LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
Department "Institut für Informatik"
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Heinrich Hußmann

Bachelor's Thesis

Gaze tracking on mobile devices

Thomas Mattusch
t.mattusch@campus.lmu.de

Bearbeitungszeitraum:	01.07.2016 bis 18.11.2016
Betreuer:	Mohamed Khamis
Verantw. Hochschullehrer:	Prof. Dr. Florian Alt

Zusammenfassung

Gaze bietet bereits neue Eingabe- und Interaktionsmöglichkeiten sowohl an öffentlichen Displays, als auch an Computerbildschirmen. Tagtäglich werden weitere Formen der Interaktion, die auf Gaze basieren, erforscht. Mit der stetig steigenden Präzision der Hardware, werden auch immer häufiger mobile Szenarien untersucht. Während es unter Laborbedingungen möglich ist, die Positionen der Nutzer zu fixieren und die Hardware für jeden Einzelnen zu kalibrieren, ist diese zeitaufwendige Vorbereitung nicht anwendbar, wenn diese Art der Interaktion von der breiten Masse genutzt werden soll.

Wir verwenden ein Konzept, welches steigende Popularität erhält, obwohl es erst Ende 2013 vorgestellt wurde. Es heißt Pursuits und nutzt gleichmäßige Augenbewegungen, die entstehen, wenn eine Person einem sich bewegendem Objekt hinter herschaut. In diesem Konzept werden die Augenbewegungen des Nutzers mit der dynamischen Bewegung von Objekten auf einem Display in Korrelation gesetzt. Es ermöglicht bereits eine spontane Interaktion mit öffentlichen Displays, aber es bietet auch die notwendigen Eigenschaften, um in mobilen Szenarien genutzt werden zu können, da es keine Kalibrierung benötigt.

Wir implementieren Pursuits auf einem Android basiertem System und erforschen die korrekte Erkennungsrate zwischen den unterschiedlichen Laufbahnen der Objekte in einer Teststudie. Zusätzlich erforschen wir die Benutzerfreundlichkeit von Gesten und Pursuits in einer zweiten Nutzerstudie.

Unsere Ergebnisse zeigen, dass obwohl die korrekte Erkennungsrate für eine Anwendung im Alltag zu niedrig ausfällt, die Teilnehmer die Benutzerfreundlichkeit von Pursuits positiv wahrnehmen, außerdem sind neun der 17 Teilnehmer bereit, die Authentifizierungsmöglichkeit, die auf unserem zweiten Pursuit Modell basiert, auch im Alltag nutzen.

Abstract

Gaze has already offered new input and interaction opportunities on public displays as well as on computer screens. Day by day further forms of interaction based on gaze are researched. With increasing precision of the hardware also mobile scenarios are studied more often. While setting up fixed user positions and calibrating the hardware for each user, is acceptable in a laboratory setting, this time-consuming preparation is not applicable if the interaction form should be commonly used.

We use a concept, which gains increasing popularity but was only introduced in late 2013. It is called Pursuits and makes use of smooth pursuit eye movements. In this concept the user's eye movements are correlated with the dynamic movements of the objects on the display. It offers spontaneous interaction with public displays already, but also provides the necessary features to be used in mobile scenarios, since it does not need calibration.

We implement Pursuits on an Android based system and explore the correct detection rates between different trajectories in a pilot study. Additionally, we study the usability of both eye gestures and Pursuits in a second user study.

Our results showed that although the detection rates with on-phone gaze estimation were still too low for real-world applications, the participants received Pursuit well in the usability study and 9 out of 17 participants would use an authentication system based on Pursuit Two on a

daily basis.

Problem Statement

Advances in processors and front-facing cameras of mobile devices have enabled basic tracking of the user's eyes on smartphones and tablets. This opens the door to many possibilities, for example, users can unlock their phones using their eye movements for higher security.

The goal of this thesis is to explore the feasibility and potential of eye tracking on mobile devices. In particular, this thesis will focus on the Pursuits technique, which was already shown to be promising on desktop and public display platforms. Based on an existing gaze tracking framework for mobile devices, the main task is to see if smooth pursuit eye movements can be detected using the front-facing camera of mobile devices, and exploring the possible mobile applications of such a new interaction modality.

In a first step, an existing framework for eye tracking on mobile devices will be extended to detect Pursuits movements. If detection of Pursuits is feasible, the following task will be to implement and evaluate an application that would make use of Pursuits as an input, otherwise we'll explore to what extent Pursuits can be utilized on mobile devices, assuming perfect gaze detection.

Tasks
Review of related literature in eye tracking on mobile devices
Extending the gaze tracking framework to detect Pursuits
Evaluating Pursuits on mobile devices
Evaluating sample applications that use Pursuits as an input method

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, November 18, 2016

. . . .

Contents

1 Introduction

Early observation of eye movements date back to the 1800s, where Louis Émile Javal recognized that the eyes do not move smoothly when following the text, but in a series of fixations and saccades. While this analysis is based purely on observation, at the end of the 19th century early eye tracking devices were introduced. They were based on mechanical lever systems. As reported by Wade et al. [17], the first successful non-invasive apparatus, used for eye tracking, was developed by Orschansky in 1899. It used a small mirror which was attached to an aluminum eye-cup. To observe the eye position, it recorded the reflection of light from the mirror.

While back then human observers were needed to evaluate these eye movements, nowadays we have powerful processors, which can process the data delivered by modern gaze trackers in real-time. Eye tracking systems are now commonly used to study user behavior. Modern fields for eye tracking are user centric research for marketing and web design [15]. In the environment of web development, eye tracking can tell how much time the user focuses on the shown elements. It can be detected which elements get the most attention and in which order the user looks at the elements. This can be used for creating better interfaces but also to place ads.

Eye tracking is not only of commercial interest, but is also used in medical research. One example is the use of detecting reading disorders like dyslexia [12]. Nowadays gaze is not only used to observe user behavior but also as an input modality. The idea to use gaze as an input modality dates back to 1987, when Ware et al. [18] proposed to use eye tracking as input.

Especially paralyzed subjects could profit from gaze based interaction, because their eyes stay functional in most cases. Being paralyzed can block them from easy communication with others and stop them from moving freely. A prototype by Wastlund et al. [19] uses gaze to control a power wheelchair. The users were able to navigate and gave positive feedback. Another project which is aimed to support disabled people is done by Lankford [9]. He proposes an interaction system for windows which can do all inputs in windows via the eyes.

Less serious but almost omni present in our society are video games. As eye tracking systems advance and increase the speed of their detection rates and become more precise at the same time, they also move into the field of video games. Saltuk [14] uses a game called ‘Whack-A-Mole’ as means for testing his calibration method in his master thesis. Further research with the focus on gaze based interaction for video games was done by Jie et al. [5]. This work shows which factors are significant in attention allocation and applies the gathered data to a video game shooter.

The presented methods either gather new knowledge about human behavior, help raising the living standard or are used to bring fun to people, but they all still have a common problem, which stops them from spreading among society. In the presented scenarios, additional hardware for gaze tracking is required, which is firstly still expensive and secondly needs calibration to work. In this context, a new field for research is opening. Today’s smartphones and tablets become more powerful and offer the basic hardware necessary for eye tracking. While low-cost gaze estimation systems based on smartphones and tablet PCs are promising, they come with major disadvantages. For once their front-facing cameras are bound to RGB images, where commercial eye trackers have infrared illumination, additionally, the front-facing cameras have a lower resolution than dedicated eye tracking cameras, the processing power is far lower than on the PC and the screen is smaller compared to a desktop setting, requiring higher accuracy. Also neither the screen nor the user stays in a fixed position. Despite these challenges Android based system have already been developed and show that the devices are capable to deliver basic gaze estimation [4].

A novel concept called Pursuits has shown that it is capable of approaching the calibration issue. In this work, we adapt Pomp's [11] eye tracking framework and implement Pursuits on a smartphone with an Android operation system. Our application runs on unmodified android devices and neither needs additional eye tracking hardware nor calibration. In a technical prototype we evaluate the correct detection rates for two moving objects on a Nexus 5 [10]. We use the real-time detection function, offered by the framework, for the gaze estimation. Further we propose four authentication methods in a Wizard of Oz experiment and evaluate its usability.

2 Background And Related Work

This section gives an overview of gaze tracking and presents work from researchers in the same field.

2.1 EyeTab

Andre Pomp's framework and the algorithms used by our application, are based on EyeTab. EyeTab, developed by Wood et al. [20], introduces a model-based approach for binocular gaze estimation. It runs on unmodified tablets and is based on a set of established image processing and computer vision algorithms. It offers near real-time gaze estimation. In addition to the presentation of a new framework, they also evaluate its precision. The precision of EyeTab was further studied by Hohlfeld et al. [4]. They measured the influence of different lighting conditions, view distance, the effect of wearing classes and different algorithms for both pupil detection and gaze tracking.

The most mentionable finding is that for the system to deliver low error rates, a distance to the face of about 20 cm should be kept at most. This is an unnatural position for operating a mobile device and it might prove to be difficult for the users in the pilot study to keep. Since it was shown that glasses do not play a significant role for the gaze detection we let the participants wear them.

To detect an on-screen gaze point, EyeTab has three stages eye localization, limbal ellipse fitting and gaze geometry. First the software detects the rough eye positions and then creates a rough eye-region of interest (ROI)) around them. In a second step this region is refined and the precise eye-centers are located. In the limbal ellipse fitting stage, within the ROI the limbus and edge points are detected and an ellipse fitted around the limbus. This is then projected from 2-D to a 3-D circle. Out of this 3-D circle a smoothed gaze point is derived.

2.2 Smooth pursuit eye movements and Pursuits

The distinction between smooth pursuit eye movement was made by Dodge in 1903 and explored by Robinson [13]. The idea of using smooth pursuit eye movements as an input modality was proposed by Vidal et al. [16]. In this concept called Pursuits the eye movements are correlated with moving objects on the screen. This method approaches the issue of calibration. Because of the correlation between the movements it does not matter where in the virtual room the eye's gaze is located, as long as it does the same motion as an object on the screen, the eye's motion can be matched to the object.

Khamis et al. [6] created a game based on Pursuits, where the users had to select fish 'swimming' over the screen, by following their movements with their eyes and conducted a field study to test the robustness of the system in the wild.

Based on Pursuits, Esteves et al. [2] presented Orbits. It uses Pursuits to offer hands-free interaction on a smart watch. Their work shows that with current generation eye trackers, eye movements following motions that are as small as the display of a smart watch can be detected and used as an input modality.

Another adaption of Pursuits was done by Khamis et al. [7], here Pursuits were used to calibrate the system with a text-based stimulus. The calibration was done while the user 'pursues' the shown text. The advantage of this system, called Read2-Calibrate, is that it can be used in a natural fashion without standing out as a calibration stage. They also introduced a prototype application which allows the user to select text with the help of Pursuits. This application does not require calibration and is called EyeVote.

2.3 Authentication Methods And Shoulder Surfing

Shoulder surfing means to observe someone when this person is filling in data, which is visible from the outside. This is for example used when entering a PIN in an ATM, but also when authenticating mobile devices. And it is also a serious threat for ones personal information. Kumar et al. [8] propose a gaze based input modality for a higher security when entering PINs on a smartphone. This application is called EyePassword. They could not only show that the presented method uses only slightly more time than a keyboard based system but also that the error rates where close.

Another system called Cued Gaze-Points, was introduced by Forget et al. [3]. In this system positions on an image are selected with the user's gaze. This offers many input possibilities, which are not bound by numbers or letters.

3 Challenges

The first challenge was to find the location in the used framework where the gaze points were calculated and to thin down the framework, to make it suitable for our own application. The next step was to implement the moving objects and to collect the data necessary for the use of Pearson-moment correlation. At first we implemented the calculation on the mobile phone itself. At this stage the objects were marked in a different color, when a user was following a moving object with their gaze. As this did seem to work more often than not, but not consistently enough to be used in a real-world application, we wanted to gain data, that gives us more information about the detection rates. Therefore we switched from on-phone calculation to saving the data in CSV files, which were then analyzed using Google's sheets. At this point we used the prototype to conduct a pilot study to gather more data and explore the gaze precision. In the pilot study, many gaze points were taken and the phone's detection rate became slower the longer each setting was running. This showed another limitation with current generation mobile phones, since only around the first 20 to 30 gaze points can be collected before the detection rates start to diminish. It showed us that the software cannot be used for a higher amount of time in combination with the used hardware. Everything until this point was tested with pursuits, but since the results of the pilot study told us the correct detection rate was far too low, we then switched to an eye gesture based input method, where the precision of the detection was less important for a correct result. In this state of the application, we found a problematic behavior of the system. To use eye gestures instead of pursuits, we needed some kind of calibration, so we used the last few detected gaze points as the basis and then calculated the difference for each axis. The calculation was again done on the phone. When the calibration was correctly done in the center and the user for example looked in the top left corner, the eye movement was correctly detected, but when the user looked back to the center, it first detected the motion to the center but then went further to the bottom right corner, while the eyes were still fixated on the center. The framework was multiple calculated gaze estimations behind the actual eye movement but still calculated them in order. In addition to this way of receiving the gaze points, the system itself became slower the more gaze points were estimated, which only increased the negative effect. At this point we gave up on the framework with this hardware, since we did not have much time for additional experiments and the results were not encouraging to go on with the current hardware.

4 Pursuits

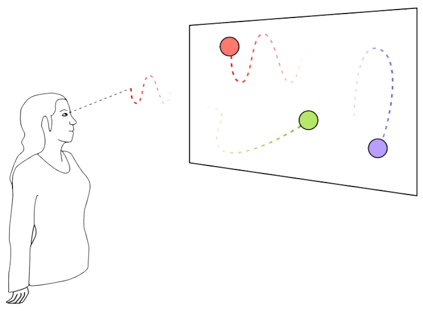


Figure 4.1: This graphic shows the concept of Pursuits [16], where a user is following the dynamic objects on the screen with the eyes. With the dotted lines, the different trajectories are indicated.

Pursuits are based on smooth pursuit eye movements, where eyes only move smoothly when following a moving object. The goal of Pursuits, when they were introduced, was to enable spontaneous interaction with eye tracking based interfaces. The basic idea of Pursuits, is to match these smooth eye movements with the movements of an object on the screen. This leads to different premises. First a dynamic interface is necessary, to match eye movements to objects on the screen. If we have no moving objects on the screen we cannot match the eye movements. Further, if an object moves up on the screen, the Y coordinate of the object increases, then gaze tracking device needs to detect the eye's Y coordinate to also increase. Another benefit is the systems independence of target size, because the trajectories are compared the target size has no influence on the detection. This means also small targets can be used, which can be difficult for gaze fixation based systems to detect. To correlate both movements the Pearson product-moment correlation coefficient is used.

4.1 Pearson product-moment correlation

The formula was developed by Karl Pearson. It indicates the degree of linear correlation between two variables. The result of the correlation can be values between 1 and -1, either showing a strong positive or negative linear correlation. If the result is null it indicates, that the compared variables do not have a correlation at all.
The formula for the Pearson's correlation is:

$$Correlation = \frac{\sum_{i=1}^{N}(x-\bar{x})(y-\bar{y})}{\sqrt{\sum_{i=1}^{N}(x-\bar{x})^2(y-\bar{y})^2}} \tag{1}$$

The Pearson's correlation is used in both the on-phone calculation which is used in the first and last attempt as well as in the calculations done with Excel.

5 Required software

In this section a short overview over the used software is given. This is important, because to be able to further develop and work on this system, certain software versions are necessary.

5.1 Eclipse with CDT 4.4.2

Eclipse is an open source development environment, which was originally used as an integrated development environment for Java. It was further developed and now offers a variety of plugins for different programming languages. We used Eclipse Luna with the CDT plugin, which supports C and C++. Once the necessary plugins are installed Eclipse offers comfortable functions like autocomplete. The editor is highly customizable and offers different views, for class exploration, functions, variables or navigation, which can be organized by the user. The version used is Eclipse Luna 4.4.2 with CDT.

5.2 OPENCV 4 Android

OPENCV (Open Source Computer Vision) is an open source software under a BSD license. It is free of use for both commercial and non-commercial use. It supports different operating systems, such as Windows, Linux, iOS and Android. It has interfaces for C++, C, Python and Java. In our setup we use OpenCV-2.4.10. It offers libraries for image processing and is aimed at real-time computer vision.

5.3 Threading Building Blocks (Intel® TBB)

Intel® TBB is used to write parallel C++ programs, this supports the programmer to make easier use of the available cores. Since android smart phones have more but weaker cores this is an essential part for real-time gaze detection on mobile devices. Used is the version Intel TBB 4.4 Update 3.

5.4 Android SDK and NDK

As for the android NDK we use android-ndk-r10e, because the older versions do not support the needed methods for André Pomp's, but newer version do not support eclipse anymore. After android-ndk-r10e, Google dropped its support for external development environments. We wanted to use eclipse, because André Pomp's framework is not a Gradle built software as used by Android-Studio. The SDK was AndroidSDK2441. The Android build target is version 4.4.2.

5.5 OpenCSV

OpenCSV is Java library used to create comma-separated values, which we used for our data collection in both the pilot and the usability study. It is released under a Apache 2.0 license and free for private and commercial use.

5.6 Google Docs, Sheets, Slides and Forms

Google offers multiple online tools for creating and editing text documents, tables, presentations and questionnaires. They also offer the creation of diagrams based on the collected data. This was used in combination with OpenCSV to access, analyze and visualize the collected data.

6 Implementation

After making sure that the eye detection does work on the device and does at least detect the rough direction the user is looking in, the frame work was then used as a basis for our approach to Pursuits on mobile phones. The first attempt, to implement pursuits on mobile phone was with on-phone correlation calculation, which showed live results. We took 6 gaze points before we gave the first result and then updated it with every gaze point taken. The last 6 gaze points and object coordinates were used to detect the selected object. But the marker changed too often even when looking at the same object. As a consequence, the system was exchanged for the Pilot study, with the aim to analyze how many gaze points need to be taken for the best results and which influence the trajectories have.

6.1 Application Used In The Pilot Study

To get analyzable data the application was remodeled. Instead of on-phone calculation, a database system was implemented. The X and Y coordinates of both moving objects at the moment the frames are taken and the name of the visible object is saved. This is used to identify correct detection later on. The frame work itself delivered two different values for each the X and Y coordinates for detected gaze locations. One was recalculated to better match the screen and the other set was the raw coordinates ranging into imaginary space around the screen. Additionally, the time stamp is taken.

In this state of the application three different trajectories were used and tested against each other to see which combination will yield the highest correct detection rate. We used linear, elliptic and curve movements. Each combination is selectable in the home screen. The tests are named after the trajectories they use. Overall there are 6 tests.

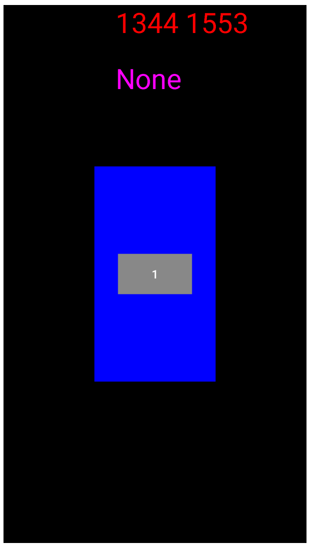
André Pomp's framework offers additional functions which are not needed for the live gaze tracking, like methods to record the screen and the user's eye for post processing or for logging sensor data. Our application only uses the necessary components to use the live gaze tracking features. We used one of his settings for live gaze detection, but implemented our own tests. When a test is selected, a new ShowActivity is started which creates a service called RecordService. The RecordService creates the necessary instances used for the gaze application. It holds the classes for sensor logging, screen recording, the test settings and the for us important part the DisplayManager. The DisplayManager holds the classes that are used for presentation of the applications. In one of these classes, called the LiveGazeWindowManager, the images taken by the front camera are passed to the gaze tracker, which calculates the gaze points. The class LiveGazeWindowManager is also used to display our application.

The application we implemented to display the moving objects, consists of the following parts, the class MovingDotsDrawView, which holds and creates instances of the moving objects. The moving objects are derived from either LinearTwoPointMovement, CircularClockwise or BezierCurveMovement. The classes themselves are Threads, so they update their positions themselves, but to reduce the amount of redraws an external class is used. It is called Surface-ViewUpdater. To switch the visibility, we use the class VisibilitySwitcher, which is done after collecting the necessary amount of gaze points and not after a certain amount of time has passed.

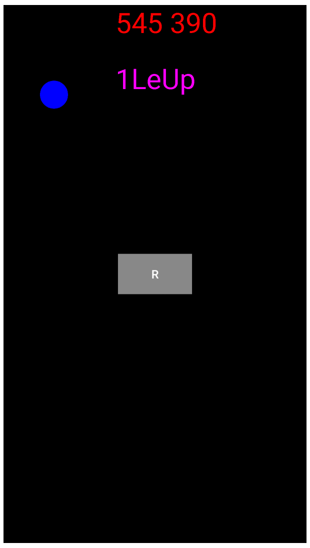
6.1.1 Trajectories

We used three different trajectories in the prototype used for the pilot study and combined two for each test. In the end, every possible combination of two of the three trajectories was tested. We decided to use trajectories which would differ most in the nature of their movement. For the test we used linear, circular and curve movements.

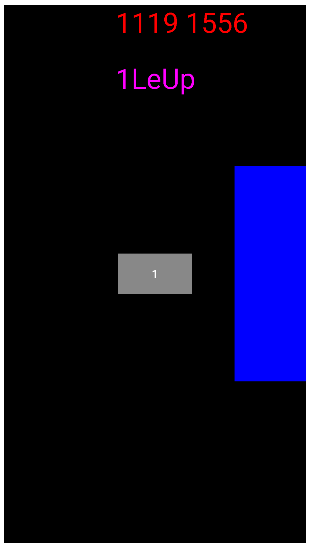
6.2 Gestures Instead Of Pursuits



(a) No eye gesture has been detected before. The application is in its start up position.



(b) Now an eye gesture to the left upper corner has been detected.



(c) A new gesture detection has been initialized, while the last result of the gesture detection is still shown. The threshold for the X axis has been exceeded, which is indicated by the rectangle

Figure 6.1: This shows the basic states of the gesture prototype

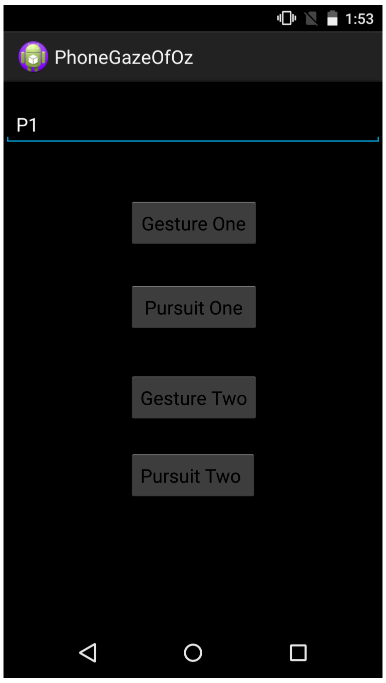
In comparison to Pursuits, eye gestures need less gaze points to operate. Since we found out that we have an ever-growing latency for the calculation time of the gaze estimation, with eye gestures a less demanding scheme is tested. Additionally, it is easier to experiment with different settings for the distance threshold needed, to detect a gesture. The application based on gestures consists of only one test. The test shows the X and Y coordinates of the last detected gaze point at the top and also the last selected gaze direction, if one has been selected before. At the center is a button to initiate the calibration and detection of the eye gestures. The gaze points taken before the button is pressed are used to calibrate the application. The idea was that the user will look at the button before pressing it. After pressing the button the difference between the calibration position and the newest gaze point is used, to calculate the direction the user is looking in. Because the device was getting slower over time, it was of interest to get the detection as fast as possible. The other option would have been to collect multiple gaze points and calculate the average to get a more consistent result. Additionally, to the coordinates the application will show which direction is detected, using rectangles which are drawn on the background. The goal is to detect an eye movement in the direction of one corner. This time the distance need to be greater than a threshold to detect an eye movement, in opposition to the pilot study where we did not use a threshold for selecting an object. If one corner is selected as the corner the person is looking at, the program will not show another position until it is restarted by pressing the button in the center. This will create a new calibration position which is used for the next gesture recognition. If the position in one of the corners is detected a circle is shown instead of a rectangle. The rectangles indicate either if the difference for one of the axes is already high enough to detect a movement for either up or down, left or right (6.1c) or if none the thresholds are exceeded (6.1a). The detection of a gesture is fulfilled if the application detects an eye movement to either of the corners, which means if the difference in both X and Y coordinates are greater than the used thresholds (6.1b).

6.3 The Parser

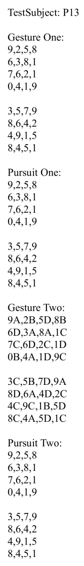
The data collected in the pilot study was saved in CSV files, but since the users had to go back manually after the visible object switched for a certain amount of times, the first object was then shown again and additional gaze points were taken. These gaze points were deleted to have the same amount of data taken for each participant. Because the amount of data was too great to efficiently work through manually, the functions used by Google's spreadsheets were parsed into the CSV files. This was used to avoid manual effort to program the functions within spreadsheets. The parser added the functions for the calculation of the correlation between the detected gaze point's X and Y values and the object's X and Y values. Afterwards the average correlation for X and Y axes were calculated. The object visible at the time was also stored. Based on the highest average correlation the selected object was chosen and if it was the same as the visible object at the time, then the correct detection was saved as 'Good' else as 'Bad'.

6.4 Sample Application

The sample application used for the usability study was designed as a Wizard of Oz experiment. To explore the usability of the presented input modalities the participants were told that the input does work, but that the system does not give feedback. The application presents four input modalities, which are called Gesture One, Gesture Two, Pursuit One and Pursuit Two. 6.2b shows one of the entry forms each participant received before they started. The entry form contains an anonymous identifier, the PINs the participants should enter and the order the tests are performed in. After selecting one test setting and before entering each PIN the user has to press start and after the PIN is entered stop. The user then has the option to either redo the test, if the participant thinks the entry was incorrect or confirm the PIN if the test subject thinks the input was correct. The digits in each test are ranging from 0 to 9 and the letters in Gesture Two from A-D. For Pursuit One and Pursuit Two, linear and circular trajectories are used. We decided to use the two trajectories with the highest distinguishability and study how the users perceive the tests.



(a) Homescreen



```
TestSubject: P13

Gesture One:
9,2,5,8
6,3,8,1
7,6,2,1
0,4,1,9

3,5,7,9
8,6,4,2
4,9,1,5
8,4,5,1

Pursuit One:
9,2,5,8
6,3,8,1
7,6,2,1
0,4,1,9

3,5,7,9
8,6,4,2
4,9,1,5
8,4,5,1

Gesture Two:
9A,2B,5D,8B
6D,3A,8A,1C
7C,6D,2C,1D
0B,4A,1D,9C

3C,5B,7D,9A
8D,6A,4D,2C
4C,9C,1B,5D
8C,4A,5D,1C

Pursuit Two:
9,2,5,8
6,3,8,1
7,6,2,1
0,4,1,9

3,5,7,9
8,6,4,2
4,9,1,5
8,4,5,1
```

(b) Sheet with PINs, test subject name and order in which the tests are done

Figure 6.2: Homescreen Of The Sample Application And PIN Entry Form

6.4.1 Gesture One

In this setup, the PINs are placed in a clockwise ascending order around the edges of the screen as shown in 6.3a. To enter a PIN the user has to look at the center of the screen and then to the digit he wishes to select. After the PIN was entered the digits are reordered. The position of the first digit is randomized and then the others are added in clockwise ascending order.

6.4.2 Gesture Two

6.3b shows the second input method. This input modality is different from the others, because it has two-part way to enter a PIN. First the user presses a button with the wished digit on it and then looks to the letter he wants to enter. The letters are placed around the edges of the screen. After completing one PIN entry the buttons stay in the same position, but the letters are reordered in the same manner the digits are reordered in.

6.4.3 Pursuit One

As shown in 6.3c the digits themselves are moving either in circles or linear trajectories. To enter a PIN the user has to just follow the moving objects with their eyes for a short amount of time. The objects trajectories are centered around one of ten preselected positions which are placed similar to the PIN entry fields on a smart phone. After a PIN is entered the position of the digits are randomized.

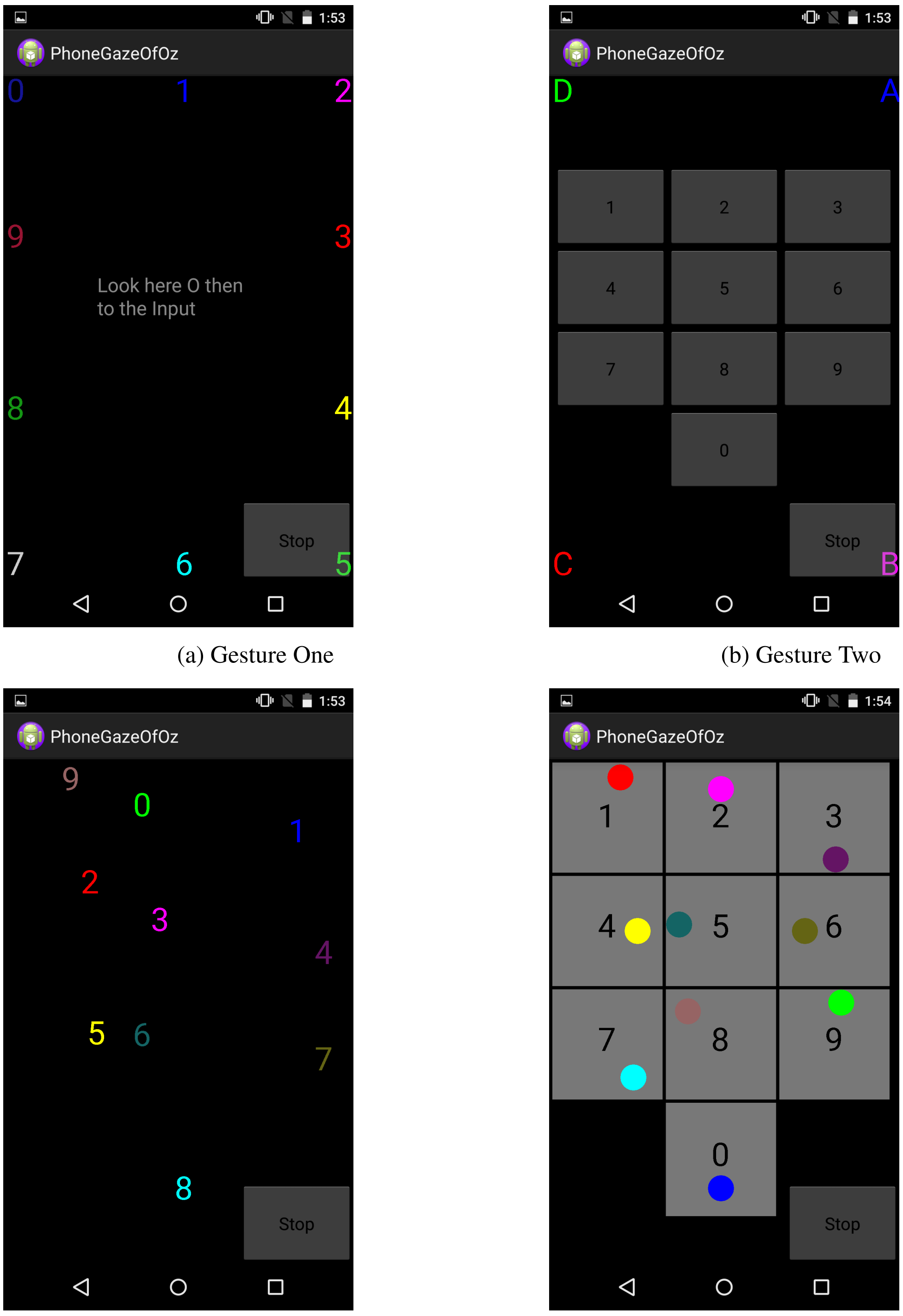
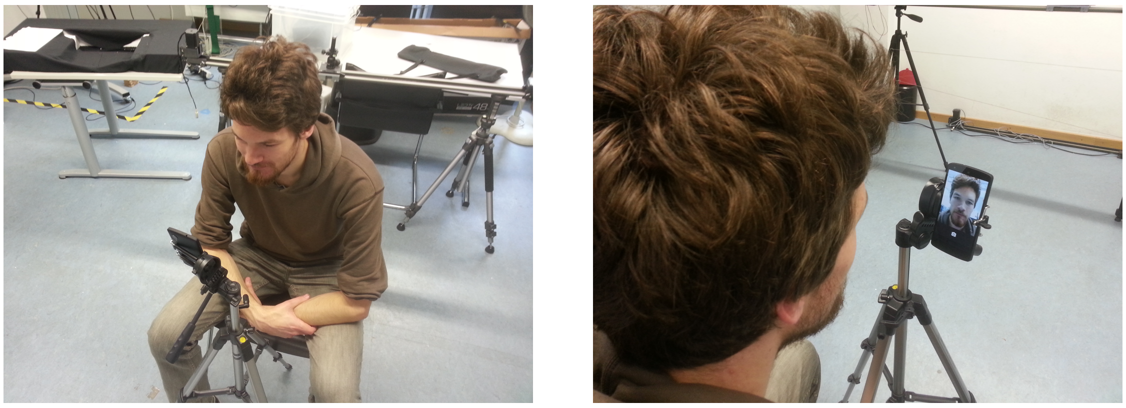


Figure 6.3: The Four Tests

6.4.4 Pursuit Two

In 6.3d circles are moving above rectangles with the number written on the rectangles. To select a digit the user has to follow the moving object above the wanted digit with his eyes. After the PIN is entered the digits will stay in the same location. Only the moving objects will change their position, this means the trajectory needed to follow for the entry is a different one after each test.

7 Pilot Study



(a) User Position (b) The mobile phone is being oriented

Figure 7.1: The participant has already been introduced to his task and now the smartphone is oriented. The aim is to position the participant in a manner that the face fills most of the screen and is around 20 cm away from the camera. When this is done, the application is started.

7.1 Goals

The lab study was designed to test the accuracy of the live gaze detection on the mobile phone and explore the burden on the user.

7.2 Apparatus

In this study a Nexus 5 was used. Its processor is a Snapdragon 800 with 4 cores and a CPU tact of 2.26 GHz. It has a screen resolution of 1920x1080 pixel on a screen size of 4.95-inch. The front camera has a maximum resolution of 1280x960 pixel, but the framework was set to use 720p. This smartphone was mounted on a tripod used for photography or video recording and was fixed in an improvised holder.

7.3 Participants

For the pilot study, we recruited 10 participants (7 female, 3 male) aged between 16 and 27 years (M=23.5). The participants were mostly students and acquire via personal contact. The participants had different professions including law student, European ethnology, one industrial business woman, one person still going to school and media informatics students. Out of the 10 participants, 6 worked with gaze based interaction systems before.

7.4 Procedure

At first the participants were introduced to the task they had to fulfill. Afterwards they signed a consent form and then were asked to sit on a chair. Then the smart phone was oriented, so that the user's face filled most of the screen, as shown in 7.1a.

The user then had to enter an anonymous test subject name and then start the first test. Since only one object was visible at a time, they only had to follow the visible object with their eyes in each test. After both of the objects were visible for two iterations, they had to manually press back and select the next task, until all test have been done. Each iteration collected 20 gaze points. After all tests were fulfilled an online survey was done, in which the participants had to answer questions regarding their person and the perceived difficulties of the application.

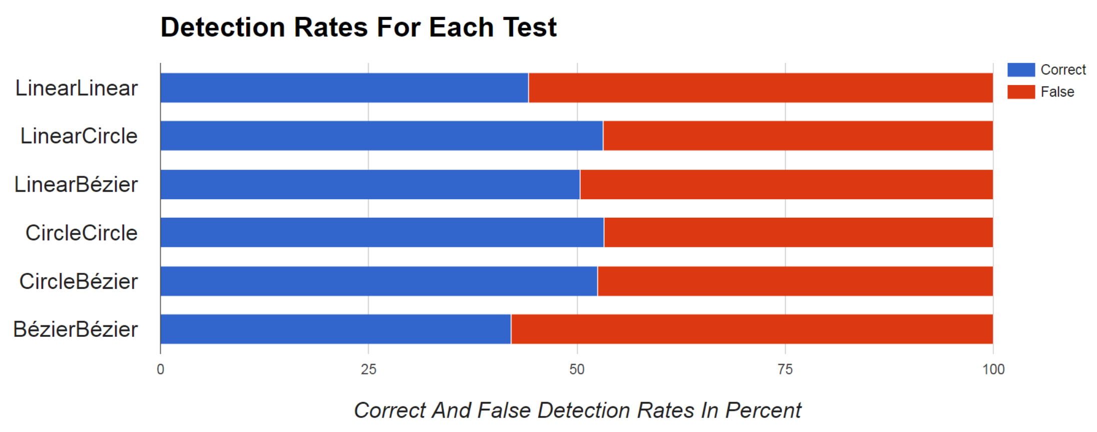


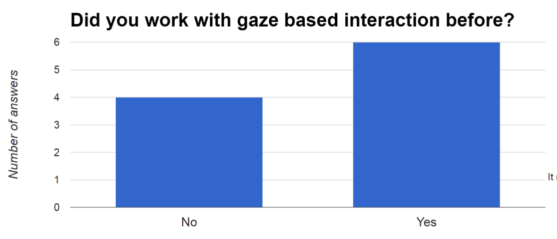
Figure 7.2: This graph shows the percentage of correct and false detection rates for each test.

7.5 Results

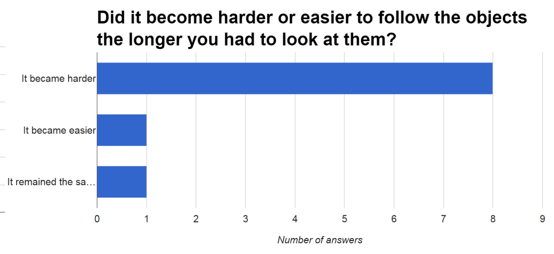
In this study, we collected 800 gaze points per test. In total 4800 gaze points were collected. The correlation was then calculated via the Pearson momentum correlation. This was done in Google's spreadsheets. After calculating the correlation for X and Y Axis, the average of both values was used to decide which object was selected. This was calculated for each object. In the next step the object with the highest correlation was compared to the visible object at that time, if it was the same, the detection was deemed correct.
The graph in 7.2 represents the correct detection rate for each test. This includes the detection rate after 20 gaze points for each object, the amount of gaze points an object was visible. As the graphs have shown the results are around 50%. This was a huge draw back, because it showed that the framework in combination with the hardware does not grant reliable gaze estimation.

After realizing, that the application does not deliver usable results when taking in all collected gaze points, we looked at the performance with each new gaze point taken. 7.4 shows the average correct detection rate per test and per number of taken gaze points. The Pearson correlation needs a minimum of three values to calculate a correlation, which means, that the X Axis actually starts by 3 taken gaze points. For the figure 7.4 to show only one test e.g. LinearLinear, the average correlation is always the one for visible object, which means both objects in the test LinearLinear are included in the average.

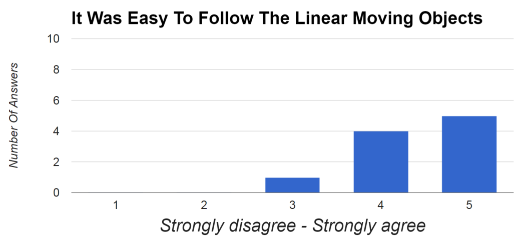
In the next step, we looked at the development of the correlation changing with the amount of collected gaze points. Regarding the correct detection rates, we recognized that after taking the first 6 gaze points and object coordinates, the detection rates have stabilized. Using more than the initial 6 gaze points does not improve the performance of the detection, for 4 out of 6 scenarios. In the two cases where there was a slight improvement, the necessary amount of gaze points would take too long to be applicable in a real-world application.



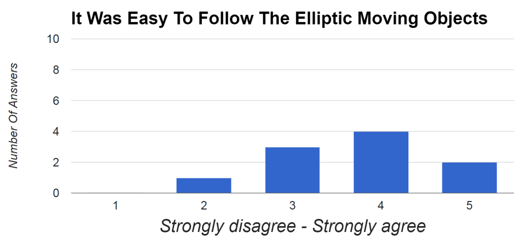
(a)



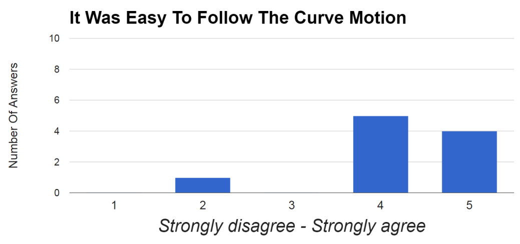
(b)



(c)



(d)



(e)

Figure 7.3: The graphs shown above are part of the survey done in the pilot study.

7.6 Qualitative Feedback

The participants wished mostly for the system to stop automatically, after collecting the necessary amount of gaze Points, so that they have one less thing to memorize. Another issue was the jump between the objects when switching visibility. One participant wished for a smoother transition between the visible objects and that the distance was smaller, because some objects were out of the range of their view at the time the visibility switched. Another part was the waiting for the hardware to cool down in between the tests, one user wished to have shorter breaks. Another participant proposed to use a head rest, similar to the head rests used when taking x-rays of teeth, but we decided beforehand that we wanted to use a more realistic test setting.

7.7 Summary

While the pilot study has proven, that the implemented system does not perform as wished, the Likert scales show that although 40% of the participants did not work with gaze based interac-

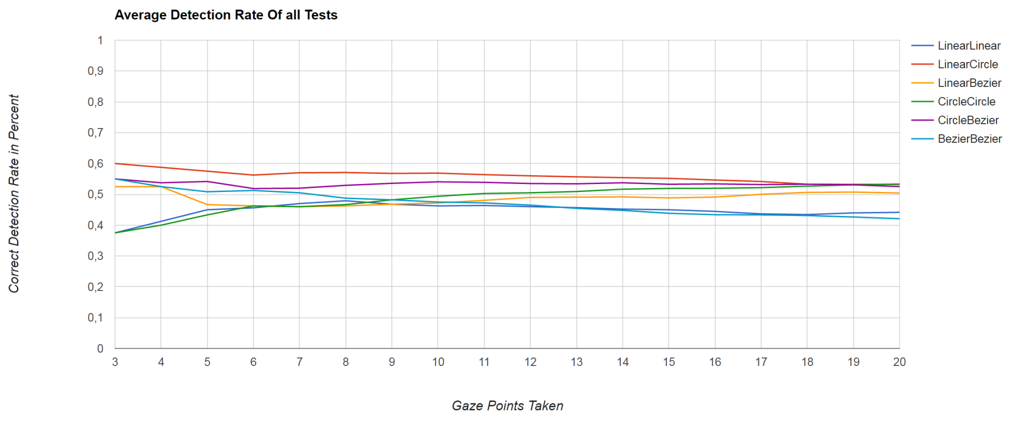


Figure 7.4: Pilot Study Average Correct Detection Rate Of All Tests Per Collected Gaze Points

tion before, most of the participants did not have much difficulty following the dynamic objects (7.3c,7.3d,7.3e). Another aspect of minor concern for Pursuits in general, is that 80% of the user found it harder to look at the moving objects the longer the test went on, as shown in 7.3b.

8 Usability Study

8.1 Goals

The goal of this study was to explore the usability of the formerly introduced input modalities in the terms of physical and mental demand.

8.2 Apparatus

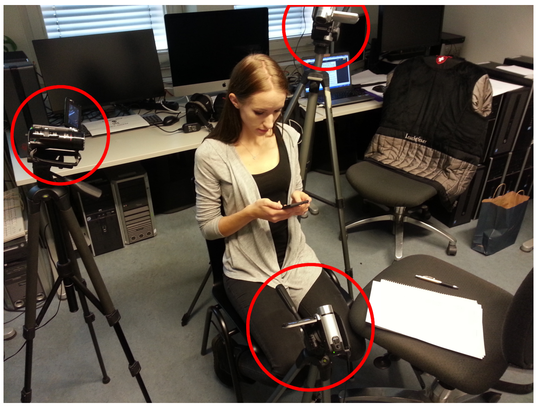
As shown in 8.1a the participants were filmed from three different angles. The front video camera was aiming at the eyes of the individuals, the side video camera was filming both the side of the eye and the display of the mobile phone and the third camera was filming the mobile phone from behind. The cameras are all models from Canon. While the front and the back camera have a lower resolution, the side camera is a high definition camera. The participants held the mobile device in one hand, to not block the side view. The procedure was filmed, with the aim to have a further study in the future, which evaluates the security of presented entry methods.

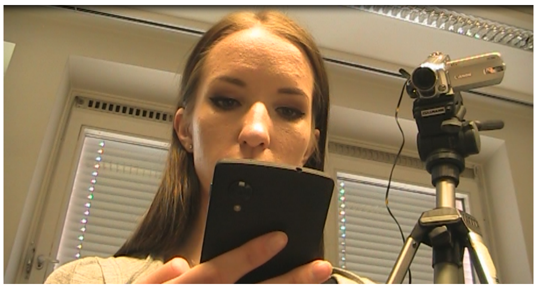
8.3 Participants

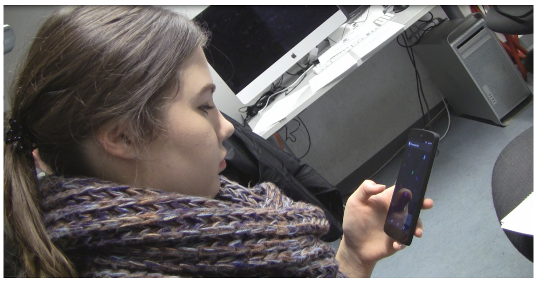
We recruited 17 participants (10 female, 7 male) aged between 16 and 52 years (M=25.76). The participants were recruited via an online mailing list and were mostly students.

8.4 Procedure

Upon arrival, the participants were informed that the study was filmed by three cameras and asked for their approval. After giving their consent, the participants were informed about the general task and then signed a consent form. Afterwards, the different input modalities were explained step by step. Then the participants were seated on a chair between the three video cameras and the video cameras were oriented. When everything was set up, each of the participants received an entry form on which the PINs they should enter were printed. A PIN had either four digits or


(a) User in setup position


(b) View of the front camera


(c) View of the side camera

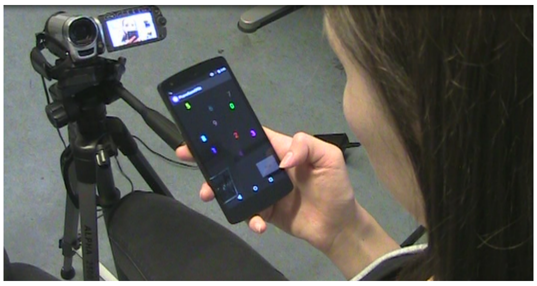

(d) View of the back camera

Figure 8.1

four combinations of a digit and a letter. Each participant had to enter eight PINs for each test, with four tests in total. After each test the participant had to fill a NTLX questionnaire and when they were done with all tests they had to fill an additional online questionnaire.

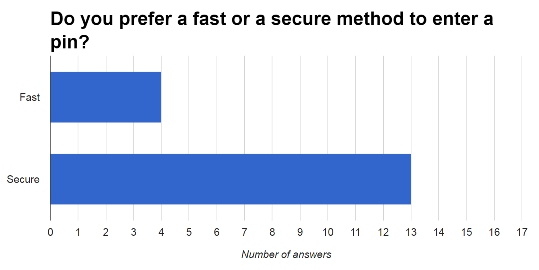
8.5 Results

We logged the time taken for each PIN to be entered by the user. Each user had to enter 8 PINs for each of the four input methods. In total, we took the time of 512 PIN entries (One User did not do a test, so we have valid data for only 16 participants). Additionally, we have three videos per participant for each of the 17 participants.

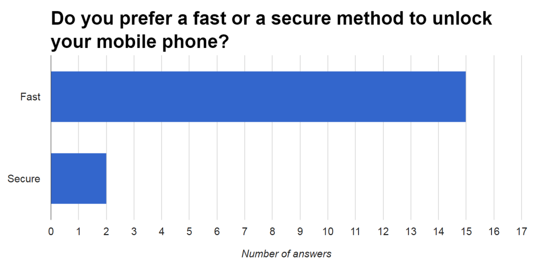
Out of the 17 participants, 7 are using a PIN to unlock their phone, 4 use a finger print sensor, 4 use a pattern and 2 just swipe. To enter their PINs, 13 participants prefer a secure over a fast input method while 4 prefer a fast input. When asked which method they prefer to unlock their mobile phones, the answers are more diverse. Now only two prefer a secure unlock method while 15 want a fast method to unlock their mobile phones. When asked which of the presented methods they would use if it offers a higher security than the standard PIN entry method, Pursuit Two was the most favored. When asked which input method they would use on a daily basis, Pursuit Two was still the most appreciated, but the usability was suddenly more important, resulting in less overall acceptance.

8.6 Perceived Workload

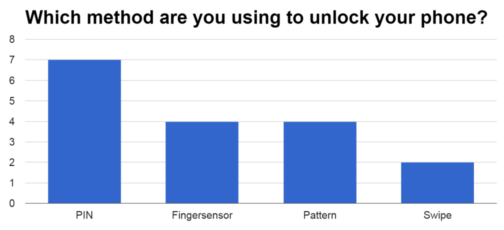
The results of the Nasa TLX confirm the results of the questionnaire. With 2.733 on the mental demand Pursuit two is the least demanding of all tested input modalities in case of mental demand, followed by Gesture One with 3.19. Pursuit Two was not only the most favored out of the four



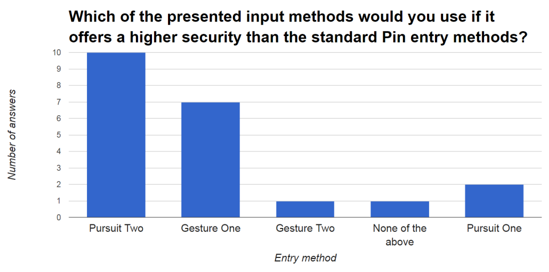
(a)



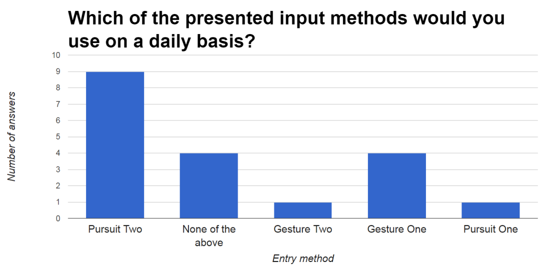
(b)



(c)

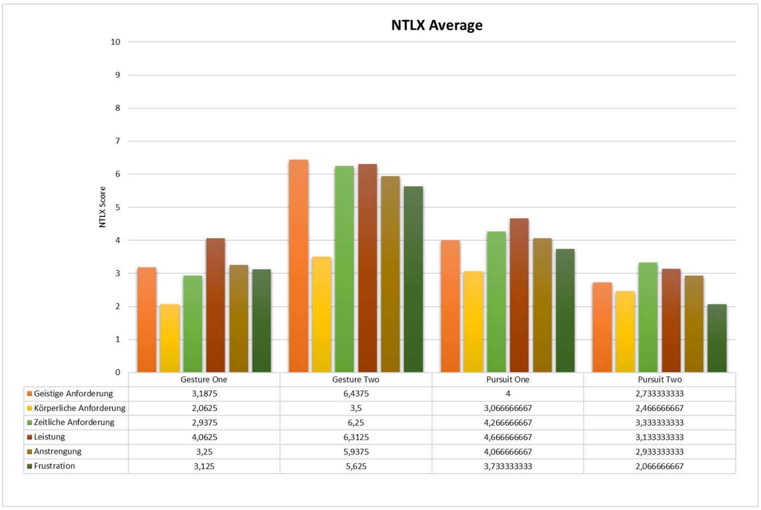


(d)



(e)

Figure 8.2: Results of the usability study



	Gesture One	Gesture Two	Pursuit One	Pursuit Two
■ Geistige Anforderung	3,1875	6,4375	4	2,733333333
■ Körperliche Anforderung	2,0625	3,5	3,066666667	2,466666667
■ Zeitliche Anforderung	2,9375	6,25	4,266666667	3,333333333
■ Leistung	4,0625	6,3125	4,666666667	3,133333333
■ Anstrengung	3,25	5,9375	4,066666667	2,933333333
■ Frustration	3,125	5,625	3,733333333	2,066666667

Figure 8.3: The NASA Task Load Index assesses the perceived workload, from subjects working with a human-computer interaction system [1]

input modalities, in case of the questionnaire, but also has the lowest level of necessary effort and also frustration. Gesture One takes the lead in physical (2.06) and temporal (2.94) demand. The combined input Gesture Two had the highest level of mental, physical and temporal demand. This undermines the mood shown in the questionnaire, where it was the least liked. It also has the highest effort and frustration level of all the presented methods.

The detailed result can be seen in figure 8.3.

8.7 Qualitative feedback

As shown in the questionnaire the participants vastly preferred fast entry methods over secure for unlocking their smart phones. This is also a concern mentioned in the qualitative feedback. It was said that the new methods need too much time and the standard PIN entry would be faster. When looking at the average time used for each entry method and the two preferred input methods were also the two fastest. Another participant liked the colorful design and wrote that this would help the system to gain a higher acceptance among new users. Another one thought that Gesture One and Gesture Two would be usable and was not content about the smooth pursuit moving objects. It was also mentioned that Gesture two was harder, because the PINs were longer and they thought it was too much to enter. A Concern with much higher importance was the false detection of entries which will lead to the device being locked, if too many false entries are made.

9 Discussion

Using Pursuits on mobile devices opens up new contact less interaction opportunities. Not needing additional calibration, makes Pursuits suitable for mobile scenarios, where the orientation of both the device and the user's head is not fixated and often changes. It also offers new means for authentication with a possible higher security to shoulder surfing attacks, since both the user's face and the display are needed to be seen, to detect the entered PINs.

In the first application used in the pilot study, 6 different combinations of trajectories were tested, with the result that a combination of linear and circular moving objects have the highest distinguishability. But even in the best-case scenario the detection rates were too low for a real-world application. This is due to different factors. First the participant's heads were not fixated with the optimal distance of 20 cm to the display. This lead the participants to move their heads in between the tests and leave the optimal range to the display. The idea behind this decision was to test the application in a more natural environment. Second we used a smart phone with a 4,95-inch screen instead of a tablet, so a higher accuracy is necessary for the same results, since the eye movements are also smaller. Third the lack of processing power, because the gaze estimation was done on the smart phone, fewer gaze points could be detected per second, than in a desktop environment and the front-facing camera also had a lower resolution than commercial counterparts.

When tested with eye gestures the effect of the latency becomes apparent. Since the latency of the detected eye movements was following behind the actual eye movements and the system needed some kind of calibration again, this was only usable at all for people who know the exact way the application works. It shows clearly that the devices still lack the needed processing power to have a fast-enough gaze estimation for real-time usage.

Although the results of the first study were negative, the usability study showed that not only Pursuits have a low demand in case of the NTLX, but the concept shown in Gesture One and especially Pursuit Two were perceived well. Over half of the participants would use a system with Pursuit Two on a daily basis without further benefits, as for example higher security for their input. When the hardware catches up to the demands for real-time gaze estimation, this might be of interest for future studies.

10 Future Work

In this section, different options for extending the work done in this thesis are suggested.

Algorithms Andre Pomp's framework offers to interchange the algorithms for gaze estimation. We tested the influence of different trajectories on the detection rates. This can be further developed if it is tested with different pupil detection and gaze estimation algorithms.

Post Processing The frame work offers synchronized screen and camera caption, used for post processing. To explore further limits of the hardware, in particular the limited resolution of the front-facing camera, the introduced tests should be redone with post processing.

Retesting with new hardware The capabilities of mobile devices improve with an astonishing speed. Since many of our issues seem to be mainly hardware related, it will be interesting to see how the application will perform with a higher resolution front camera and more processing power. Because the application exists already, this does not require much additional work an can be done in a future user study.

Input Modalities In the second lab study four different input modalities were presented. Since commercial eye tracking devices offer higher precision and in combination with the computation power of desktop PCs have proven to deliver real-time gaze estimation, the entry methods can be retested and studied in case of error rates.

Security study In the usability study, we have presented four different input modalities. The participants were filmed from three different angles while performing the different Pin entries. The collected video material is intended to be used in a future study, to research how secure the presented input methods are in case of shoulder surfing.

11 Conclusion

In this work, we implemented Pursuits into an existing low-cost computer vision based framework and conducted a user study in a lab setting, to explore its feasibility. Because Pursuits offer spontaneous and natural interaction and have proven to also offer robust interaction on public displays [6], in theory this concept is also suited for interaction with smart phones. But in the end the framework could not make the gaze estimation work with a high enough precision, to gain reliable detection rates for the selected objects. Neither the different combination of trajectories could improve the results for the Pursuit based interaction nor was the gaze estimation precise enough to make the eye gestures work reliably. It seems not viable to work on today's mobile Android devices. But because the processing power of the devices grows rapidly it is interesting to test it with a newer generation of processors and higher resolution front-facing cameras.

In a second study, where we conducted a Wizard of Oz experiment, we explored the usability of Pursuits and eye gestures. Although the implementation of Pursuits and eye gestures proved not to work, due to slow processing and low accuracy, both concepts could be presented in a manner which finds high acceptance from the participants. In these setups, the input modalities Gesture One and Pursuit Two, have shown low mental and physical demands, which also makes them suitable for daily use.

Contents of the enclosed DVD

- This thesis as a PDF and tex file
- Software project for the pilot study
- Software project for the usability study
- Software prototype for gesture detection
- Data from both studies, including CSV files
- The diagrams used in this thesis

References

[1] NASA Task Load Index nasa task load index.

[2] Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software &#38; Technology*, UIST '15, pages 457–466, New York, NY, USA, 2015. ACM.

[3] Alain Forget, Sonia Chiasson, and Robert Biddle. Shoulder-surfing resistance with eye-gaze entry in cued-recall graphical passwords. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1107–1110, New York, NY, USA, 2010. ACM.

[4] Oliver Hohlfeld, André Pomp, Jó Ágila Bitsch Link, and Dennis Guse. On the applicability of computer vision based gaze tracking in mobile scenarios. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 427–434, New York, NY, USA, 2015. ACM.

[5] Li Jie and James J. Clark. Video game design using an eye-movement-dependent model of visual attention. *ACM Trans. Multimedia Comput. Commun. Appl.*, 4(3):22:1–22:16, September 2008.

[6] Mohamed Khamis, Florian Alt, and Andreas Bulling. A field study on spontaneous gaze-based interaction with a public display using pursuits. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, pages 863–872, New York, NY, USA, 2015. ACM.

[7] Mohamed Khamis, Ozan Saltuk, Alina Hang, Katharina Stolz, Andreas Bulling, and Florian Alt. Textpursuits: Using text for pursuits-based interaction and calibration on public displays. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, pages 274–285, New York, NY, USA, 2016. ACM.

[8] Manu Kumar, Tal Garfinkel, Dan Boneh, and Terry Winograd. Reducing shoulder-surfing by using gaze-based password entry. In *Proceedings of the 3rd Symposium on Usable Privacy and Security*, SOUPS '07, pages 13–19, New York, NY, USA, 2007. ACM.

[9] Chris Lankford. Effective eye-gaze input into windows. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*, ETRA '00, pages 23–27, New York, NY, USA, 2000. ACM.

[10] LG. Nexus 5 lg product page.

[11] André Pomp. Using mobile gaze detection for user centric research. Master's thesis, RWTH Aachen, Germany, 2014.

[12] Luz Rello and Miguel Ballesteros. Detecting readers with dyslexia using machine learning with eye tracking measures. In *Proceedings of the 12th Web for All Conference*, W4A '15, pages 16:1–16:8, New York, NY, USA, 2015. ACM.

[13] DA Robinson. The mechanics of human smooth pursuit eye movement. *The Journal of Physiology*, 180(3):569, 1965.

[14] Ozan Saltuk. Experimenting with Pursuit Calibration on Public Displays. Master's thesis, LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN, Germany, 2016.

[15] Juni Nurma Sari, Ridi Ferdiana, Paulus Insap Santosa, and Lukito Edi Nugroho. An eye tracking study: Exploration customer behavior on web design. In *Proceedings of the International HCI and UX Conference in Indonesia*, CHIuXiD '15, pages 69–72, New York, NY, USA, 2015. ACM.

[16] Mélodie Vidal, Andreas Bulling, and Hans Gellersen. Pursuits: Spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 439–448, New York, NY, USA, 2013. ACM.

[17] Nicholas Wade and Benjamin W Tatler. *The moving tablet of the eye: The origins of modern eye movement research*. Oxford University Press, USA, 2005.

[18] Colin Ware and Harutune H. Mikaelian. An evaluation of an eye tracker as a device for computer input2. *SIGCHI Bull.*, 17(SI):183–188, May 1986.

[19] Erik Wästlund, Kay Sponseller, and Ola Pettersson. What you see is where you go: Testing a gaze-driven power wheelchair for individuals with severe multiple disabilities. In *Proceedings of the 2010 Symposium on Eye-Tracking Research &#38; Applications*, ETRA '10, pages 133–136, New York, NY, USA, 2010. ACM.

[20] Erroll Wood and Andreas 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, New York, NY, USA, 2014. ACM.

Appendices

Pursuits On Mobilephones

* Erforderlich

1. **Date of birth**

   ....

   Beispiel: 15. Dezember 2012

2. **Gender**

   Markieren Sie nur ein Oval.

   ⬭ Male
   ⬭ Female
   ⬭ Other

3. **Field of Study / Profession**

   ....

4. **Did you work with gaze based interaction before?** *

   Markieren Sie nur ein Oval.

   ⬭ Yes
   ⬭ No

5. **Length of your hair?**

   Markieren Sie nur ein Oval.

   ⬭ Short
   ⬭ Medium
   ⬭ Long

6. **It was easy to follow the linear moving objects** *

   Markieren Sie nur ein Oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strongly agree

7. **It was easy to follow the elliptic moving objects** *

   Markieren Sie nur ein Oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strongly agree

8. **It was easy to follow the curve motion** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

9. **Did it become harder or easier to follow the objects the longer you had to look at them?**
Markieren Sie nur ein Oval.

○ It became harder
○ It remained the same
○ It became easier

10. **How much time did you need to focus on the object again after they switched?** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
very long	○	○	○	○	○	very short

11. **How did you perceive the switching between objects?** *
Markieren Sie nur ein Oval.

	1	2	3	4	5	
very disturbing	○	○	○	○	○	very pleasant

12. **Test subject abbreviation** *

....

13. **What would you improve or change ?**

....
....
....

PhoneGazeOfOzRoundUp

* Erforderlich

1. **TestSubject**

   ....

2. **Do you prefer a fast or a secure method to enter a pin?** *

   Markieren Sie nur ein Oval.

   - ( ) Fast
   - ( ) Secure

3. **Do you prefer a fast or a secure method to unlock your mobile phone?** *

   Markieren Sie nur ein Oval.

   - ( ) Fast
   - ( ) Secure

4. **Which of the presented input methods would you use if it offers a higher security than the standard Pin entry methods?** *

   Wählen Sie alle zutreffenden Antworten aus.

   - [ ] Gesture One
   - [ ] Pursuit One
   - [ ] Gesture Two
   - [ ] Pursuit Two
   - [ ] None of the above

5. **Which of the presented input methods would you use on a daily basis?** *

   Wählen Sie alle zutreffenden Antworten aus.

   - [ ] Gesture One
   - [ ] Pursuit One
   - [ ] Gesture Two
   - [ ] Pursuit Two
   - [ ] None of the above

6. **What method are you using to unlock your phone?** *

7. **What do you think about the presented entry methods?** *

....

....

....

....