressed by this chapter. 71
4.2 Illustration of eye and world view with sampled data of the eye position and the on-screen marker position. 75
4.3 Experiment Setup: A 23-inch computer monitor placed on a table at a height of 120 cm. The participant was standing at a distance of 50 or 100 cm in front of the display, wearing a head-mounted eye tracking device. 78
4.4 Difference in gaze estimation error w.r.t. the full calibration mapping M_{Second}. The results are based on the data recorded during the second test run. 81

4.5 Difference in the gaze estimation error w.r.t. M_{Second}. The results are presented per experiment block and number of re-calibration points based on the data recorded during the second test run. 82
4.6 The nine points used for recalibration, labeled with their identifiers (0 to 8). 83
4.7 Gaze estimations based on different mapping functions of the re-calibration combinations for the data of the second test run. Numbers in subscript indicate the calibration points that were used in this re-calibration. . . 85

5.1 *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space highlighting the complexity ((K x L) : M : 1) that is addressed by this chapter. 95
5.2 GazeProjector enables seamless gaze-based interaction with multiple displays from arbitrary locations and orientations, such as wall-sized displays (1), horizontal screens (2), and handheld devices (3) without active recalibration. 98
5.3 Examples of tracking a person's spatial relationship to a display: (a) Proximity Toolkit [115] based on a instrumentation of the environment, (b) TouchProjector [4] using a mobile phone's camera to accomplish this task. 100
5.4 Concept of *GazeProjector*: (1) Eye tracker calibration and tracking the user's current field of view and (2) the spatial relationship between eye tracker and display in order to (3) map the user's gaze accordingly . . 101
5.5 2D homography maps a point x from image plane P to point x' in plane P' (adapted from [27]). 103
5.6 Gaze estimation on a display using the eye tracker's (a) eye and (b) scene camera and the display's content (d): the approach combines the eye tracker calibration (c) with the determined projective transformation between the scene camera's image plane and the display (f), based on image feature tracking and matching (e), to obtain the location on the display (g). 104
5.7 Architecture of *GazeProjector* with three components: Tracker, Engine and Trackable 105
5.8 First example application: after a user selects an event from a public calendar (a), that information is shown on the personal mobile device, where gaze estimation also works (b). 112

5.9 Second example application: when users select from a time table of departing flights (1), the corresponding information is shown on their mobile device (2). 112
5.10 Experimental setup showing all locations (L1-L6) and orientations relative to the display, the nine different positions (T1-T9) of on-screen visual targets, and the background image used for feature tracking and marker tracking respectively. 114
5.11 Mean gaze estimation error for every location for MT-near, MT-far, HO-near, HO-far, GP-near and GP-far. Error bars indicate ± standard error of the mean. 117
5.12 Visualization of the mean gaze error (ellipses) for the three modes MT, HO and GP and all calibrations averaged over all targets. Black circles visualize the mean gaze points. 118
5.13 Our setup showing the three screens (including their background images for feature tracking) used during the experiment, as well as the placement of the nine targets per screen. Note that all participants were free to choose a location within the blue area throughout the experiment. 120
5.14 Mean gaze estimation error of each mode for each display. Error bars indicate ± standard error of the mean. 123
5.15 Visualization of the mean gaze error (ellipses) for all different modes, MT and GP, and all calibrations averaged over all targets over all screens. Additionally, black circles visualize the mean gaze estimations. 124
5.16 Concept of the system using *GazeProjector* for gaze estimation on objects in the environment: The head-mounted eye tracker (a,b) enables gaze estiamtion in the user's field of view obtained through the scene camera (c). With the GPS location, possible candidates for the environment the user is facing are estimated (d). The orientation sensor (compass) reduces the environment images to the correct one according to the user's head orientation (e). *GazeProjector* matches the user's gazepoint from the scene camera image to the selected environment (f). 131
5.17 Architecture of the system using *GazeProjector* for gaze estimation on objects in the environment. 133
5.18 Head-mounted device consisting of Pupil Labs eye tracker and a wireless inertial measurement unit to detect the user's head orientation; iPhone SE used for GPS location. 134

5.19 Playback and analysis of the recorded data: (a) The standard Pupil Player is able to play back the data recorded by the eye tracker. (b) The developed custom plugin is able to correlate GPS location, head orientation and eye tracking data. For manual analysis, the path on a Google Map is shown (1), as well as the Street View image (2). The user is able to set some parameters for the visualization (3). (c) Using feature tracking, the plugin determines the transformation between the scene camera's image and the Google Street View image. 138
5.20 The route that was walked by the participants. Each person started through the park, beginning and ending in front of the railway station at the red marker. 140
5.21 Left: Mean eye movement in degrees, horizontal vs. vertical direction. Right: Mean head movements in degrees, horizontal vs. vertical direction. 141

6.1 *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space highlighting the complexity ((K x L) : M : N) that is addressed by this chapter. 146
6.2 Basic idea of using the corneal images (i.e. the environment reflected on the human eye) for gaze estimation in the current scene. 147
6.3 Corneal images, showing (1) a computer monitor, (2) faces, (3) a poster and (4) an iPhone6 display. 149
6.4 Limbus detection is done via ellipse fitting (5, marked by the green rectangle and the orange ellipse) after a polar transformation (2) and radial derivation (3). The eye center is extracted using image gradients (4). The final region of interest (6) contains the limbus and the eye center location (yellow circle). 151
6.5 *EyeMirror*'s two approaches for gaze estimation on surfaces, here displays: (1) Gaze is approximated by transforming the extracted eye center onto the display using a homography matrix. (2) Gaze is approximated by clustering the image feature pairs of the display's content and the corneal image. 153
6.6 Building steps: (1) Extraction of camera board; (2) removing glue around the lens to adjust focus; (3) camera is built into a custom rotatable and movable enclosure and (4) mounted on a glasses frame, connected to a (5) RaspberryPi. On the right, the final wearable and fully functional prototype is shown. 155

6.7 Calibration with chessboard pattern (1) to undistort the corneal image (2). Computation of a homography may be more exact (3). 160
6.8 Study setup: (a) A large projected screen and the Kinect sensor v2. In addition, all locations (*near-left, near-right, far-left, far-center and far-right*) and targets (T1-T9) are shown. (b) The prototype combining the Pupil Labs eye tracker with EyeMirror. 161
6.9 Mean gaze estimation error for every location and mode. Error bars indicate ± standard error of the mean. 164
6.10 The different display contents used in the experiment: C1 with 9262, C2 with 6988, C3 with 3314, C4 with 9476, C5 with 4239 and C6 with 934 features. The right picture shows the setup for the second experiment. 166
6.11 Mean gaze estimation error for the desktop content images C1–C5, combined with the number of features. 167
6.12 Face detection and its 3D orientation (2) using the pre processed camera frame (1). 173
6.13 Display Detection and Tracking (1)–(3) in images reflected on a person's eye; detected displays are marked with a green rectangle. 173
6.14 Gaze estimation on posters (1,2) using the feature tracking algorithm. Gaze estimation on a bookshelf (6) and a book (7); on other real-world objects (3,4,5). 174
6.15 Gaze estimation on car hifi system (1), and detection of retailer logos (2) and road sign (3). 174
6.16 The developed *hEYEbrid* concept used to build a mobile and wearable device for spontaneous competitive gaze estimation at any place and any time. The head-mounted device is connected to a Nexus 6P that processes both camera streams in real time. 176
6.17 Basic processing pipeline of *hEYEbrid*, which combines infrared eye and corneal images. The pupil is tracked in the IR image and mapped onto the corneal image to finally crop the reflection within the pupil. The mapped pupil center coincides with the gaze point. 179
6.18 Estimating a mapping between the two eye camera image planes based on epipolar geometry. In *hEYEbrid*, the problem is reduced from 3D to 2D and solved via a planar homography mapping matrix. 180
6.19 The extension *hEYEbrid-3C* connects the result of *hEYEbrid*'s processing pipeline with the scene image to project the pupil center into the scene image plane. 182

6.20 Building steps to integrate *hEYEbrid* into the Pupil Labs head-mounted device. The corneal camera is affixed to a custom mount together with the device's IR camera. The mount is movable and rotatable. 183
6.21 Integration of *hEYEbrid* and *hEYEbrid-3C* into the Pupil Labs framework. The Capture software is able to record data, whereas the Player is used for analysis and gaze computation in the different approaches. 184
6.22 Setting up the eye cameras by computing the transformation between the infrared and corneal image planes. Image features are extracted and matched to compute a homography matrix, used to map a point from one image into the other. 185
6.23 The study protocol showing the procedure of the experiment as well as the gaze targets used for the gaze pointing task. 187
6.24 The study setup showing the distribution of the tangible objects (attached with AR markers) in the room and a participant gazing at the book. In addition, the projection including all targets is shown. 188
6.25 Mean gaze estimation error for each mode for each of the three gaze pointing tasks. Each column shows the mean gaze estimation for each mode after the procedure done before the gaze pointing task, i.e. after the initial *Pupil* calibration, the *Pupil* re-calibration and taking off the device and putting it back on. 191
6.26 Example applications: 1. Gaze estimation on a display, based on natural feature tracking. 2. Object detection in the corneal image using a CNN. 196
6.27 Architecture of the mobile system usable with *hEYEbrid*, *hEYEbrid-3C* and a Pupil Labs eye tracker. The hardware is made up of a head-mounted device containing two or three cameras, depending on the mode. The software side contains three main parts. 198
6.28 Progress made within the development of a device for mobile calibration-free gaze estimation. 203
6.29 (a) The data format used to extract fixation data; (b) the stitching approach used to cluster image data 207
6.30 Panorama stitching to cluster multiple fixations. The resulting image contains all three fixations mapped onto the stitched image. 208

7.1 *Ubiquitous Computing for Eye Tracking Continuum*: 3-dimensional space ((Displays x Objects) : Locations : Users) highlighting the complexity of scenarios from a simple desktop setting with one display at one location for a single user (1 : 1 : 1) to a pervasive/ubiquitous setting with multiple digital and physical objects distributed at many locations for multiple users (K x L : M : N). The developed approaches are classified within the continuum which increases along the three axes.212

List of Tables

3.1 Comparison between remote and head-mounted eye tracking concerning five major attributes. 44

3.2 Comparing different aspects of the presented re-calibration approaches. None of the existing methods provides a general solution, (• fulfilled, ◦ not fulfilled). 61

3.3 An overview of existing gaze estimation approaches based on corneal imaging. This shows the main differences between remote and head-mounted techniques, (• fulfilled, ◦ not fulfilled). 66

4.1 Design of the experiment blocks and test runs (• device was took off, ◦ device was not took off). 79

4.2 Accuracy of mapping functions M_{First} and M_{Second} per experiment block in degrees. Evaluated for the second test data set. 82

4.3 Results of Welch's unequal variances t-test of the covered polygon area between $NS_{|R|}$ and $S_{|R|}$. 85

4.4 Point combinations yielding the best results (mean and standard deviation of the gaze estimation error in degree), chosen based on the two rules defined above. In addition, the gain in accuracy and the time in seconds needed for re-calibration are shown. 88

4.5 Mean gaze estimation error in degrees for each adaptation method when performed with the twelve superior point distributions. 89

5.1 Mean, standard deviation and variance for x,y-coordinates of normalized gaze positions in the participants' field of view. 119

6.1 Means and standard deviations of the overall, horizontal and vertical gaze estimation for all modes and locations. 163

6.2 Means and standard deviations for the computed gaze estimation errors for all modes on the two target groups. 190

Bibliography

[1] *PETMEI '18: Proceedings of the 7th Workshop on Pervasive Eye Tracking and Mobile Eye-Based Interaction*. ACM, 2018.

[2] J. L. Austin. *Sense and Sensibilia*. Oxford University Press USA, 1962.

[3] M. Backes, M. Dürmuth, and D. Unruh. Compromising reflections or how to read lcd monitors around the corner. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP '08, pages 158–169. IEEE Computer Society, 2008.

[4] R. Ballagas, J. Borchers, M. Rohs, and J. G. Sheridan. The smart phone: A ubiquitous input device. *IEEE Pervasive Computing*, 5(1):70–77, 2006.

[5] T. Baltrusaitis, P. Robinson, and L.-P. Morency. Openface: An open source facial behavior analysis toolkit. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–10, 2016.

[6] L. Bartram, C. Ware, and T. Calvert. Moticons: detection, distraction and task. *International Journal of Human-Computer Studies*, 58(5):515 – 545, 2003. Notification User Interfaces.

[7] M. Barz, F. Daiber, and A. Bulling. Prediction of gaze estimation error for error-aware gaze-based interfaces. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '16, pages 275–278. ACM, 2016.

[8] R. Bates, M. Donegan, H. O. Istance, J. P. Hansen, and K. J. Räihä. Introducing cogain - communication by gaze interaction. In *Designing Accessible Technology*, pages 77–84. Springer, 2006.

[9] D. Baur, S. Boring, and S. Feiner. Virtual projection: Exploring optical projection as a metaphor for multi-device interaction. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '12, pages 1693–1702. ACM, 2012.

[10] T. R. Beelders and P. J. Blignaut. Gaze and speech: Pointing device and text entry modality. In M. Horsley, M. Eliot, B. A. Knight, and R. Reilly, editors, *Current Trends in Eye Tracking Research*, pages 51–75. Springer, 2014.

[11] J. R. Bergstrom and A. J. Schall. *Eye Tracking in User Experience Design*. Morgan Kaufmann, 2014.

[12] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: The see-through interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 73–80. ACM, 1993.

[13] P. Blignaut, K. Holmqvist, M. Nyström, and R. Dewhurst. Improving the accuracy of video-based eye tracking in real time through post-calibration regression. In M. Horsley, M. Eliot, B. A. Knight, and R. Reilly, editors, *Current Trends in Eye Tracking Research*, pages 77–100. Springer, 2014.

[14] F. Bockes, L. Edel, M. Ferstl, and A. Schmid. Collaborative landmark mining with a gamification approach. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, MUM '15, pages 364–367. ACM, 2015.

[15] R. A. Bolt. Gaze-orchestrated dynamic windows. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '81, pages 109–119. ACM, 1981.

[16] S. Boring, D. Baur, A. Butz, S. Gustafson, and P. Baudisch. Touch projector: Mobile interaction through video. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '10, pages 2287–2296. ACM, 2010.

[17] J. Breuninger, C. Lange, and K. Bengler. Implementing gaze control for peripheral devices. In *Proceedings of the 1st International Workshop on Pervasive Eye Tracking Mobile Eye-based Interaction*, PETMEI '11, pages 3–8. ACM, 2011.

[18] A. Bulling. Pervasive attentive user interfaces. *Computer*, 49(1):94–98, 2016.

[19] A. Bulling and H. Gellersen. Toward mobile eye-based human-computer interaction. *IEEE Pervasive Computing*, 9(4):8–12, 2010.

[20] A. Bulling, J. A. Ward, H. Gellersen, and G. Troester. Eye movement analysis for activity recognition using electrooculography. 33(4):741 – 753, 2011-04. Manuscript received 26 November 2009, Accepted 26 February 2010, Published online 30 March 2010.

[21] A. Bulling, J. A. Ward, H. Gellersen, and G. Tröster. Robust recognition of reading activity in transit using wearable electrooculography. In *International Conference on Pervasive Computing*, pages 19–37. Springer, 2008.

[22] J. J. Cerrolaza, A. Villanueva, and R. Cabeza. Study of polynomial mapping functions in video-oculography eye trackers. *ACM Trans. Comput.-Hum. Interact.*, 19(2):10:1–10:25, 2012.

[23] J. J. Cerrolaza, A. Villanueva, M. Villanueva, and R. Cabeza. Error characterization and compensation in eye tracking systems. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 205–208. ACM, 2012.

[24] B. S. Chaparro and C. Russell. Hotspots and hyperlinks: Using eye-tracking to supplement usability testing, 2005.

[25] I. Chatterjee, R. Xiao, and C. Harrison. Gaze+gesture: Expressive, precise and targeted free-space interactions. In *Proceedings of the International Conference on Multimodal Interaction*, ICMI '15, pages 131–138. ACM, 2015.

[26] J. Chen and Q. Ji. Probabilistic gaze estimation without active personal calibration. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 609–616. IEEE Computer Society, 2011.

[27] G. B. Christiano Gava. Lecutre on 2d projective transformations.

[28] T. N. Cornsweet and H. D. Crane. Accurate two-dimensional eye tracker using first and fourth purkinje images. *Journal of the Optical Society of America*, 63(8):921–928, 1973.

[29] F. L. Coutinho and C. H. Morimoto. Improving head movement tolerance of cross-ratio based eye trackers. *International Journal of Computer Vision*, 101(3):459–481, 2013.

[30] E. S. Dalmaijer, S. Mathôt, and S. Van der Stigchel. Pygaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. *Behavior Research Methods*, 46(4):913–921, 2014.

[31] W. Delamare, T. Han, and P. Irani. Designing a gaze gesture guiding system. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, pages 26:1–26:13. ACM, 2017.

[32] A. H. Dictionary. *The American Heritage Medical Dictionary*. Houghton Mifflin Company, 2007.

[33] R. Dodge and T. S. Cline. The angle velocity of eye movements. *Psychological Review*, 8(2):145–157, 1901.

[34] M. Donegan, J. D. Morris, F. Corno, I. Signorile, A. Chió, V. Pasian, A. Vignola, M. Buchholz, and E. Holmqvist. Understanding users and their needs. *Universal Access in the Information Society*, 8(4):259, 2009.

[35] T. D'Orazio, M. Leo, G. Cicirelli, and A. Distante. An algorithm for real time eye detection in face images. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 3 of *ICPR' 04*, pages 278–281. IEEE, 2004.

[36] H. Drewes and A. Schmidt. Interacting with the computer using gaze gestures. In *Proceedings of the 11th International Conference on Human-computer Interaction*, INTERACT'07, pages 475–488. Springer, 2007.

[37] A. T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer, 2007.

[38] A. T. Duchowski, N. Cournia, and H. Murphy. Gaze-contingent displays: A review. *CyberPsychology & Behavior*, 7(6):621–634, 2004.

[39] M. Duckham, S. Winter, and M. Robinson. Including landmarks in routing instructions. *J. Locat. Based Serv.*, 4(1):28–52, 2010.

[40] M. L. Dybdal, J. S. Agustin, and J. P. Hansen. Gaze input for mobile devices by dwell and gestures. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 225–228. ACM, 2012.

[41] C. Ehmke and S. Wilson. Identifying web usability problems from eye-tracking data. In *Proceedings of the 21st British Conference on People and Computers*, BCS-HCI '07, pages 119–128. British Computer Society, 2007.

[42] D. R. Flatla, C. Gutwin, L. E. Nacke, S. Bateman, and R. L. Mandryk. Calibration games: Making calibration tasks enjoyable by adding motivating game elements. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 403–412. ACM, 2011.

[43] W. Fuhl, T. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci. Excuse: Robust pupil detection in real-world scenarios. In G. Azzopardi and N. Petkov, editors, *Computer Analysis of Images and Patterns: 16th International Conference, CAIP 2015, Valletta, Malta, September 2-4, 2015 Proceedings, Part I*, pages 39–51. Springer, 2015.

[44] W. Fuhl, T. C. Santini, T. Kübler, and E. Kasneci. Else: Ellipse selection for robust pupil detection in real-world environments. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '16, pages 123–130. ACM, 2016.

[45] W. Fuhl, M. Tonsen, A. Bulling, and E. Kasneci. Pupil detection for head-mounted eye tracking in the wild:an evaluation of the state of the art. *Machine Vision and Applications*, 27(8):1275–1288, 2016.

[46] I. Giannopoulos, P. Kiefer, and M. Raubal. Gazenav: Gaze-based pedestrian navigation. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 337–346. ACM, 2015.

[47] J. H. Goldberg and X. P. Kotval. Computer interface evaluation using eye movements: methods and constructs. *International Journal of Industrial Ergonomics*, 24(6):631 – 645, 1999.

[48] E. D. Guestrin and M. Eizenman. Remote point-of-gaze estimation requiring a single-point calibration for applications with infants. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '08, pages 267–274. ACM, 2008.

[49] D. Guitton and M. Volle. Gaze control in humans: eye-head coordination during orienting movements to targets within and beyond the oculomotor range. *Journal of Neurophysiology*, 58(3):427–459, 1987.

[50] L. E. Hafi, T. Ogasawara, M. Ding, and J. Takamatsu. Gaze tracking using corneal images captured by a single high-sensitivity camera. pages 33–43, 2016.

[51] R. Hammoud. *Passive Eye Monitoring: Algorithms, Applications and Experiments*. Springer, 1st edition, 2008.

[52] 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.*, 32(3):478–500, 2010.

[53] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[54] H. Hartridge and L. C. Thomson. Methods of investigating eye movements. *The British journal of ophthalmology*, 32 9:581–91, 1948.

[55] J. Helgath, S. Provinsky, and T. Schaschek. Landmark mining on a smartwatch using speech recognition. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, MUM '15, pages 379–383. ACM, 2015.

[56] C. Hennessey and J. Fiset. Long range eye tracking: Bringing eye tracking into the living room. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 249–252. ACM, 2012.

[57] L. Herbert, N. Pears, D. Jackson, and P. Olivier. Mobile device and intelligent display interaction via scale-invariant image feature matching. In *Proceedings of the International Conference on Pervasive and Embedded Computing and Communication Systems*, PECCS' 11, 2011.

[58] R. Herpers, M. Michaelis, K.-H. Lichtenauer, and G. Sommer. Edge and keypoint detection in facial regions. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 212–217. IEEE, 1996.

[59] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. van de Weijer. *Eye Tracking: A Comprehensive Guide to Methods and Measures*. Oxford: Oxford University Press, 2011.

[60] S. Hoppe, T. Loetscher, S. Morey, and A. Bulling. Recognition of curiosity using eye movement analysis. In *Proc. UbiComp 2015*, pages 185–188, 2015.

[61] A. J. Hornof and T. Halverson. Cleaning up systematic error in eye-tracking data by using required fixation locations. *Behavior Research Methods, Instruments, & Computers*, 34(4):592–604, 2002.

[62] M. X. Huang, T. C. Kwok, G. Ngai, S. C. Chan, and H. V. Leong. Building a personalized, auto-calibrating eye tracker from user interactions. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '16, pages 5169–5179. ACM, 2016.

[63] A. Huckauf and M. H. Urbina. Gazing with peyes: Towards a universal input for various applications. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '08, pages 51–54. ACM, 2008.

[64] A. Hyrskykari, H. Istance, and S. Vickers. Gaze gestures or dwell-based interaction? In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 229–232. ACM, 2012.

[65] A. Hyrskykari, P. Majaranta, and K.-J. Räihä. Proactive response to eye movements. In *INTERACT*, volume 3, pages 129–136. IOS press Amsterdam, 2003.

[66] T. Imai, K. Sekine, K. Hattori, N. Takeda, I. Koizuka, K. Nakamae, K. Miura, H. Fujioka, and T. Kubo. Comparing the accuracy of video-oculography and the scleral search coil system in human eye movement analysis. *Auris Nasus Larynx*, 32(1):3 – 9, 2005.

[67] Y. Ishiguro, A. Mujibiya, T. Miyaki, and J. Rekimoto. Aided eyes: Eye activity sensing for daily life. In *Proceedings of the 1st Augmented Human International Conference*, AH '10, pages 25:1–25:7. ACM, 2010.

[68] H. Istance and A. Hyrskykari. Gaze-aware systems and attentive applications. *Gaze Interaction and Applications of Eye Tracking*, 58:175–195, 2011.

[69] H. Istance, A. Hyrskykari, L. Immonen, S. Mansikkamaa, and S. Vickers. Designing gaze gestures for gaming: An investigation of performance. In *Proceedings of the Symposium on Eye-Tracking Research and Applications*, ETRA '10, pages 323–330. ACM, 2010.

[70] N. J Wade. Pioneers of eye movement research. *i-Perception*, 1:33–68, 11 2010.

[71] R. J. K. Jacob. What you look at is what you get: Eye movement-based interaction techniques. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '90, pages 11–18. ACM, 1990.

[72] R. J. K. Jacob and K. S. Karn. Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises. *The Mind's eye: Cognitive The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, pages 573–603, 2003.

[73] H. Jarodzka, T. Balslev, K. Holmqvist, K. Scheiter, M. Nystrm, and B. Eika. Learning perceptual skills for medical diagnosis via eye movement modeling examples on patient video cases. The 5th Scandinavian Workshop on Applied Eye-Tracking, SWAET 2010, pages 11–11. Lund University, 2010.

[74] A.-H. Javadi, Z. Hakimi, M. Barati, V. Walsh, and L. Tcheang. Set: A pupil detection method using snusoidal approximation. *Frontiers in Neuroengineering*, 8:4, 2015.

[75] Q. Ji and X. Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5):357–377, 2002.

[76] M. Kandemir and S. Kaski. Learning relevance from natural eye movements in pervasive interfaces. In *Proceedings of the 14th International Conference on Multimodal Interaction*, ICMI '12, pages 85–92. ACM, 2012.

[77] J. Kangas, D. Akkil, J. Rantala, P. Isokoski, P. Majaranta, and R. Raisamo. Gaze gestures and haptic feedback in mobile devices. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '14, pages 435–438. ACM, 2014.

[78] M. Kassner, W. Patera, and A. Bulling. Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Adjunct Proceedings of UbiComp 2014*, UbiComp '14 Adjunct, pages 1151–1160. ACM, 2014.

[79] A. E. Kaufman, A. Bandopadhay, and B. D. Shavi. An eye tracking computer user interface. 1993.

[80] D. H. Kelly. *Visual Science and Engineering: Models and Applications*. Optical Science and Engineering. Taylor & Francis, 1994.

[81] M. Khamis, F. Alt, and A. Bulling. Challenges and design space of gaze-enabled public displays. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, pages 1736–1745. ACM, 2016.

[82] M. Khamis, O. Saltuk, A. Hang, K. Stolz, A. Bulling, and F. Alt. Textpursuits: Using text for pursuits-based interaction and calibration on public displays. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, pages 274–285. ACM, 2016.

[83] P. Kiefer, I. Giannopoulos, D. Kremer, C. Schlieder, and M. Raubal. Starting to get bored: An outdoor eye tracking study of tourists exploring a city panorama. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, pages 315–318. ACM, 2014.

[84] P. Kiefer, I. Giannopoulos, and M. Raubal. Where am i? investigating map matching during self-localization with mobile eye tracking in an urban environment. *Trans. GIS*, 18:660–686, 2014.

[85] K. Klamka, A. Siegel, S. Vogt, F. Göbel, S. Stellmach, and R. Dachselt. Look & pedal: Hands-free navigation in zoomable information spaces through gaze-supported foot input. In *Proceedings of the International Conference on Multimodal Interaction*, ICMI '15, pages 123–130. ACM, 2015.

[86] M. Klauck, Y. Sugano, and A. Bulling. Noticeable or distractive?: A design space for gaze-contingent user interface notifications. In *Proceedings of the Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, pages 1779–1786. ACM, 2017.

[87] C. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. In L. M. Vaina, editor, *Matters of Intelligence: Conceptual Structures in Cognitive Neuroscience*, pages 115–141. Springer, Dordrecht, 1987.

[88] O. V. Komogortsev and J. I. Khan. Eye movement prediction by kalman filter with integrated linear horizontal oculomotor plant mechanical model. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '08, pages 229–236, New York, NY, USA, 2008. ACM.

[89] R. Kothari and J. L. Mitchell. Detection of eye locations in unconstrained visual images. In *Image Processing, 1996. Proceedings., International Conference on*, volume 3, pages 519–522. IEEE, 1996.

[90] M. Land and B. W. Tatler. *Looking and acting: vision and eye movements in natural behaviour*. Oxford University Press, 2009.

[91] C. Lander. Methods for calibration free and multi-user eye tracking. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI '16, pages 899–900. ACM, 2016.

[92] C. Lander, N. Coenen, S. Biewer, and A. Krüger. Kollaboratives text lesen: Adaptive text scroll geschwindigkeit. *Mensch und Computer 2015–Proceedings*, 2015.

[93] C. Lander, S. Gehring, A. Krüger, S. Boring, and A. Bulling. Gazeprojector: Accurate gaze estimation and seamless gaze interaction across multiple displays. In *Proceedings of the 28th Symposium on User Interface Software and Technology*, UIST '15, pages 395–404. ACM, 2015.

[94] C. Lander, S. Gehring, M. Löchtefeld, A. Bulling, and A. Krüger. Eyemirror: Mobile calibration-free gaze approximation using corneal imaging. In *Proceedings of the 16th International Conference on Mobile and Ubiquitous Multimedia*, MUM '17, pages 279–291. ACM, 2017.

[95] C. Lander, F. Kerber, T. Rauber, and A. Krüger. A time-efficient re-calibration algorithm for improved long-term accuracy of head-worn eye trackers. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '16, pages 213–216. ACM, 2016.

[96] C. Lander, F. Kosmalla, F. Wiehr, and S. Gehring. Using corneal imaging for measuring a human's visual attention. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the International Symposium on Wearable Computers*, UbiComp '17, pages 947–952. ACM, 2017.

[97] C. Lander, A. Krüger, and M. Löchtefeld. "the story of life is quicker than the blink of an eye": Using corneal imaging for life logging. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, pages 1686–1695. ACM, 2016.

[98] C. Lander, M. löchtefeld, and A. Krüger. heyebrid: A hybrid approach for mobile calibration-free gaze estimation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4):149:1–149:29, 2018.

[99] C. Lander, M. Speicher, F. Kerber, and A. Krüger. Towards fixation extraction in corneal imaging based eye tracking data. In *Extended Abstracts of the Conference on Human Factors in Computing Systems*, CHI EA '18, pages LBW558:1–LBW558:6. ACM, 2018.

[100] C. Lander, M. Speicher, D. Paradowski, N. Coenen, S. Biewer, and A. Krüger. Collaborative newspaper demo: Exploring an adaptive scrolling algorithm in a multi-user reading scenario. In *Proceedings of the 4th International Symposium on Pervasive Displays*, PerDis '15, pages 271–272. ACM, 2015.

[101] C. Lander, M. Speicher, D. Paradowski, N. Coenen, S. Biewer, and A. Krüger. Collaborative newspaper: Exploring an adaptive scrolling algorithm in a multi-user reading scenario. In *Proceedings of the 4th International Symposium on Pervasive Displays*, PerDis '15, pages 163–169. ACM, 2015.

[102] C. Lander, F. Wiehr, N. Herbig, A. Krüger, and M. Löchtefeld. Inferring landmarks for pedestrian navigation from mobile eye-tracking data and google street view. In *Proceedings of the Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, pages 2721–2729. ACM, 2017.

[103] C. Lankford. Effective eye-gaze input into windows. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '00, pages 23–27. ACM, 2000.

[104] D. Li, D. Winfield, and D. J. Parkhurst. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 79–79. IEEE, 2005.

[105] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–. IEEE Computer Society, 1999.

[106] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[107] I. S. MacKenzie. An eye on input: Research challenges in using the eye for computer input control. In *Proceedings of the Symposium on Eye-Tracking Research and Applications*, ETRA '10, pages 11–12. ACM, 2010.

[108] J. Magee, T. Felzer, and I. S. MacKenzie. Camera mouse + clickeraid: Dwell vs. single-muscle click actuation in mouse-replacement interfaces. In M. Antona and C. Stephanidis, editors, *Universal Access in Human-Computer Interaction. Access to Today's Technologies*, pages 74–84, Cham, 2015. Springer International Publishing.

[109] P. Majaranta, U.-K. Ahola, and O. Špakov. Fast gaze typing with an adjustable dwell time. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '09, pages 357–360. ACM, 2009.

[110] P. Majaranta and A. Bulling. *Eye tracking eye-based human-computer interaction.* 2014.

[111] P. Majaranta and K.-J. Räihä. Twenty years of eye typing: Systems and design issues. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '02, pages 15–22. ACM, 2002.

[112] M. Mansouryar, J. Steil, Y. Sugano, and A. Bulling. 3d gaze estimation from 2d pupil positions on monocular head-mounted eye trackers. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '16, pages 197–200, 2016.

[113] D. Mardanbegi and D. W. Hansen. Mobile gaze-based screen interaction in 3d environments. In *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications*, NGCA '11, pages 2:1–2:4. ACM, 2011.

[114] D. Mardanbegi and D. W. Hansen. Parallax error in the monocular head-mounted eye trackers. In *Proceedings of the 2012 Conference on Ubiquitous Computing*, UbiComp '12, pages 689–694. ACM, 2012.

[115] N. Marquardt, R. Diaz-Marino, S. Boring, and S. Greenberg. The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the 24th Symposium on User Interface Software and Technology*, UIST '11, pages 315–326. ACM, 2011.

[116] A. J. May, T. Ross, S. H. Bayer, and M. J. Tarkiainen. Pedestrian navigation aids: information requirements and design implications. *Personal and Ubiquitous Computing*, 7(6):331–338, 2003.

[117] A. B. Mélodie Vidal and H. Gellersen. Pursuits: Spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 439–448. ACM, 2013.

[118] B. Miller, C. Keane, and M. O'Toole. *Miller-Keane Encyclopedia & Dictionary of Medicine, Nursing & Allied Health*. ENCYCLOPEDIA AND DICTIONARY OF MEDICINE, NURSING, AND ALLIED HEALTH. Saunders, 1997.

[119] D. Miniotas, O. Špakov, I. Tugoy, and I. S. MacKenzie. Speech-augmented eye gaze interaction with small closely spaced targets. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '06, pages 67–72. ACM, 2006.

[120] D. Model and M. Eizenman. A general framework for extension of a tracking range of user-calibration-free remote eye-gaze tracking systems. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 253–256. ACM, 2012.

[121] C. H. Morimoto and M. R. M. Mimica. Eye gaze tracking techniques for interactive applications. *Comput. Vis. Image Underst.*, 98(1):4–24, 2005.

[122] S. A. Mosquera, S. Verma, and C. McAlinden. Centration axis in refractive surgery. *Eye and Vision*, 2(1):4, 2015.

[123] O. Mubin, T. Lashina, and E. van Loenen. How not to become a buffoon in front of a shop window: A solution allowing natural head movement for interaction with a public display. In T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. Palanque, R. O. Prates, and M. Winckler, editors, *Human-Computer Interaction – INTERACT 2009*, pages 250–263, Berlin, Heidelberg, 2009. Springer.

[124] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, VISAPP '09, pages 331–340, 2009.

[125] J. Müller, D. Wilmsmann, J. Exeler, M. Buzeck, A. Schmidt, T. Jay, and A. Krüger. Display blindness: The effect of expectations on attention towards digital signage. In H. Tokuda, M. Beigl, A. Friday, A. J. B. Brush, and Y. Tobe, editors, *Pervasive Computing*, pages 1–8. Springer, 2009.

[126] Y. Nakanishi, T. Fujii, K. Kiatjima, Y. Sato, and H. Koike. Vision-based face tracking system for large displays. In *Proceedings of the 4th International Conference on Ubiquitous Computing*, UbiComp '02, pages 152–159. Springer, 2002.

[127] A. Nakazawa and C. Nitschke. Point of gaze estimation through corneal surface reflection in an active illumination environment. In *Proceedings of the European Conference on Computer Vision - Volume Part II*, ECCV'12, pages 159–172. Springer, 2012.

[128] A. Nakazawa, C. Nitschke, and T. Nishida. Non-calibrated and real-time human view estimation using a mobile corneal imaging camera. In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.

[129] A. Nakazawa, C. Nitschke, and T. Nishida. Registration of eye reflection and scene images using an aspherical eye model. *Journal of the Optical Society of America A*, 33(11):2264–2276, 2016.

[130] B. L. Nguyen, Y. Chahir, M. Molina, C. Tijus, and F. Jouen. Eye gaze tracking with free head movements using a single camera. In *Proceedings of the Symposium on Information and Communication Technology*, SoICT '10, pages 108–113. ACM, 2010.

[131] K. Nguyen, C. Wagner, D. Koons, and M. Flickner. Differences in the infrared bright pupil response of human eyes. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '02, pages 133–138. ACM, 2002.

[132] J. Nielsen. Noncommand user interfaces. *Communications of the ACM - Special issue on graphical user interfaces*, 36(4):83–99, 1993.

[133] J. Nielsen. Usability 101: Introduction to usability, 2003.

[134] K. Nishino, P. N. Belhumeur, and S. K. Nayar. Using eye reflections for face recognition under varying illumination. In *Tenth IEEE International Conference on Computer Vision (ICCV' 05)*, volume 1, pages 519–526. IEEE, 2005.

[135] K. Nishino and S. Nayar. Eyes for relighting. In *Proceedings of International Conference on Computer Graphics and Interactive techniques)*, SIGGRAPH '04, pages 704–711. ACM, 2004.

[136] K. Nishino and S. Nayar. The world in an eye. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '04, pages 444–451, 2004.

[137] K. Nishino and S. Nayar. Corneal imaging system: Environment from eyes. *International Journal of Computer Vision*, 70(1):23–40, 2006.

[138] C. Nitschke, A. Nakazawa, and T. Nishida. I see what you see: Point of gaze estimation from corneal images. In *Proceedings of Asian Conference on Pattern Recognition*, ACPR '13, pages 298–304. IEEE Computer Society, 2013.

[139] C. Nitschke, A. Nakazawa, and H. Takemura. Display-camera calibration using eye reflections and geometry constraints. *Computer Vision and Image Understanding*, 115(6):835–853, 2011.

[140] C. Nitschke, A. Nakazawa, and H. Takemura. Image-based eye pose and reflection analysis for advanced interaction techniques and scene understanding. *Computer Vision and Image Media (CVIM)*, pages 1–16, 2011.

[141] C. Nitschke, A. Nakazawa, and H. Takemura. Corneal imaging revisited: An overview of corneal reflection analysis and applications. *Information and Media Technologies*, 8(2):389–406, 2013.

[142] M. Nyström, R. Andersson, K. Holmqvist, and J. van de Weijer. The influence of calibration method and eye physiology on eyetracking data quality. *Behavior Research Methods*, 45(1):272–288, 2013.

[143] K. Ooms, L. Dupont, L. Lapon, and S. Popelka. Accuracy and precision of fixation locations recorded with the low-cost eye tribe tracker in different experimental set-ups. *Journal of Eye Movement Research*, 8(1), 2015.

[144] R. Ortiz. Freak: Fast retina keypoint. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 510–517. IEEE Computer Society, 2012.

[145] A. B. Pablo Alcantarilla, Jesus Nuevo. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *Proceedings of the British Machine Vision Conference*, pages 13.1–13.11. BMVA Press, 2013.

[146] N. Pears, D. G. Jackson, and P. Olivier. Smart phone interaction with registered displays. *IEEE Pervasive Computing*, 8(2):14–21, 2009.

[147] A. Pérez, M. L. Córdoba, A. García, R. Méndez, M. L. Muñoz, J. L. Pedraza, and F. M. Sánchez. A precise eye-gaze detection and tracking system. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, WSCG' 03, 2003.

[148] K. Pfeuffer, J. Alexander, M. K. Chong, and H. Gellersen. Gaze-touch: Combining gaze with multi-touch for interaction on the same surface. In *Proceedings of the 27th Symposium on User Interface Software and Technology*, UIST '14, pages 509–518. ACM, 2014.

[149] K. Pfeuffer, J. Alexander, and H. Gellersen. Partially-indirect bimanual input with gaze, pen, and touch for pan, zoom, and ink interaction. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '16, pages 2845–2856. ACM, 2016.

[150] K. Pfeuffer, M. Vidal, J. Turner, A. Bulling, and H. Gellersen. Pursuit calibration: Making gaze calibration less tedious and more flexible. In *Proceedings of the 26th Symposium on User Interface Software and Technology*, UIST '13, pages 261–270. ACM, 2013.

[151] A. Plopski, Y. Itoh, C. Nitschke, K. Kiyokawa, G. Klinker, and H. Takemura. Corneal-imaging calibration for optical see-through head-mounted displays. *IEEE transactions on visualization and computer graphics*, 21(4):481–490, 2015.

[152] D. Pohl, X. Zhang, A. Bulling, and O. Grau. Concept for using eye tracking in a head-mounted display to adapt rendering to the user's current visual field. In *Proceedings of the 22Nd Conference on Virtual Reality Software and Technology*, VRST '16, pages 323–324. ACM, 2016.

[153] P. Qvarfordt. The handbook of multimodal-multisensor interfaces. chapter Gaze-informed Multimodal Interaction, pages 365–402. Association for Computing Machinery and Morgan and Claypool, 2017.

[154] K.-J. Räihä, A. Hyrskykari, and P. Majaranta. 7 tracking of visual attention and adaptive applications. *Human Attention in Digital Environments*, page 166, 2011.

[155] J. Rantala, J. Kangas, P. Isokoski, D. Akkil, O. Špakov, and R. Raisamo. Haptic feedback of gaze gestures with glasses: Localization accuracy and effectiveness. In *Adjunct Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, pages 855–862. ACM, 2015.

[156] T. Rauber. *A time-efcient re-calibration algorithm for improved long time accuracy of head-worn eye trackers.* Universitt des Saarlandes, 2015.

[157] K. Rayner. *Eye movements and visual cognition: Scene perception and reading.* Springer, 2012.

[158] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger, 2016.

[159] M. J. Reinders, R. Koch, and J. J. Gerbrands. Locating facial features in image sequences using neural networks. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 230–235. IEEE, 1996.

[160] K.-F. Richter and S. Winter. *Landmarks: GIScience for intelligent services.* Springer, 2014.

[161] D. A. Robinson. A method of measuring eye movement using a scleral search coil in a magnetic field. *IEEE Transactions on Bio-medical Electronics*, 10:137–45, 1963.

[162] D. D. Salvucci and J. H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '00, pages 71–78. ACM, 2000.

[163] J. San Agustin, J. P. Hansen, and M. Tall. Gaze-based interaction with public displays using off-the-shelf components. In *Proceedings of the International Conference Adjunct Papers on Ubiquitous Computing - Adjunct*, UbiComp '10 Adjunct, pages 377–378. ACM, 2010.

[164] T. Santini, W. Fuhl, and E. Kasneci. Calibme: Fast and unsupervised eye tracker calibration for gaze-based pervasive human-computer interaction. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '17, pages 2594–2605. ACM, 2017.

[165] S. Schenk, M. Dreiser, G. Rigoll, and M. Dorr. Gazeeverywhere: Enabling gaze-only user interaction on an unmodified desktop pc in everyday scenarios. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '17, pages 3034–3044. ACM, 2017.

[166] D. Schnieders, X. Fu, and K.-Y. K. Wong. Reconstruction of display and eyes from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1442–1449. IEEE, 2010.

[167] S. K. Schnipke and M. W. Todd. Trials and tribulations of using an eye-tracking system. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, pages 273–274. ACM, 2000.

[168] D. Scott, J. Findlay, W. Hursley Human Factors Laboratory, and I. U. H. H. F. Laboratory. *Visual Search, Eye Movements and Display Units*. HF–147. IBM UK Hursley Human Factors Laboratory, 1991.

[169] J. S. Shell, T. Selker, and R. Vertegaal. Interacting with groups of computers. *Commun. ACM*, 46(3):40–46, 2003.

[170] D. Shepard. A Two-dimensional Interpolation Function for Irregularly-Spaced Data. In *Proc. ACM '68*, ACM '68, pages 517–524, 1968.

[171] L. E. Sibert and R. J. K. Jacob. Evaluation of eye gaze interaction. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '00, pages 281–288. ACM, 2000.

[172] A. Sippl, C. Holzmann, D. Zachhuber, and A. Ferscha. Real-time gaze tracking for public displays. In *Proceedings of the 1st International Joint Conference on Ambient Intelligence*, AmI'10, pages 167–176. Springer, 2010.

[173] S. A. Sirohey and A. Rosenfeld. Eye detection in a face image using linear and nonlinear filters. *Pattern Recognition*, 34(7):1367 – 1391, 2001.

[174] J. D. Smith, R. Vertegaal, and C. Sohn. Viewpointer: Lightweight calibration-free eye tracking for ubiquitous handsfree deixis. In *Proceedings of the 18th Symposium on User Interface Software and Technology*, UIST '05, pages 53–61. ACM, 2005.

[175] J. S. Stahl. Amplitude of human head movements associated with horizontal saccades. *Experimental Brain Research*, 126(1):41–54, 1999.

[176] D. M. Stampe. Heuristic filtering and reliable calibration methods for video-based pupil-tracking systems. *Behavior Research Methods, Instruments, & Computers*, 25(2):137–142, 1993.

[177] D. M. Stampe and E. M. Reingold. Selection by looking: A novel computer interface and its application to psychological research. In J. M. Findlay, R. Walker, and R. W. Kentridge, editors, *Eye Movement Research*, volume 6 of *Studies in Visual Information Processing*, pages 467 – 478. North-Holland, 1995.

[178] J. Steil and A. Bulling. Discovery of everyday human activities from long-term visual behaviour using topic models. In *Proc. UbiComp 2015*, pages 75–85, 2015.

[179] J. Steil and A. Bulling. Discovery of everyday human activities from long-term visual behaviour using topic models. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 75–85. ACM, 2015.

[180] S. Stellmach and R. Dachselt. Look & touch: Gaze-supported target acquisition. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '12, pages 2981–2990. ACM, 2012.

[181] S. Stellmach and R. Dachselt. Still looking: Investigating seamless gaze-supported selection, positioning, and manipulation of distant targets. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '13, pages 285–294. ACM, 2013.

[182] S. Stellmach, S. Stober, A. Nürnberger, and R. Dachselt. Designing gaze-supported multimodal interactions for the exploration of large image collections. In *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications*, NGCA '11, pages 1:1–1:8. ACM, 2011.

[183] Y. Sugano and A. Bulling. Self-calibrating head-mounted eye trackers using egocentric visual saliency. In *Proceedings of the 28th Symposium on User Interface Software; Technology*, UIST '15, pages 363–372. ACM, 2015.

[184] Y. Sugano, Y. Matsushita, and Y. Sato. Appearance-based gaze estimation using visual saliency. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(2):329–341, 2013.

[185] L. Świrski, A. Bulling, and N. Dodgson. Robust real-time pupil tracking in highly off-axis images. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 173–176. ACM, 2012.

[186] L. Świrski and N. Dodgson. A fully-automatic, temporal approach to single camera, glint-free 3d eye model fitting [abstract]. In *Proceedings of ECEM 2013*, 2013.

[187] K. Takemura, S. Kimura, and S. Suda. Estimating point-of-regard using corneal surface image. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, pages 251–254. ACM, 2014.

[188] K. Takemura, T. Yamakawa, J. Takamatsu, and T. Ogasawara. Estimating focused object using corneal surface image for eye-based interaction. In *3rd International Workshop on Pervasive Eye Tracking and Mobile Eye-Based Interaction*, 2013.

[189] B. W. Tatler, I. D. Gilchrist, and M. F. Land. Visual memory for objects in natural scenes: From fixations to object files. *The Quarterly Journal of Experimental Psychology Section A*, 58(5):931–960, 2005.

[190] B. Tessendorf, A. Bulling, D. Roggen, T. Stiefmeier, M. Feilner, P. Derleth, and G. Tröster. Recognition of hearing needs from body and eye movements to improve hearing instruments. In *Proceedings of the 9th International Conference on Pervasive Computing*, Pervasive'11, pages 314–331. Springer, 2011.

[191] F. Timm and E. Barth. Accurate eye centre localisation by means of gradients. volume 11, pages 125–130, 2011.

[192] A. Tom and M. Denis. Referring to landmark or street information in route directions: What difference does it make? In W. Kuhn, M. F. Worboys, and S. Timpf, editors, *Spatial Information Theory. Foundations of Geographic Information Science*, pages 362–374. Springer, 2003.

[193] J. Turner, J. Alexander, A. Bulling, and H. Gellersen. Gaze+rst: Integrating gaze and multitouch for remote rotate-scale-translate tasks. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '15, pages 4179–4188. ACM, 2015.

[194] J. Turner, A. Bulling, and H. Gellersen. Extending the visual field of a head-mounted eye tracker for pervasive eye-based interaction. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 269–272. ACM, 2012.

[195] B. Velichkovsky, A. Sprenger, and P. Unema. Towards gaze-mediated interaction: Collecting solutions of the "midas touch problem". In S. Howard, J. Hammond, and G. Lindgaard, editors, *Human-Computer Interaction INTERACT '97: IFIP TC13 International Conference on Human-Computer Interaction, 14th–18th July 1997, Sydney, Australia*, pages 509–516. Springer US, Boston, MA, 1997.

[196] R. Vertegaal et al. Attentive user interfaces. *Communications of the ACM*, 46(3):30–33, 2003.

[197] A. Villanueva and R. Cabeza. A novel gaze estimation system with one calibration point. *Trans. Sys. Man Cyber. Part B*, 38(4):1123–1138, 2008.

[198] O. Špakov and D. Miniotas. Gaze-based selection of standard-size menu items. In *Proceedings of the 7th International Conference on Multimodal Interfaces*, ICMI '05, pages 124–128. ACM, 2005.

[199] N. J. Wade. An essay upon single vision with two eyes. In *Destined for Distinguished Oblivion: The Scientific Vision of William Charles Wells (1757–1817)*, pages 71–117. Springer, 2003.

[200] M. Weiser. The computer for the 21st century. *Mobile Computing and Communications Review*, 3(3):3–11, 1999.

[201] M. Weiser and J. S. Brown. Beyond calculation. chapter The Coming Age of Calm Technolgy, pages 75–85. Copernicus, 1997.

[202] E. W. Weisstein. Least squares fitting. `http://mathworld.wolfram.com/LeastSquaresFitting.html`.

[203] F. Wenczel, L. Hepperle, and R. von Stlpnagel. Gaze behavior during incidental and intentional navigation in an outdoor environment. *Spatial Cognition & Computation*, 17(1-2):121–142, 2017.

[204] M. Wolfensberger and K.-F. Richter. A mobile application for a user-generated collection of landmarks. In J. Gensel and M. Tomko, editors, *Web and Wireless Geographical Information Systems*, pages 3–19. Springer, 2015.

[205] E. Wood and A. Bulling. Eyetab: Model-based gaze estimation on unmodified tablet computers. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, pages 207–210. ACM, 2014.

[206] D. Young, H. Tunley, and R. Samuels. *Specialised hough transform and active contour methods for real-time eye tracking.* University of Sussex, Cognitive & Computing Science, 1995.

[207] L. R. Young and D. Sheena. Survey of eye movement recording methods. *Behavior Research Methods & Instrumentation*, 7(5):397–429, 1975.

[208] L. H. Yu and M. . A new methodology for determining point-of-gaze in head-mounted eye tracking systems. *IEEE Transactions on Biomedical Engineering*, 51(10):1765–1773, 2004.

[209] L. H. Yu and M. Eizenman. A new methodology for determining point-of-gaze in head-mounted eye tracking systems. *IEEE transactions on bio-medical engineering*, 51(10):17651773, 2004.

[210] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2):99–111, 1992.

[211] S. Zhai, C. Morimoto, and S. Ihde. Manual and gaze input cascaded (magic) pointing. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '99, pages 246–253. ACM, 1999.

[212] X. Zhang and I. S. MacKenzie. Evaluating eye tracking with iso 9241 - part 9. In *Proceedings of the 12th International Conference on Human-computer Interaction: Intelligent Multimodal Interaction Environments*, HCI'07, pages 779–788. Springer, 2007.

[213] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. Appearance-based gaze estimation in the wild. *CoRR*, abs/1504.02863, 2015.

[214] Y. Zhang, A. Bulling, and H. Gellersen. Sideways: A gaze interface for spontaneous interaction with situated displays. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '13, pages 851–860. ACM, 2013.